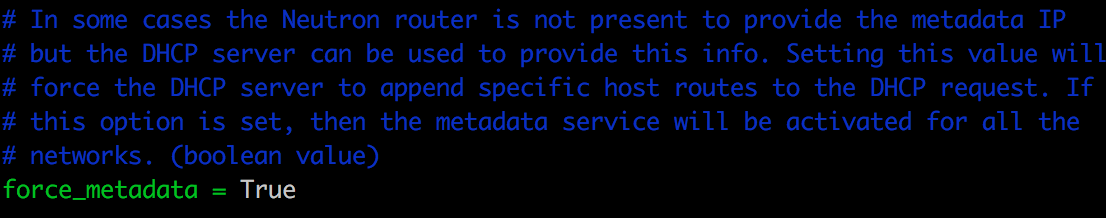
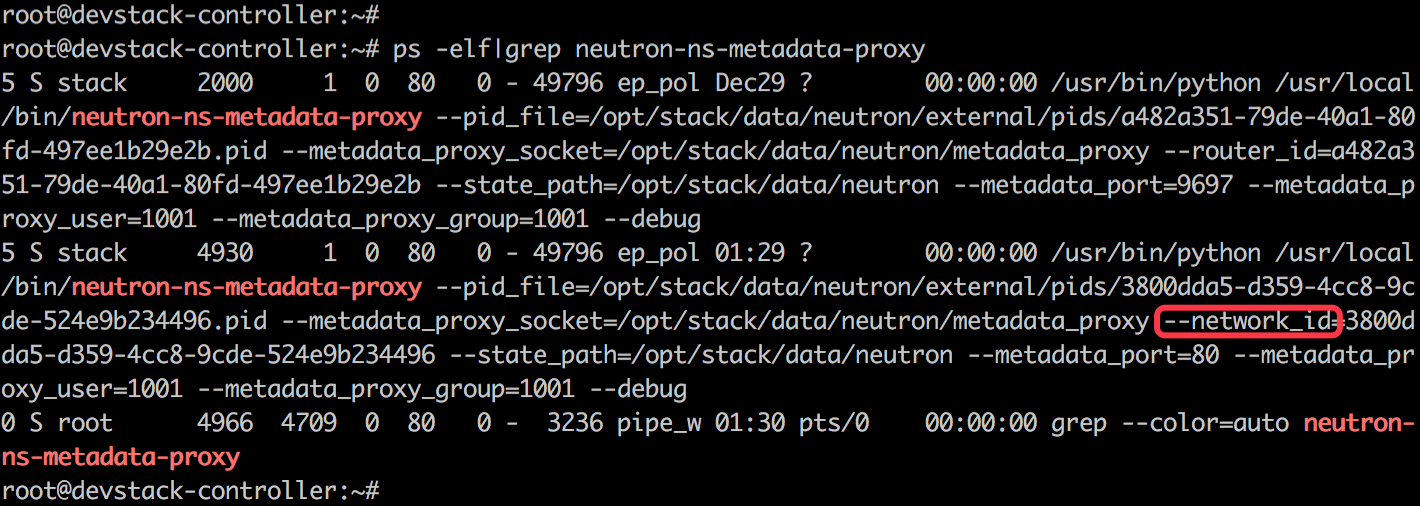
OpenStack 默认通过 l3-agent 创建和管理 neutron-ns-metadata-proxy，进而与 nova-metadata-api 通信。但不是所有环境都有 l3-agent，比如直接用物理 router 的场景。这时就需要走另一条路：让 dhcp-agent 来创建和管理 neutron-ns-metadata-proxy。

打开 /etc/neutron/dhcp\_agent.ini，设置 force\_metadata

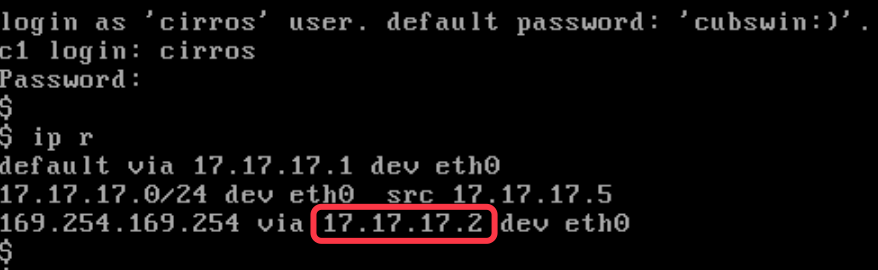


重启 dhcp-agent 后，可以看到控制节点上多了一个 neutron-ns-metadata-proxy 进程。

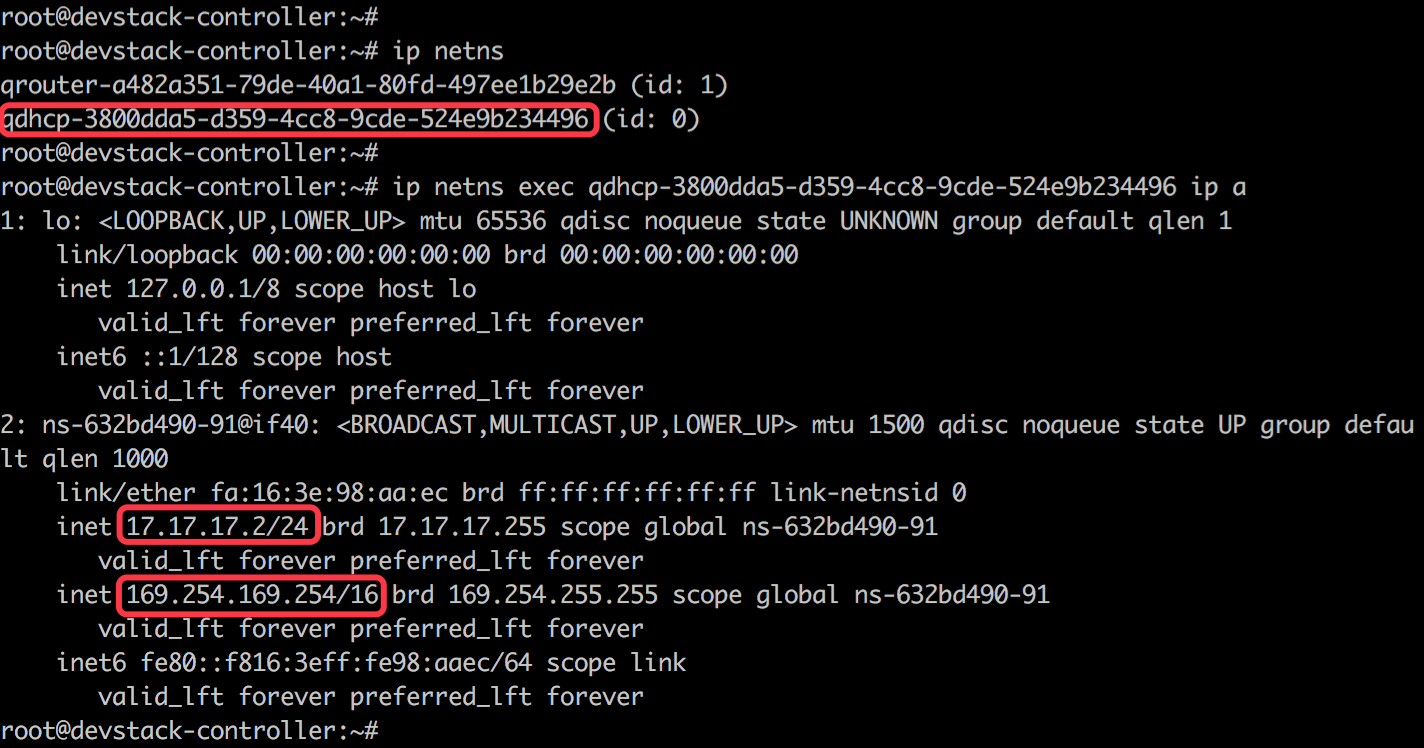


此进程通过 --network\_id 关联到 test\_net，这就是 dhcp-agent 启动的 neutron-ns-metadata-proxy，用于接收 test\_net 网络上 instance 的 metadata 请求。每一个 network 都有一个与之对应的 neutron-ns-metadata-proxy。

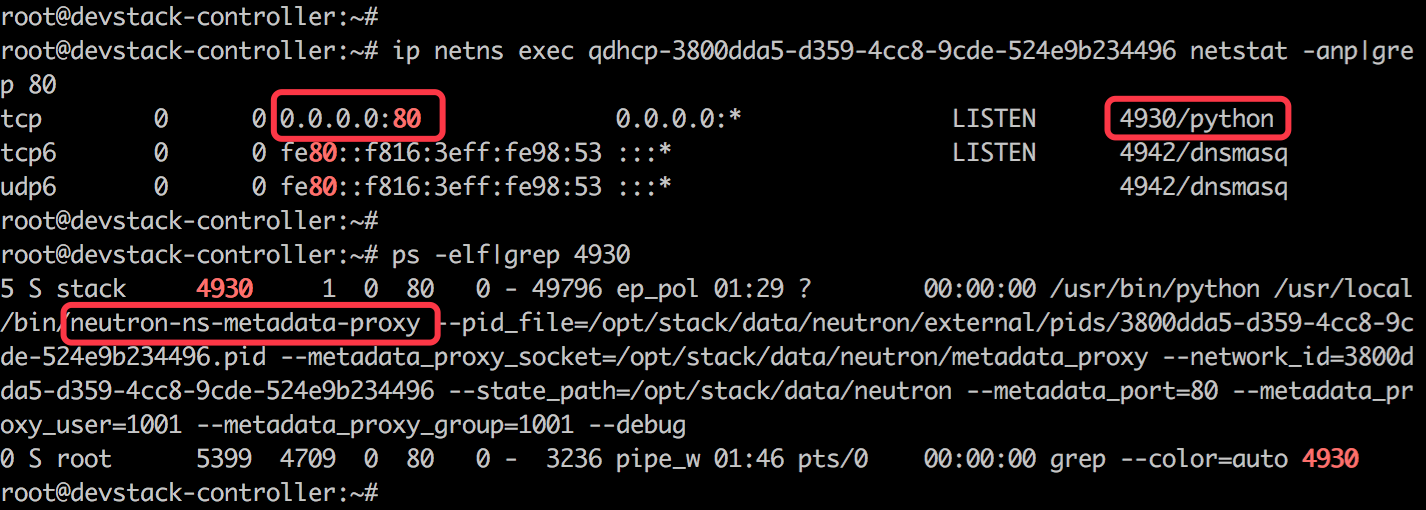
重启 instance c1，查看路由表。



请注意，现在访问 169.254.169.254 的路由已由之前的 17.17.17.1变为 17.17.17.2。这里的 17.17.17.2 是 dhcp-agent 在test\_net 上的 IP。这条路由是由 dhcp-agent 添加进去的。正是因为这条路由的存在，即便 l3-agent 与 dhcp-agent 同时提供 neutron-ns-metadata-proxy 服务，metadata 请求也只会发送给 dhcp-agent。



同时我们也看到，dhcp-agent 已经将 IP 169.254.169.254 配置到了自己身上。也就是说：c1 访问 metadata 的请求 [http://169.254.169.254](http://169.254.169.254/) 实际上是发送到了 dhcp-agent 的 80 端口。而监听 80 端口的正是 dhcp-agent 启动的 neutron-ns-metadata-proxy 进程。



后面的数据流向就与 l3-agent 的场景一样了：neutron-ns-metadata-proxy 将请求通过 unix domain socket 发给 neutron-metadata-agent，后者再通过管理网络发给 nova-api-metadata。

到这里，我们已经分别讨论了通过 l3-agent 和 dhcp-agent 访问 metadata 的实现方法。对于 169.254.169.254：

1. l3-agent 用 iptables 规则来转发。
2. dhcp-agent 则是将此 IP 配置到自己的 interface 上。

不知道大家有没有这样一个疑问：

nova-api-metadata 是怎么知道应该返回哪个 instance 的 metadata？c1 只是向 169.254.169.254 发送了一个 http 请求，nova-api-metadata 怎么就知道应该返回 c1 的 metadata 呢？  
  
下节咱们详细分析这个问题。