**Amazon Simple Storage Service (Amazon S3)**

**and**

**Amazon Glacier Storage**

* Amazon S3 provides developers and IT teams with secure, durable, and highly-scalable cloud storage.
* Amazon S3 is easy-to-use *object storage* with a simple web service interface

Common use cases for Amazon S3 storage include:

* Backup and archive for on-premises or cloud data
* Content, media, and software storage and distribution
* Big data analytics
* Static website hosting
* Cloud-native mobile and Internet application hosting
* Disaster recovery

**Object Storage versus Traditional Block and File Storage**

* two kinds of storage dominate: *block storage* and *file storage*.
* Block storage operates at a lower level—the raw storage device level—and manages data as a set of numbered, fixed-size blocks.
* File storage operates at a higher level—the operating system level—and manages data as a named hierarchy of files and folders
* Amazon S3 storage is independent of a server and is accessed over the Internet.
* Each Amazon S3 object contains both data and metadata

**Amazon S3 Basics**

* A *bucket* is a container (web folder) for objects (files) stored in Amazon S3
* Buckets form the top-level namespace for Amazon S3, and bucket names are global.
* Bucket names must be unique across all AWS accounts.
* You can have up to 100 buckets per account by default.
* It is a best practice to use bucket names that contain your domain name and conform to the rules for DNS names.
* each Amazon S3 bucket is created in a specific region that you choose.
* data in an Amazon S3 bucket is stored in that region unless you explicitly copy it to another bucket located in a different region.
* *Objects* are the entities or files stored in Amazon S3 buckets.
* An object can store virtually any kind of data in any format.
* Objects can range in size from 0 bytes up to 5TB
* Single bucket can store an unlimited number of objects.
* Each object consists of data (the file itself) and *metadata* (data about the file).
* Amazon S3 doesn’t know or care what type of data you are storing, and the service doesn’t act differently for text data versus binary data.
* There are two types of metadata: system metadata and user metadata
* System metadata is created and used by Amazon S3 itself, and it includes things like the date last modified, object size, MD5 digest, and HTTP Content-Type.
* User metadata is optional, and it can only be specified at the time an object is created. You can use custom metadata to tag your data with attributes that are meaningful to you.
* Every object stored in an S3 bucket is identified by a unique identifier called a ***key***.
* A key can be up to 1024 bytes of Unicode UTF-8 characters
* Keys must be unique within a single bucket
* The combination of bucket, key, and optional version ID uniquely identifies an Amazon S3 object.

**Object URL**

* every Amazon S3 object can be addressed by a unique URL formed using the web services endpoint, the bucket name, and the object key
* For example, with the URL:

http://mybucket.s3.amazonaws.com/jack.doc

mybucket is the S3 bucket name, and jack.doc is the key or filename.

If another object is created, for instance:

http://mybucket.s3.amazonaws.com/fee/fi/fo/fum/jack.doc

then the bucket name is still mybucket, but now the key or filename is the string fee/fi/fo/fum/jack.doc.

* A key may contain delimiter characters like slashes or backslashes to help you name and logically organize your Amazon S3 objects, but to Amazon S3 it is simply a long key name in a flat namespace.
* There is no actual file and folder hierarchy
* **a bucket is a single flat namespace of keys with no structure.**

**Amazon S3 Operations**

* handful of common operations
  + Create/delete a bucket
  + Write an object
  + Read an object
  + Delete an object
  + List keys in a bucket

**REST Interface**

* The native interface for Amazon S3 is a REST (Representational State Transfer) API
* Always use HTTPS for Amazon S3 API requests to ensure that your requests and data are secure.
* Amazon S3 originally supported a SOAP (Simple Object Access Protocol) API in addition to the REST API, but you should use the REST API.

**Durability and Availability**

* Durability addresses the question, “Will my data still be there in the future?”
* Availability addresses the question, “Can I access my data right now?”
* S3 standard storage is designed for 99.999999999% durability and 99.99% availability of objects over a given year
* Amazon S3 achieves high durability by automatically storing data redundantly on multiple devices in multiple facilities within a region.
* It is designed to sustain the concurrent loss of data in two facilities without loss of user data.
* Even though Amazon S3 storage offers very high durability at the infrastructure level, it is still a best practice to protect against user-level accidental deletion or overwriting of data by using additional features such as versioning, cross-region replication, and MFA Delete.

**Data Consistency**

* S3 is an eventually consistent system.
* changes in your data may take some time to propagate to all locations.
* For PUTs to new objects, Amazon S3 provides read-after write consistency.
* for PUTs to existing objects (object overwrite to an existing key) and for object DELETEs, Amazon S3 provides eventual consistency.
* Eventual consistency means that if you PUT new data to an existing key, a subsequent GET might return the old data.

**Access Control**

* Amazon S3 is secure by default.
* when you create a bucket or object in Amazon S3, only you have access.
* Amazon S3 ACLs allow you to grant certain coarse-grained permissions: READ, WRITE, or FULL-CONTROL at the object or bucket level.
* S3 bucket policies are the recommended access control mechanism for Amazon S3 and provide much finer-grained control.
* S3 Bucket Policy Vs IAM
  + They are associated with the bucket resource instead of an IAM principal.
  + They include an explicit reference to the IAM principal in the policy. This principal can be associated with a different AWS account, so Amazon S3 bucket policies allow you to assign cross-account access to Amazon S3 resources.

**Static Website Hosting**

* A static website means that all of the pages of the website contain only static content and do not require server-side processing such as PHP, ASP.NET, or JSP.

**Amazon S3 Advanced Features**

**Prefixes and Delimiters**

* This feature lets you organize, browse, and retrieve the objects within a bucket hierarchically.
* This feature lets you logically organize new data and easily maintain the hierarchical folder-and-file structure of existing data uploaded or backed up from traditional file systems.
* Use delimiters and object prefixes to hierarchically organize the objects in your Amazon S3 buckets, but always remember that **Amazon S3 is not really a file system.**

**Storage Classes**

* **Amazon S3 Standard** offers high durability, high availability, low latency, and high performance object storage for general purpose use.
* **Amazon S3 Standard – Infrequent Access** (Standard-IA) offers the same durability, low latency, and high throughput as Amazon S3 Standard, but is designed for long-lived, less frequently accessed data.
* Standard-IA has a lower per GB-month storage cost than Standard, but the price model also includes a minimum object size (128KB), minimum duration (30 days), and per-GB retrieval costs, so it is best suited for infrequently accessed data that is stored for longer than 30 days.
* **Amazon S3 Reduced Redundancy Storage (RRS)** offers slightly lower durability (4 nines) than Standard or Standard-IA at a reduced cost.
* RRS is most appropriate for derived data that can be easily reproduced, such as image thumbnails.
* **Amazon Glacier** storage class offers secure, durable, and extremely low-cost cloud storage for data that does not require real-time access, such as archives and long-term backups.
* Amazon Glacier is optimized for infrequently accessed data where a retrieval time of several hours is suitable.
* To retrieve an Amazon Glacier object, you issue a restore command using one of the Amazon S3 APIs; three to five hours later, the Amazon Glacier object is copied to Amazon S3 RRS.
* Note that the restore simply creates a copy in Amazon S3 RRS; the original data object remains in Amazon Glacier until explicitly deleted.
* Also be aware that Amazon Glacier allows you to retrieve up to 5% of the Amazon S3 data stored in Amazon Glacier for free each month

**Object Lifecycle Management**

* Amazon S3 Object Lifecycle Management is roughly equivalent to automated storage tiering in traditional IT storage infrastructures
* Lifecycle configurations are attached to the bucket and can apply to all objects in the bucket or only to objects specified by a prefix.
* Object lifecycle management policies can be used to automatically move data between storage classes based on time.

**Encryption**

* It is strongly recommended that all sensitive data stored in Amazon S3 be encrypted, both in flight and at rest.
* **SSE-S3 (AWS-Managed Keys):** 
  + AWS handles the key management and key protection for Amazon S3.
  + Every object is encrypted with a unique key.
  + The actual object key itself is then further encrypted by a separate master key.
  + A new master key is issued at least monthly, with AWS rotating the keys.
* **SSE-KMS (AWS KMS Keys)**
  + This is a fully integrated solution where Amazon handles your key management and protection for Amazon S3, but where you manage the keys.
  + Using SSE-KMS, there are separate permissions for using the master key, which provide protection against unauthorized access to your objects stored in Amazon S3 and an additional layer of control.
  + AWS KMS also provides auditing, so you can see who used your key to access which object and when they tried to access this object.
* **SSE-C (Customer-Provided Keys)**
  + This is used when you want to maintain your own encryption keys but don’t want to manage or implement your own client-side encryption library.
* **Client-Side Encryption**
  + Client-side encryption refers to encrypting data on the client side of your application before sending it to Amazon S3

**Versioning**

* Amazon S3 versioning helps protects your data against accidental or malicious deletion by keeping multiple versions of each object in the bucket, identified by a **unique version ID**.
* If a user makes an accidental change or even maliciously deletes an object in your S3 bucket, you can restore the object to its original state simply by referencing the version ID in addition to the bucket and object key.
* Versioning is turned on at the **bucket level**.
* Once enabled, versioning **cannot be removed** from a bucket; it can only be **suspended**.

**MFA Delete**

* MFA Delete requires additional authentication in order to permanently delete an object version or change the versioning state of a bucket.
* Note that MFA Delete can only be enabled by the root account.

**Pre-Signed URLs**

* All Amazon S3 objects by default are private, meaning that only the owner has access.
* the object owner can optionally share objects with others by creating a *pre-signed URL*, using their own security credentials to grant time-limited permission to download the objects.
* When you create a pre-signed URL for your object, you must provide your security credentials and specify a bucket name, an object key, the HTTP method (GET to download the object), and an expiration date and time.
* The pre-signed URLs are **valid only for the specified duration**.

**Multipart Upload**

* This allows you to upload large objects as a set of parts, which generally gives better network utilization (through parallel transfers), the ability to pause and resume, and the ability to upload objects where the size is initially unknown.
* Three-step process: initiation, uploading the parts, and completion (or abort).
* Parts can be uploaded independently in arbitrary order, with retransmission if needed.
* you *should* use multipart upload for objects larger than 100 Mbytes
* you *must* use multipart upload for objects larger than 5GB.

**Range GETs**

* It is possible to download (GET) only a portion of an object in both Amazon S3 and Amazon Glacier by using something called a *Range GET*.
* This can be useful in dealing with large objects when you have poor connectivity or to download only a known portion of a large Amazon Glacier backup.

**Cross-Region Replication**

* Allows you to asynchronously replicate all new objects in the source bucket in one AWS region to a target bucket in another region.
* Any metadata and ACLs associated with the object are also part of the replication.
* To enable cross-region replication, versioning must be turned on for both source and destination buckets, and you must use an IAM policy to give Amazon S3 permission to replicate objects on your behalf.
* commonly used to reduce the latency required to access objects in Amazon S3
* Existing objects will not be replicated and must be copied to the new bucket via a separate command.

**Logging**

* Logging is off by default, but it can easily be enabled
* you must choose where the logs will be stored
* Logs include information such as:
  + Requestor account and IP address
  + Bucket name
  + Request time
  + Action (GET, PUT, LIST, and so forth)
  + Response status or error code

**Event Notifications**

* Event notifications enable you to run workflows, send alerts, or perform other actions in response to changes in your objects stored in Amazon S3.
* event notifications are set up at the bucket level

**Best Practices, Patterns, and Performance**

* To support higher request rates, it is best to ensure some level of random distribution of keys, for example by including a **hash as a prefix to key names.**
* If you are using Amazon S3 in a GET-intensive mode, such as a static website hosting, for best performance you should consider using an Amazon CloudFront distribution as a caching layer in front of your Amazon S3 bucket.

**Amazon Glacier**

* an extremely low-cost storage service
* Designed for **infrequently accessed** data where a retrieval time of **three to five hours** is acceptable.
* can store an **unlimited amount** of virtually any kind of data, in any format
* Common use cases for Amazon Glacier include **replacement of traditional tape** solutions for long-term backup and archive and **storage of data required for compliance purposes**.
* Extremely durable, storing data on multiple devices across multiple facilities in a region.
* Amazon Glacier is designed **for 99.999999999% durability** of objects over a given year.

**Archives**

* Data is stored in ***archives****.*
* An archive can contain up to **40TB** of data.
* You can have an **unlimited** number of archives.
* Each archive is assigned a **unique archive ID** at the time of creation.
* All archives are **automatically encrypted**
* Archives are **immutable**—after an archive is created, it **cannot be modified.**

**Vaults**

* *Vaults* are containers for archives
* Each AWS account can have up to 1,000 vaults.
* You can control access to your vaults and the actions allowed using IAM policies or vault access policies.

**Vaults Locks**

* You can easily deploy and enforce compliance controls for individual Amazon Glacier vaults with a ***vault lock***policy.
* You can specify controls such as **Write Once Read Many** (WORM) in a vault lock policy and **lock the policy from future edits**.
* Once locked, the policy **can no longer be changed**.

**Data Retrieval**

* You can retrieve up to 5% of your data stored in Amazon Glacier for free each month, calculated on a daily prorated basis.

**Amazon Glacier versus Amazon Simple Storage Service**

* Amazon Glacier supports 40TB archives versus 5TB objects in Amazon S3.
* Archives in Amazon Glacier are identified by system-generated archive IDs, while Amazon S3 lets you use “friendly” key names.
* Amazon Glacier archives are automatically encrypted, while encryption at rest is optional in Amazon S3.
* by using Amazon Glacier as an Amazon S3 storage class together with object lifecycle policies, you can use the Amazon S3 interface to get most of the benefits of Amazon Glacier without learning a new interface.