



## About 云

零基础学习 openstack (完整篇) 【中级篇】

( 第四期 )

about 云为热爱云开发技术人员提供最全面的信息传播和服务平台，内容包括云技术文档，视频、云技术学习指导，解疑等。是大数据、云技术爱好者的学习分享基地。

about 云，本着活到老学到老的精神，为了广大云技术爱好者获取更多知识，在文章开头，都有几个问题，因此 about 云亦为学问社区。同样我们准备了每日一读，为了就是每天进步一点，每天能够学到新的内容。



QQ 群: 39327136、322273151、371358502、90371779

每日一读汇总腾讯认证空间：

<http://user.qqzone.qq.com/278595437/main>

## 问题导读

- 1.你是如何学习 **openstack** 的？
- 2.你对 **openstack** 的组件了解多少？
- 3.你认为 **openstack** 该如何学习？



本文链接: <http://www.aboutyun.com/thread-10306-1-1.html>

一直想写关于 openstack 的方面的内容，今天终于整理完成。算是完成一桩心事，内容整合来自：

[零基础学习 openstack（上）【中级篇】](#)

[零基础学习 openstack（下）【中级篇】](#)

是在

[零基础学习 openstack【初级篇】](#) 基础上的一个继续：

初级篇，我们主要是有这么一个概念，openstack 的组成 openstack 由哪些部分来组成：

- Identity(代号为“Keystone”)
- Dashboard(代号为“Horizon”)
- Image Service(代号为“Glance”)
- Network(代号为“Quantum”)
- Object Storage(代号为“Swift”)
- Block Storage(代号为“Cinder”)

及它们的初步认识，这篇，我们将深入这些概念，及对 openstack 的进一步的认识。

我们初级篇中，知道了如何部署集群，如何使用集群，但是遇到了很多的问题，

1. 什么是 floating ip?
2. 什么是管理网络?
3. 为什么会获取不到 ip?
4. 为什么网络会不通?
5. 为什么虚拟机 ping 通, 外网 ping 不同?
6. 出现问题了, 会看日志了, 可是还是找不出问题的原因?

这些都困扰着我们。

那么我们为什么会产生这些问题, 并且遇到这些问题, 还解决不了, 到处求救, 可是并非每次都那么幸运。

我们的学习方法, 通常有两个极端:

1. 先看书
2. 不管三七二十一, 先动手在说。

## 看书

看书是没有错的, 但是切勿较真, 因为很多书并不像《春秋》、《大学》、《易经》, 那么值得推敲, 现在的你看的更多的是一种框架, 一本书籍, 如果能够保证 98%以上都是正确的, 就已经很不错了。但是个人认为看书总比不看书要好一些, 毕竟花费了作者大量时间和精力去整理。

再回来, 我们在看书的时候, 有时候, 并不能一次就能把书看透, 使劲也是没用的, 因为水平在那。所以建议看书的时候, 先整体了解, 有了初步概念和自己的理解, 然后动手实践。

## 实践


有些同学, 注重实践, 因为实践才能获取知识, 其实这个没有错, 但是我们经常会遇到问题, 并且不知该如何解决。比如在大数据、云技术中, 搭建 **hadoop** 集群, **openstack** 集群, 调试的信息, 都在日志中了, 我们也知道看日志, 但是眼睁睁看日志, 却还是不知道问题出现在什么问题。**这时候你该怎么做?**

有三种办法?

1. 在 qq 群求助

这种效果一般不怎么好

## 2.在论坛社区发帖

论坛发帖一般也不怎么靠谱，当然除了 about 云以外 ，about 云对于发的帖子都会及时的回答。当我们遇到问题，解决问题的时候，我们就需要考虑，我们是否该看书了

## 3.回头看书

回头看书，这时候看书，你的收获很大，而且还会对以前的知识有一个重新理解和定义。

# 平时积累（爱好）

在读书和实践的过程中，其实还有更重要的一点，就是我们平时积累，这是任何学习方法都无法比拟的。没有比爱好更重要的。因为爱好所以琢磨，因为琢磨，所以有深度。

about 云也会为云技术、大数据爱好者，提供每日一读，网站和群（90371779、322273151）每天都会有相关内容

[about 云每日一读汇总](#)

以上方法是针对学习云技术、大数据，当然同样适合其它 IT 技术。

同时这里推荐一篇很不错的文章

[想学大数据、云技术、IT 人、大学生必读的一篇文章：如何快速掌握一门技术](#)

# 云计算、openstack 的理解

首先我们还是来说，什么是 openstack，什么是云计算。

引用百度：

云计算是一种通过网络以服务的方式提供动态可伸缩的虚拟化的资源的计算模式。

举个例子，你要做个网站，希望有一台独立的服务器，以前你可能得自行购买一台服务器并托管在 IDC 机房，不仅得花很多钱买服务器，而且每年要花很多钱托管（当然你也可以租一台服务器）。而现在，你可以在云计算服务商那里租一台同样由你掌握的“服务器”，你一样可以对它格式化，安装自己喜欢的操作系统和软件，但它并不是一台物理上的服务器，而且云计算平台上为你提供的一台虚拟机。

因此，云计算是由一系列可以动态升级和被虚拟化的资源组成，这些资源被所有云计算的用户共享并且可以方

便地通过网络访问，用户无需掌握云计算的技术，只需要按照个人或者团体的需要租赁云计算的资源。如果你真想了解，可以从[虚拟机](#)入手。简单讲，虚拟机是[云计算](#)的基础。

云计算的特点：

云计算拥有以下特点：

- 虚拟化 and 自动化
- 服务器，存储介质，网络等资源都可以随时替换
- 所有的资源都由云端统一管理
- 高度的伸缩性以满足业务需求
- 集中于将服务传递给业务

引用：[云计算是什么](#)

上边便是[云计算](#)的解释，但是我们仍旧感觉很模糊。

那么我们从 openstack 的角度来理解，说到 openstack 我们必须说一下，openstack 的发展，openstack 的初期，nova 是主要的组件，但是由于不断的扩展，所以逐渐的从 nova 中分离出来。

比如：

## 网络组件 nova-network

发展如下：

nova network→Quantum→Neutron

Openstack 在 2010 年正式发布它的第一个版本 Austin 的时候，nova-network 作为它的核心组件被包含其中原先网络有 nova network 来承担，后来逐渐分离出来，改名为 Quantum.

Quantum 是随 Openstack 的 Folsom 版本正式发布的，其实它已经作为试用组件包含在之前的 Essex 版本中。在 Grizzly 里功能得到了增强。

为什么引入 Quantum？答案非常简单，Quantum 功能更强大，满足更多需求。

## Neutron

因为商标侵权的原因，Openstack 在 Havana 版本上将 Quantum 更名为 Neutron，所以 Neutron 并不是什么新的东西。在 Havana 版里，Neutron 也只增加和增强了少数功能。

更详细信息参考

[OpenStack 网络组件 Neutron 的研究](#)

## nova-volume 组件

Essex 将 nove 的卷管理 api 独立化后，Folsom 终于将卷管理服务抽离成了 Cinder；Cinder 管理所有的块存储设备，块设备可以挂接在虚机的实例中，然后虚机里的 guest 系统可以像操作本地卷一样操作块存储设备；

Cinder 需要处理的主要问题应该是接入各种块设备，如本地磁盘、LVM 或各大厂商提供的设备如 EMC、NetApp、HP、HuaWei，还有如 Vmware 提供的虚拟块设备等。

从上面我们认识，nova 为 openstack 的重要组件，而 nova 中 nova-compute 则可以创建虚拟机。它也是云计算的核心。

所谓的云计算，从技术角度来讲，**其实就是能够灵活的创建和删除虚拟机。**

你或许会有很多的疑问或则不相信，为什么创建和删除个虚拟机就被称之为云计算，我们创建和删除虚拟机这不是很平常的一件事情吗？openstack 为什么会如此的火热。

到这里，让我们在来看看什么是云计算，或许有更进一步的认识。

引用百度：

云计算是一种通过网络以服务的方式提供动态可伸缩的虚拟化的资源的计算模式。

举个例子，你要做个网站，希望有一台独立的服务器，以前你可能得自行购买一台服务器并托管在 IDC 机房，不仅得花很多钱买服务器，而且每年要花很多钱托管（当然你也可以租一台服务器）。而现在，你可以在云计算服务商那里租一台同样由你掌握的“服务器”，你一样可以对它格式化，安装自己喜欢的操作系统和软件，但它并不是一台物理上的服务器，而且云计算平台上为你提供的一台**虚拟机**。

因此，**云计算**是由一系列可以动态升级和被虚拟化的资源组成，这些资源被所有云计算的用户共享并且可以方便地通过网络访问，用户无需掌握云计算的技术，只需要按照个人或者团体的需要租赁云计算的资源。

如果你真想了解，可以从虚拟机入手。**简单讲，虚拟机是云计算的基础。**

云计算的特点：

云计算拥有以下特点：

- 虚拟化 and 自动化
- 服务器，存储介质，网络等资源都可以随时替换
- 所有的资源都由云端统一管理
- 高度的伸缩性以满足业务需求
- 集中于将服务传递给业务

## 了解 openstack

当我们想学习大数据、云技术的时候，部署往往是我们的第一步，这样能够对 openstack 有一个直观的认识，比如那个文件需要修改，部署完成后该如何使用，详细参考：

[openstack 零基础入门：OpenStack Grizzly 安装指导（1）](#)

[openstack 零基础入门：OpenStack Grizzly 安装指导（2）](#)

更多内容参考

[零基础学习 openstack【初级篇】](#)

当然上面部署是一套 openstack 的部署，如果想单个部署，网上资料也还是不少的。

## openstack 开发

### 1.环境搭建

对于 openstack 开发，开发环境还是比较重要的，有了开发环境，我们可以阅读源码，同样可以修改里面的环境。那么我们该如何搭建开发环境。

比较可靠的方法，可以参考

[about 云课程 5：配置 Linux 中的 eclipse 环境，导入 openstack keystone 源码](#)

当然网上流行了比较多就是使用 dev 来搭建开发环境，由于各种原因，使用 dev 搭建过程中，可能会遇到比较多的问题，参考

[使用 DevStack 安装和配置 OpenStack 开发环境](#)

[基于 DevStack 的 Openstack folsom 版开发环境搭建（1-2）](#)

[基于 davstack 搭建 openstack folsom 开发环境（3-4）](#)

[OpenStack Nova 开发与测试环境搭建](#)

[准备 OpenStack 开发环境遇到的问题及解决办法](#)

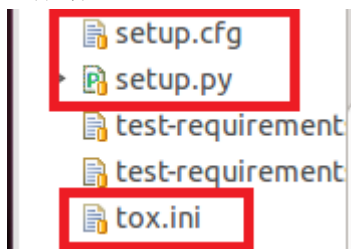
[建立 openstack quantum 开发环境](#)

## 2. 源码架构

当我们拿到[源码](#)的时候，我们如果直接每个文件查看源码，这个难度是相当大的，首先我们需要搞清楚源码

python 工程

一般有

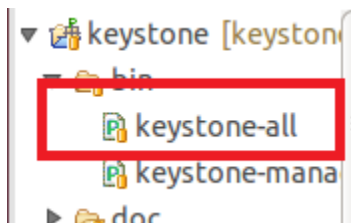


setup 文件

setup.cfg 配置文件

tox.nin 测试文件

bin 目录



keystone-all

keystone-manager

当项目启动时，[keystone-all](#) 为项目总入口

## openstack 各组件

我们了解了[云计算](#)，在我们技术人眼里，其实根本不是什么云计算，就是创建个虚拟机，而 openstack 就能完成这件事情。云计算之所以这么称呼当然有它的道理，当然咱们只关心技术，



所以咱们需要对 openstack 有一个深度的认识。

对于英语比较好的同学，学习可以直接访问官网，详细参考：

[新手指导：OpenStack 官网介绍](#)

官方地址：<https://github.com/openstack>

孵化项目：<https://github.com/stackforge>

首先我们需要对各个组件有一个认识，包括从原理、内部结构、部署、[源码](#)等角度。

## 1.了解认识 Nova

这个是最核心的，Nova 最开始的时候，可以说是一套虚拟化管理程序，还可以管理网络和存储。不过从 Essex 版本后，Nova 开始做减法，和网络相关的内容，包括安全组，交给 Neutron 负责，存储相关的交给 Cinder 负责。调度有关的内容，会交给新的项目 [Marconi](#)。

以前还有一个 nova common，这其实是各个组件都使用相同的东西，现在也专门成立一个项目：oslo，已经是核心项目。

未来 Nova 对各种 Hyperv 的支持是有差异的，KVM 和 XEN，基本是最好的。微软的 Hyper-V 算是很不错，微软投入再研发。计算节点，不直接查询数据库，而是通过 rpc 的方式，据说这是一大进步。

Nova 的稳定性，其实取决于 libvirt, qemu，希望未来可以能更加稳定。功能现在其实已经不是大问题。

那么我们再了解一下 nova。

nova 是一个很复杂的组件，而且内容很多。

认识 nova

nova 可以说是一套虚拟化管理程序，为什么这么说，因为 nova 可以创建、删除虚拟机、重启虚拟机等，openstack 的之所以能够搭建云平台，也是因为它能够创建虚拟机，其它的组件，比如 Neutron 则是为了让虚拟机之间、虚拟机与外网之间能够互通，Cinder 则是为了增加虚拟机的存储空间。可见 nova 在 openstack 中作用是非常大的。

更多内容，可以参考下面内容。

[大家谈 OpenStack-Nova 组件理解](#)

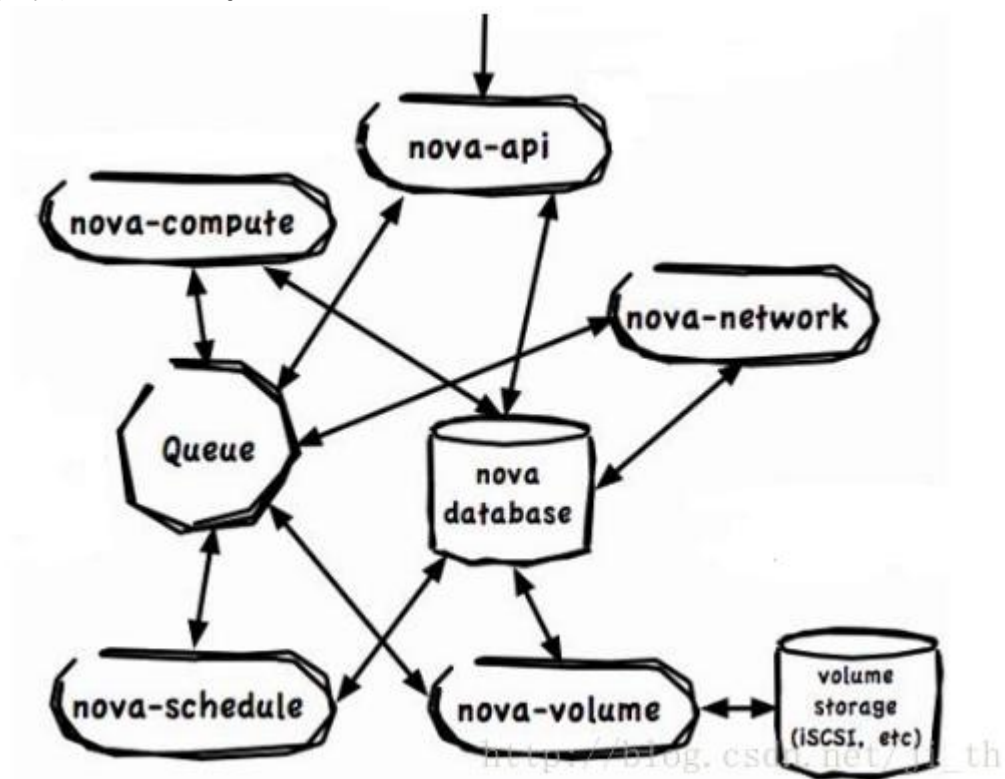
[关于 OpenStack 中 Nova 的几个基本概念](#)

[Openstack 核心, nova 详细介绍](#)

[OpenStack Compute\(Nova\)功能分析](#)

### nova 结构

nova 是云主机控制器。它包含了很多组件, API 服务器 (nova-api), 计算服务器 (nova-compute), 网络控制器 (nova-network), 调度器 (nova-schedule), 卷控制器 (nova-volume), 消息队列以及 DashBoard。



至于 nova 的发展, 上文我们已经介绍。

对于 nova 各个组件的作用, 及它们之间是如何通信的, 详细可以参考：

Nova 各个组件介绍以及功能分析(逻辑架构, 运行架构, 开发架构以及数据库)

<http://www.aboutyun.com/thread-10069-1-1.html>

### nova 命令行：

我们认识 nova 和了解了 nova, 那么 nova 具体该如何使用, 如何删除虚拟机、添加虚拟机、启

动虚拟机等，参考下面帖子。

openstack nova 用户管理

<http://www.aboutyun.com/thread-8717-1-1.html>

openstack nova 命令行指令大全

<http://www.aboutyun.com/thread-6373-1-1.html>

## nova 源码及开发

对于 nova 有了一定的认识，如果我们对 nova 二次开发

- 1.我们首先搭建开发环境
- 2.然后阅读源码修改源码

至于开发环境的搭建参考上文 openstack 开发，[源码](#)的阅读，这里有一些帖子供大家参考.对于内容，有的是对源码的整体认识，及[虚拟机](#)启动源码分析，更多的内容，相信能从标题可以看到。

## OpenStack Nova 源码结构解析

<http://www.aboutyun.com/thread-10105-1-1.html>

- 1.处理虚拟机磁盘镜像由哪个文件来完成？
- 2.调度器中的主机权重在哪个文件中？
- 3./nova/scheduler/host\_manager.py 文件的作用是什么？

## OpenStack 基于 Libvirt 的虚拟化平台调度实现----Nova 虚拟机动态迁移源码分析

<http://www.aboutyun.com/thread-10108-1-1.html>

- 1.实现[虚拟机](#)动态迁移主要实现的语句是什么？
- 2.方法\_update 实现了哪方面的内容？
- 3.live\_migration 方法的作用是什么？

## NOVA 源码分析——NOVA 中的 RabbitMQ 解析

<http://www.aboutyun.com/thread-10107-1-1.html>

- 1.终端用户（DevOps、Developers 和其他 OpenStack 组件）如何与 openstack 系统互动？
- 2.Nova 守护进程之间如何执行 API 请求？
- 3.RabbitMQ 是什么？
- 4.构成 AMQP 的三个关键要素，那么它们之间是如何工作的呢？

#### OpenStack Nova-cell 服务的源码解析（1）

<http://www.aboutyun.com/thread-10104-1-1.html>

- 1.nova-cell 服务的具体实现包含哪些流程？
- 2.哪个类定义了当路由信息到特定的 cell 上时，需要调用的方法？
- 3.schedule\_run\_instance 实现了什么？

#### OpenStack Nova-cell 服务的源码解析（2）

<http://www.aboutyun.com/thread-10103-1-1.html>

#### OpenStack Nova-scheduler 组件的源码解析（1）

<http://www.aboutyun.com/thread-10102-1-1.html>

- 1.哪个文件实现了基于随即选取主机节点的方式的调度器？
- 2./nova/scheduler/manager.py 文件作用是什么？
- 3./nova/scheduler/filters/affinity\_filter.py 定义了那四个过滤器？

#### OpenStack Nova-scheduler 组件的源码解析（2）

<http://www.aboutyun.com/thread-10121-1-1.html>

- 1.host\_state.update\_from\_compute\_node(compute)这条语句实现了什么功能？
- 2.哪一个函数循环实现了为每一个实例获取合适的主机后，返回选择的主机列表？
- 3.\_schedule 实现有哪三步？

OpenStack 基于 Libvirt 的虚拟化平台调度实现----Nova 虚拟机启动源码实现 (1)

<http://www.aboutyun.com/thread-10100-1-1.html>

- 1.Nova-Compute 中 Libvirt 默认调用的底层虚拟化平台是什么？
- 2.Libvirt 是什么？
- 3.Libvirt 哪些底层虚拟化平台？
- 4.一台虚拟机随着用户需求的改变可能会经历哪些状态？
- 5.哪个方法实现了确定来宾系统的磁盘映射信息？

OpenStack 基于 Libvirt 的虚拟化平台调度实现----Nova 虚拟机启动源码实现 (2)

<http://www.aboutyun.com/thread-10111-1-1.html>

- 1.类 Image 下的方法 cache 实现了什么功能？
- 2.哪个方法实现下载镜像文件？
- 3.方法 download 由那两部分组成？

OpenStack 基于 Libvirt 的虚拟化平台调度实现----Nova 虚拟机动态迁移源码分析

<http://www.aboutyun.com/thread-10108-1-1.html>

OpenStack 基于 Libvirt 的虚拟化平台调度实现----Nova 虚拟机启动源码实现 (3)

<http://www.aboutyun.com/thread-10110-1-1.html>

- 1.哪个方法实现了获取元数据？
- 2.对文件注入代码了解多少？
- 3.哪个方法实现向磁盘镜像注入不同的文件信息？

OpenStack 基于 Libvirt 的虚拟化平台调度实现----Nova 虚拟机启动源码实现 (4)

<http://www.aboutyun.com/thread-10109-1-1.html>

- 1.\_create\_domain\_and\_network 你认为完成了什么？
- 2.inst\_path = libvirt\_utils.get\_instance\_path(instance)语句的作用是什么？
- 3.domain.createWithFlags(launch\_flags)实现什么功能？

openstack nova 源码分析 1-setup 脚本

<http://www.aboutyun.com/thread-10090-1-1.html>

openstack nova 源码分析 2 之 nova-api,nova-compute

<http://www.aboutyun.com/thread-10091-1-1.html>

openstack nova 源码分析 3-nova 目录下的 service.py、driver.py

<http://www.aboutyun.com/thread-10092-1-1.html>

1.nova 下的 service.py 的源码主要完成什么任务？

2.driver.py 位于哪个目录下？

openstack nova 源码分析 4-1 -nova/virt/libvirt 目录下的 connection.py

<http://www.aboutyun.com/thread-10094-1-1.html>

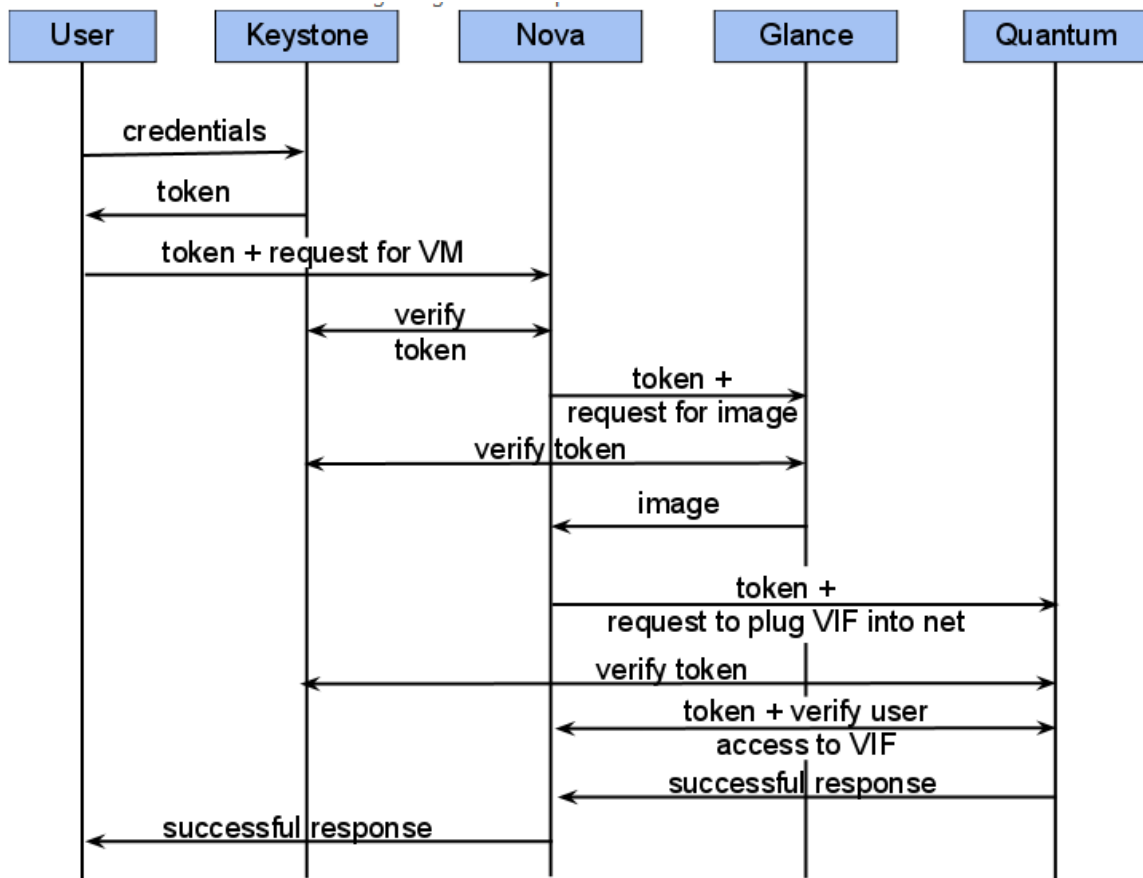
openstack nova 源码分析 4-2 -nova/virt/libvirt 目录下的 connection.py

<http://www.aboutyun.com/thread-10095-1-1.html>

## 2.了解认识 keystone

这是提供身份认证和授权的组件。任何系统，身份认证和授权，其实都比较复杂。尤其 Openstack 那么庞大的项目，每个组件都需要使用统一认证和授权。

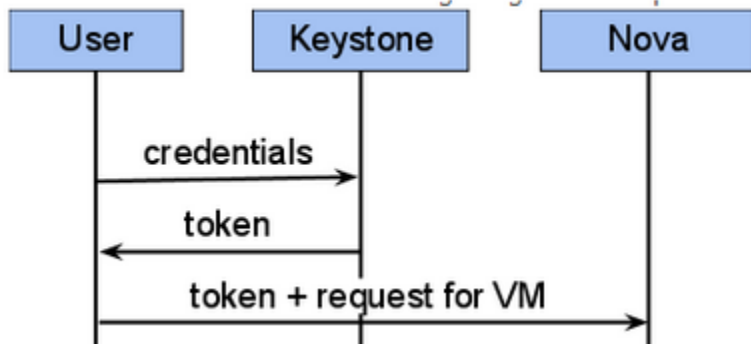
认识 keystone



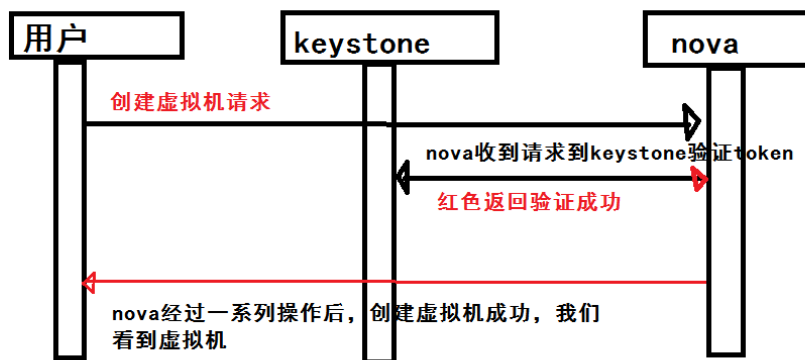
我们或许知道 **keystone** 是用来身份验证的，但是它是如何身份验证的，很多人或许不太清楚。上面的图示，当用户进行操作的时候，用户首先发送用户名和密码给 **Keystone**，（这里的用户名和密码，则是我们配置的环境变量，）然后获取 **token**，**token** 是什么？我们称之为令牌，有了这个令牌在请求资源，那么就畅通无阻了，我们为什么需要令牌，用户名和密码不也一样吗？如果作为一名程序员，我们都知道两个字段的对比与一个字段的对比在性能上是不一样的，何况是**云计算**组件之间通信是非常频繁的，所以个人认为了身份验证方便，所以产生了令牌（**token**）。

令牌的作用是什么，为什么需要令牌，我们就需要仔细看上图了。  
比如

下图用户请求创建虚拟机，截图如下，然后 **nova** 最后经过上图中操作，最后操作成功



我们将上图简化



用户带着 token 到 Nova 去请求虚拟机，nova 这时候需要验证这个 token 是否有效，自己无法判断，所以必须去 keystone 去验证，由于 keystone 记录了由它产生的 token，所以对照一下，就能知道是否有效，如果有效，返回 nova 验证成功。这时候 nova 经过一系列的操作，创建虚拟机，最后创建成功。

## keystone 包含的概念

### 1. User

User 即用户，他们代表可以通过 keystone 进行访问的人或程序。Users 通过认证信息（credentials，如密码、API Keys 等）进行验证。



## 2. Tenant

**Tenant** 即租户，它是各个服务中的一些可以访问的资源集合。例如，在 **Nova** 中一个 **tenant** 可以是一些机器，在 **Swift** 和 **Glance** 中一个 **tenant** 可以是一些镜像存储，在 **Quantum** 中一个 **tenant** 可以是一些网络资源。**Users** 默认的总是绑定到某些 **tenant** 上。

## 3. Role

**Role** 即角色，**Roles** 代表一组用户可以访问的资源权限，例如 **Nova** 中的虚拟机、**Glance** 中的镜像。**Users** 可以被添加到任意一个全局的 或 租户内的角色中。在全局的 **role** 中，用户的 **role** 权限作用于所有的租户，即可以对所有的租户执行 **role** 规定的权限；在租户内的 **role** 中，用户仅能在当前租户内执行 **role** 规定的权限。

## 4. Service

**Service** 即服务，如 **Nova**、**Glance**、**Swift**。根据前三个概念（**User**，**Tenant** 和 **Role**）一个服务可以确认当前用户是否具有访问其资源的权限。但是当 **user** 尝试着访问其租户内的 **service** 时，他必须知道这个 **service** 是否存在以及如何访问这个 **service**，这里通常使用一些不同的名称表示不同的服务。在上文中谈到的 **Role**，实际上也是可以绑定到某个 **service** 的。例如，当 **swift** 需要一个管理员权限的访问进行对象创建时，对于相同的 **role** 我们并不一定也需要对 **nova** 进行管理员权限的访问。为了实现这个目标，我们应该创建两个独立的管理员 **role**，一个绑定到 **swift**，另一个绑定到 **nova**，从而实现对 **swift** 进行管理员权限访问不会影响到 **Nova** 或其他服务。

## 5. Endpoint

**Endpoint**，翻译为“端点”，我们可以理解它是一个服务暴露出来的访问点，如果需要访问一个服务，则必须知道他的 **endpoint**。因此，在 **keystone** 中包含一个 **endpoint** 模板（**endpoint template**，在安装 **keystone** 的时候我们可以在 **conf** 文件夹下看到这个文件），这个模板提供了所有存在的服务 **endpoints** 信息。一个 **endpoint template** 包含一个 **URLs** 列表，列表中的每个 **URL** 都对应一个服务实例的访问地址，并且具有 **public**、**private** 和 **admin** 这三种权限。**public url** 可以被全局访问（如 <http://compute.example.com>），**private url** 只能被局域网访问（如 <http://compute.example.local>），**admin url** 被从常规的访问中分离。

很多人觉得比较难以，更多可以参考 [openstack 中 tenant 的作用到底是什么](#)

### keystone 命令

**keystone** 都有哪些操作，**keystone** 可以创建租户、角色、用户，详细参考 [openstack 之 role 篇](#)

[openstack 之 user 篇](#)

## [openstack 之 tenant 篇](#)

### keystone 源码

源码的阅读，如果感兴趣，可以自己有一个理解，然后与作者进行对比，这样才会对自己的思想有所提高，也算是与作者的思想交流。当然作者也会有错的，所以通过彼此的角，这样达到提高的目的

#### [openstack][G 版]keystone 源码记录

<http://www.aboutyun.com/thread-10136-1-1.html>

- 1.在 G 版中密码和 token 两个验证方法由哪个文件来实现？
- 2.WSGImiddleware 在 keystone 应用中的作用是什么？

#### Openstack 之 keystone 源代码分析 1--WSGI 接口流程分析

<http://www.aboutyun.com/thread-10137-1-1.html>

- 1.keystone 是怎么通过 WSGI 接口访问其中的服务的？
- 2.你认为 add\_routes 作用是什么？

#### Openstack 之 keystone 源代码分析 2--Controller->Manager->Driver

<http://www.aboutyun.com/thread-10138-1-1.html>

- 1.Driver 在那个配置文件中可配置？
- 2.Manager 怎么调用 conf 下面配置的或者默认的 driver 的？

#### Openstack 源代码分析之 keystone 服务 (keystone-all)

<http://www.aboutyun.com/thread-10139-1-1.html>

- 1.keystone-all.py 的作用是什么？
- 2.Python 应用程序使用 WSGI (Web Server Gateway Interface) 协议来做什么？

### 3.了解认识 Neutron

Neutron 是 OpenStack 核心项目之一，提供[云计算](#)环境下的虚拟网络功能。Neutron 的功能日益强大，并在 Horizon 面板中已经集成该模块。作为 Neutron 的核心开发者之一，个人觉得 Neutron 完全代替 Nova Network 模块作为云计算网络管理中心是必然趋势。要使用好 OpenStack,了解 Neutron 概念及其相应操作就显得格外重要。

Neutron 对于开发人员为什么难以理解：

初学者很难理解 Neutron，这是因为网络不在是实实在在的网线、路由等，都是通过命令来实现的。

例如使用 openvswitch 创建网桥

```
1. brctl addbr qbr02
```

在比如添加路由 router01

```
1. ip netns add router01
```

这些都是虚拟化的，也就是说在[虚拟机](#)之间，也就是在云中，网络都是虚拟化，所以我们才会觉得难以理解。

同样对于一些概念也比较模糊，比如

#### 固定 IP

私有 IP 地址，用于租户实例间通信

#### 浮动 IP

公共 IP 地址，用于实例与外部或 Internet 的通信

特别是浮动 IP 很多不太理解，一个网卡如果赋予的 ip 能够与外部 Internet 通信，那么它就是浮动 ip。

公共 IP 地址不一定是 Internet 上可路由的地址，也可以是站点内部或局域网的地址

私有地址和公共地址的关系以及必要的路由由 nova-network 来处理，实例不必考虑此问题。

在我們有了一定的理解，在回頭看進行一些，我們又會對 openstack 有一個新的認識。

下面我們從基礎開始認識 Neutron

## Neutron 基本概念

### 網絡

在普通人的眼里，網絡就是網線和供網線插入的端口，一個盒子會提供這些端口。對於網絡工程師來說，網絡的盒子指的是交換機和路由器。所以在物理世界中，網絡可以簡單地被認為包括網線，交換機和路由器。當然，除了物理設備，我們還有軟的物件：IP 地址，交換機和路由器的配置和管理軟件以及各種網絡協議。要管理好一個物理網絡需要非常深的網絡專業知識和經驗。

Neutron 網絡目的是（為 OpenStack 雲更靈活地）劃分物理網絡，在多租戶環境下提供給每個租戶獨立的網絡環境。另外，Neutron 提供 API 來實現這種目標。Neutron 中“網絡”是一個可以被用戶創建的对象，如果要和物理環境下的概念映射的話，這個对象相當於一個巨大的交換機，可以擁有無限多個動態可創建和銷毀的虛擬端口。

### 端口

在物理網絡環境中，端口是用於連接設備進入網絡的地方。Neutron 中的端口起着類似的功能，它是路由器和[虛擬機](#)掛接網絡的着附點。

### 路由器

和物理環境下的路由器類似，Neutron 中的路由器也是一個路由選擇和轉發部件。只不過在 Neutron 中，它是可以創建和銷毀的軟部件。

### 子網

簡單地說，子網是由一組 IP 地址組成的地址池。不同子網間的通信需要路由器的支持，這個 Neutron 和物理網絡下是一致的。Neutron 中子網隸屬於網絡。

## 為什麼引入 Quantum？

答案非常簡單，Quantum 功能更強大，滿足更多需求。下面列幾條主要功能。

- 提供面向租戶的 API，以便控制 2 層網絡和管理 IP 地址
- 支持插件式網絡組件，像 Open vSwitch, Cisco, Linux Bridge, Nicira NVP 等等
- 支持位於不同的 2 層網絡的 IP 地址重疊
- 支持基本的 3 層轉發和多路由器
- 支持隧道技術（Tunneling）

- 支持 3 层代理和 DHCP 代理的多节点部署，增强了扩展性和可靠性
- 提供负载均衡 API （试用版本）

**Neutron** 主要有以下几部分组成。

**Neutron Server**：这一部分包含守护进程 `neutron-server` 和各种插件 `neutron-*-plugin`，它们既可以安装在控制节点也可以安装在网络节点。`neutron-server` 提供 API 接口，并把对 API 的调用请求传给已经配置好的插件进行后续处理。插件需要访问数据库来维护各种配置数据和对应关系，例如路由器、网络、子网、端口、浮动 IP、安全组等等。

**插件代理（Plugin Agent）**：虚拟网络上的数据包的处理则是由这些插件代理来完成的。名字为 `neutron-*-agent`。在每个计算节点和网络节点上运行。一般来说你选择了什么插件，就需要选择相应的代理。代理与 **Neutron Server** 及其插件的交互就通过消息队列来支持。

**DHCP 代理（DHCP Agent）**：名字为 `neutron-dhcp-agent`，为各个租户网络提供 DHCP 服务，部署在网络节点上，各个插件也是使用这一个代理。

**3 层代理（L3 Agent）**：名字为 `neutron-l3-agent`，为客户机访问外部网络提供 3 层转发服务。也部署在网络节点上。

下面这张图取自官网，很好的反映了 **Neutron** 内部各部分之间的关系。（SDN 服务在这里是额外的外部功能，可以暂时略过。）

上面简单的介绍，下面内容可以参考：

[openstack---Neutron 网络入门](#)

[OpenStack Neutron 解析](#)

[OpenStack 网络组件 Neutron 的研究](#)

[OpenStack Neutron 运行机制解析概要](#)

[OpenStack: 网络组件 Neutron](#)

[Openstack 之 neutron 入门一](#)

[Openstack 之 neutron 入门二](#)

[Openstack 之 neutron 入门三](#)

[openstack 网络,外部网络、内部网络、管理网络作用介绍](#)

[openstack neutron 创建多个外网](#)

[开发人员必读 openstack 网络基础 1:什么是 L2、L3](#)

[开发人员必读 openstack 网络基础 2:交换机、路由器、DHCP](#)

[开发人员必读 openstack 网络基础 3: iptables 详解](#)

[开发人员必读 openstack 网络基础 4:Dnsmasq、网络混杂模式](#)

[开发人员必读 openstack 网络基础 5:网络叠加模式 VLAN、VxLAN、GRE](#)

[开发人员必读 openstack 网络基础 6:什么是 Tap/Tun、网桥](#)

[开发人员必读 openstack 网络基础 7:到底什么是 Open vSwitch](#)

[源码分析参考](#)

[Neutron 分析（1）——neutron-server 启动过程分析](#)

[openstack Neutron 分析（2）—— neutron-l3-agent](#)

[openstack Neutron 分析（3）—— neutron-dhcp-agent 源码分析](#)

[openstack Neutron 分析（4）—— neutron-l3-agent 中的 iptables](#)

[openstack Neutron 分析（5）-- neutron openvswitch agent](#)

[OpenStack Neutron DVR L2 Agent 的初步解析（一）](#)

[OpenStack J 版 Neutron-server 服务加载与启动源码分析（一）](#)

[OpenStack J 版 Neutron-server 服务加载与启动源码分析（二）](#)

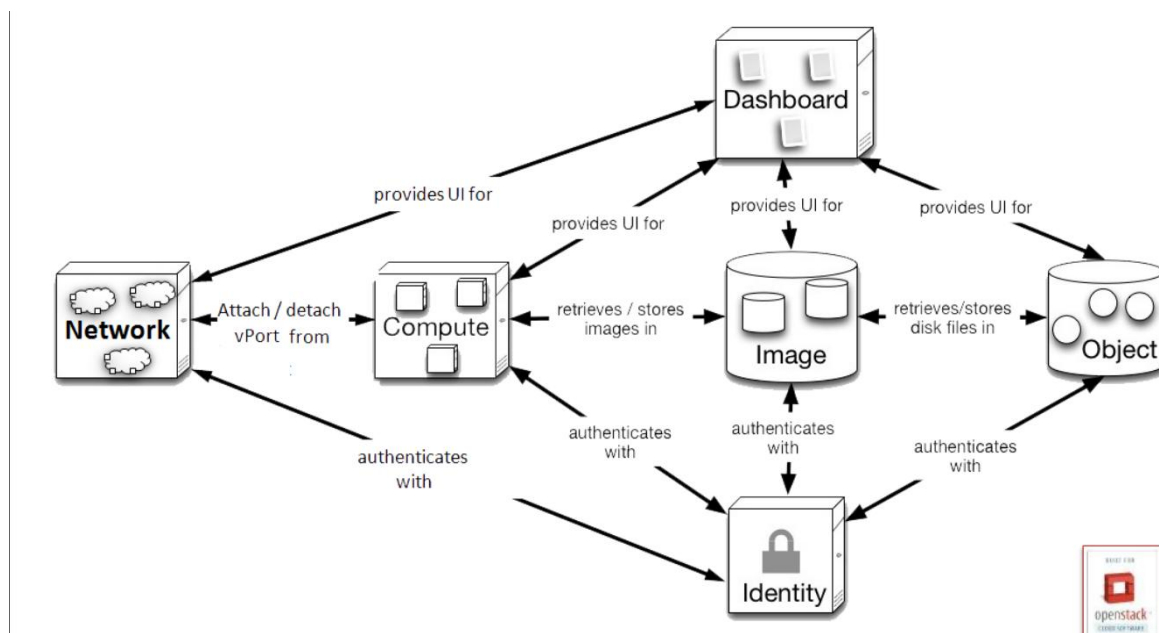
[Openstack Neutron-server 服务加载与启动源码分析（三）](#)

## 4.了解 Swift

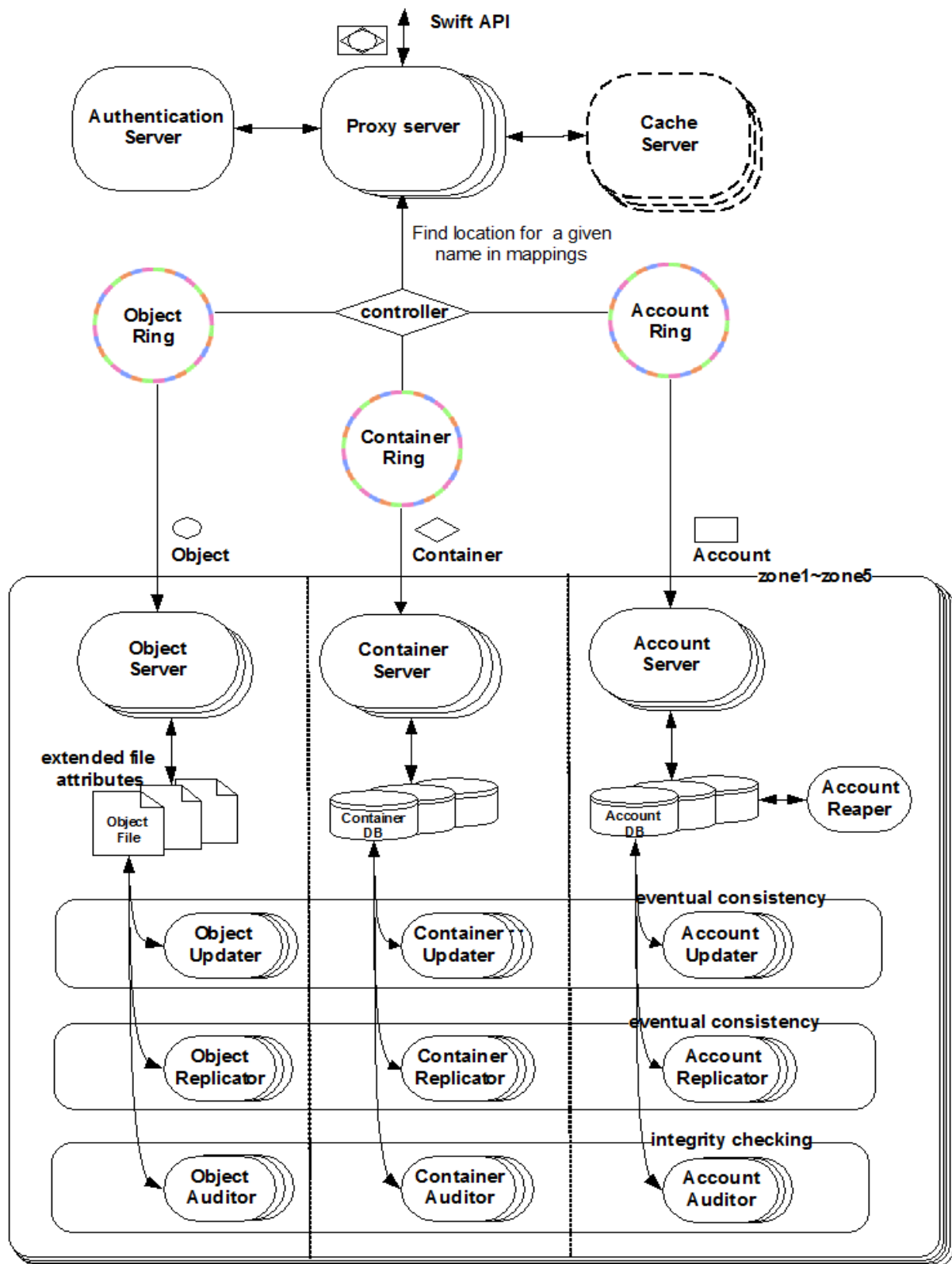
这是对象存储的组件。对于大部分用户来说，**swift** 不是必须的。你只有存储数量到一定级别，而且是非结构化数据才有这样的需求。很多人都问一个相同的问题：是否可以把**虚拟机**的存储放在 **swift** 上。简单回答：不行。你需要搞明白对象存储是干啥，擅长那些地方，那些是不行的。**swift** 是 Openstack 所有组件了最成熟的,可以在线升级版本,各种版本可以混合在一起,也就是说,1.75 版本的 **swift** 可以和 1.48 的在一个群集里.这个是很难得的。

### **swift** 是什么及在 openstack 的作用

OpenStack Object Storage (Swift) 是开源的，用来创建可扩展的、冗余的、对象存储（引擎）。**swift** 使用标准化的服务器存储 PB 级可用数据。但它并不是文件系统（file system），实时的数据存储系统(real-time data storage system)。**swift** 看起来更像是一个长期的存储系统(long term storage system)，为了获得、调用、更新一些静态的永久性的数据。比如说，适合存储一些类型的数据：虚拟机镜像，图片存储，邮件存储，文档的备份。没有“单点”或者主控结点 (master point of control)，**swift** 看起来具有更强的扩展性、冗余和持久性。



swift 结构





### Swift 组件包括：

- 代理服务（Proxy Server）：对外提供对象服务 API，会根据环的信息来查找服务地址并转发用户请求至相应的账户、容器或者对象服务；由于采用无状态的 REST 请求协议，可以进行横向扩展来均衡负载。
- 认证服务（Authentication Server）：验证访问用户的身份信息，并获得一个对象访问令牌（Token），在一定的时间内会一直有效；验证访问令牌的有效性并缓存下来直至过期时间。
- 缓存服务（Cache Server）：缓存的内容包括对象服务令牌，账户和容器的存在信息，但不会缓存对象本身的数据；缓存服务可采用 Memcached 集群，Swift 会使用一致性散列算法来分配缓存地址。
- 账户服务（Account Server）：提供账户元数据和统计信息，并维护所含容器列表的服务，每个账户的信息被存储在一个 SQLite 数据库中。
- 容器服务（Container Server）：提供容器元数据和统计信息，并维护所含对象列表的服务，每个容器的信息也存储在一个 SQLite 数据库中。
- 对象服务（Object Server）：提供对象元数据和内容服务，每个对象的内容会以文件的形式存储在文件系统中，元数据会作为文件属性来存储，建议采用支持扩展属性的 XFS 文件系统。
- 复制服务（Replicator）：会检测本地分区副本和远程副本是否一致，具体是通过对比散列文件和高级水印来完成，发现不一致时会采用推式（Push）更新远程副本，例如对象复制服务会使用远程文件拷贝工具 rsync 来同步；另外一个任务是确保被标记删除的对象从文件系统中移除。
- 更新服务（Updater）：当对象由于高负载的原因而无法立即更新时，任务将会被序列化到在本地文件系统中进行排队，以便服务恢复后进行异步更新；例如成功创建对象后容器服务器没有及时更新对象列表，这个时候容器的更新操作就会进入排队中，更新服务会在系统恢复正常后扫描队列并进行相应的更新处理。
- 审计服务（Auditor）：检查对象，容器和账户的完整性，如果发现比特级的错误，文件将被隔离，并复制其他的副本以覆盖本地损坏的副本；其他类型的错误会被记录到日志中。
- 账户清理服务（Account Reaper）：移除被标记为删除的账户，删除其所包含的所有容器和对象。

上面只是简单的介绍，或许并不能让你真正明白什么是 swift，如果感兴趣可以了解更多内容

[Openstack Swift 原理、架构与 API 介绍](#)

[openstack 入门之 swift 基础一：什么是对象存储](#)

[openstack 入门之 swift 基础二：三种存储类型比较-文件、块、对象存储](#)

[openstack 入门之 swift 基础三：swift 能干什么，不能干什么及相关概念](#)

[对象存储系统 Swift 技术详解：综述与概念（上）](#)

[对象存储系统 Swift 技术详解：综述与概念（下）](#)

[swift 初见](#)

[Ubuntu 12.04 OpenStack Swift 单节点部署指导](#)

[Object Storage \(Swift\)和 Block Storage \(Cinder\)有什么区别？](#)

[hadoop 中 HDFS 与 opesntack 的 swift 有何不同](#)

**单独部署**

[Swift 能单独使用吗？如何单独部署？](#)

Swift 源码想开发和了解原理的途径之一

**Swift 源码分析**

[Swift 源码分析----swift-object-auditor（1）](#)

[Swift 源码分析----swift-object-auditor\(2\)](#)

[Swift 源码分析----swift-container-auditor](#)

[Swift 源码分析----swift-account-auditor](#)

[Swift 源码分析----swift-account-audit\(1\)](#)

[Swift 源码分析----swift-account-audit\(2\)](#)

[OpenStack Swift 源码分析（1） ----swift 服务启动源码分析之一](#)

[OpenStack Swift 源码分析（2） ----swift 服务启动源码分析之二](#)

[OpenStack Swift 源码分析（3） ----swift 服务启动源码分析之三](#)

[OpenStack Swift 源码分析（4） ----swift-ring-builder 源代码解析之一](#)

[OpenStack Swift 源码分析（5） ----swift-ring-builder 源代码解析之二](#)

[Swift 源码分析----swift-object-updater](#)

[Swift 源码分析----swift-object-info](#)

[Swift 源码分析----swift-object-replicator\(1\)](#)

[Swift 源码分析----swift-object-replicator\(2\)](#)

[Swift 源码分析----swift-proxy 实现请求 req 的转发](#)

[Swift 源码分析----swift-container-info](#)

[Swift 源码分析----swift-proxy 与 swift-account](#)

[Swift 源码分析----swift-account-reaper\(1\)](#)

[Swift 源码分析----swift-account-reaper\(2\)](#)

[Swift 源码分析----swift-proxy 与 swift-container](#)

[Swift 源码分析----swift-account-replicator](#)

[Swift 源码分析----swift-container-replicator](#)

[Swift 源码分析----swift-proxy 与 swift-object](#)

[Swift 源码分析----swift-container-updater](#)

## 5.了解 Cinder

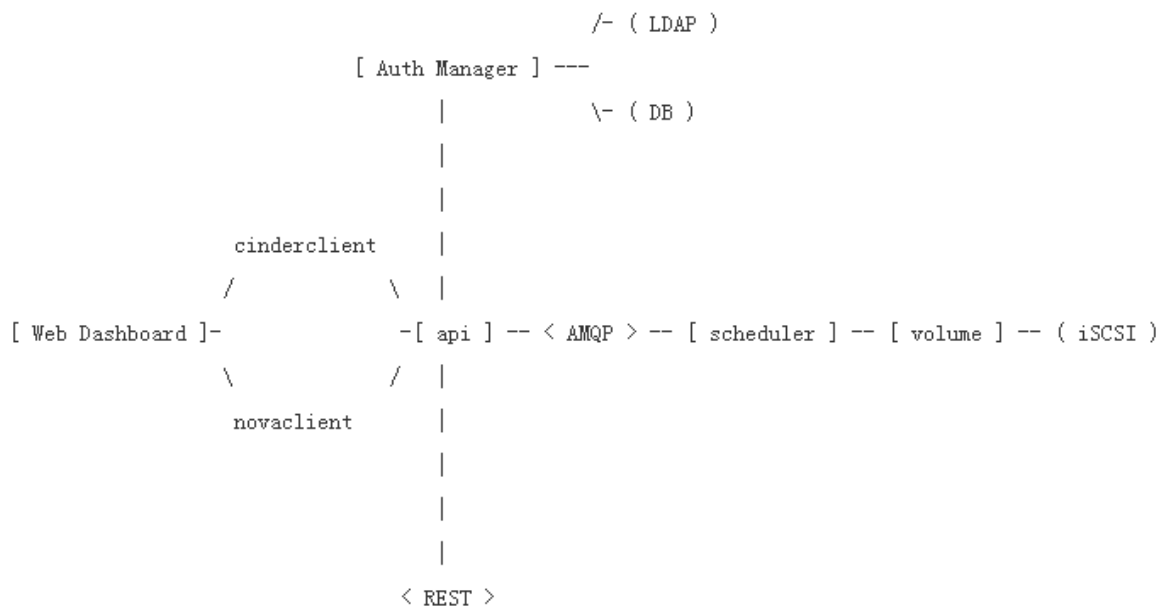
这是存储管理的组件。一直以来，很多人都很纠结 AWS 的 EBS 的实现。Openstack 也终于推出了自己的存储管理组件。

Cinder 存储管理主要是指虚拟机的存储管理。目前支持开源和商业化产品。开源的 **sheepdog**, **Ceph** 等。商业存储的支持, 目前 I B M 是最积极的。未来如果商业存储厂商都支持 **Cinder**, 对 **Openstack** 的商业化还是非常有利的。

对于企业来说, 使用分布式作为**虚拟机**的存储, 并不能真正节省成本, 维护一套分布式存储, 成本还是很高的。目前虚拟机的各种高可用, 备份的问题, 其实都可以把问题交给商业存储厂商来解决。

我们知道 **Openstack** 从 **Folsom** 开始使用 **Cinder** 替换原来的 **Nova-Volume** 服务, 为 **Openstack** 云平台提供块存储服务。

**cinder** 架构如下 :



## Cinder 服务

- **API service** : 负责接受和处理 **Rest** 请求, 并将请求放入 **RabbitMQ** 队列。Cinder 提供 **Volume API V2**, 我没有找到格式很好的在线文档, 大体可以参见 [Openstack block storage API V1](#)
- **Scheduler service**: 处理任务队列的任务, 并根据预定策略选择合适的 **Volume Service** 节点来执行任务。目前版本的 **cinder** 仅仅提供了一个 **Simple Scheduler**, 该调度器选择卷数量最少的一个活跃节点来创建卷。
- **Volume service**: 该服务运行在存储节点上, 管理存储空间。每个存储节点都有一个 **Volume Service**, 若干个这样的存储节点联合起来可以构成一个存储资源池。为了支持不同类型和型号的存储, 当前版本的 **Cinder** 为 **Volume Service** 如下 **drivers**。当然在 **Cinder** 的 **blueprints** 当中还有一些其它的 **drivers**, 以后的版本可能会添加进来。

- 本地存储：LVM, Sheepdog
- 网络存储：NFS, RBD (RADOS)
- IBM: XIV, Storwize V7000, SVC storage systems
- Netapp: NFS 存储 ;ISCSI 存储则需要 OnCommand 5.0 和 Data ONTAP 7-mode storage systems with installed iSCSI licenses
- EMC: VNX, VMAX/VMAXe
- Solidfire: Solidfire cluster

### **cinder** 还有更多内容

1.Cinder 有哪些服务？

2.Cinder 有哪些部署命令？

3.Cinder 在 IT 环境中的主要有哪些问题？

详细参考：

[Openstack 之 Cinder 介绍](#)

更多内容参考

[Openstack 之 Cinder 介绍](#)

[OpenStack IceHouse 版 cinder 模块新增加功能](#)

[OpenStack-Icehouse \(nova-network\) 多节点块存储服务 Cinder 部署](#)

[Openstack Cinder 安装向导：在 Unitestack 的 UOS 下测试](#)

[OpenStack Cinder 与 Ceph 使用进阶篇（基于 Icehouse 版本）](#)

[cinder 中删除僵尸卷（error deleting）的方法](#)

当然如果想了解**源码**，同样附上源码：

**源码分析：**

[Openstack Cinder 中建立 volume 过程的源码解析（1）](#)

[Openstack Cinder 中建立 volume 过程的源码解析（2）](#)

[Openstack Cinder 中建立 volume 过程的源码解析 \(3\)](#)

[Openstack Cinder 中建立 volume 过程的源码解析 \(4\) ----以及 taskflow 相关解析](#)

[Openstack Cinder 中建立 volume 过程的源码解析 \(5\) ----以及 taskflow 相关解析](#)

[Openstack Cinder 中建立 volume 过程的源码解析 \(6\) ----以及 taskflow 相关解析](#)

[Openstack Cinder 中建立 volume 过程的源码解析 \(7\) ----以及 taskflow 相关解析](#)

[Openstack Cinder 中建立 volume 过程的源码解析 \(8\)](#)

[Openstack Cinder 中建立 volume 过程的源码解析 \(9\)](#)

[OpenStack Cinder 服务启动过程中的资源加载和扩展源码解析之一](#)

[Paste Deployment 简介以及 cinder-api-paste.ini 的解析 \(1\)](#)

[cinder 服务启动源码分析](#)

[cinderclient 源码解析之一](#)

[cinderclient 源码解析之二](#)

[OpenStack Cinder 源码分析之一](#)

[OpenStack Cinder 源码分析之二](#)

[OpenStack Cinder 源码分析之三](#)

[OpenStack Cinder 源码分析之四](#)

[OpenStack Cinder 源码分析之五](#)

[OpenStack Cinder 源码分析之六](#)

[OpenStack Cinder 源码分析之七](#)

[OpenStack Cinder 源码分析之八](#)

## 6.了解 Glance

Glance 比较简单，是一个虚拟机镜像的存储。向前端 nova（或者是安装了 Glance-client 的其他虚拟管理平台）提供镜像服务，包括存储，查询和检索。这个模块本身不存储大量的数据，需要挂载后台存储（Swift，S3。。。）来存放实际的镜像数据。

OpenStack 镜像服务器是一套虚拟机镜像发现、注册、检索系统，我们可以将镜像存储到以下任意一种存储中：

- 本地文件系统（默认）
- OpenStack 对象存储
- S3 直接存储
- S3 对象存储（作为 S3 访问的中间渠道）
- HTTP（只读）

### 功能及特点

提供镜像相关服务

### Glance 构件

- Glance 控制器
- Glance 注册器

从上面我们看出 glance 是可以存储的，但是其重心是管理镜像，存储则由比如 swift、S3 等来完成。如果感觉还有疑惑，参考：

[让你真正明白 cinder 与 swift、glance 的区别](#)

我们知道了 glance 能干什么，那么我们该如何使用，glance 命令该如何使用，api 该如何使用，详细参考 [openstack 之 glance 篇](#)

更多参考：

[翻译：Openstack 镜像服务（glance）用法的高级例子](#)

[建立高可用 OpenStack 云系列--建立 HA Openstack 云（五）：安装 Glance](#)

同样附上

## 源码分析

[Glance 源码架构分析（一）](#)

[Glance 源码架构分析（二）](#)

[Glance 源码架构分析（三）](#)

[Glance 源码架构分析（四）](#)

## 7.了解 Horizon

严格意义来说，Horizon 不会为 Openstack 增加一个功能，他更多的是一个演示，demo。不过对于很多用户来说，了解 Openstack 基本都是从 Horizon，dashboard 开始。从这个角度来看，他在 Openstack 各个项目里，显得非常重要。

Horizon 的开发者，应该是最累的。需要和各个项目打交道。每个项目的功能很多都是需要通过 Dashboard 来展现。

大家需要注意的是：Horizon 只是使用了 Openstack 部分 API 功能，很多功能，你可以根据你的需求去实现。

### OpenStack 管理的 Web 接口----Horizon

Horizon 是一个用以管理、控制 OpenStack 服务的 Web 控制面板，它可以管理实例、镜像、创建密钥对，对实例添加卷、操作 Swift 容器等。除此之外，用户还可以在控制面板中使用终端（console）或 VNC 直接访问实例。

总之，Horizon 具有如下一些特点：

- 实例管理：创建、终止实例，查看终端日志，VNC 连接，添加卷等
- 访问与安全管理：创建安全群组，管理密钥对，设置浮动 IP 等
- 偏好设定：对虚拟硬件模板可以进行不同偏好设定
- 镜像管理：编辑或删除镜像



- 查看服务目录
- 管理用户、配额及项目用途
- 用户管理：创建用户等
- 卷管理：创建卷和快照
- 对象存储处理：创建、删除容器和对象
- 为项目下载环境变量

上面如果作为初学者，可能还是感觉两个字“模糊”，好吧，其实 Horizon 可以理解为 openstack 界面，我们既可以通过界面操作 openstack，也可以通过 shell 的方式操作 openstack。但是界面在前些版本中还不够完善，很多操作通过命令行的方式更方便些。不过 Horizon 目前功能在不断的完善。

Horizon 无须过多的介绍，因为当我们安装完毕之后，我们自然能够看到界面。那么该如何使用它，比如创建项目、用户创建网络，详细参考

[测试 OpenStack Icehouse Horizon —— 创建项目、用户创建网络](#)

更多内容：

[Openstack Horizon Icehouse Blueprint 简介](#)

[horizon 界面修改以及扩展](#)

[openstack dashboard 使用的是什么语言，如何搭建开发环境，如何本地化（汉化）](#)

[OpenStack Dashboard 二次开发--简明教程如何设置 OpenStack Horizon 开发环境 Part 2](#)

[给 horizon 添加分配指定 floating IP 的功能](#)

## 8.Ceilometer

这是实现监控和计量的组件。这应该算是 Grizzly 的孵化项目。对他的了解其实很少。在 Grizzly 版本里，你应该可以在 Dashboard 里看到这个组件。

监控和计费一直是一个难题，尤其用户希望知道 cpu 和内存的使用情况。看看他如何解决这个问题。到时候看看同事如何调用 api 来解决监控和计量的问题。解决计量，计费就简单的。

<http://wiki.openstack.org/Ceilometer>

这个组件目前大家讨论的不多，但是可能会后期发力，因为云平台搭建之后，我们该如何计费，这是个问题，所以有些同学云平台该怎么计费

[请问楼主，云主机根据不同配置进行计费的功能是用什么软件实现的？有开源的软件吗？](#)

那么研究下这个这个组件就可以了，我们对它有了了解，那么

Ceilometer 如何部署

参考

[部署 Ceilometer 到已有环境中](#)

Ceilometer 的概念，

可参考：

[OpenStack 监控项目 Ceilometer 的一些术语](#)

Ceilometer API 说明参考

[OpenStack 监控测量服务 Ceilometer 安装及 API 说明](#)

其它

[OpenStack 里数据采集（监控数据、计费数据）基础设施--Ceilometer](#)

关于[源码](#)：

[云计算计费：Ceilometer 的 alarm 模块代码分析](#)

[OpenStack Ceilometer Collector 代码解读](#)

## 9.了解 Heat

这个项目是要解决[虚拟机](#)的软件部署的问题。你的虚拟机创建好，os 准备好，你还需要做很多配置才能使用，如何能实现把所有繁琐的操作简化呢？亚马逊上有一个专门的工具：AWS cloudformation。目前 Openstack 上，希望通过 Heat 来实现类似的功能。

关于这个项目，还是有很多争议。不过这个项目是 Redhat 发起。他们的功力是不容置疑，等 Openstack 成熟后，这个项目的重要性就会体现出来。

<http://wiki.openstack.org/Heat>

真的要实现弹性扩展，自动部署，都是需要指望这个。

上面整体介绍，下面我们介绍

### 什么是 Heat

Heat 是一套业务流程平台，旨在帮助用户更轻松配置以 OpenStack 为基础的云体系。利用 Heat 应用程序，开发人员能够在程序中使用模板以实现资源的自动化部署。Heat 能够启动应用、创建[虚拟机](#)并自动处理整个流程。它还拥有出色的跨平台兼容性，能够与 Amazon Web Services 业务流程平台 CloudFormation 相对接——这意味着用户完全可以将 AWS 模板引入 OpenStack 环境当中。

### 为什么会产生 Heat

Openstack 对应于云计算的概念，是实现了 IaaS（Infrastructure as a Service），即基础设施即服务，提供对云的基础设施运行环境的管理。有了基础设施就可以在其上部署和运行相关的应用，如 web 群集，paas，数据库等等相关的服务和应用。对于这些软件运行环境的构建需要进行相关的部署过程，当然部署的过程可以手工的完成，但是面对于快速构建应用的普遍需求来说，手工部署并不能满足要求，并且云环境下的群集部署对于普通的非专业的用户来说是很困难的，所以就实现一种自动化的通过简单定义和配置就能实现部署的云部署方式。Heat 项目就是提供了一种通过模版定义的协同部署方式，实现云基础设施软件运行环境的自动化部署。

上面我们对 Heat 有了初步的认识，那么 Heat 如何安装部署、命令如何使用等更多内容：

[openstack\(G 版、icehouse 版本\)中 Heat 介绍](#)

[heat 安装和配置及命令使用](#)

[在 OpenStack 中通过 Heat 来使用 Docker Containers](#)

[OpenStack 中的 Heat 分析](#)

[OpenStack Heat 服务再介绍（二）](#)

[OpenStack Heat 模板学习一 之 hello world](#)

[OpenStack Heat 模板学习二 之 LBaaS\(负载均衡即服务\)](#)

对于下面组件，目前研究的人不多，简单了解即可

## 10.Lbaas

Load Balancer as a Service(LBaaS)，负载均衡即服务，是 OpenStack 在其网络组件 Neutron 中提供的一种将负载均衡器/软件/设备纳入到 Neutron 体系中的框架，纳入到 Neutron 中的负载均衡以服务的形态供用户使用。用户可以在 OpenStack 中自行创建负载均衡器，进行相关配置，并对自己在 OpenStack 上的 instance 进行负载均衡。此功能类似于 AWS 和阿里云中为用户提供的负载均衡服务。

SEnginx 目前实现了对 OpenStack LBaaS 的支持，可以为 OpenStack 中的 instance 提供负载均衡服务，这是通过提供了一个 LBaaS 的 driver 来实现的，具体使用方法详见：

<https://github.com/NeusoftSecurity/SEnginx-LBaaS-Driver>

目前 SEnginx 只支持 OpenStack 的 Havana 版本（2013.10）

详细参考

[SEnginx 支持 OpenStack LBaaS](#)

## 11.oslo

这个项目其实就是把所有组件需要用到相同的东西，集中起来，以前叫 nova common，估计感觉不贴切，现在单独成立一个项目。日后大家开发新的组件，估计都需要用到 oslo。

其资料不多，相关内容：

[OpenStack 配置解析库——oslo.config](#)

## 12.Moniker

这是实现 dns 功能的组件。其实如果你用过 AWS，你就知道这个功能是必不可少。新浪目前的已经加上了这个功能，每个虚拟机，都会自动有一个 dns 记录。

<https://github.com/stackforge/moniker>

估计集成到 Dashboard 里，还是需要等待一段时间啊。目前该项目开发还是非常积极。

## 13.marconi

此项目用于解决 openstack 消息队列的扩展问题。据说这是 Rackspce 推出的项目，就是为了解决他们生产中遇到消息队列的问题。

## 附上 openstack 资源：

希望对大家的学习有帮助

**OpenStack Installation Guide for Ubuntu 中文翻译版**

<http://www.aboutyun.com/thread-9218-1-2.html>

**openstack installation Guide for red hat Enterprise Linux,CentOS, and Fedora**

<http://www.aboutyun.com/thread-9324-1-2.html>

**openstack operations Guide** 英文书籍

<http://www.aboutyun.com/thread-9326-1-2.html>

**openstack** 各种文档下载

<http://www.aboutyun.com/thread-8797-1-2.html>

**OpenStack** 身份服务 **API** 资料

<http://www.aboutyun.com/thread-8699-1-2.html>

各个版本 **Linux** 系统安装部署 **openstack icehouse** 在线英文文档汇总及下载

<http://www.aboutyun.com/thread-9417-1-2.html>

**Swift** 自编精品教程

<http://www.aboutyun.com/thread-9325-1-2.html>

**2014** 中国系统架构师大会：视频 **CDN** 技术分享

<http://www.aboutyun.com/thread-9327-1-2.html>

**OpenStack Get\_**介绍及基础概念

<http://www.aboutyun.com/thread-9706-1-1.html>

**openstack** 开发 **python** 教程

<http://www.aboutyun.com/thread-9388-1-1.html>

**Neutron** 防火墙

<http://www.aboutyun.com/thread-9381-1-1.html>

盛大云平台架构设计和实现

<http://www.aboutyun.com/thread-5568-1-1.html>

openstack 最新版--- junos 版最新官网文档

<http://www.aboutyun.com/thread-9423-1-1.html>

运维社区-openstack 源码安装资源分享

<http://www.aboutyun.com/thread-5566-1-1.html>

《OpenStack Juno 版》资源分享

<http://www.aboutyun.com/thread-9888-1-1.html>

openstack-ice-house 云环境构建

<http://www.aboutyun.com/thread-9323-1-1.html>

系统讲解 Openvswitch (138 页) ppt 分享

<http://www.aboutyun.com/thread-9839-1-1.html>

Zabbix 中文使用手册分享

<http://www.aboutyun.com/thread-10059-1-1.html>

Openstack 入门基础知识 51 页 ppt 【推荐】

<http://www.aboutyun.com/thread-10057-1-1.html>

OpenStack-Icehouse 版 多节点部署资源

<http://www.aboutyun.com/thread-8999-1-1.html>

cloudstack 介绍及开发环境设置(windows、centos)文档分享

<http://www.aboutyun.com/thread-8283-1-1.html>

OpenStack 企业应用之路

<http://www.aboutyun.com/thread-8975-1-1.html>

openstack 实践 pdf 分享

<http://www.aboutyun.com/thread-10056-1-1.html>

华为云计算解决方案

<http://www.aboutyun.com/thread-8978-1-1.html>

openstack junos 版发布文档汇总：各个版本安装、API、配置使用、管理员英文文档

<http://www.aboutyun.com/thread-9568-1-1.html>

openstack redhat 两小时安装部署

<http://www.aboutyun.com/thread-9365-1-1.html>

openstack 实践、HA、商业模式探讨、新浪应用文档下载

<http://www.aboutyun.com/thread-8419-1-2.html>

swift 安装及开发调试环境及 Apple Swift 编程语言入门教程

<http://www.aboutyun.com/thread-8613-1-2.html>

原创：基于 Ubuntu 上 OpenStack IceHouse 版详细安装资源分享（中文翻译）

<http://www.aboutyun.com/thread-9102-1-2.html>

OpenStack 资源分享

<http://www.aboutyun.com/thread-9209-1-2.html>

基于 openstack 的 docker 开发

<http://www.aboutyun.com/thread-9186-1-4.html>

openstack icehouse 部署视频

<http://www.aboutyun.com/thread-8723-1-4.html>

云里雾里云计算

<http://www.aboutyun.com/thread-8201-1-1.html>

openstack 开发, Python 系列最全文档书籍下载

<http://www.aboutyun.com/thread-7853-1-1.html>

台湾辅仁大学--Python 视频分享

<http://www.aboutyun.com/thread-8173-1-1.html>

老外 python 视频 30 讲、python100 例、python100 习题等系列汇总

<http://www.aboutyun.com/thread-8095-1-1.html>

openstack 相关文档分享

<http://www.aboutyun.com/thread-6741-1-1.html>

混合云管理平台 CloudForms 简介

<http://www.aboutyun.com/thread-8008-1-1.html>

在 Debian 上部署 OpenStack 官方文档翻译 1

<http://www.aboutyun.com/thread-7255-1-1.html>

在 Debian 上部署 OpenStack 官方文档翻译 2

<http://www.aboutyun.com/thread-7665-1-1.html>

在 Debian 上部署 OpenStack 官方文档翻译 3

<http://www.aboutyun.com/thread-7961-1-1.html>

OpenStack 开发之 Python 资料大全汇总

<http://www.aboutyun.com/thread-7950-1-1.html>

下面是其资源汇总：

1、[openstack 编程, Python 系列文档下载](#)

2、[Python 核心编程下载分享](#)



- 3、[openstack 编程：Python 标准库分享](#)
- 4、[openstack 编程：Python 学习手册 第3版](#)
- 5、[Python Cookbook（第2版）中文版](#)
- 6、[openstack 开发，Python 系列最全文档书籍下载](#)

**OpenStack Icehouse** 版本官方安装教程-英文版

<http://www.aboutyun.com/thread-7617-1-1.html>

**openstack-H 版 Centos6.4** 下安装单网卡安装

<http://www.aboutyun.com/thread-7426-1-1.html>

**CloudComputing** 相关英文文档

<http://www.aboutyun.com/thread-7096-1-1.html>

**openstack** 安装英文版 **openstack-install-guide-ubuntu12\_04-apt-trunk**

<http://www.aboutyun.com/thread-7052-1-1.html>

**openstack** 入门、建设公有云、私有云、商业模式、云存储等系列文档下载

<http://www.aboutyun.com/thread-8199-1-1.html>

[openstack 中文、英文安装文档下载](#)

[openstack 入门视频](#)

[Python 核心编程下载分享](#)

[openstack 编程：Python 标准库分享](#)

[openstack 编程：Python 学习手册 第3版](#)

[OpenStack 简介，入门书籍](#)

[openstack 实战手册指导](#)

[OpenStack 在 Ubuntu12.04X64 系统上的安装](#)

[在 CentOS 6.x 通过 RPM 包安装 OpenStack 多节点环境文档下载](#)

[VMware 镜像完整版](#)

[vSphere5 介绍（55 视频）包括文档下载](#)

[IT 相关架构文档下载](#)

更多大数据、**openstack**、云平台资料：

**about** 云汇总帖：包括资源，指导，文档，视频等

<http://www.aboutyun.com/thread-7178-1-1.html>

**about** 云资源汇总 V1.2：包括 **hadoop,openstack,nosql,虚拟化**

<http://www.aboutyun.com/thread-5928-1-1.html>

**about** 云资源汇总指引 V1.3：包括 **hadoop,openstack,nosql,虚拟化**

<http://www.aboutyun.com/thread-6150-1-1.html>

**about** 云资源汇总指引 V1.4:包括 **hadoop,openstack,nosql,虚拟化**

<http://www.aboutyun.com/thread-6730-1-1.html>

**about** 云资源汇总指引 V1.5:包括 **hadoop,openstack,storm,spark** 等视频文档书籍汇总

<http://www.aboutyun.com/thread-8203-1-1.html>

**about** 云资源汇总指引 V1.6:包括 **hadoop,openstack,storm,spark** 等视频文档书籍汇总

<http://www.aboutyun.com/thread-10302-1-1.html>