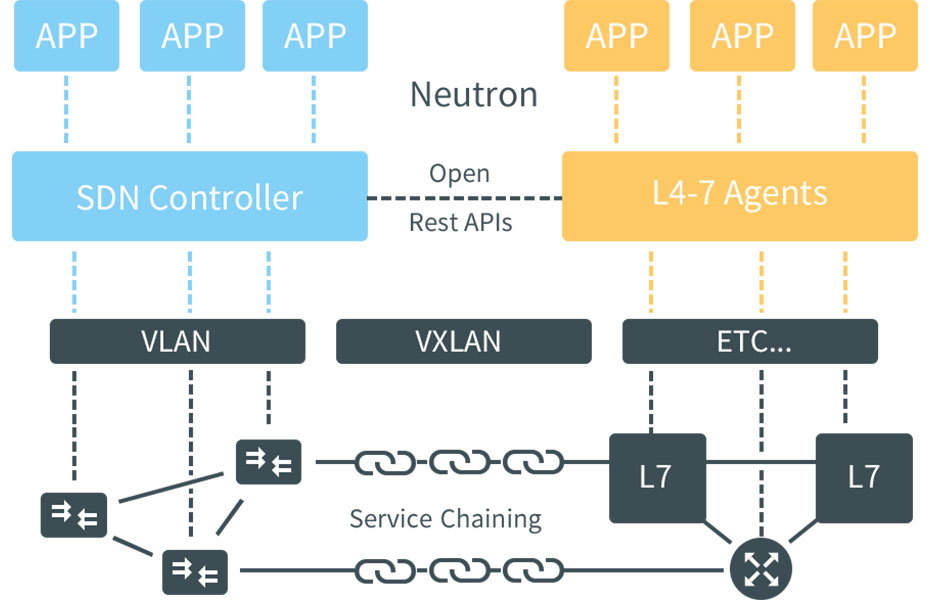
通过demo学习OpenStack开发所需的基础知识 — API服务(1)

使用OpenStack服务的方式

OpenStack项目作为一个IaaS平台，提供了三种使用方式：

* **通过Web界面**，也就是通过Dashboard（面板）来使用平台上的功能。
* **通过命令行**，也就是通过*keystone*, *nova*, *neutron*等命令，或者通过最新的*openstack*命令来使用各个服务的功能（社区目前的发展目标是使用一个单一的*openstack*命令替代过去的每个项目一个命令的方式，以后会只存在一个*openstack*命令）。
* **通过API**，也就是通过各个OpenStack项目提供的API来使用各个服务的功能。

上面提到的三种方式中，**通过API**这种方式是基础，是其他两种方式可行的基础。

通过Web界面使用OpenStack服务这种方式是通过OpenStack的**Horizon**项目提供的。Horizon项目是一个Django应用，实现了一个面板功能，包含了前后端的代码（除了Python，还包括了CSS和JS）。Horizon项目主要是提供一种交互界面，它会通过API来和各个OpenStack服务进行交互，然后在Web界面上展示各个服务的状态；它也会接收用户的操作，然后调用各个服务的API来完成用户对各个服务的使用。

通过命令行是用OpenStack服务的方式是由一系列项目来提供的，这些项目一般都命名为*python-projectclient*，比如*python-keystoneclient*，*python-novaclietn*等。这些命令行项目分别对应到各个主要的服务，为用户提供命令行操作界面和Python的SDK。比如*python-keystoneclient*对应到*keystone*，为用户提供了*keystone*这个命令，同时也提供了*keyston*项目的SDK（其实是在SDK的基础上实现了命令行）。这些client项目提供的SDK其实也是封装了对各自服务的API的调用。由于每个主要项目都有一个自己的命令行工具，社区觉得不好，于是又有了一个新的项目*python-openstackclient*，用来提供一个统一的命令行工具*openstack*（命令的名字就叫做openstack），这个工具实现了命令行，然后使用各个服务的client项目提供的SDK来完成对应的操作。

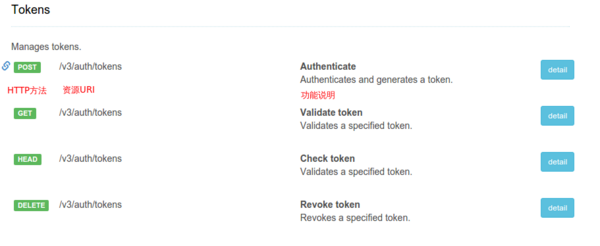
通过API使用OpenStack的方式是由各个服务自己实现的，比如负责计算的nova项目实现了计算相关的API，负责认证的keystone项目实现了认证和授权相关的API。这些API都是有统一的形式的，**都是采用了HTTP协议实现的符合REST规范的API**。OpenStack中如何实现这些API就是本文重点要将的内容。

基于HTTP协议的RESTful API

REST的全称是**Representational State Transfer**，中文翻译过来是**表征状态转移**，是*Roy Fielding*在他的博士论文[\*\*Architectural Styles and the Design of Network-based Software Architectures\*\*](https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm" \t "_blank)提出的一种软件架构风格。可以先到[wikipedia](https://zh.wikipedia.org/zh/REST" \t "_blank)页面了解一下这个风格的特点。一般会把满足这种设计风格的API成为**RESTful API**。由于这种软件设计风格非常适合采用HTTP协议来实现，因此HTTP协议是目前实现RESTful API的主要方案。

OpenStack就是基于HTTP协议和JSON来实现自己的RESTful API（之前OpenStack还有采用XML来表示数据的，现在都已经转到JSON了）。当一个服务要提供API时，它就会启动一个HTTP服务端，用来对外提供RESTful API。

OpenStack的API都是有详细的文档记录的，可以在[http://docs.openstack.org/](http://docs.openstack.org/" \t "_blank)看到所有的API的文档。每个API的文档形式如下：



当然，你可以点开**detail**看到详细的说明。从上面这个API的文档来看，你会觉得这个和开发网站时使用的GET方法和POST方法差不多，实际上也是差不多的，只不过对HTTP协议的使用方法做了满足REST风格的规定而已。

Python如何实现RESTful API

因为Python能够进行Web开发，所以用来开发RESTful API也就不成问题，这两者的技术基础是一样。在Python下开发RESTful API应用，无非是解决两个问题：

1. 服务如何部署？
2. 用什么框架开发？

服务如何部署？

说到Python的Web服务部署这个问题，就不得不提到WSGI。目前Python有两种方式来开发和部署一个Web应用：**用WSGI**和**不用WSGI**。如果你不了解WSGI，那么你需要先看下另外这篇关于WSGI的文章：[WSGI简介](http://segmentfault.com/a/1190000003069785)。

OpenStack的API服务都是使用WSGI的方式来部署的。在生产环境中部署WSGI，一般会考虑使用**Web服务器 + 应用服务器 + 应用(框架)**的方案。OpenStack官方推荐的是使用**Apache + mod\_wsgi**的方案，不过这个要换成其他方案也很容易，你也可以选**nginx + uWSGI**。对于开发调试的目的，有些项目也会提供使用**eventlet**的单进程部署方案，比如Keystone项目的**keystone-all**命令。采用eventlet这种异步架构来进行应用开发也是一个比较大的话题，本文不覆盖这方面的内容。

当然，也可以不用WSGI。在Python中，如果不使用WSGI的化，一般开发者会选择一些专门的服务器和框架，比如[Tornado](http://www.tornadoweb.org/en/stable/" \t "_blank)，或者最新最潮的[aiohttp](http://aiohttp.readthedocs.org/en/stable/" \t "_blank)。不过在OpenStack的项目中我还没见过不使用WSGI的。

用什么框架开发

Python的Web开发框架很多，最出名自然是Django了。基本上，还活跃的框架都支持RESTful API的开发，有些框架还专门为RESTful API的开发提供了便利的功能（比如Pecan），有些框架则通过第三方模块来提供这种便利，比如Django和Flask都有不少和REST相关的第三方库。

对于框架选择，也没有什么特别好的标准，一般都是比较性能、文档、社区是否活跃等。在我看来，选择流行的一般就不会错。

OpenStack中的RESTful API开发

上面已经谈到了OpenStack都是使用WSGI，也提到了部署方式。这一章来说一下OpenStack中使用的框架。

OpenStack项目倾向于**不重新发明轮子**，一般都会选择现有的库和框架来使用，除非现有的框架不满足需求。因为Web框架的选择很多，而且都满足需求，所以OpenStack项目到目前为止都是使用现成的Web框架。

OpenStack早期的项目并没有使用一个框架，而是使用了几个不同的模块来组合出一个框架：**Paste + PasteDeploy + Routes + WebOb**，这几个不同的模块分别负责应用的WSGI化、URL路由和请求处理等功能。Nova, Glance, Neutron, Keystone等早期的项目都是使用这样的架构来实现RESTful API的。

早期的这种技术选型带来的好处是”框架”具备足够的灵活性，缺点则是要把这几个模块组合起来实现一个REST服务，需要写很多代码，连WSGI的入口函数都要自己实现（比如Keystone项目的keystone/common/wsgi.py文件中的class Application）。因为灵活性的好处不是很明显，而代码量大的坏处很明显，比如上面那个class Application需要在每个项目中复制一遍，所以社区的新项目就开始使用新的Web框架**Pecan**。

Pecan是一个基于对象路由的框架，即灵活又简单。Pecan主要实现了URL路由功能，支持RESTful API。Pecan没有实现模板、session管理和ORM等功能，但是这些功能可以通过其他的模块来实现。对于OpenStack来说，Pecan是一个很好的选择，因为OpenStack项目中统一使用sqlalchemy来实现ORM，API的实现也不需要模板功能，安全控制则基于Keystone体系。使用Pecan来开发REST服务，代码量很少，代码结构也清晰。Ceilometer项目就是使用了Pecan。