

本节开始，我们将详细讲解 Nova 的各个子服务。

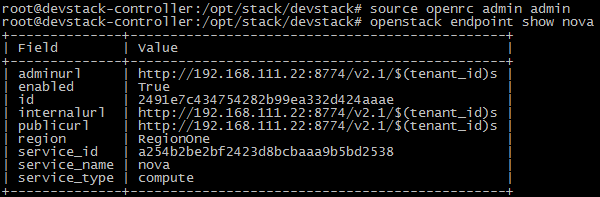
前面架构概览一节知道 Nova 有若干 nova-\* 的子服务，下面我们将依次学习最重要的几个。  
今天先讨论 nova-api 和 nova-conductor。

**nova-api**

Nova-api 是整个 Nova 组件的门户，所有对 Nova 的请求都首先由 nova-api 处理。

Nova-api 向外界暴露若干 HTTP REST API 接口。

在 keystone 中我们可以查询 nova-api 的 endponits。



客户端就可以将请求发送到 endponits 指定的地址，向 nova-api 请求操作。

当然，作为最终用户的我们不会直接发送 Rest AP I请求。

OpenStack CLI，Dashboard 和其他需要跟 Nova 交换的组件会使用这些 API。

Nova-api 对接收到的 HTTP API 请求会做如下处理：

1. 检查客户端传人的参数是否合法有效

2. 调用 Nova 其他子服务的处理客户端 HTTP 请求

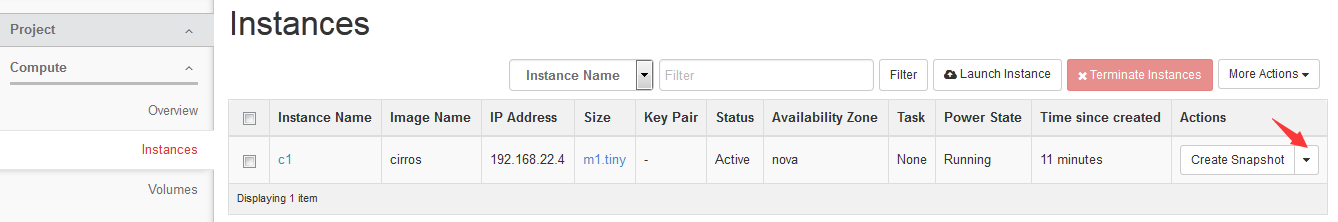
3. 格式化 Nova 其他子服务返回的结果并返回给客户端

nova-api 接收哪些请求？

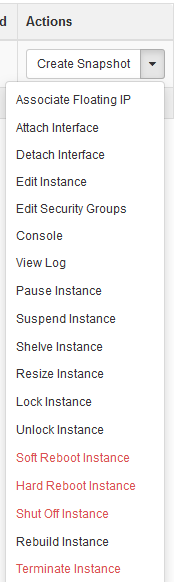
简单的说，只要是跟虚拟机生命周期相关的操作，nova-api 都可以响应。

大部分操作都可以在 Dashboard 上找到。

打开Instance管理界面



点击下拉箭头，列表中就是 nova-api 可执行的操作。

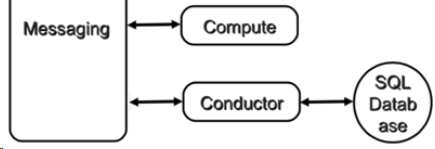


OpenStack 用术语 “Instance” 来表示虚拟机，后面我们将统一使用这个术语。

**nova-conductor**

nova-compute 需要获取和更新数据库中 instance 的信息。

但 nova-compute 并不会直接访问数据库，而是通过 nova-conductor 实现数据的访问。



这样做有两个显著好处：

1. 更高的系统安全性
2. 更好的系统伸缩性

**更高的安全性**

在 OpenStack 的早期版本中，nova-compute 可以直接访问数据库，但这样存在非常大的安全隐患。

因为 nova-compute 这个服务是部署在计算节点上的，为了能够访问控制节点上的数据库，就必须在计算节点的 /etc/nova/nova.conf 中配置访问数据库的连接信息，比如

[database]

connection = mysql+pymysql://root:secret@controller/nova?charset=utf8

试想任意一个计算节点被黑客入侵，都会导致部署在控制节点上的数据库面临极大风险。

为了解决这个问题，从 G 版本开始，Nova 引入了一个新服务 nova-conductor，将 nova-compute 访问数据库的全部操作都放到 nova-conductor 中，而且 nova-conductor 是部署在控制节点上的。

这样就避免了 nova-compute 直接访问数据库，增加了系统的安全性。

**更好的伸缩性**

nova-conductor 将 nova-compute 与数据库解耦之后还带来另一个好处：提高了 nova 的伸缩性。

nova-compute 与 conductor 是通过消息中间件交互的。

这种松散的架构允许配置多个 nova-conductor 实例。

在一个大规模的 OpenStack 部署环境里，管理员可以通过增加 nova-conductor 的数量来应对日益增长的计算节点对数据库的访问。  
  
下一节我们讨论计算节点调度服务 nova-scheduler.