OpenStack Object Storage (Swift) Essentials

OpenStack is a rapidly growing open source cloud software; and Swift is one such project. Swift allows users to build cloud storage with the help of inexpensive commodity hardware. With the latest OpenStack Juno release, developers and experts have access to hundreds of new features to build robust cloud infrastructures.

This book teaches you the fundamentals of cloud storage and OpenStack Swift and gives you the skillset to build and operate your own cloud storage. This book also explains the usefulness of OpenStack Swift for cloud storage and how it provides an architectural review of Swift. Next, the book covers the installation and management of Swift, along with other new topics including quality of service and storage policies. Hardware choice, Swift tuning, and use cases will round out your skills. This edition also features two new chapters covering OpenvStorage and Sahara. The book then demonstrates monitoring and new packaging methods, including Docker. Get a head start in the world of cloud storage using this content-rich book loaded with topics.

Who this book is written for

If you are an IT administrator and you want to enter the world of cloud storage using OpenStack Swift, then this book is ideal for you. Basic knowledge of Linux and server technology is beneficial to get the most out of the book.

What you will learn from this book

- Learn the intricacies of cloud storage and discover the new features of Swift
- Understand OpenStack Swift's architecture and install a multicluster environment
- Use new packaging techniques such as Docker
- Perform basic and advanced operations for day-to-day management using REST APIs
- Know how to choose the right hardware configuration for your needs
- Tune Swift for your particular workload and use case
- Explore new Swift interfaces such as Cinder and Sahara

OpenStack Object Storage (Swift) Essentials

Design, implement, and successfully manage your cloud storage using OpenStack Swift

Amar Kapadia         Kris Rajana        Sreedhar Varma

In this package, you will find:

- The authors biography
- A preview chapter from the book, Chapter 1 'Cloud Storage – Why Can't I Be Like Google?'
- A synopsis of the book’s content
- More information on OpenStack Object Storage (Swift) Essentials

About the Authors

**Amar Kapadia** is a storage technologist and blogger based in the San Francisco Bay Area. He is currently the senior director of product marketing for Mirantis, the #1 pure-play OpenStack company. Prior to Mirantis, he was the senior director of strategy for EVault's Long-Term Storage Service, a public cloud storage offering based on OpenStack Swift. He has over 20 years of experience in storage, server, and I/O technologies at Emulex, Philips, and HP. Amar's current passion is in cloud and object storage technologies. He holds a master's degree in electrical engineering from the University of California, Berkeley.

When not working on OpenStack Swift, Amar can be found working on technologies such as Kubernetes, MongoDB, PHP, or jQuery. His blogs can be found at www.buildcloudstorage.com.

**Kris Rajana** is a technologist and serial entrepreneur, passionate about building globally distributed teams to deliver innovative infrastructure solutions. His areas of interest include data infrastructure and fast-emerging open source cloud storage technologies, such as OpenStack, Cloud Foundry, Dockers/Containers, and big data. As the CEO of Vedams and Biarca (an offshoot of Vedams), he takes immense pride in his team and its development, which leads to excellence in execution. Kris has over 20 years of experience in managing engineering teams in fields such as space, aviation, storage at BFGoodrich Aerospace, Snap Appliance (currently Overland Storage), Adaptec, Xyratex, and Sullego. His current passion is DevOps, and he likes to leverage leading open source cloud technologies to make enterprises more agile, speed up the development and deployment of modern enterprise applications, and make IT operations more efficient. Kris earned his doctorate in engineering science from Pennsylvania State University.
He is a member of the board of the Pratham Bay Area Chapter. Along with the Vedams team, he is a sponsor of an urban learning center in Hyderabad. He is a student and sevak of the San Jose Chinmaya mission.

Sreedhar Varma has more than 15 years of experience in the storage industry, and has worked on various storage technologies such as SCSI, SAS, SATA, FC HBA drivers (Adaptec, Emulex, Qlogic, Promise, and so on), RAID, storage stacks of various operating systems, and system software for fault-tolerant and high-availability systems. He has good experience with SAN, NAS, and iSCSI networks; various storage arrays (Dothill, IBM, EMC, Netapp, Oracle Pillar, and so on); object storage implementations (Swift and Ceph); and software development using the corresponding REST APIs.

Sreedhar is currently working for Vedams software providing storage engineering services. In the past, he has worked for Stratus Technologies, Compaq, Digital Equipment Corp, and IBM. He has a master's degree in computer science from the University of Massachusetts.
OpenStack Object Storage (Swift) Essentials

CIOs around the world are asking their teams to take advantage of cloud technologies as a way to cut costs and improve usability. OpenStack is a piece of fast-growing open source cloud software with a number of projects, and OpenStack Swift is one such project that allows users to build cloud storage. With Swift, users can not only build storage using inexpensive commodity hardware, but also use the public cloud storage built using the same technology. Starting with the fundamentals of cloud storage and OpenStack Swift, this book will provide you with the skills required to build and operate your own cloud storage or use a third-party cloud. This book is an invaluable tool if you want to get a head start in the world of cloud storage using OpenStack Swift. You will be equipped to build an on-premise private cloud, manage it, and tune it.

What This Book Covers

Chapter 1, Cloud Storage – Why Can’t I Be Like Google?, introduces the need for cloud storage, the underlying technology of object storage, and an extremely popular open source object storage project called OpenStack Swift.

Chapter 2, OpenStack Swift Architecture, discusses the internals of the Swift architecture in detail, and shows you how elegantly Swift converts commodity hardware into reliable and scalable cloud storage.

Chapter 3, Installing OpenStack Swift, walks you through all the necessary steps required to perform a multinode Swift installation, and show you how to set it up along with the Keystone setup for authentication.

Chapter 4, Using Swift, describes the various ways in which you can access Swift object storage. This chapter also provides examples for the various access methods.

Chapter 5, Additional Swift Interfaces, describes the interfaces available for using Swift object storage as data stores (block storage), as well as the Swift interface within Sahara.

Chapter 6, Monitoring and Managing Swift, provides details on the various options that are available for monitoring and managing a Swift cluster. Some of the topics covered in this chapter are StatsD metrics, handling drive failures, node failures, and migrations.

Chapter 7, Docker Intercepts Swift, describes dockerization of Swift services and how to deploy a dockerized Swift image.
Chapter 8, *Choosing the Right Hardware*, provides you with the information necessary to make the right decision in selecting the required hardware for your cloud storage cluster.

Chapter 9, *Tuning Your Swift Installation*, walks you through a performance benchmarking tool and the basic mechanisms available for tuning a Swift cluster. Users utilizing Swift will need to tune their installation to optimize performance, durability, and availability, based on their unique workload.

Chapter 10, *Additional Resources*, explores several use cases of Swift and provides pointers on operating systems, virtualization, and distribution tools used across various Swift installations.

Appendix, *Swift CLI Commands*, provides details on various commands that can be run from a Swift CLI session.
Cloud Storage – Why Can't I Be Like Google?

If you could build your IT systems and operations from scratch today, would you recreate what you have? That's the question Geir Ramleth, CIO of the construction giant Bechtel, asked himself in 2005. The answer was obviously not, and Bechtel ended up using the best practices from four Internet forerunners of that time—YouTube, Google, Amazon, and Salesforce—to create their next set of data centers.

This is exactly the same question CIOs and IT administrators around the world are asking themselves! In this book, you will learn about a revolutionary new storage system called cloud storage that uses the best practices (though not the exact technologies) of these web giants. This will cut the total cost of ownership (TCO) of storage by more than 10 times compared to traditional enterprise block or file storage.

This book will show you how you can implement cloud storage using a leading open source storage software stack called OpenStack Swift. Let's first explore some key elements that constitute cloud storage:

- Dramatic reduction in TCO
- Unlimited scalability
- Elasticity achieved by virtualization
- On-demand; that is, pay for what you use
- Universal, that is, access from anywhere
- Multitenancy, which means sharing storage hardware with other departments or companies
- Data durability and availability, even with partial failures of the storage system
What constitutes cloud storage?
Let’s review each of these elements of cloud storage in more detail.

Reduced TCO
Reduced TCO is the crux of cloud storage. Unless this new storage cuts storage cost by more than 10 times, it is not worth switching from block or file storage and dealing with something new and different. By total cost of ownership, we mean the total of capital expenditure (CAPEX) which involves equipment and operational expenditure (OPEX) in the form of IT storage administrators, electricity, power, cooling, and so on. This TCO reduction must be achieved without sacrificing durability (keeping data intact) or availability.

Unlimited scalability
Whether the cloud storage offering is public (that is, offered by a service provider) or private (that is, offered by central IT), it must have unlimited scalability. As we will see, cloud storage is built on distributed systems, which means that it scales very well. Traditional storage systems typically have an upper limit, making them unsuitable for cloud storage.

Elastic
Storage virtualization decouples and abstracts the storage pool from its physical implementation. This means that you can get an elastic (grow and shrink as required) and unified storage pool, when in reality, the underlying hardware is neither. IT professionals who have spent endless hours forecasting data growth and then waiting for their equipment will appreciate the magnitude of this benefit.

On-demand
Consumers do not reserve blocks of electricity and pay for it upfront, yet we routinely pay for storage upfront, whether we use it or not. Cloud storage uses a pay-as-you-go model, where you pay only for the data stored and the data accessed. For a private cloud, there is a minimal cluster to start with, beyond which it is on-demand. This can result in huge cost savings for the storage user.
Universal access
Existing enterprise storage has limitations in terms of access. Block storage is very limiting; a server has to be on the same storage area network, and storage volumes cannot be shared. Network-attached-storage (NAS) must be mounted to access it. This creates limitations on the number of clients and requires LAN access.

Cloud storage is extremely flexible—there is no limit on the number of users or from where you can access it. This is possible since cloud storage systems usually use a REST API over HTTP (GET, PUT, POST, and DELETE) instead of the traditional SCSI or CIFS/NFS protocols.

Multitenancy
Cloud storage is typically multi-tenant. The tenants may be different organizations in a public cloud or different departments in a private cloud. The benefit is centralized management and higher storage utilization, which reduces costs. Security, often an issue with multi-tenant systems, is addressed comprehensively in cloud storage through strong authentication, access controls, and various encryption options.

Data durability and availability
Cloud storage is able to run on commodity hardware, yet it is highly durable and available. This is even more impressive in that durability and availability is maintained in the face of a partial system failure. As with many modern distributed systems, the burden of data durability and availability is on the software layer rather than the underlying hardware layer.

Limitations of cloud storage
While cloud storage has numerous benefits, there are some limitations in the areas of performance and new APIs.

Performance
Storage systems have struggled to balance reliability, cost, and performance. Generally, you can get two out of these three aspects. Cloud storage optimizes reliability and cost, but not performance. In fact, as we will see later, reliability in cloud storage is better than the traditional RAID when you reach a large scale. By the way RAID works, you are at a very high risk of getting an unrecoverable failure during a RAID rebuild when operating at-scale. Cloud storage uses different techniques such as replication or erasure coding to provide high reliability.
This means that cloud storage is well suited for applications such as web servers and application servers, but not for databases or high-performance computing. It is also suitable for tier 2/3 storage, for example, backup, archival (photos, documents, videos, logs, and so on), and creating an additional copy for disaster recovery.

**New APIs**

Cloud storage affects applications in two ways: its interface with storage and its behavior. Firstly, applications need to port to a new and different storage interface utilizing HTTP instead of SCSI or CIFS/ NFS. Secondly, applications need to handle an eventually consistent storage system. The second part requires explanation.

Cloud storage is built using distributed systems that are governed by a theorem called the **CAP theorem**, which states that out of the following three points, it is impossible to guarantee more than two:

- **Consistency**: For cloud storage, this means that a request to any region or node returns the same data
- **Availability**: For cloud storage, this signifies that a request is successfully acknowledged with a response other than no response or an error
- **Tolerance to partial failures**: For cloud storage, this implies that the architecture is able to withstand failures in connectivity or parts of the system

Most cloud storage systems guarantee availability and tolerance to partial failures at the expense of consistency, making the system eventually consistent. This means that an operation such as an update may not be reflected to all nodes at the same time. Traditional applications expect strict consistency and may need to be modified.

If an application has not ported to cloud storage, is that a dead end? Fortunately not. There is a class of devices called cloud gateway that provides file or block interfaces to an application (for example, CIFS, NFS, iSCSI, or FTP/SFTP) and performs protocol conversion on the cloud. These gateways provide other functions as well, such as caching, WAN optimization, optional compression, encryption, and deduplication. They also eliminate the need for an application to handle the eventual consistency problem.
Object storage

How do you build a cloud storage system? The most suitable underlying technology is object storage.

Object storage is different from block or file storage as it allows a user to store data in the form of objects (essentially files) in a flat namespace using REST HTTP APIs. Object storage completely virtualizes the physical implementation from the logical presentation. It is similar to check-in luggage versus carry-on luggage, where once you put your check-in luggage in the system, you really don't know where it is. You simply get it back at your destination. With carry-on luggage, you have to know exactly where you have kept it at all times.

Object storage is built using scale-out distributed systems. Each node, most often, actually runs on a local filesystem. As we will see, object storage architectures allow for the use of commodity hardware, as opposed to specialized, expensive hardware used by traditional storage systems. The most critical tasks of an object storage system are as follows:

- Data placement
- Automating management tasks, including durability and availability

Typically, a user sends their HTTP GET, PUT, POST, HEAD, or DELETE request to any one node out from a set of nodes, and the request is translated to physical nodes by the object storage software. The software also takes care of the durability model by doing any one of the following: creating multiple copies of the object, chunking it, creating erasure codes, or a combination of these.

The durability model is not RAID because, as discussed earlier, RAID simply does not scale beyond hundreds of terabytes. The second critical task deals with management, such as periodic health checks, self-healing, and data migration. Management is also made easy by using a single flat namespace, which means that a storage administrator can manage the entire cluster as a single entity.
Let's evaluate through the following table how object storage meets the aforementioned cloud storage benefits:

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Ability to meet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low TCO</td>
<td>Storage nodes have no special requirements such as high availability, management, or special hardware such as RAID.</td>
</tr>
<tr>
<td></td>
<td>A single flat namespace with automated management features allows you to cut operational expenses (OPEX).</td>
</tr>
<tr>
<td></td>
<td>A full analysis of how this cuts the TCO by 10 times or more is beyond the scope of this book.</td>
</tr>
<tr>
<td>Unlimited scalability</td>
<td>A distributed architecture allows capacity and performance to scale.</td>
</tr>
<tr>
<td>Elasticity</td>
<td>A fully virtualized approach allows data to grow and shrink as necessary.</td>
</tr>
<tr>
<td>On-demand</td>
<td>A fully virtualized approach with centralized management allows storage to be offered as an on-demand self-service resource.</td>
</tr>
<tr>
<td>Universal access</td>
<td>REST HTTP APIs provide access from wherever the user is, with no restriction on the number of users.</td>
</tr>
</tbody>
</table>

The importance of being open

Although the need for software to be open is not a technical requirement, it is increasingly becoming a business requirement. Open means three things:

- **Open source**: While there are numerous benefits of open source software, the key advantages are the users' ability to influence the direction of the project, the velocity of innovation, reduced license fees, and the ability to switch vendors.

- **Open APIs**: To avoid vendor lock-in, the APIs must be open. Often, proprietary APIs are enticing upfront but lock users in.

- **Agnostic to underlying hardware choices**: To reduce hardware costs and maintain users' preferences, the software needs to be hardware agnostic.
OpenStack Swift

OpenStack Swift is a leading open source object storage project that meets the mentioned object storage and open technology requirements, and is the topic of this book. Let us first look at what the OpenStack project is about, and then specifically what OpenStack Swift (also referred to as just Swift) is.

OpenStack, a project launched by NASA and Rackspace in 2010, is currently the fastest growing open source project, and its mission is to produce a cloud computing platform useful for both public and private implementations. Its two core principles are simplicity and scalability. OpenStack has numerous subprojects under its umbrella, ranging from computing and storage to networking, among others. The object storage project is called Swift and is a highly available, durable, distributed, masterless, and eventually consistent software stack.

The Swift project, in particular, came out of Rackspace’s cloud files platform. The project was unique because it utilized a DevOps methodology, where the engineers and ops professionals worked together to create and operate it. This resulted in a very powerful storage system that is simple, yet easy to manage. Rackspace made Swift open source in 2010, and the leading contributors include SwiftStack, Rackspace, Red Hat, HP, Intel, IBM, and others.

In addition to sharing the mentioned generic object storage characteristics, OpenStack Swift has some unique additional functionality, as follows:

- **Open source**: Comes with no license fees, as mentioned previously.
- **Open standards**: Using HTTP REST APIs with SSL for optional encryption. The combination of open source and open standards eliminates any potential vendor lock-in.
- **Account container object structure**: OpenStack Swift incorporates rich naming and organization capacity, unlike a number of object storage systems that offer a primitive interface, where the user gets a key upon submitting an object. In these other systems, the burden of mapping names to keys and organizing them in a reasonable manner is left to the user. Swift, on the other hand, handles the organization of data along with multitenancy.
- **Global cluster capability**: This allows replication and distribution of data around the world. This functionality helps with disaster recovery, distribution of hot data, and so on.
Cloud Storage – Why Can’t I Be Like Google?

- **Storage policies**: This feature allows sets of data (stored in separate containers) to be optionally stored on different types of underlying storage using different durability models. For example, a valuable set of digital assets can be stored on high-quality hardware using triple replication, while less important assets can be stored on lower quality hardware with a lower level of replication. Hot data could be stored on SSDs.

- **Partial object retrieval**: For example, you want just a portion of a movie object or a TAR file.

- **Middleware architecture**: This allow users to add functionality. A great example of this is integrating with an authentication system.

- **Large object support**: Objects of any size can be stored.

- **Additional functionality**: This includes object versioning, causing objects to expire, rate limiting, temporary URL support, CNAME lookup, domain remap, account-to-account data copy, quota support, and static web mode. This list is constantly growing as a consequence of Swift being an open source project.

**Summary**

In this chapter, we saw why cloud storage is a new way of building storage systems that cuts the total cost of ownership significantly. It uses a technology called object storage. A high-quality and open source object storage software stack to consider is OpenStack Swift. OpenStack Swift uses a dramatically different architecture from traditional enterprise storage systems by using a distributed architecture on commodity servers. The next chapter explains this architecture in detail.
Where to buy this book

You can buy OpenStack Object Storage (Swift) Essentials from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.