OpenStack 架构

yeasy.github.com

v0.2: 2013-04-08

添加服务架构说明

v0.1: 2013-04-02

完成基于 folsom 版本的初始版本

1.1 组件

目前（folsom 版本），OpenStack 包括 7 个核心部件，包括前端面板、计算、对象、镜像、鉴权、网络和块存储。各部分功能为

- 前端面板 项目名称为 Horizon，为所有的 OpenStack 服务提供 web 前端接口界面。利用 web 界面，可以进行大部分的云操作，包括运行实例、分配 IP 和设置访问控制等。
- 计算 项目名称为 Nova，提供虚拟机服务。基于 Nova，HP 和 Rackspace 都开发有商用的计算服务方案，同时在 Mercado Libre 和 NASA（发起人）等公司内部都有使用。
- 对象 项目名称为 Swift 允许储存或获取文件（但不能像文件服务器一样挂载目录）。基于 Swift，数家公司开发了商业的存储服务方案，包括 KT，Rackspace（发起人）和 Internap。Swift 同时在多家大型企业内部应用负责数据存储。
- 镜像 项目名称为 Glance，提供登记和虚拟机镜像的管理，这些镜像主要是支持计算服务的。
- 鉴权 项目名称为 Keystone，为 OpenStack 所有的服务提供认证和授权，同时提供了服务的登记管理。
- 网络 项目名称为 Quantum，在网络接口设备之间提供连接服务。允许用户创建自由网络并挂载端口。架构开放，支持插件。从 Folsom 版本时引入。
- 块存储 项目名称为 Cinder，为 guest 虚拟机提供永久性块设备，前身为 nova-volume。所提供的块存储并非类似 NFS 或 CIFS share 的文件系统。从 Folsom 版本时引入。

与 Amazon 的 AWS 相比，OpenStack 大部分的工具和 API 等都有很好的兼容性。包括

- Nova 在概念上类似 EC2。可以有多种方式来支持 EC2 的 API。
- Swift 在概念上类似于 S3.在 WSGI 中间件上实现了部分 S3 API。
- Glance 提供了 Amazon 的 AMI 登记服务类似的很多特性。
- Cinder 提供了类似于 EBS 的块设备。

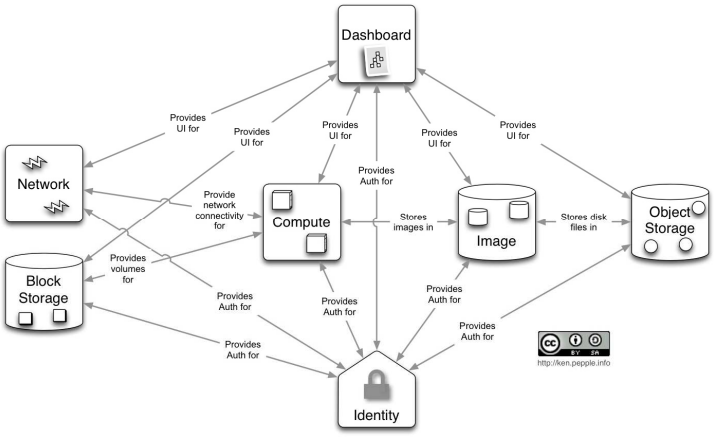
1.2 架构

1.2.1 概念架构

OpenStack 在整体设计上是“提供大量的可扩展的云服务的操作系统”，为了实现这点，各个组成的服务被设计一起协作提供“架构即服务（IaaS）”。这些协作通过服务提供的 API 之间

进行操作实现。这些 API 允许服务之间的调用，并且相互之间松耦合。

概念架构如图表 1 所示。

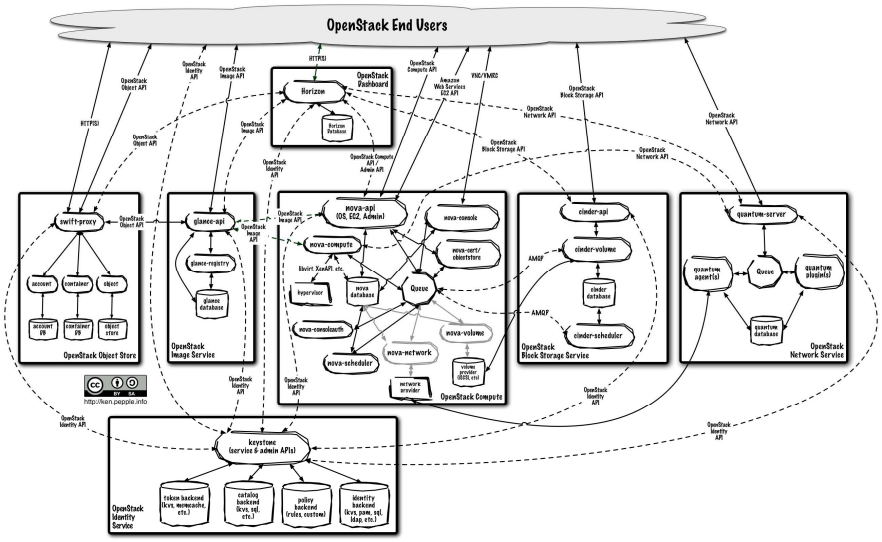


图表 1 OpenStack 概念架构

其中，Horizon 提供用户的 web 前端；Nova 存储和获取虚拟机，并且在 Glance 中维护元数据；Quantum 为 Nova 提供虚拟网络；Cinder 为 Nova 提供块存储；Glance 在 Swift 中保持实际的虚拟机硬盘文件；所有的服务都需要 Keystone 进行认证。

1.2.2 逻辑架构

逻辑架构比概念架构复杂的多。图表 2 中给出了最常见的一些架构。



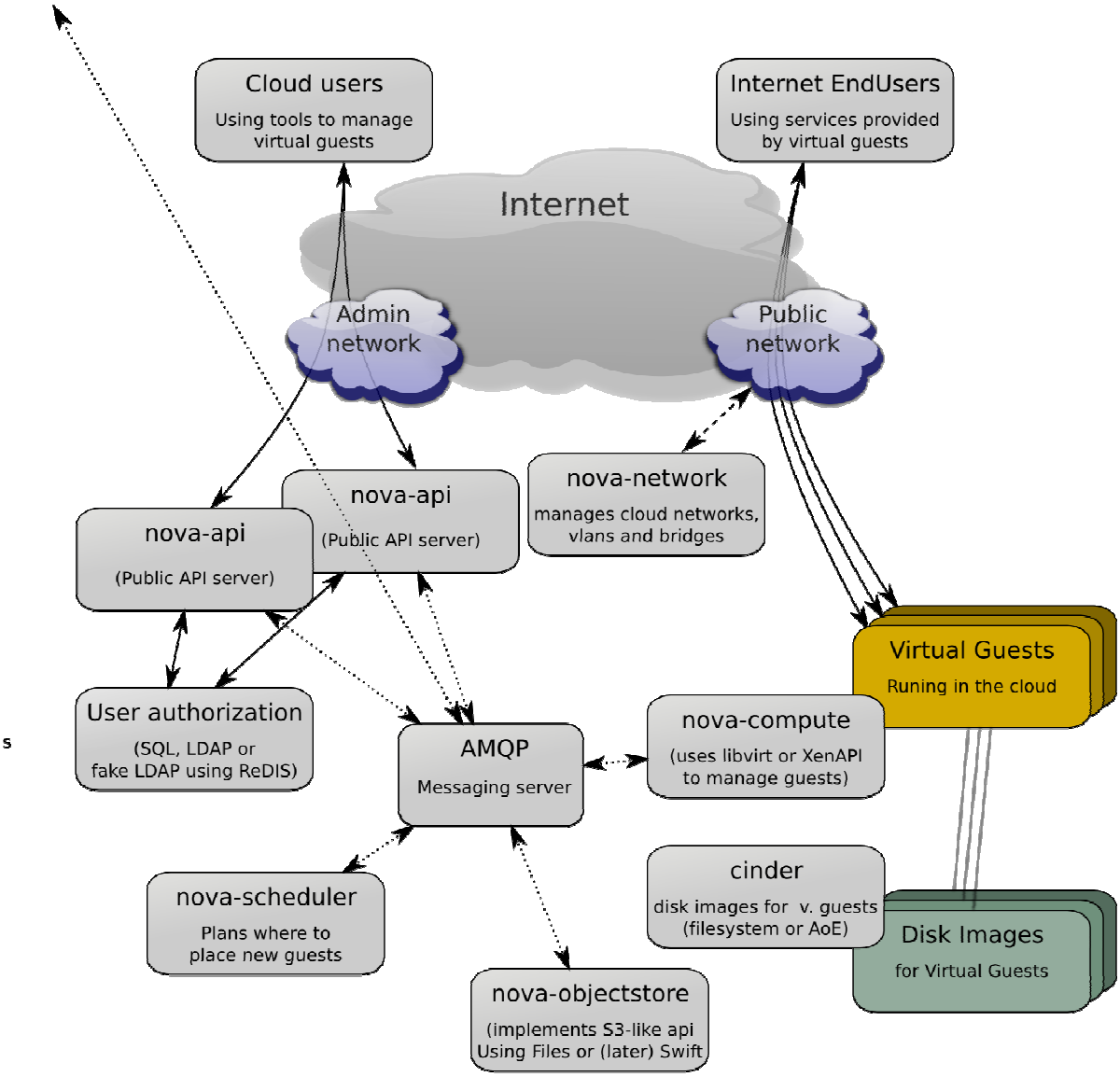
图表 2 OpenStack 逻辑架构

用户可以通过 Horizon 跟其他服务交互，或者通过其他服务提供的 API。

所有的服务都需要通过 Keystone 来进行认证。

部分服务通过公开的 API 跟其他服务进行交互。

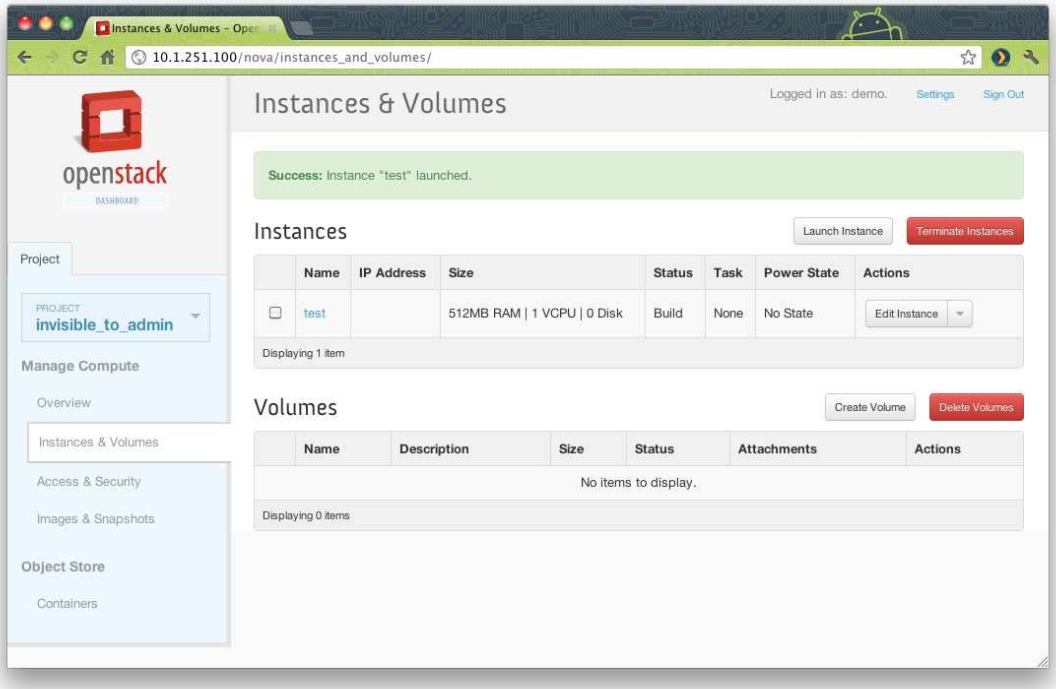
1.2.3 服务架构



图表 3 服务和通信架构

1.3 Horizon-前端面板

Horizon 是一个模块化的 Django web 应用，它为用户和管理员提供访问 OpenStack 服务的界面接口，如图表 4。



图表 4 Horizon 提供用户接口界面

跟大部分的 web 应用类似，Horizon 的架构比较简单：

- Horizon 一般通过 Apache 中的 mod_wsgi 进行部署。它的代码独立成为一个可重用的 python 模块，包括大部分的逻辑（跟不同的 OpenStack API 交互）和展示层（为不同站点提供容易的定制）。
- 一个数据库。因为大部分数据都是依赖自其它服务，自身存储的数据很少。

从网络架构的角度，Horizon 服务需要是用户可访问的，同时能跟其他服务的公开 API 交互。如果同时需要一些管理功能（例如，管理其他服务），则 Horizon 需要能够访问到其他服务的管理 API（这些 API 一般是用户不能直接访问的）。

1.4 Nova-计算

Nova 是最复杂，同时也是最分散的一个组件。该组件包括大量的进程合作来把用户的 API 请求发给运行中的虚拟机。包括如下的进程：

- Nova-api 接受和响应终端用户的计算 API 请求。支持 OpenStack 的计算 API，Amazon 的 EC2 API 和一些特定的管理 API（给超级用户提供管理操作）。并且发起大部分的协调（orchestration）操作，比如运行一个虚拟机实例。此外，还包括对策略的支持（主要是配额检查）。
- Nova-compute 主要是个工作守护进程（worker daemon），通过 hypervisor 的 API（包括 XenServer/XCP 的 XenAPI，KVM 和 QEMU 的 libvirt，VMWare 的 VMWareAPI 等）来创建和关闭虚拟机实例。进程要做的事情很复杂，但是工作过程却比较直观：不断从队列中获取请求并响应，执行一系列的系统命令（例如运行一个 KVM 实例），同时

更新数据库的状态。

- Nova-volume 负责管理为计算实例提供创建、挂载和卸载永久性存储（类似于 Amazon 的 Elastic Block Storage）。该进程支持很多类型的存储，包括 iSCSI、Ceph 中的 Rados Block Device。新出现的 Cinder 项目将最终取代 Nova-volume，在 Folsom 版本中，两者提供了类似的功能。
- Nova-schedule 该进程在概念上是 Nova 中最简单的一个进程：从队列中获取创建虚拟机实例的请求，并且决定在哪里运行它（特别的，运行在哪一台物理机上）。
- Queue 提供一个中央的 hub，来在各个 daemon 之间传递消息。基于 RabbitMQ 实现，但同时支持任何兼容 AMPQ 消息的 queue 机制（包括 Apache Qpid 和 Zero MQ）。
- SQL 数据库 存储架构中大部分的创建和运行时状态。包括可用、在用、网络可用的实例类型和项目等。理论上，Nova 可以支持被 SQL_Alchemy 支持的任意数据库，目前广泛应用的包括 sqlite3（推荐仅用于测试和开发）、MySQL 和 PostgreSQL。
- Nova 还提供了一些控制台服务，让用户可以通过一个 proxy 来访问虚拟实例的控制台。包括若干 daemon（nova-console、nova-vncproxy 和 nova-consoleauth）。

Nova 跟 OpenStack 大部分的其他服务都有交互。例如需要 KeyStone 提供认证、Glance 提供镜像管理、Horizon 提供 web 接口。与 Glance 的交互是核心。API 进程可以上传和查询 Glance，同时 nova-compute 可以通过 Glance 下载镜像以运行。

1.5 Swift-对象

Swift 在架构上也是分布式的，以避免单点故障（single point of failure）和支持横向扩展，包括如下的子组件。

- Swift-proxy-server 接受通过 OpenStack 对象 API 或原始 HTTP 的来访请求。接受包括文件上传、元信息修改和容器创建（container creation）。另外，为 web 浏览器提供文件或列出容器服务。该自组件通常采用可选的 cache（一般部署 memcache）来提高性能。
- Account servers 管理被对象存储服务（object storage service）定义的账户。
- Container servers 管理在对象储存服务（object store service）中容器（例如文件夹）的映射。
- Object servers 管理存储节点上的实际对象（例如对象）。

此外，还有一些周期性的进程，负责大数据仓库中一些管理维护工作。其中最重要的是冗余服务（replication services），来确保 cluster 中数据的一致性和可用性。其他的周期性进程包括审计（auditors）、更新（updaters）和收获（reapers）。

对象仓促可以通过 HTTP 来提供静态的网页或对象。例如图像、多媒体等。

认证是通过可配置的 WSGI 中间件来负责的，即 Keystone。

1.6 Glance-镜像

Glance 组件的架构从 Cactus 版本以来就一直很稳定。最大的改变包括添加了认证。Glance 主要包括四个主要部分：

- Glance-api 接受镜像 API 调用，包括发现镜像、获取镜像和存储镜像等。
- Glance-registry 存储、处理和获取镜像的元数据（包括尺寸、类型等）。

- 数据库 存储镜像的元数据。数据库类型可选（i 一般推荐 MySQL 或 SQlite）。
- 镜像文件的存储 一般镜像存储是在 Swift 中，但这是可以配置的。Glance 支持文件系统、RADOS 块设备、Amazon S3 和 HTTP。但部分仅支持读操作。

类似 Swift，Glance 中也包括一些周期性进程来支持 caching 和冗余等。

Glance 在整个 OpenStack 的概念架构上，起到了 IaaS 中的中心作用。它接受用户或 Nova 对镜像的请求 API，并且存储磁盘文件到对象服务 Swift 中。

1.7 Keystone-鉴权

Keystone 提供了一个对 OpenStack 中策略（policy）、登记（catalog）、口令（token）和认证（authentication）的支持。

- 处理 API 请求，包括提供可配置的登记、策略、口令和身份服务。
- 每个 Keystone 的功能，后面都是一个可插拔式（pluggable）的后端，从而可以采用多种不同的服务。大部分支持的标准后端包括 LDAP、SQL 和 Key Value Store（KVS）。

Keystone 一般常被用来提供认证服务。

1.8 Quantum-网络

Quantum 试图在 OpenStack 其他服务（大部分情况下是 Nova）管理的接口设备之间提供“网络连接即服务（network connectivity as a service）”。Quantum 允许用户创建自由的网络，同时将接口连接上去。跟 OpenStack 中很多服务类似，Quantum 的架构也是可配置的，支持插件（plug-in）。这些插件适应不同的网络设备和软件。因此，Quantum 的架构和部署都是可以快速调整的。

- Quantum-server 接受接受 API 请求，转发给合适的 Quantum 插件上。
- Quantum 插件和代理 执行实际的操作，包括插上、拔下端口、创建网络、划分子网和管理 IP 地址。这些插件和代理可以来自不同的提供商。Quantum 自身支持或已配置的插件和代理包括 Cisco 虚拟和物理交换机，Nicira 的 NVP 产品，NEC OpenFlow 产品、Open vSwitch、Linux bridging 和 Ryu 网络控制器。Midokua 提供了一个插件来帮助整合。常见的代理包括 L3、DHCP 和其他指定的插件代理。
- 大部分情况下，Quantum 采用一个消息队列来在 Quantum-server 和各种代理之间传递消息，同时采用数据库来为一些特定插件存储网络状态。

Quantum 大部分交互都是跟 Nova 进行的，为虚拟机提供网络连通服务。

1.9 Cinder-块存储

Cinder 将 nova-volume 服务单独剥离出来，提供永久性块存储服务。提供 API 来支持操作存储卷、存储类型和快照等。

- Cinder-api 接受 API 请求，并转发给 cinder-volume。
- Cinder-volume 通过读写 Cinder 数据库来响应请求，包括维护状态、跟其他进程打交道、提供软件和硬件等。通过驱动层，可以跟不同类型的存储设备交互，包括 IBM、SolidFire、NetApp、Nexenta、Zadara、Linux iSCSI 等。
- Cinder-scheduler 选取最优的块设备节点来创建存储卷。

- Cinder 采用一个消息队列来在 Cinder 各个进程之间传递消息，同时采用数据库类存储存储卷状态。

与 Quantum 类似，Cinder 大部分交互都是跟 Nova 进行的，为虚拟实例提供存储服务。

1.10 未来版本中的部件

这些项目可能在今后版本的 OpenStack 中出现，包括

Ceilometer 提供一些测量（metering）信息，让提供接口展现 OpenStack 中的一些内部活动。该项目并非计费项目。要实现计费，需要测量、评估和计费。该项目让用户可以更好的了解哪些操作被执行了，评估则负责价格和条目、计费则计算费用，并发给用户。

Heat 提供 REST API 来协调多个实现标准的云应用，例如 ASW 的 CloudFormation。

1.11 参考

http://ken.pepple.info/openstack/2012/09/25/openstack-folsom-architecture/
http://docs.openstack.org/trunk/openstack-compute/install/apt/content/index.html