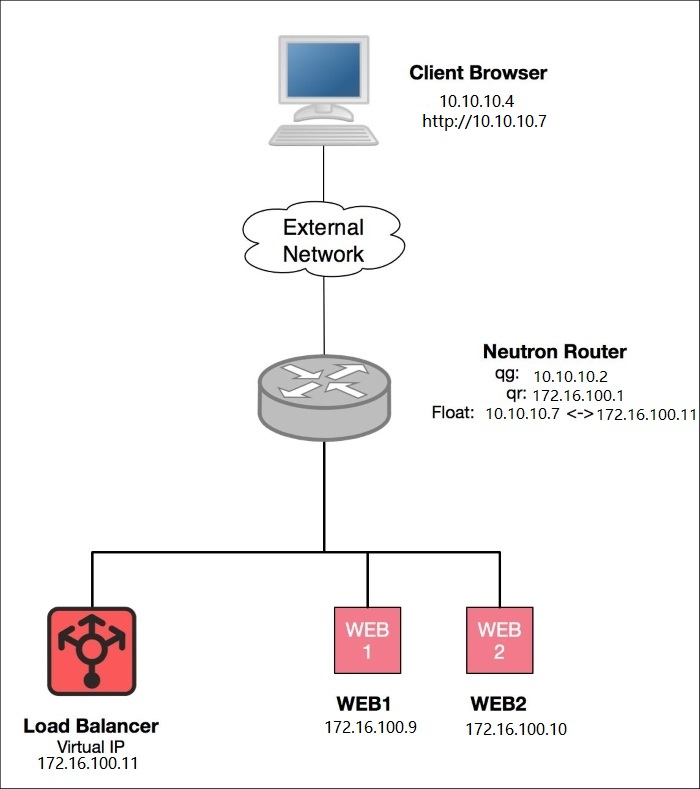


上节完成了 LBaaS 配置，今天我们开始实现如下 LBaaS 环境。

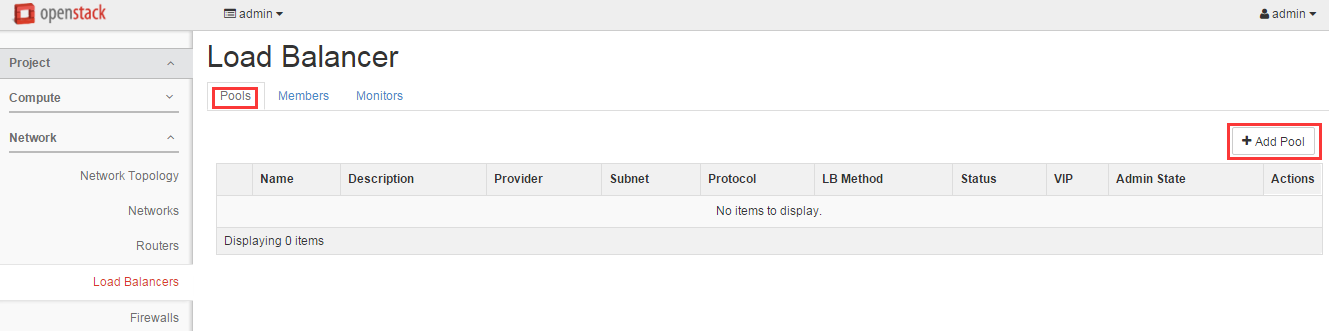


环境描述如下：  
1. 创建一个 Pool “web servers”。  
2. 两个 pool member “WEB1” 和 “WEB2”，均为运行 Ubuntu cloud image 的 instance。  
3. load balancer VIP 与 floating IP 关联。  
4. 位于外网的 client 通过 floating IP 外网访问 web server

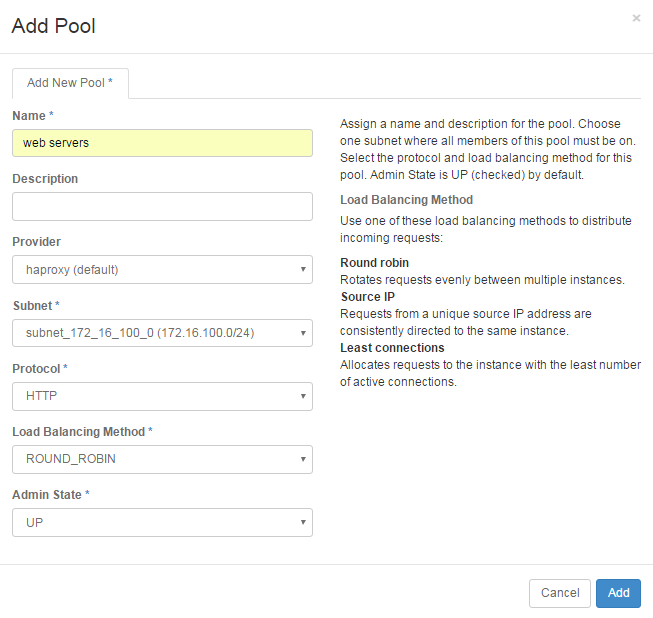
我们从第一步开始。

创建 Pool

点击菜单 Project -> Network -> Load Balancers，点击 Pools 标签页中的 “Add Pool” 按钮。

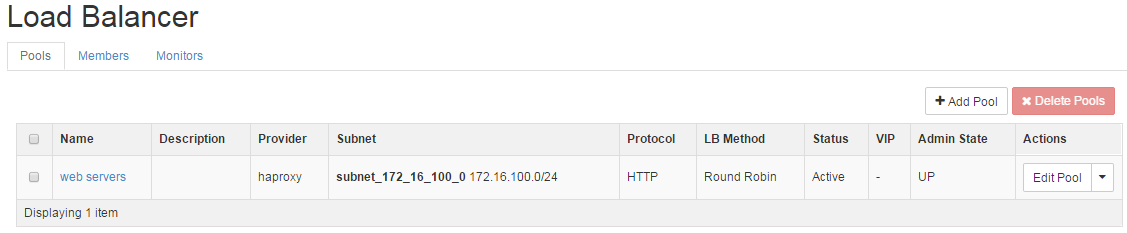


显示 Pool 创建页面。



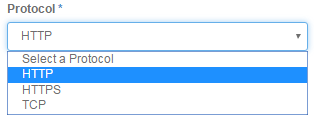
将 Pool 命名为“web servers”。   
Provider 选择默认的 “haproxy”。   
Subnet 选择 “172.16.100.0/24”。   
Protocol 选择 “HTTP”。   
Load Balancing Method 选择 “ROUND\_ROBIN”。

点击 “Add” 按钮，“web servers” 创建成功。



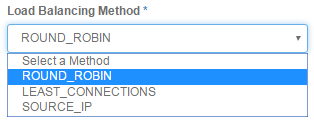
这里对 Pool 的几个属性进行一下说明。

LBaaS 支持如下几种 Protocol:



因为我们用 web server 做实验，所以这里需要选择 “HTTP”

LBaaS 支持多种 load balance method:



**ROUND\_ROUBIN**  
如果采用 round robin 算法，load balancer 按固定的顺序从 pool 中选择 member 相应 client 的连接请求。 这种方法的不足是缺乏机制检查 member 是否负载过重。 有可能出现某些 member 由于处理能力弱而不得不继续处理新连接的情况。 如果所有 pool member 具有相同处理能力、内存容量，并且每个连接持续的时间大致相同，这种情况非常适合 round robin，每个 member 的负载会很均衡。

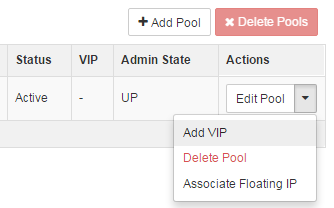
**LEAST\_CONNECTIONS**  
如果采用 least connections 算法，load balancer 会挑选当前连接数最少的 pool  member。 这是一种动态的算法，需要实时监控每个 member 的连接数量和状态。 计算能力强的 member 能够更快的处理连接进而会分配到更多的新连接。

**SOURCE\_IP**  
如果采用 source IP 算法，具有相同 source IP 的连接会被分发到同一个 pool member。 source IP 算法对于像购物车这种需要保存状态的应用特别有用，因为我们希望用同一 server 来处理某个 client 连续的在线购物操作。

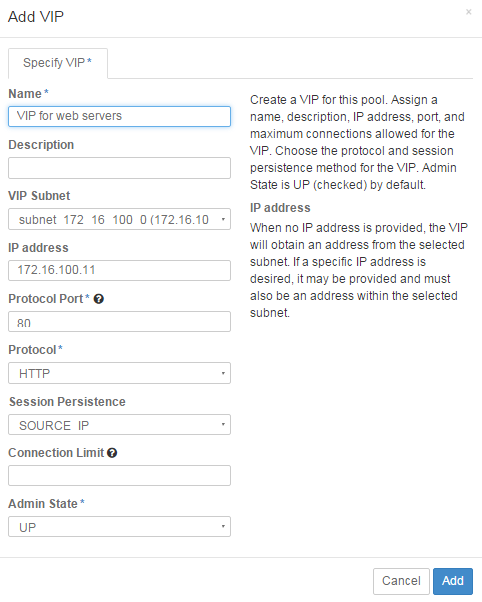
在我们的实验中选择的是 ROUND\_ROUBIN 算法。

为 Pool 添加 VIP

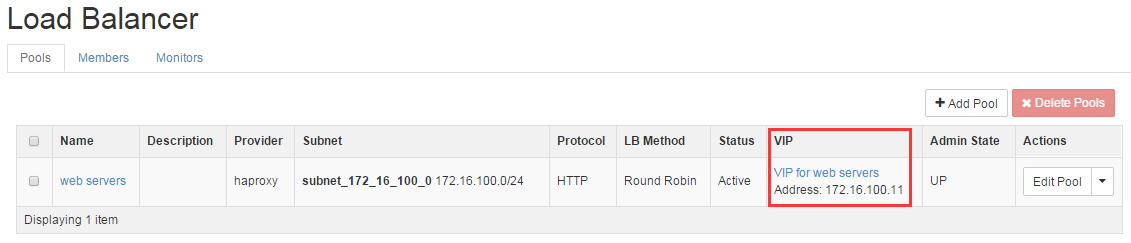
现在 Pool 已经就绪，接下需要为其设置 VIP。 在 “web servers” 的操作列表中点击 “Add VIP”。



VIP 命名为 “VIP for web servers”。   
VIP Subnet 选择 “172.16.100.0/24”，与 pool 一致。   
指定 VIP 为 172.16.100.11，如果不指定，系统会自动从 subnet 中分配。   
指定 HTTP 端口 80。   
Session Persistence 选择 “SOURCE IP”。   
可以通过 Connection Limit 限制连接的数量，如果不指定则为不加限制。

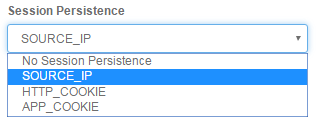


点击 “Add”，VIP 创建成功。



通常我们希望让同一个 server 来处理某个 client 的连续请求。 否则 client 可能会由于丢失 session 而不得不重新登录。

这个特性就是 Session Persistence。 VIP 支持如下几种 Session Persistence 方式：



**SOURCE\_IP**  
这种方式与前面 load balance 的 SOURCE\_IP 效果一样。 初始连接建立后，后续来自相同 source IP 的 client 请求会发送给同一个 member。 当大量 client 通过同一个代理服务器访问 VIP 时（比如在公司和学校上网），SOURCE\_IP 方式会造成 member 负载不均。

**HTTP\_COOKIE**

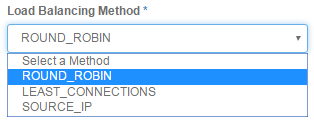
HTTP\_COOKIE 的工作方式如下： 当 client 第一次连接到 VIP 时，HAProxy 从 pool 中挑选出一个 member。 当此 member 响应请求时，HAProxy 会在应答报文中注入命名为 “SRV” 的 cookie，这个 cookie 包含了该 member 的唯一标识。 client 的后续请求都会包含这个 “SRV” cookie。 HAProxy 会分析 cookie 的内容，并将请求转发给同一个 member。

HTTP\_COOKIE 优于 SOURCE\_IP，因为它不依赖 client 的 IP。

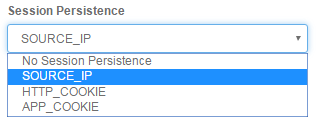
**APP\_COOKIE**  
app cookie 依赖于服务器端应用定义的 cookie。 比如 app 可以通过在 session 中创建 cookie 来区分不同的 client。  
HAProxy 会查看报文中的 app cookie，确保将包含 app cookie 的请求发送到同一个 member。  
如果没有 cookie（新连接或者服务器应用不创建 cookie），HAProxy 会采用 ROUND\_ROUBIN 算法分配 member。

**比较 Load Balance Method 和 Session Persistence**

前面我们介绍了三种 Load Balance Method：



这里还有三种 Session Persistence：



因为两者都涉及到如何选择 pool member，所以很容易混淆。 它们之间的最大区别在于选择 pool member 的阶段不同：

1. Load Balance Method 是为新连接选择 member 的方法
2. Session Persistence 是为同一个 client 的后续连接选择 member 的方法

例如这里我们的设置为：   
Load Balance Method -- ROUND\_ROUBIN  
Session Persistence -- SOURCE\_IP

当 client A 向 VIP 发送第一个请求时，HAProxy 通过 ROUND\_ROUBIN 选择 member1*。*对于 client A 后续的请求，HAProxy 则会应用 SOURCE\_IP 机制，仍然选择 member1 来处理请求。

Pool 创建完毕，下一节我们向 Pool 添加 Member