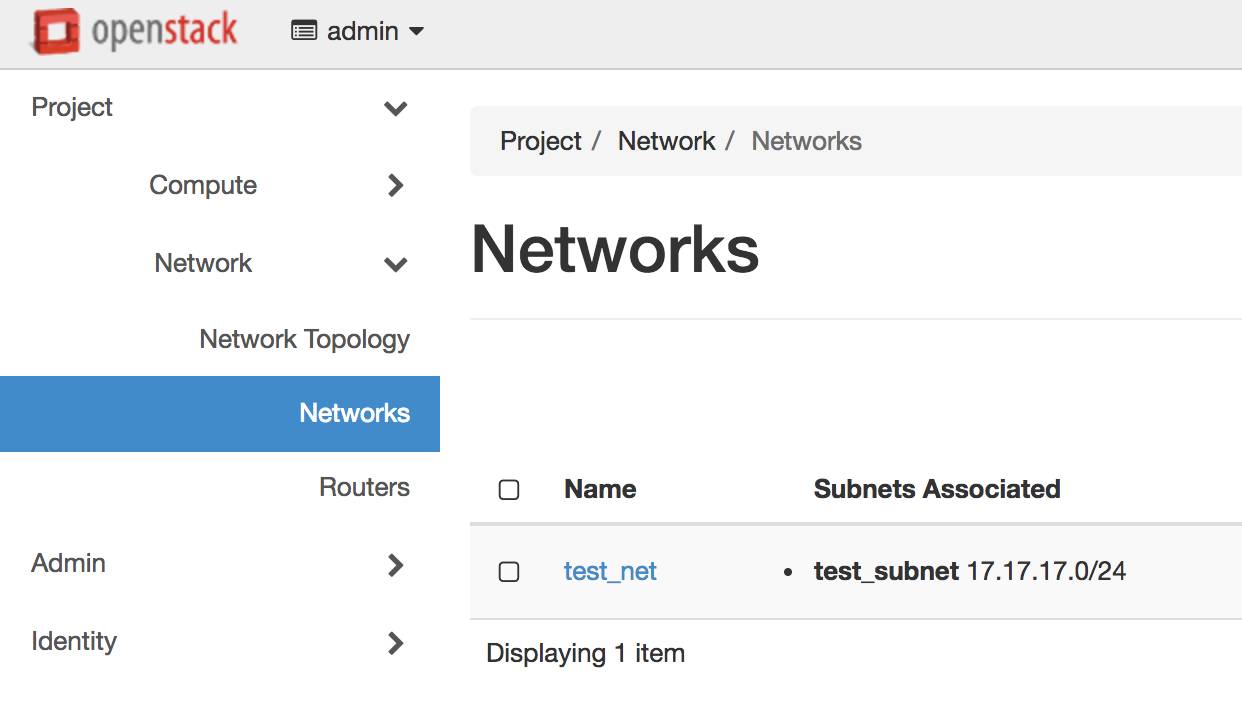
我们将通过实验详细分析 instance 从 nova-api-metadata 获取信息的完整过程。

**环境介绍**

1. 一个 all-in-one 环境（多节点类似）。

2. 已创建 neutron 网络 test\_net，DHCP 已启动。在这个 metadata 实验中， test\_net 的 type 不重要，flat、vlan、vxlan 都可以。

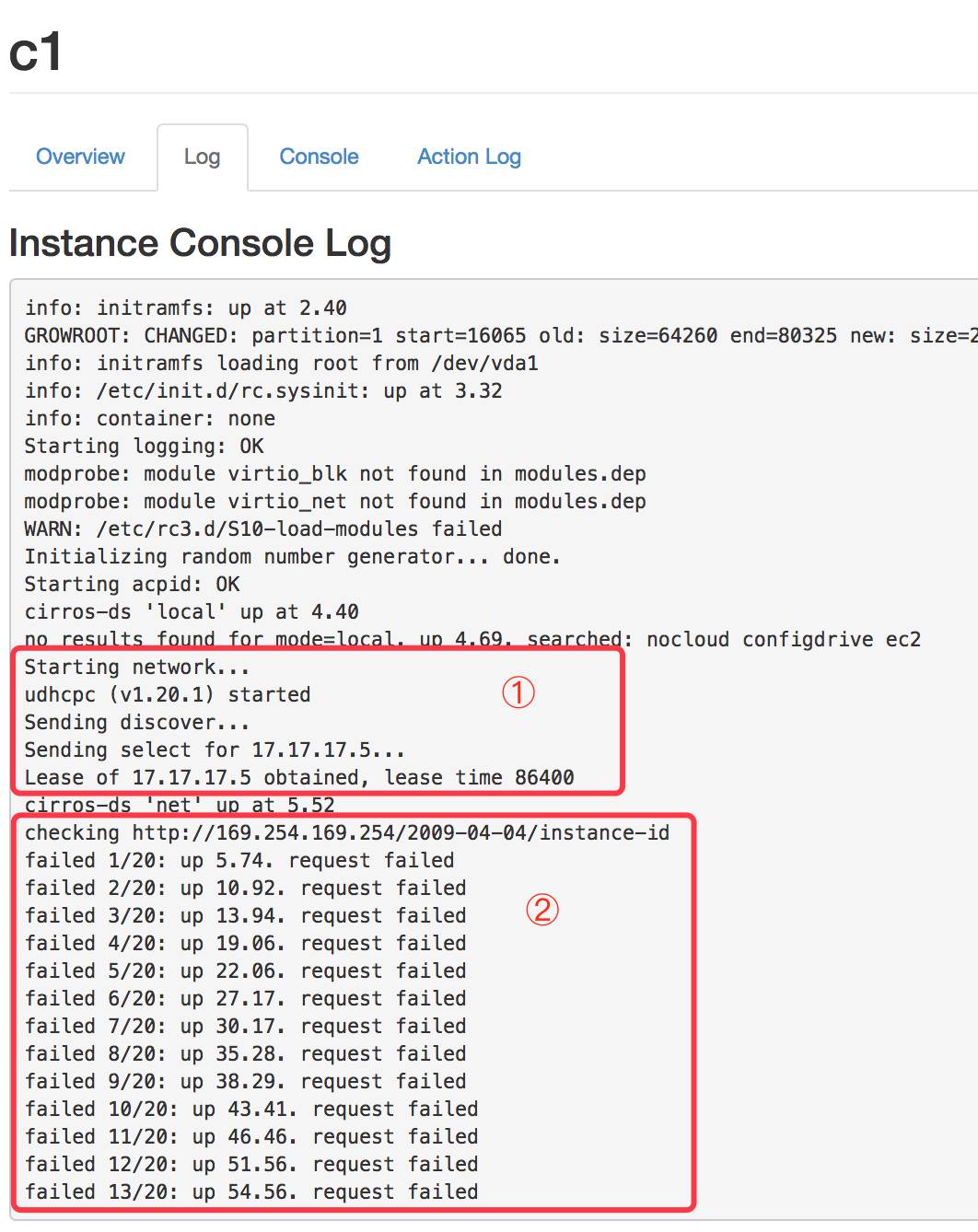


3. 暂无 neutron router。

准备就绪，开始实验。

**启动 instance**

通过 cirros 镜像部署一个 instance，命名为 c1，选择网络 test\_net。启动过程中，查看 instance 的启动日志。



上面的 log 中我们看到两个信息：

① instance 从 DHCP 拿到了 IP 17.17.17.5，这个好理解，因为我们在test\_net 上开启的 DHCP 服务。

② instance 会去访问 [http://169.254.169.254/2009-04-04/instance-id，尝试了](http://169.254.169.254/2009-04-04/instance-id%EF%BC%8C%E5%B0%9D%E8%AF%95%E4%BA%86) 20 次都失败了。

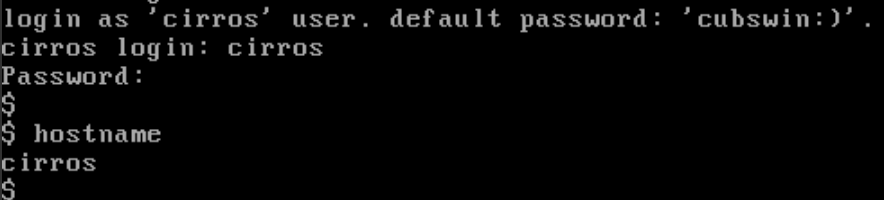
**神奇的 169.254.169.254**

169.254.169.254 是个什么地址？

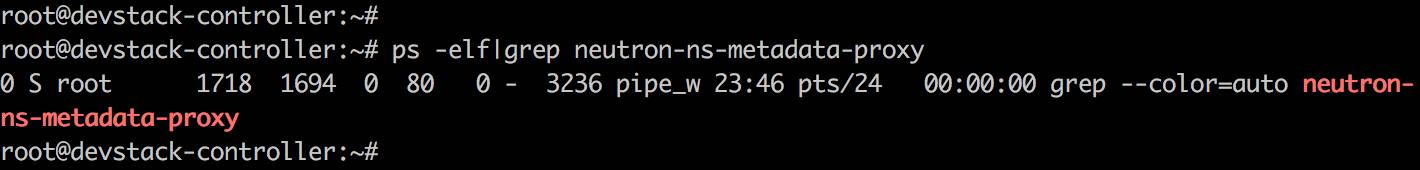
这是 metadata service 的 IP。

这个地址来源于 AWS，当年亚马逊在设计公有云的时候，为了让 instance 能够访问 metadata，就将 169.254.169.254 这个特殊的 IP 作为 metadata 服务器的地址，instance 启动时就会向 169.254.169.254 请求 metadata。OpenStack 之后也沿用了这个设计。

我们现在遇到的问题是 169.254.169.254 没法访问啊！cirros 的 cloud-init 显然是没有拿到 metadata 的，这点至少可以从 instance 的 hostname 没有被设置为 c1 判断出来。



前面我们在 Metadata Service 架构部分介绍了，instance 首先会将 metadata 请求发送给 DHCP agent 或者 L3\_agent 管理的 neutron-ns-metadata-proxy。那目前到底是谁在管理 neutron-ns-metadata-proxy 呢？我们先在控制节点上查看一下 neutron-ns-metadata-proxy 的进程。

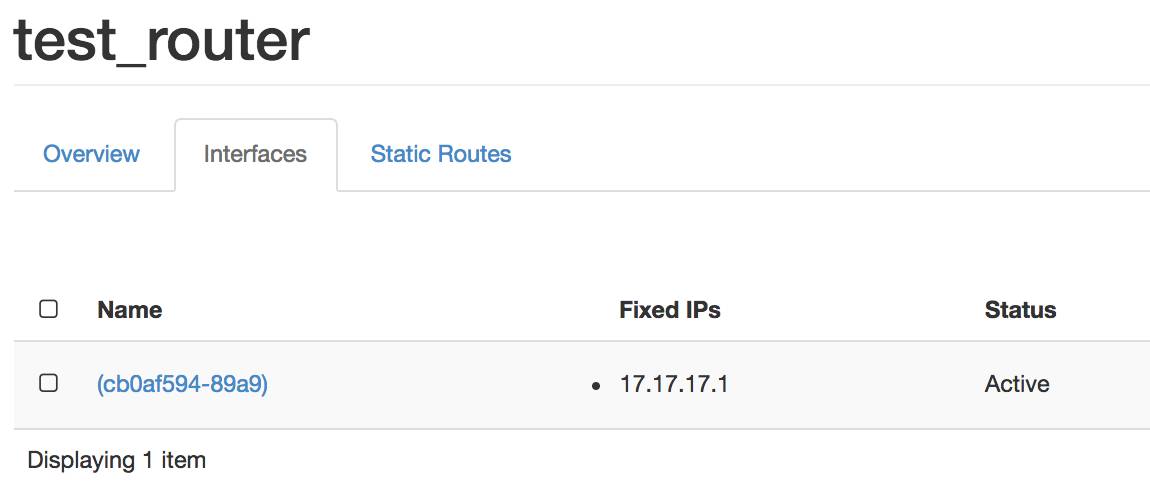


尽然没有 neutron-ns-metadata-proxy 在运行！

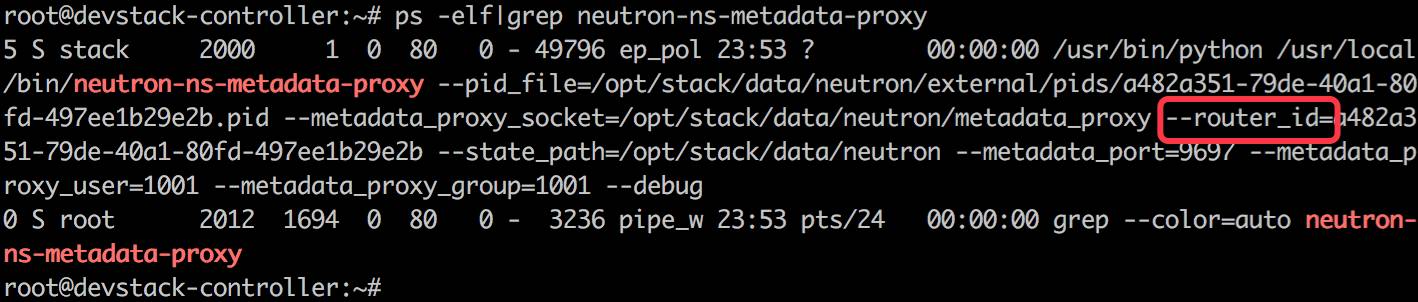
其原因是：默认配置下，neutron-ns-metadata-proxy 是由 L3\_agent 管理的（后面会讨论让 DHCP 来管理），由于当前 test\_net 并没有挂在 neutron router 上，所以没有启动 neutron-ns-metadata-proxy。

**添加 router**

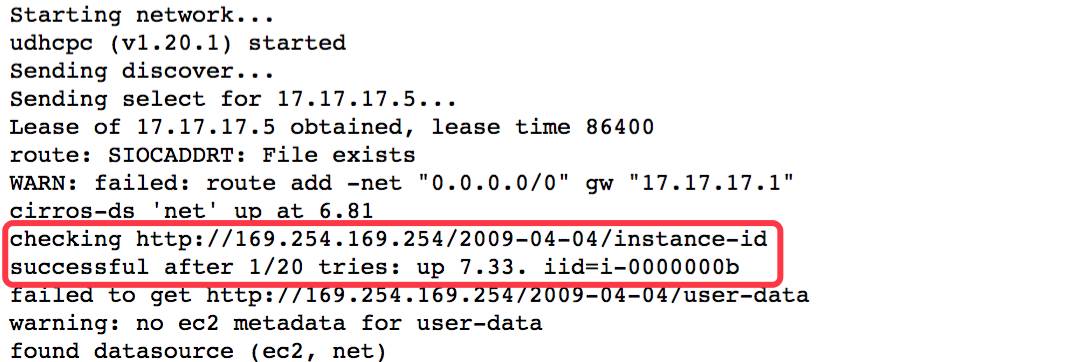
要解决这个问题很简单：创建虚拟路由器 test\_router 并连接 test\_net。



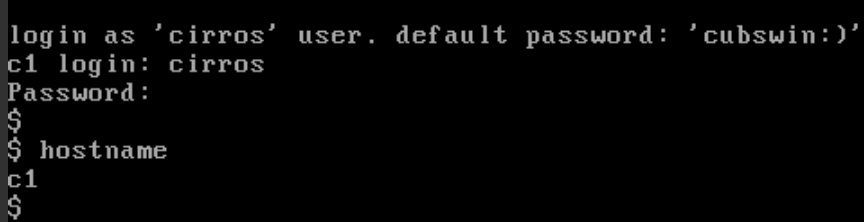
现在控制节点上已经能够看到 test\_router 管理的 neutron-ns-metadata-proxy 了。



重启 instance c1，看会发生怎样的变化。



instance 成功访问到 169.254.169.254。从结果看，cloud-init 已经获取到 metadata，因为 hostname 已经设置为 c1。



下一节我们详细分析 c1 是如何拿到 metadata 的。