OpenStack Object Storage (Swift)

目 录

1. OpenStack 项目介绍

OpenStack 是由 NASA（美国国家航空航天局）和 Rackspace 共同发起的，获 Apache 许可证授权，是一个自由软件和开放源代码项目。它是一种 IaaS（基础设施即服务）云平台，让任何人都可以自行建立和提供云计算机服务，即可用它建立公共云平台，也可以建立私有云平台。OpenStack 由 Python 语言编写，目前发布周期为 3 个月。

由三个子项目组成，分别是：

OpenStack 计算（Nova）——云控制器，提供网络协调时所需的软件、控制面板和 API，其中包括运行实例、管理网络和控制访问。

OpenStack 对象存储(Swift)—— 使用标准的服务器集群为数千万亿字节的存取数据提供冗余的、可伸缩的数据存储。

OpenStack 镜像服务（Glance）——为虚拟硬盘镜像提供发现、注册和传递等服务。

这三个项目相互独立，可以单独安装，其中的对象存储（Swift）是我们要关注的内容。

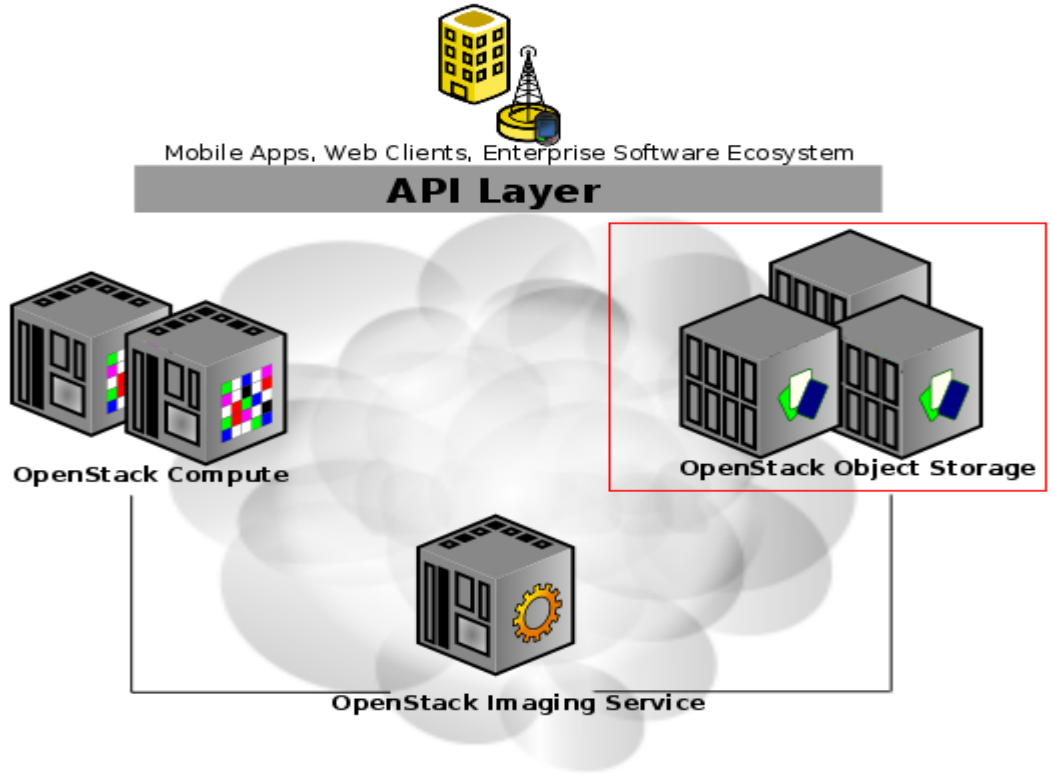

图 1 OpenStack 组件图

2 OpenStack 对象存储（Swift）概况

Swift 是一种可伸缩的对象存储系统，采用标准化的服务器存储 PB 级可用数据。内建冗余和数据恢复机制，没有主控节点，因而具有极好的性能以及可扩展性。图 2 是 OpenStack 对象存储概念图。Swift 不能像传统文件系统那样进行挂载和访问，只能通过 ReST API 接口来访问数据，并且这些 API 与亚马逊的 S3 服务 API 是兼容的。Swift 不同于传统文件系统和实时数据存储系统，它适用于存储、获取、一些静态的永久性的数据并在需要的时候进行更新，比如说，虚拟机镜像，图片，邮件，文档的备份等。

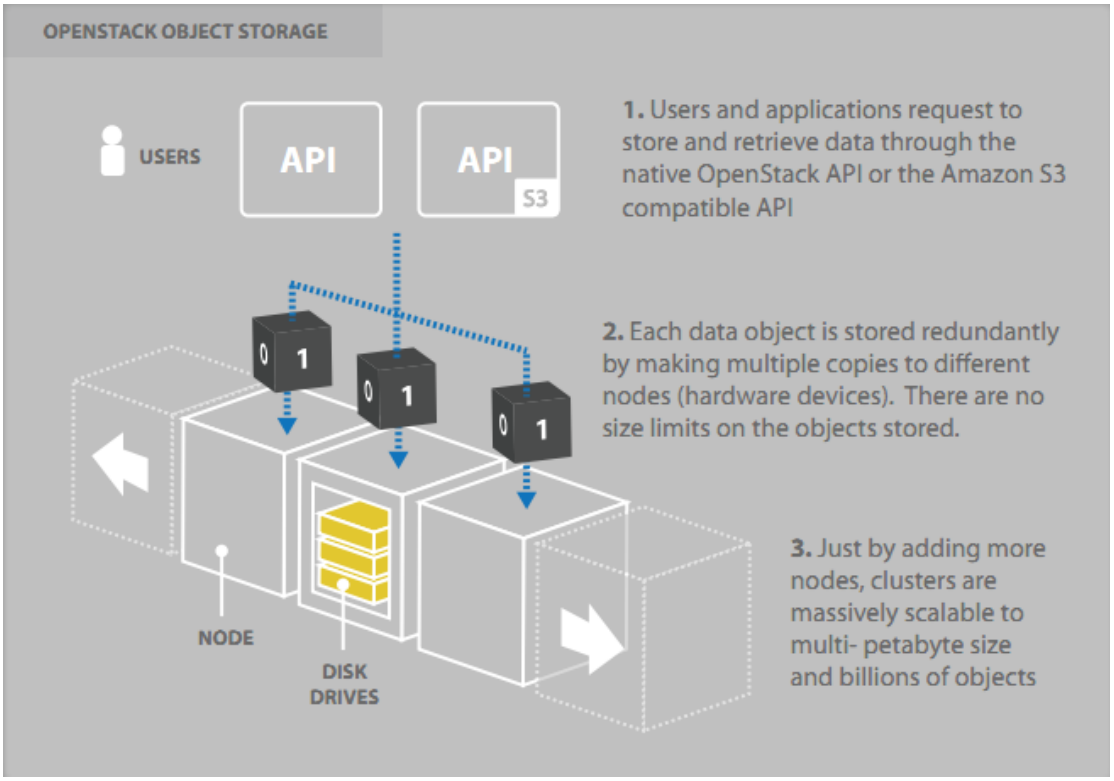

图 2 OpenStack 对象存储概念图

Swift 的特点和好处如表 1 所示：

表 1 Swift 的特点和好处

Feature	Benefit
Store and Manage files programmatically via API 通过 API 编程实现文件存储和管理	Automates resource nagement/provisioning 资源管理可以自动化
Create Public or Private containers	Better control. Allows to share data publicly or keep it private

能够创建公共和私有容器	更自由的访问权限控制。既允许用户间共享数据，也可以保存隐私数据
Leverages Commodity hardware 充分利用商用硬件	No lock-in, lower price/GB 节约单位存储成本
HDD/node failure agnostic 透明的硬盘/节点故障处理能力	Self-healing. Reliability, Data redundancy protecting from failures 系统故障能够自我修复，内建冗余机制使得部分节点故障不影响数据取用
Unlimited Storage 无限的存储容量	Huge & flat namespace, highly scalable read/write access Ability to serve content directly from storage system 巨大、平级的命名空间，高度的可扩展性，不经中间节点直接访问存储设备
Multi-dimensional scalability (scale out architecture) Allows to scale vertically and horizontally-Distributed storage 多维的扩展能力，允许横向、纵向扩展存储空间	Backup/Archive large amounts of data with linear performance 可以高效地备份、归档大规模数据
Account/Container/Object structure no nesting, not a traditional file system 账号/容器/对象架构，不能嵌套，不同于传统文件系统	Optimized for scale Allows to scale to multiples Peta-bytes, billions of objects 优化扩展性能，能过存储 PB 级别数据，数十亿对象
Built-in Replication(N copies of accounts, container, objects) 3x+ data redundancy compared to 2x on RAID 内建冗余机制 RAID 技术只做两个备份，而 Swift 最少有 3 个备份	High Availability 高可靠性
Easily add capacity unlike RAID resize 可以方便地进行存储扩容	Elastic data scaling with ease 方便的扩容能力
No central database 没有中心节点	Higher performance, No bottlenecks 高性能，无瓶颈限制
RAID not required 不需要使用 RAID 技术	Allows to handle lots of small, random reads and writes efficiently

	允许高效的处理大量的小文件随机读写
Built-in Mgmt. utilities 内建 Mgmt 能力	Acct. Management: Create, add, verify, delete users. Container Management: upload, download, verify. Monitoring: Capacity, Host, Network, Log trawling, cluster health 账号管理：创建、添加、验证、删除用户；容器管理：上传、下载、验证；监控容量、主机、网络、日志、集群运行的稳定性

3. Swift 中的若干基本概念

Account

出于访问安全性考虑，使用 Swift 系统，每个用户必须有一个账号（Account）。只有通过 Swift 验证的账号才能访问 Swift 系统中的数据。提供账号验证的节点被称为 Account Server。Swift 中由 Swauth 提供账号权限认证服务。

用户通过账号验证后将获得一个验证字符串（authentication token.），后续的每次数据访问操作都需要传递这个字符串。

Container

Swift 中的 container 可以类比 Windows 操作系统中的文件夹或者 Unix 类操作系统中的目录，用于组织管理数据，所不同的是 container 不能嵌套。数据都以 Object 的形式存放在 container 中

Object

Object（对象）是 Swift 中的基本存储单元。一个对象包含两部分，数据和元数据（metadata）。其中元数据包括对象所属 container 名称，对象本身名称以及用户添加的自定义数据属性（必须是 key-value 格式）。

对象名称在 URL 编码后大小要求小于 1024 字节。用户上传的对象最大是 5GB，最小是 0 bytes。用户可以通过 Swift 内建的大对象支持技术获取超过 5GB 的大对象。对象的元数据不能超过 90 个 key-value 对属性，并且这些属性的总大小不能超过 4KB。

Account、Container、Object 是 Swift 系统中的 3 个基本概念，三者的层次关系是一个 Account 可以创建拥有任意多个 Container，一个 Container 中可以包含任意多个 Object。

在 Swift 系统中，集群被划分成多个区（zone），区可以是一个磁盘，一个服务器，一台机柜甚至一个数据中心，每个区中有若干个节点（Node）。Swift 将 Object 存储在节点（Node）上，每个节点都是由多个硬盘组成的，并保证对象在多个节点上有备份（默认情况下，Swift 会给所有数据保存 3 个复本）以及这些备份之间的一致性。备份将均匀地分布在集群服务器上，并且系统保证各个备份分布在不同区的存储设备上，这样可以提高系统的稳定性和数据的安全性。它可以通过增加节点来线性的扩充存储空间。当一个节点出现故障，Swift 会从其它正常节点对出故障节点的数据进行备份。

4. Swift 中使用的相关技术

4.1 对象存储

对象是系统中数据存储的基本单位，一个对象实际上就是文件的数据和一组属性信息（metadata）的组合，这些属性信息可以定义基于文件的 RAID 参数、数据分布和服务质量等，如图 3 所示。而传统的存储系统中用文件或块作为基本的存储单位，在块存储系统中还需要始终追踪系统中每个块的属性。对象通过与存储系统通信维护自己的属性。

在对象存储中，存储的不仅是数据，还有与丰富的数据相关的属性信息。图 4 为文件存储与对象存储所包含的属性信息数量的对比。系统会给每一个对象分配一个唯一的 ID。对象本身是平等的，所有的 ID 都属于一个平坦的地址空间，而并非文件系统那样的树状逻辑结构。

这种存储结构带来的好处是可以实现数据的智能化管理，因为对象本身包含了元数据信息，甚至更多的属性，我们可以根据这些信息对对象进行高效的管理。例如我们可以制定这样的存储策略，如果对象中包含 priority:high，这样的属性，我们就对文件做比平常文件多的备份次数（5 次）。平坦地址空间的设计使得访问对象只通过一个唯一的 ID 标识即可，不需要复杂的路径结构。

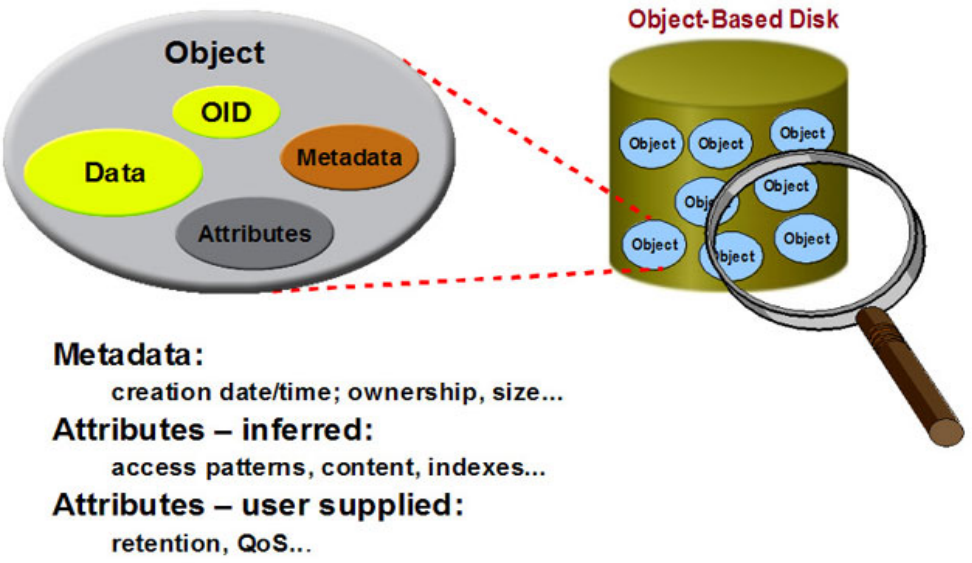

图 3 对象包含数据、对象的 OID、元数据和属性

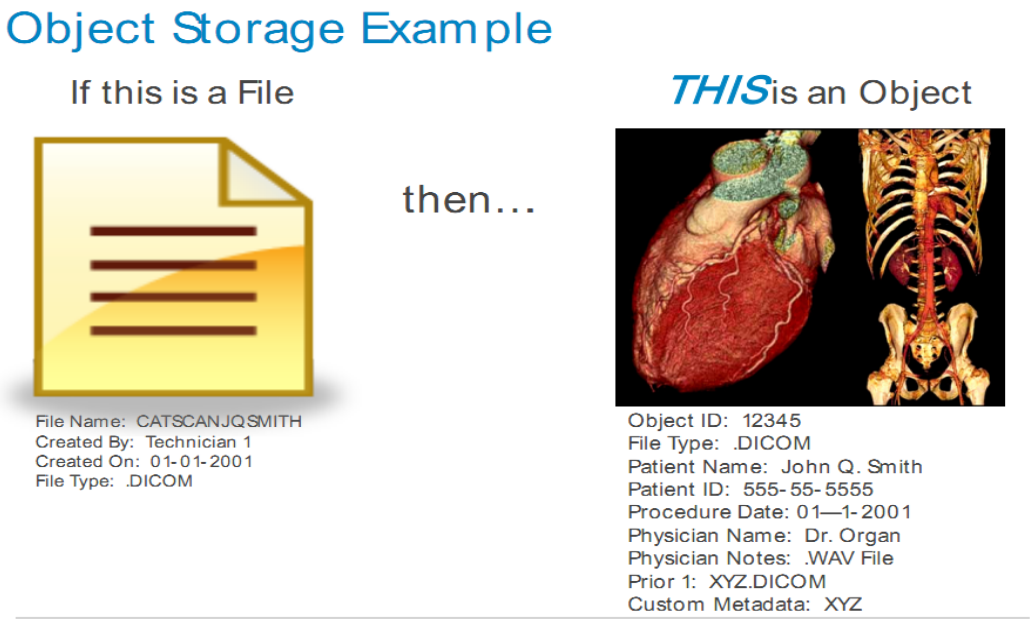

图 4 文件存储与对象存储所包含的属性信息数量的对比

4.2 REST 软件架构

REST 是 Roy Fielding 博士在 2000 年他的博士论文中提出来的一种软件架构风格。REST（Representational State Transfer）是一种轻量级的 Web Service 架构风格，其实现和操作明显比 SOAP 和 XML-RPC 更为简洁，可以完全通过 HTTP 协议实现，还可以利用 Cache 来提高响应速度，性能、效率和易用性上都优于 SOAP 协议。

REST 架构遵循了 CRUD 原则，CRUD 原则对于资源只需要四种行为：Create（创建）、Read（读取）、Update（更新）和 Delete（删除）就可以完成对其操作和处理。这四个操作是一种原子操作，即一种无法再分的操作，通过它们可以构造复杂的操作过程，正如数学上四则运算 是数字的最基本的运算一样。

REST 架构让人们真正理解我们的网络协议 HTTP 本来面貌，对资源的操作包括获取、创建、修改和删除资源的操作正好对应 HTTP 协议提供的 GET、POST、PUT 和 DELETE 方法，因此 REST 把 HTTP 对一个 URL 资源的操作限制在 GET、POST、PUT 和 DELETE 这四个之内。这种针对 网络应用的设计和开发方式，可以降低开发的复杂性，提高系统的可伸缩性。

因为其简洁方便性，越来越多的 web 服务开始采用 REST 风格设计和实现。例如，Amazon.com 提供接近 REST 风格的 Web 服务进行图书查找；雅虎提供的 Web 服务也是 REST 风格的。

因为 Swift 采用 REST 架构，我们不能像普通的文件系统那样对数据进行访问，必须通过它提供的 API 来访问操作数据，如图 5 所示，图 6 和图 7 分别展示了 Swift 的上传和下载操作。Rackspace 还对这些 api 做了不同语言的封装绑定，以方便开发者进行开放。目前支持的语言有 PHP、Python、Java、C#/.NET 和 Ruby。

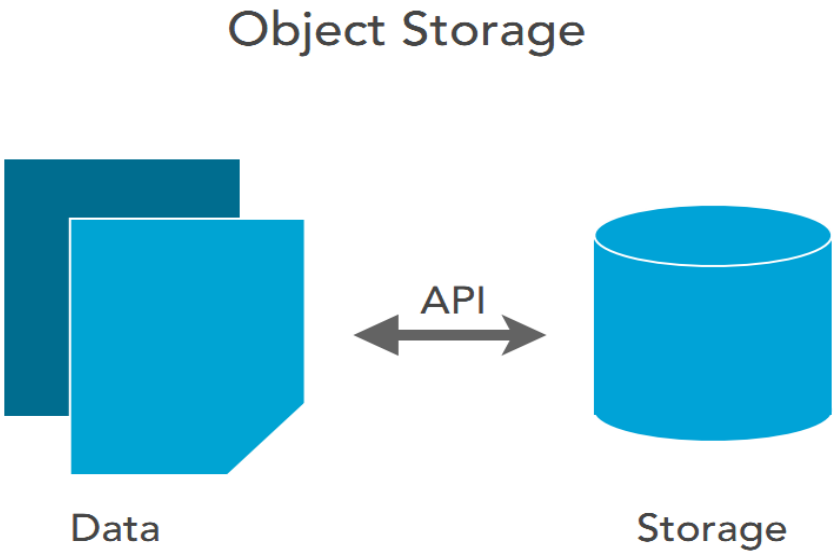

图 5 通过 api 来和 Swift 存储系统进行交互

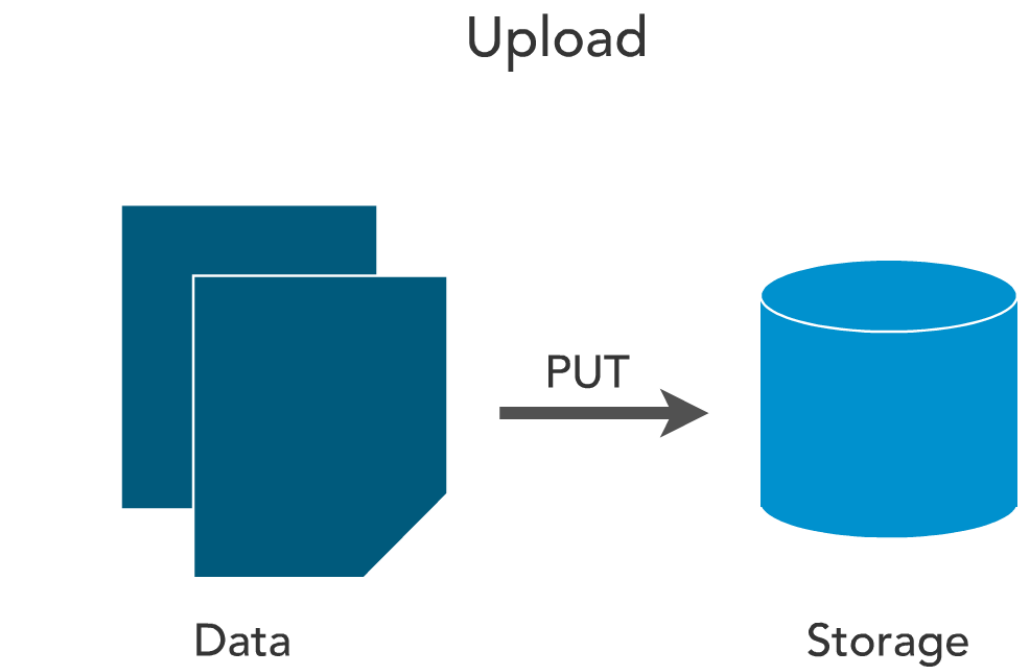

图 6 通过 http 的 put 方法上传数据

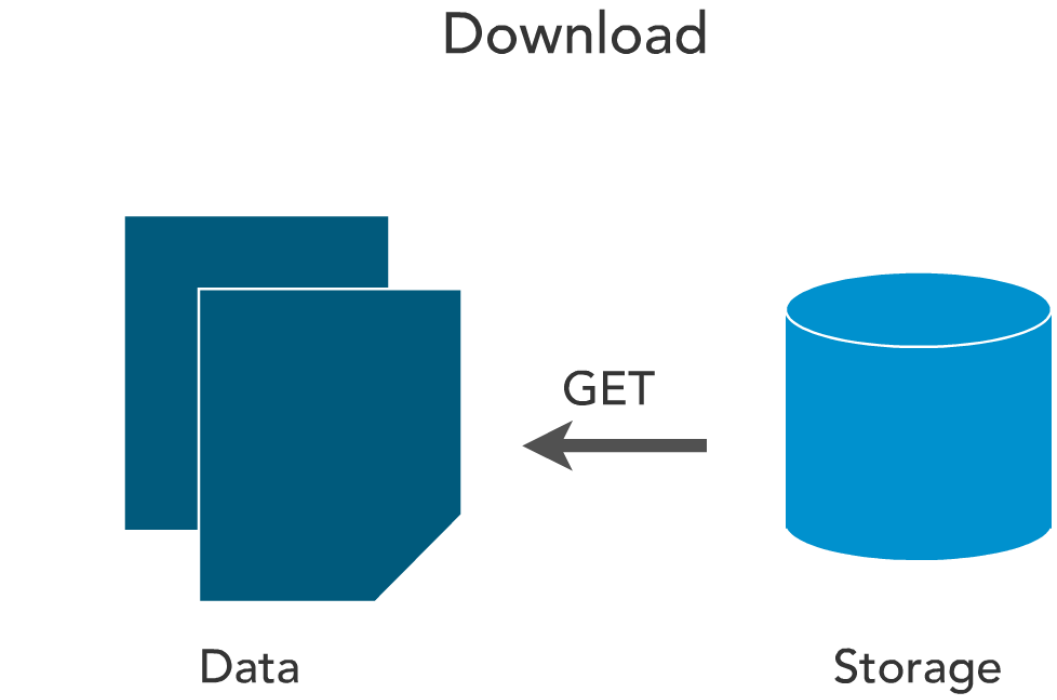

图 7 通过 http 的 get 方法下载数据

5 安装 Swift 硬件及系统要求

具体安装操作详见官网上的文档（http://www.openstack.org/），下面简单介绍安装所需的硬件和系统要求。图 8 为一个最小的 Swift 系统架构图。

硬件：商用服务器即可。服务器内置存储不需要也不建议采用 RAID 技术。Swift 的磁盘模式非常不适合采用 RAID 技术，使用后将极大的降低存储性能。

操作系统： Ubuntu 系统

网络：推荐采用 1000Mbps 网络。

数据库： SQLite 数据库是 OpenStack 对象存储账号和容器管理进程的组成部分。

权限： 在安装时你可以选择安装成 root 用户或者是拥有 sudo 权限的普通 user 用户。

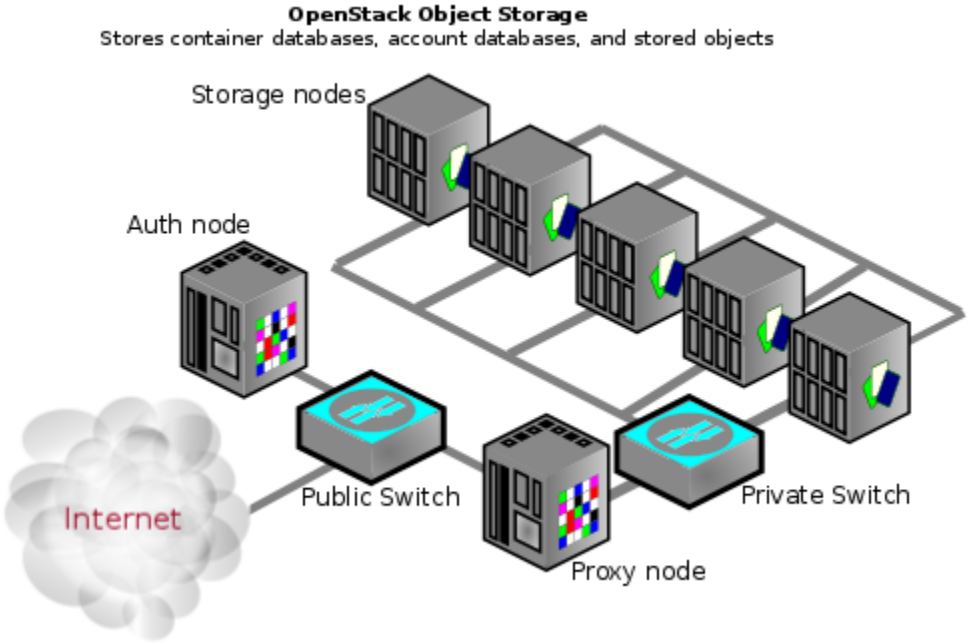

图 8 一个最小的 Swift 系统架构图

6 使用现状

在国外，有商用的成功案例：Rackspace 公司通过结合 Swift 和 nova 提供 IaaS 云服务，http://www.rackspace.com/cloud/；微软的 SharePoint 的后端支持，http://sharepoint.microsoft.com；韩国电信公司（KT）推出的云服务，https://cs.ucloud.com/。

在国内尚未有商用的成功案例，大部分都处于实验阶段，上海交大信息中心今年用 OpenStack 做了一个私有云，50 台服务器，500GB SSD storage, 100TB block storage and 400TB object storage。

2011 年 9 月 6 日首届开源云 OpenStack 峰会在上海举行，可见 openstack 在国内的还是研究的比较热的。

7 总结

- ✓ *Swift 特点小结*

1) Swift 提供 RESTful api，及相关的语言绑定，编程方便，另外它的 API 和亚马逊的 S3 兼容，所以用于 S3 的应用同样可以用于 Swift。
2) Swift 使用对象存储，可以实现数据的智能管理。适合存储长期静态数据。
3) 可以通过简单的增加节点来扩充存储容量。
4) 通过多次备份数据以及数据的自动修复，可以保证数据的稳定；
5) 根据 Swift 的官方文档，上传文件的上限是 5G，下限是 0。没有发现在文件大小偏好方面的说明，如适合大文件还是小文件。

- ✓ *使用 Swift 技术上的可行性*

Rackspace 用 Swift 已经有多年时间。如过我们使用 Swift 来提供存储服务。现有的可利用资源有完全开源的代码，规范的代码注释和风格，现有的系统部署和管理文档，Rackspace 公布的一些简单的注意事项， 现有的面向开发者的 REST 接口文档，现有的多种语言封装之后的库。

8 参考资料

1) OPENSTACK OBJECT STORAGE ADMINISTRATOR MANUAL - CACTUS
http://docs.openstack.org/cactus/openstack-object-storage/admin/content/index.html
2) OpenStack 维基 http://zh.wikipedia.org/wiki/OpenStack
3) OpenStack 项目介绍
http://hi.baidu.com/chenshake/blog/item/5fcb2b7a4307c3fa2f73b3fb.html
4) 共享存储的分类
http://hi.baidu.com/chenshake/blog/item/9b74c43398327f55ac4b5ff5.html
5) 什么是对象存储？1 http://wangxu.me/site/node/33
6) 什么是对象存储？2 ttp://www.chinastor.com/a/jishu/OSD.html
7) Sun Object-Based Storage Devices
http://developers.sun.com/solaris/articles/osd.html
8) Dell DX Object Storage Platform

http://www.slideshare.net/PeterMorel/dell-dx-objectstorageplatform-r
9) Object Storage——A Fresh Approach to Long-Term File Storage, A Dell Technical White Paper
10) Openstack 9 月 6 号在上海峰会 PPT
http://www.slideshare.net/ben_duyujie/presentations
11) OpenStack APIs: Present and Future (Beta Talk)
http://www.slideshare.net/wademinter/openstack-apis-present-and-future-beta-talk
12) openstack Swift 典型架构和 openstack Swift 简要说明
http://blog.sina.com.cn/s/blog_6b98772b0100pk7p.html
13) 基于 REST 架构的 Web Service 设计
http://www.williamlong.info/archives/1728.html
14) Learn REST: A Tutorial http://rest.elkstein.org/
15) RESTful Web services: The basics
http://www.ibm.com/developerworks/webservices/library/ws-restful/