STORAGE
IN
AWS

Storage: Background

In this section we will discuss theoretical concepts related to the storage of data, the three major types of cloud storage, and their characteristics. A concise list of the topics to be discussed is given below:

- Memory, Storage, and how do they differ from one another.
- On-prem storage, its advantages and disadvantages.
- Cloud storage, and its three different types.
- Object Storage and its use cases
- Block Storage and its use cases
- File Storage and its use cases

If the reader believes themselves to already be familiar with the listed concepts, then they can feel free to skip this section and move on to the rest of the chapter.

Memory vs Storage

As technology has become more ubiquitous in modern society, it must come as little surprise that technical terms have also found their way into the modern vocabulary. **Memory** and **Storage** are two such terms, and are often used interchangeably by people to refer to a computer's capacity to hold data. However, there do exist contrasts between the two which I find important to delve into.

Memory, sometimes also referred to as RAM, is used to refer to the hardware component(s) responsible for holding data temporarily, data typically related to the instructions that are currently being executed or operated by the computer, a task typically done by the CPU. It is quite fast, and enables quick data processing but **volatile** in nature, meaning that all data within it is lost when the device is turned off.

Storage on the other hand, is a broad term encompassing all hardware components and devices used to hold data for longer periods of time, data typically related to the OS, installed applications and personal files like documents, photos and videos. Though slower than memory, storage is non-volatile in nature and can typically handle much larger quantities of data than memory can.

Basically, while memory serves as the computer's short-term workspace, enabling rapid access to data for active tasks, storage acts as the long-term repository, preserving data permanently.

On-prem Storage and RAID

Now, there are two major ways in which large organizations and enterprises acquire and organize their storage infrastructure: **On-premises**, **Cloud** and **Hybrid**. Since On-prem is the oldest kid on the block, it would be helpful for us to learn about it before transitioning towards Cloud and Hybrid storage.

On-Premises storage environments have all their storage systems, devices and related infrastructure physically located within or near their facilities, providing direct oversight and security control over them, and minimal redundancy to boot. It has remained extremely popular among legacy organizations and organizations with high data security needs, compliance requirements, or specialized performance criteria.

Also, On-prem environments often utilize **RAID (Redundant Array of Independent Disks)** setups for the organization and management of their hardware devices, but since RAID and its multitude configuration types are a complex topic, beyond the scope of a book dedicated to Cloud computing, we will not be discussing it here. Though the topic might be worth looking into for the interested.

Cloud Storage

Cloud storage environments involve the renting of storage systems, devices and related infrastructure, usually from a cloud provider like AWS or Microsoft Azure. These rented storage systems can then be connected to and taken advantage of either through the internet or though dedicated private channels.

Unlike with traditional storage, cloud storage can scale itself according to the capacity needs, and enterprises also only have to pay for the storage capacity that they use and not a dime more, allowing them to avoid the problems of under or over-provisioning of hardware resources associated with traditional storage. These factors, combined with the variety of cloud storage types and the convenience of not having to manage any hardware on their own, make cloud storage an appealing choice for businesses with fluctuating storage needs or those aiming to avoid the high upfront costs associated with on-premises infrastructure.

Note however that it is not necessary for organizations to chose one or the other, in fact many organizations often adopt environments that utilize both on-prem and cloud storage solutions in varying capacities, usually keeping frequently accessed or low-latency access data near their facilities; An approach termed as hybrid storage.

Now there are three primary types of cloud storage: **Object Storage**, **Block Storage**, and **File Storage**. All three types, their characteristics and use cases are discussed in detail below.

Object Storage

Object Storage (as the name may suggest) organizes data by breaking it into pieces called objects, with each object containing the data itself, metadata related to it, and a unique ID. It is highly scalable and well-suited for unstructured data like images, videos, and backups.

Particularly designed for scenarios where quick access to large volumes of data is required, it is ideal for content repositories, backup, and archiving. All the objects are stored in a non-hierarchical manner in a single “folder”, often referred to as the object bucket. Additionally, the metadata associated with the objects allow for the objects to be searched through using say, SQL Queries. It should therefore, not come as a surprise to many that Object Storage is often used to power big data environments such as data lakes and data warehouses.

Block Storage

Block storage is a type of cloud storage that resembles a hard drive in traditional computers. It places data into blocks and then stores those blocks as separate pieces with each block having a unique identifier. A hierarchical storage method, it follows the nested file and folder structure similar to desktop file systems, placing the blocks of data wherever most prudent or efficient capacity wise.

File Storage

File storage is the traditional file systems we run on our desktop machines, just now being run on the cloud i.e. now being run in someone else’s data center. It is mostly used to provide shared file systems for a network of devices running the same Operating System. Examples of this include NTFS-based volumes for Windows systems and NFS-based volumes for Linux/UNIX systems.

They are often adopted due to their ability to be mounted and directly accessed by numerous computers and agents at the same time.

S3

Amazon Simple Storage Service (also called Amazon S3) is a scalable, high-speed, web-based cloud storage service designed to store and retrieve any amount of data at any time. It uses an object storage architecture, where data is managed as objects rather than file hierarchies. The industry standard object storage service, S3 streamlines the process of storing files on the cloud, while providing easy methods of accessing and managing it.

Each object in S3 is stored in a bucket and consists of data, metadata, and a unique identifier. S3's key features include high (99.999999%) durability, availability, security, and performance. It's widely used for backup and recovery, content distribution, static website hosting, etc, making it a fundamental service for many cloud-based applications.

S3 has different storage classes or categories to choose from based on the performance, availability called S3 Standard, S3 Infrequent Access, S3 Intelligent Tiering, etc each optimized for different use cases based on access frequency and cost requirements. These will be discussed in future sections.

It is also the **cheapest** of the three major forms of storage in AWS, the others being: Block Storage using EBS (Elastic Block Store), which is used in databases and File Storage using EFS. Both types of cloud storage have already been discussed in the background, though their representative services will be discussed in later sections. Usually, when talking about the cost per Gigabyte(GB) for the three major storage services in AWS, they can be organized as follows:

EBS ⇒ Cheap   **EFS** ⇒ Cheaper   **S3** ⇒ Cheapest

EBS and Fast Snapshot Restore

Amazon Elastic Block Store (EBS) is a high-performance block storage service designed for use with EC2 instances. It provides persistent block-level storage volumes that can be attached to EC2 instances, allowing you to store data and access it from any instance. EBS volumes are durable and highly available, offering features such as snapshots for data backup and replication for data protection.

Additionally, EBS Volumes can be resized without any downtime, allowing us to increase/decrease storage capacity or adjust performance characteristics as needed on the fly. However, do note that all EBS volumes are tied to a single availability zone (AZ), so point-in-time copies of the EBS volumes (called Snapshots) must be created and attached/restored to a different AZ or Region. This is usually done to enhance data durability and recovery options.

It is also not uncommon to encrypt the data in EBS Volumes at rest, which can be done using encryption keys provided by the AWS Key Management Service (KMS), granting the EBS volumes with an additional layer of security. KMS and related concepts will be discussed in detail in a later chapter.

Note:

EBS also has a feature called **Fast Snapshot Restore** which can be used to clone data from an EC2 instance store into new EBS volumes with minimal time, reducing the amount of required to recover or create new volumes.

Types of EBS Volumes

EBS offers several different volume types, each tailored to meet the performance and cost requirements of various use cases. The different EBS Volume types are as follows:

General Purpose SSD (sub-types: gp3, gp2): A balanced storage option suitable for a wide range of workloads. gp3 allows independent scaling of IOPS and throughput, while gp2 provides baseline performance with burst capabilities. Ideal for boot volumes, medium-sized databases, and general-purpose applications.

Provisioned IOPS (sub-types: io1, io2) is a storage option offered by AWS for applications requiring high-performance and consistent input/output operations per second (IOPS). It is designed for use with Amazon EBS (Elastic Block Store) and is ideal for I/O-intensive applications such as large databases, transactional systems, and other performance-sensitive workloads especially when compared to General Purpose SSD.

Throughput Optimized HDD (sub-types: st1): Low-cost, high-throughput storage optimized for large, sequential workloads. Ideal for big data processing, data warehousing, and log processing, where throughput is more important than IOPS.

Cold HDD (sub-types: sc1): The most cost-effective storage option, designed for infrequently accessed data. Best suited for archival storage, long-term backups, and datasets that require occasional access at a low cost.

EFS and Multiple Access

One of three major storage services offered by AWS, Amazon Elastic File System (EFS) provides us with a fully managed, serverless, scalable, POSIX-compatible file storage service on the cloud. As mentioned in the background, file storage systems follows the same structure and is operated in the same manner as many traditional desktop file systems.

EFS is highly available and durable, making it ideal for workloads that require shared access to file storage across multiple instances. The service is considered moderately priced, generally costing more than S3 but less than EBS (Elastic Block Store). It also has a Infrequent Access (EFS-IA) feature which functions similar to S3 Infrequent Access through its lifecycle policies with regards to IA are much more limited than its S3 counterpart.

Elastic File Systems can be easily mounted onto EC2 instances, AWS Containers, Lambda functions, or on-premise servers and are often done so in order to augment their default storage capabilities.

Finally, EFS offers two different performance and throughput modes for potential adopters:

Performance Modes:

- **General-Purpose** (Most widely used, particularly efficient for latency-sensitive use cases)
- **Max I/O** (For highly parallel applications which necessitate high throughput)

Differences between the two performance modes are discussed below:

Feature/Characteristic	**General Purpose Mode**	**Max I/O Mode**
Use Case	Ideal for latency-sensitive applications (e.g., web serving, content management)	Ideal for applications with highly parallelized workloads (e.g., big data, media processing)
Latency	Low latency	Higher latency compared to General Purpose
Throughput	Up to 7 GB/s with Bursting Throughput	Higher throughput (up to 12 GB/s) with Bursting Throughput
IOPS	Up to 500,000 IOPS	More than 500,000 IOPS
Parallelism	Suitable for moderate to high parallelism	Designed for very high parallelism
Performance Consistency	Consistent performance with low latency	Scales with load; latency can increase as load increases
Cost	Generally lower cost	May incur higher cost due to increased resource usage
Best For	File-based workloads that require consistent low latency	Applications that require high aggregate throughput and can tolerate higher latency

Throughput Modes:

- **Bursting Throughput (**Automatically scales the throughput based on file system size)
- **Provisioned Throughput (**Allows us to allocate throughput without being dependent on the file system size)

Differences between the two throughput modes are as follows:

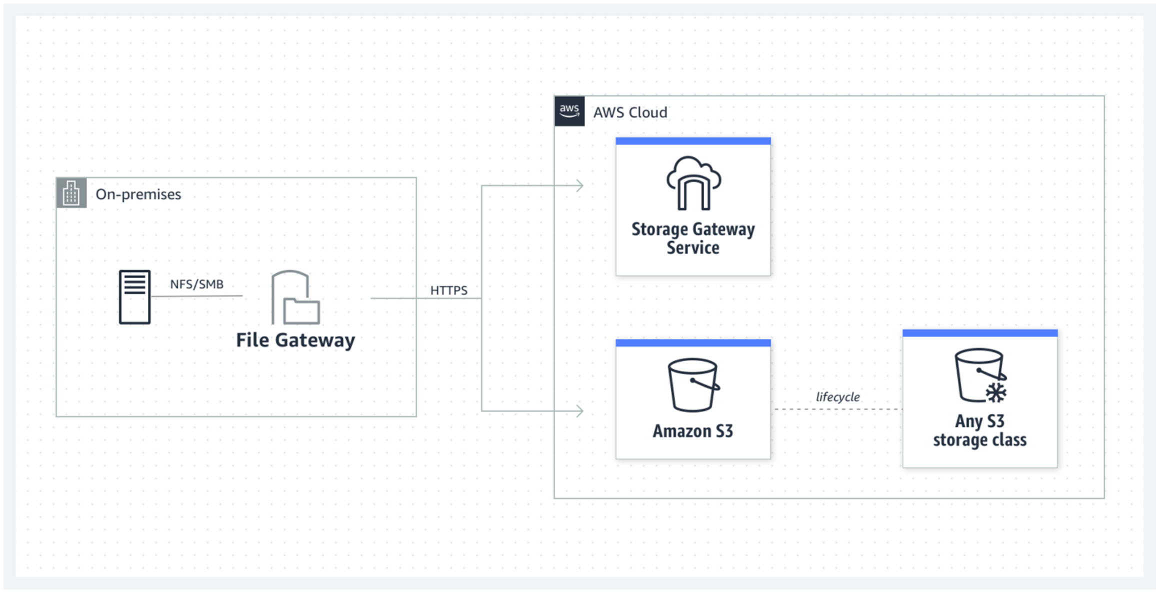
Feature/Characteristic	**Bursting Throughput Mode**	**Provisioned Throughput Mode**
Use Case	Suitable for most workloads with variable and unpredictable throughput needs	Ideal for applications requiring consistent, high levels of throughput, regardless of file system size
Throughput Allocation	Throughput scales with file system size, allowing bursts based on the amount of stored data	Throughput is provisioned independently of the file system size, ensuring a fixed throughput level
Throughput Range	Scales automatically with storage, up to 100 MB/s per TB (can burst up to 100 MB/s for file systems under 1 TB)	Customizable from 1 MB/s to 1024 MB/s, regardless of file system size
Bursting Capability	Supports bursting when workload needs exceed baseline throughput (depending on size and burst credits)	No bursting; throughput remains constant as provisioned
Performance Consistency	Variable throughput, can burst when needed but may be lower during sustained periods of heavy use	Consistent throughput, ideal for sustained high-performance needs
Cost	Cost-effective for variable workloads, as throughput is tied to file system size	May incur higher costs due to the fixed provisioned throughput, independent of storage usage
Best For	General-purpose workloads, including web serving, backups, and dev/test environments	High-performance workloads, like media processing, big data analytics, and machine learning

S3 File Gateway

Storage gateways are the services provided by AWS which enable hybrid cloud storage solutions, leveraging the best of both worlds, i.e. grant us the benefits of low-latency access associated with on-premise storage systems while retaining the virtually unlimited capacity of cloud storage storage solutions.

This is achieved by creating a cache of the cloud storage and storing it locally on premises. S3 File Gateway is one such storage gateway service, and provides a way to seamless integrate on-premise applications with Amazon S3, using a local cache to store data that would benefit the most from low-latency access while keeping the rest in S3. Asynchronous in nature, the service presents S3 buckets as NFS or SMB file shares, allowing applications to access objects stored in S3 using familiar file protocols.

A basic file gateway architecture is shown in the following diagram:



AWS DataSync vs Storage Gateway

AWS DataSync is a service provided by Amazon Web Services (AWS) that simplifies, accelerates, and automates the process of transferring data between on-premises storage systems and AWS services.

After the initial migration, AWS DataSync can be configured to maintain the consistency of data between on-premises storage and AWS storage services. In such cases, data synchronization between the two happens in either real or near real-time.

While both AWS DataSync and AWS Storage Gateway provide solutions for integrating on-premises storage environments with AWS cloud storage services, they serve different purposes and have distinct functionalities, including how they handle data synchronization. This is illustrated in the bulleted sub-sections below:

AWS DataSync

- **Purpose**: Primarily designed for high-speed, one-time data transfers and ongoing data synchronization between on-premises storage systems and AWS storage services.
- **Data Transfer Mechanism**: Utilizes optimized data transfer protocols to ensure fast and efficient transfer of large volumes of data.
- **Use Cases**: Ideal for scenarios requiring frequent updates or synchronization of data between on-premises and AWS environments, such as continuous data backup, real-time data processing, or maintaining consistent copies of data for disaster recovery purposes.

AWS Storage Gateway

- **Purpose**: Offers a hybrid storage solution that enables on-premises applications to seamlessly access data stored in AWS cloud storage services with lower latency than it otherwise could.
- **Data Synchronization**: While AWS Storage Gateway supports data migration and synchronization capabilities, its primary focus is on providing on-demand access to data stored in AWS, rather than continuous synchronization.
- **Storage Protocols**: Supports various storage protocols, including NFS, SMB, and iSCSI, allowing existing on-premises applications to interact with AWS storage as if it were local storage.
- **Use Cases**: Commonly used for extending on-premises storage capacities, disaster recovery, data archiving, and enabling cloud-based applications to access on-premises data.

TL;DR

AWS Data Sync: Used for data migration and synchronization.

AWS Storage Gateways: Used for low-latency access to data and adding capacity to existing on-prem storage.

Data transfer times

Data transfer to AWS involves moving data from your on-premises data center or other cloud environments into AWS. This process can be critical for cloud migration, data backup, disaster recovery, and large-scale data analytics. AWS offers multiple methods for data transfer, each tailored to different volumes of data, speed requirements, and cost considerations.

(Note: The following task might be hard for some in senior positions to perform)

Consider yourself as an employee has been tasked to transfer 200 TB worth of data to the Cloud with a 100 Mbps Internet connection, while taking into consideration the associated transfer costs and time taken. Now, AWS provides us with a myriad of different methods to tackle such a situation, and though a full explanation of all methods is beyond the scope of this book, some of those solutions are briefly discussed in the subsections below.

Over the Internet / Site-to-Site VPN

A Site-to-Site VPN creates a secure, encrypted connection (i.e. a tunnel) over the public internet between your on-premises network and AWS. Using a Site-to-Site VPN allows you to securely transfer data in an immediate and hassle-free manner using an internet connection already available to you. However, this method can be time-consuming, especially when transferring large volumes of data, such as 200 TB, at 100 Mbps would take approximately 185 days.

TLDR;

- Immediate to setup.
- Will take 200(TB) * 1000(GB) * 1000(MB) * 8(Mb) / 100 Mbps = 16,000,000s ⇒ ~185 days.

Over Direct Connect (1 Gbps)

AWS Direct Connect provides a dedicated, high-speed network connection between your data center and AWS, bypassing the public internet for a more reliable and consistent transfer experience. Setting up Direct Connect can take over a month due to the physical installation and configuration required, but once established, it reduces the transfer time for 200 TB of data to around 18.5 days. This method is ideal for organizations needing fast, reliable transfers and ongoing large-scale data replication but comes with higher setup costs and complexity.

TLDR;

- Time taken for initial setup is quite long (Over a month).
- Will take 200(TB) * 1000(GB) * 8(Gb) / 1 Gbps = 1,600,000s ⇒ ~18.5 days.

Over Snowball

AWS Snowball is a physical data transport solution where AWS ships you storage devices to load your data, which are then returned to AWS for uploading. This method can handle large data volumes quickly, with 200 TB taking about a week using 2 to 3 Snowballs in parallel. Snowball is particularly useful when internet-based transfers are impractical due to bandwidth limitations. Additionally, it can be combined with AWS Database Migration Service (DMS) for seamless database migrations. It’s a secure and efficient method for one-time, large-scale data migrations.

TLDR;

- Can perform tasks by utilizing multiple Snowballs in parallel.
- Time taken to complete the end-to-end transfer is about 1 week.
- Can be combined with DMS.

Snowmobile Capacity

[**Note:** This service is now retired, but this section has been retained from the first edition for the purposes of historical context.]

One of the more unhinged services made available to us, AWS Snowmobile is a secure data transport service designed for transferring extremely large amounts of data (up to 100 petabytes) to AWS. The largest of the Snow family of storage device and services, it is ideal for large-scale migrations such as data center shutdowns, archives, and content libraries, offering a fast and cost-effective way to transfer massive datasets to the cloud securely.

Designed with the purpose of transferring exabyte scale data from on-premise servers to the cloud, AWS Snowmobile is a ruggedized, tamper resistant shipping truck that is 45 feet long, 9.6 feet high, and 8 feet wide.

Fitted with 100 Petabytes worth of storage capacity, networking equipment, etc, the AWS Snowmobile is essentially a data center on wheels which can attach itself to your local network, absorb the necessary data and then deliver it physically to the desired AWS data center.

Snowball Edge vs Snowball

AWS Snowball is a data migration service that uses secure, ruggedized devices to transfer large amounts of data into and out of AWS. Created for the purpose of simplifying and accelerating data migration, and reducing the challenges posed by limited network bandwidth, and is particularly useful for quickly migrating Small to Medium sized enterprises to the cloud for the first time.

AWS Snowball Edge is an offering which extends the functionality of the basic AWS Snowball by adding compute capabilities. The AWS Snowball Edge machines come with built-in storage and compute power, enabling local data processing and analysis before transferring data to AWS with support for running EC2 instances and AWS Lambda functions, enabling complex workflows and custom applications to run on the device. These additional capacities transform the AWS Snowball Edge from a simple storage device to a versatile edge computing machine.

TLDR;

AWS Snowball Edge can run EC2 Instances and lambda functions, allowing for Edge Computing use cases which the normal Snowball cannot. You do pay a little extra for this privilege, however.

A comprehensive, tabular comparison of Snowball and Snowball Edge is presented in the next page. Feel free to utilize the chart when deciding which of the two offerings to utilize.

Feature	AWS Snowball	AWS Snowball Edge
Primary Purpose	Large-scale data migration	Data migration plus edge computing capabilities
Storage Capacity	50 to 80 TB	80TB HDD to 210TB NVMe
Compute Capabilities	No	Yes
Types of Devices	Single configuration	Snowball Edge Storage Optimized, Snowball Edge Compute Optimized, Snowball Edge with Tape Gateway
Local Data Processing	No	Yes
Encryption	256-bit encryption, AWS KMS managed	Same as Snowball
Physical Security	Tamper-resistant, ruggedized, GPS tracking	Same as Snowball
Networking	Standard network interfaces for data transfer	High-speed network interfaces for data transfer and local compute
Supported AWS Services	S3	S3, EC2, Lambda, Tape Gateway
Typical Use Cases	Data migration to AWS	Edge computing, data migration, remote analytics, and processing
Pricing	Relatively low per GB	Relatively higher per GB
Management	Managed via AWS Snow Family Management Console	Same as Snowball
Target Environment	Data centers, enterprises	24/7 Workloads, Remote locations

FSX for Lustre

Amazon FSx is a fully managed service that allows you to launch and run popular file systems in the cloud. It supports various file systems, each tailored for specific workloads, offering the flexibility to choose the right file system for your needs. FSx handles all the complexities of file system deployment, management, scaling, and optimization, allowing you to focus on your applications.

One of the supported file systems by Amazon FSx is Lustre, which is a an open-source, high-performance parallel file system designed for large-scale, data-intensive workloads. It is particularly well-known for its ability to handle massive amounts of data and provide high throughput and low-latency access to that data.

Also worth mentioning is that Lustre is a Linux-based file system, and hence is natively compatible with Linux. In fact, AWS even provides its own Lustre client for Linux instances on AWS, allowing them to more conveniently connect to and use FSx for Lustre.

This makes it especially useful when designing architecture for applications which have intense performance requirements, such as High-Performance Computing (HPC), which necessitate highly performant file systems. Therefore, FSx for Lustre is often the default choice when designing architectures in Linux and/or designing architectures for performance-intensive applications.

FSX for Windows

Another service belonging to the Amazon FSx family, Amazon FSx for Windows is a fully managed file storage service designed to seamlessly run Windows file systems in the cloud. Built on Windows Server, FSx for Windows provides highly reliable and scalable file storage that is accessible over the Server Message Block (SMB) protocol.

This service eliminates the complexity of managing and maintaining traditional file servers for the NFTS, FAT32 and FAT16 file systems used by the Windows Operating System alongside many other convenient features like automatic backups, continuous monitoring, and integration with other AWS services.

FSx for Windows is particularly suited for Windows-based applications that require shared file storage, such as Microsoft SQL Server, SAP, and user home directories. By leveraging this service, organizations can run their Windows file systems in a cloud environment with ease, benefitting from the robust infrastructure that AWS provides.

Additionally, Amazon FSx for Windows includes a File Gateway that operates similarly to the S3 File Gateway. This feature further enhances the service's capability by providing a seamless bridge between on-premises environments and the cloud, allowing for efficient data transfer and storage management.

Compliance vs Governance Object Lock

Object Lock is a feature of Amazon S3 that helps enforce compliance by preventing deletion or modification of objects for a specified retention period. It ensures that data remains immutable and protected against accidental or malicious deletion, supporting regulatory requirements and data governance policies.

The two types of Object Lock in Amazon S3 are:

1. **Retention Periods**: This mode sets a fixed retention period during which objects cannot be deleted or altered. It ensures data immutability for compliance with regulations like SEC Rule 17a-4(f) and the General Data Protection Regulation (GDPR).
2. **Legal Hold**: This mode allows you to place legal holds on objects, which prevents them from being deleted by any user until the hold is removed. It is typically used for preserving data relevant to legal or investigative proceedings.

Also, Object Lock has two modes namely:

1. **Compliance Mode**: In this mode, the data is immutable and cannot be altered or deleted by any user, including governing authorities. This ensures the highest level of data protection, often required for regulatory compliance.
2. **Governance Mode**: This mode allows certain users with special permissions, such as governing authorities, to modify or delete the data if necessary. It provides a balance between data protection and administrative flexibility.

Legal Hold vs Retention Periods

As mentioned before, the two types of Object Lock made available by Amazon S3 are Legal Holds and Retention Periods. Though they both serve the same purpose of protecting an object from being overwritten or removed, they differ from each other in several different key aspects which are discussed below:

Legal Hold allows for the indefinite protection of an object version without a predefined time frame. Once applied, it remains in effect until explicitly removed by a user with the **s3:PutObjectLegalHold** permission.

Retention Period, on the other hand, is a specific duration during which the object version is protected. After this period expires, the object can be deleted or modified according to the permissions set.

Feature	**Legal Hold**	**Retention Period**
Duration	Indefinite, until removed	Fixed, defined at the time of setting
Modification	Can be placed or removed by users with the appropriate permissions	Cannot be altered until the period expires
Purpose	Ensures indefinite protection until no longer needed	Ensures protection for a set time period
WORM Enforcement	Yes	Yes

Note: Both Object Lock methods are unified by the Write Once, Read Many (WORM) model, which ensures that data cannot be altered once written.

S3 Bucket Policy

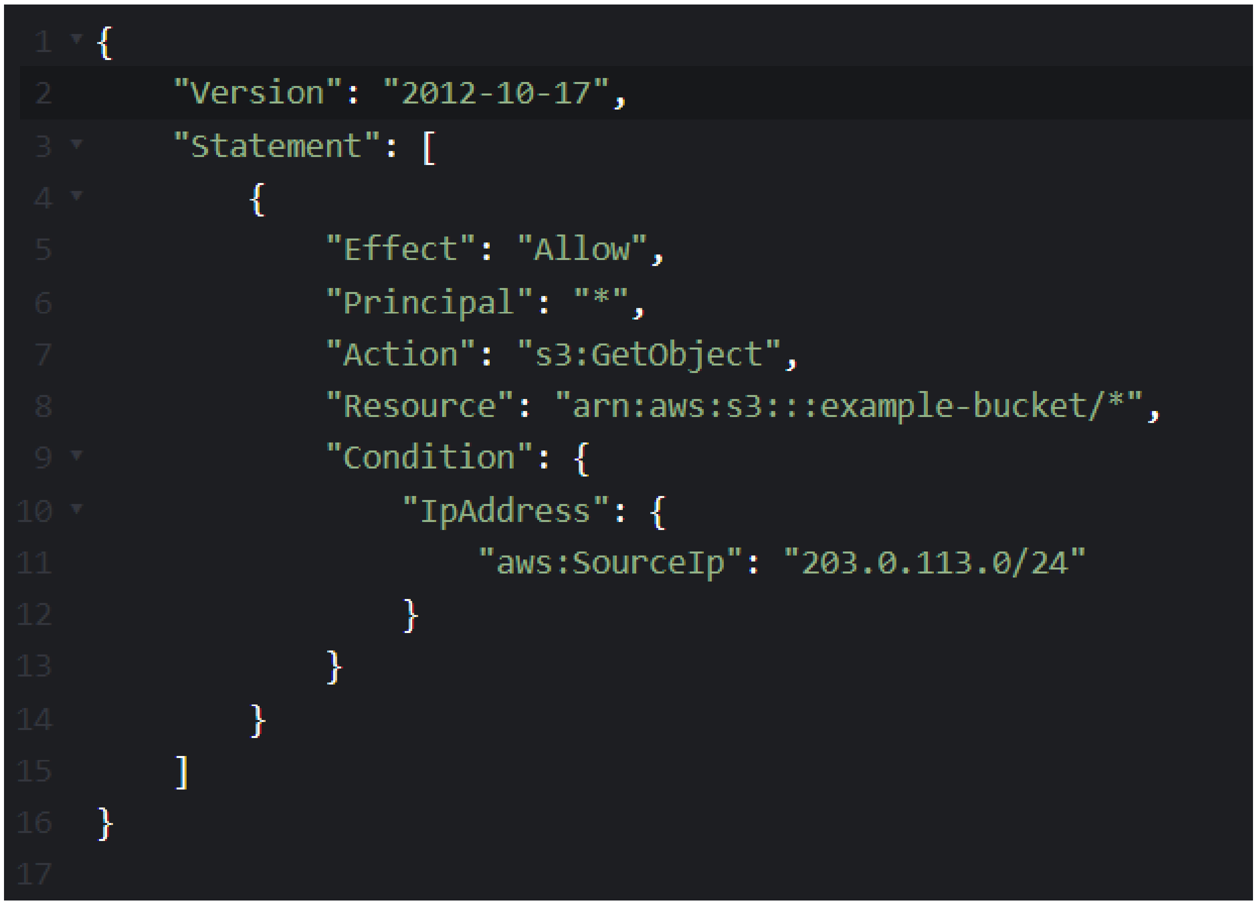
AWS Policies are JSON-based documents that define permissions for what actions can be performed on AWS resources and under what conditions. They are a key element of AWS security model, allowing fine-grained control over access to various services and resources. Each policy consists of statements that specify which principals (IAM users or AWS services) can perform certain actions (e.g., `s3:GetObject`) on specific resources (e.g., an S3 bucket) under particular conditions.

There are two subcategories of AWS policies in particular, which are crucial for managing access control in AWS, though they both serve different purposes and scopes:

1. **S3 Bucket Policies:** These are attached directly to S3 buckets and define permissions for all objects within the bucket. Bucket policies are written in JSON and can specify access permissions based on IP address, AWS account, or other conditions.
2. **IAM Policies:** These are attached to IAM identities (users, groups, roles) and specify what actions they can perform on AWS resources and are also written in JSON but are more granular, allowing fine-tuned control over specific AWS services and resources. They define permissions based on actions, resources, and conditions.

Basically, S3 bucket policies control access to S3 buckets and their contents based on conditions like IP address or AWS account, while IAM policies control access related to actions, resources, and identities within an AWS account. A detailed example of a S3 bucket policy is presented in the next page, with IAM policies receiving a similar treatment in later sections.

S3 Bucket Policy Example and Breakdown:



```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::example-bucket/*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": "203.0.113.0/24"
                }
            }
        }
    ]
}
```

Note: If you are not someone from a technical background, do not be scared by the curly braces, the JSON code you see above is extremely simple to understand and write.

Short for JavaScript Object Notation, JSON code is structured as a series of keys, (some of which may be nested within one another) with every key having a unique value associated with it. Let us examine the code one line at a time, to better understand this structure and the purpose of the code.

"Version": "2012-10-17"

This line specifies that the 2012-10-17 version of the AWS Policy Language is being used.

"Statement": [ ]

The core of our policy, this line defines the statement key, which acts as a container for any number of different individual permissions (often times also referred to as “Statements”). Consider each individual statement to be like a rule that dictates who can do what with your resources on AWS.

"Effect": "Allow"

The “Effect” field determines whether an action is allowed or denied. In our case, the specified value is "Allow", meaning that the policy grants permission to perform the mentioned action. If the specified value were to be "Deny" instead, then it would explicitly block access to the desired action.

"Principal": “*”

The “Principal” element specifies the users, accounts and/or services which the policy applies to. In our case, the asterisk (*), also known as the wildcard character, is set as the element value. This makes the policy applicable to all users, accounts and/or services (basically everyone). You could replace this with a specific AWS account or user if you wanted to limit access to said agent.

"Action": “s3:GetObject”

Used to specify the operations or API calls that are to be allowed or denied. Each AWS service has its own associated set of actions that describe the tasks that can be performed with said service. For example, the above statement deals with whether the policy allows the associated user to retrieve objects from the S3 Object Storage service.

"Resource": “arn:aws:s3:::example-bucket/*”

A self-explanatory field, the “Resource” element specifies the AWS resources (like an S3 bucket, EC2 instance, lambda function, etc) that the policy applies to. It allows us to ensure that the policy only applies to the objects we deem necessary, with the AWS resources being identified by their ARN (Amazon Resource Name).

For example, in our case, the policy applies to all objects within the S3 Bucket named `example-bucket` as specified by the ARN (*arn:aws:s3:::example-bucket/**)

"Condition": { }

An optional field, the “Condition” element lets us specify the circumstances in which the policy would be effective, allowing us to define additional constraints that must be met for the policy to apply. Mentioned conditions are written as a series of key-value pairs, for example in the mentioned snippet, the “IpAddress” condition key is used to limit the effects of the policy to requests coming from the very specific IP range of `192.0.2.0/24`, adding an extra layer of security to the S3 Buckets.

TLDR;

In our example policy, the following fields are used to:

Statement: Define the individual permissions.

> **Effect**: Specify that the purpose of the policy is to grant permission/"Allow" to perform specific actions.
>
> **Principal**: Make the policy apply universally to all users/accounts, through use of the wildcard character `"*"`.
>
> **Action**: `s3:GetObject` allows users to retrieve objects from the S3 bucket.
>
> **Resource**: Specifies the S3 bucket (`example-bucket`) and all objects inside (`/*`).
>
> > **Condition**: Limits access based on IP address. In this case, only users coming from the IP range `203.0.113.0/24` are allowed access.

Therefore, the purpose of the policy is to grant read-only access to all objects in the `example-bucket` , provided the users are accessing it from a specific IP range.

S3 Storage Classes and Lifecycle Policies

As mentioned before, S3 has a range of storage classes designed for different balances of cost, durability, availability and performance with prospective users being able to choose one based on their intended use cases. There are six major different storage classes, all of which are discussed below:

1. **S3 Standard**: For frequently accessed data. It offers high availability, low latency, and durability of 99.999999999% (11 nines). This storage class is best suited for general-purpose storage.

2. **S3 Standard-IA (Infrequent Access)**: For data that is accessed less frequently but needs fast retrieval when required. Lower storage cost but retrieval fees apply.

3. **S3 Intelligent-Tiering**: Automatically moves data between S3 Standard and S3 Infrequent Access as it sees fit using proprietary algorithms to determine which of the two tiers an object may be kept in. So objects that have a history of being more scarcely accessed would be automatically moved from S3 Standard to S3 Infrequent Access, without any extra retrieval charges. Most useful in cases where some files are accessed frequently while other files are rarely accessed in an unpredictable pattern.

4. **S3 One Zone-IA**: Similar to Standard-IA but data is stored in a single Availability Zone, making it cheaper albeit less available. This storage class is most suitable for non-critical, reproducible data.

5. **S3 Glacier**: For archival data where retrieval times of minutes to hours are acceptable. It's extremely cost-effective for long-term storage.

6. **S3 Glacier Deep Archive**: The lowest-cost storage class, designed for data that is accessed quite rarely and has the largest retrieval times of up to 12 hours.

Lifecycle Policies

With so many different storage classes available to us, there might be times when we would like to sort the objects into different buckets based on the time that has passed since the object's creation for the purposes of cost optimization, data retention, etc. Lifecycle policies provide us with this luxury.

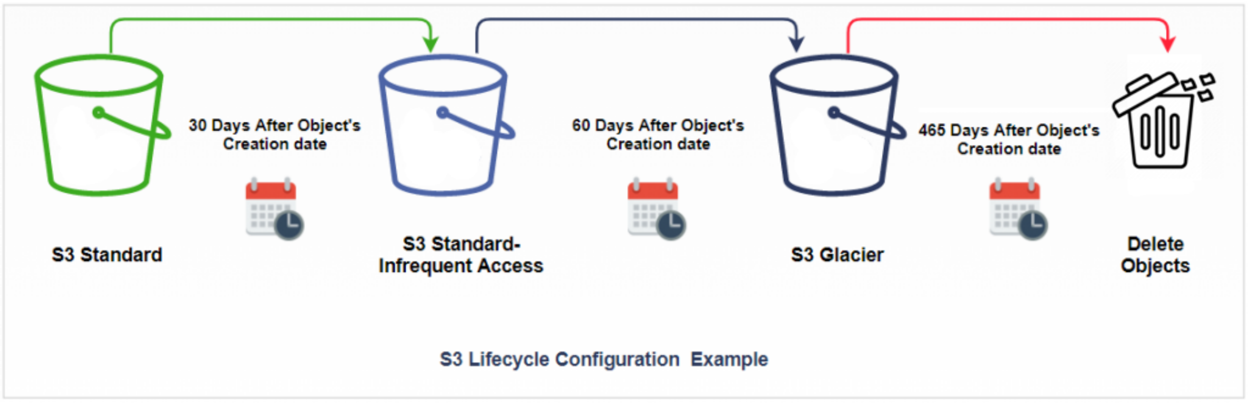
Lifecycle policies in AWS are a set of rules that allow us to manage resources (such as objects in S3 Buckets) and automate certain actions that are to be performed on the data, such as transitioning data to different storage classes or deleting them after a certain period of time.

Let us consider for example, that you are the administrator for a company that uses S3 to store all of its cash transaction logs. Said transaction logs are frequently accessed for the first month after their creation, and are sporadically accessed for the month after that. Though the transaction logs are rarely accessed after that, government regulations mandate transaction logs to be stored up till 465 days after the fact.

Now, in order to meet the restrictions laid out above, we can configure a lifecycle policy for the S3 Bucket where transaction logs are:

1. By default created and stored in S3 Standard
2. Moved from S3 Standard to S3 Infrequent Access 30 days after the object creation date

3. Moved from S3 Infrequent Access to S3 Glacier 60 days after the object creation date.
4. Retained in S3 Glacier until 465 days after the object creation date, after which it is promptly deleted.


S3 Standard
30 Days After Object's Creation date
S3 Standard-Infrequent Access
60 Days After Object's Creation date
S3 Glacier
465 Days After Object's Creation date
Delete Objects
S3 Lifecycle Configuration Example

Thus, through the use of S3 Lifecycle policies we were able to easily automate the process of managing our S3 object data based on the time that has passed since the object creation date in a hassle-free manner.

S3 Glacier

Of the different storage tiers introduced in the previous pages, there is one in particular that perhaps warrants a closer inspection namely, the Amazon S3 Glacier storage classes.

The Amazon S3 Glacier storage classes are designed to be the most cost-effective storage solutions in the cloud providing a good compromise between performance, price and flexibility. Best suited for the purposes of data-archiving, S3 Glacier has also become a sort-of industry standard for organizations that wish to retain large amounts of rarely accessed, legacy data for the long term mostly due to its relative cheapness, virtually unlimited scalability and 99.999999999% (11 nines) data durability guarantee.

There are three major S3 Glacier storage classes, each with its own unique set of strengths and drawbacks that must be taken into consideration when choosing between them. Said major S3 Glacier storage classes are as follows:

1. **Amazon S3 Glacier Instant Retrieval:** Built specifically for cases where the stored data may need to be accessed within a more immediate timeframe, S3 Glacier Instant Retrieval provides the lowest-cost storage option with millisecond-level retrieval times, making it suitable for workloads that require fast, frequent access to their stored data (examples include: medical archives, genomics data, etc) offering a good middle ground for users who need a combination of cost-effectiveness and fast access.

2. **Amazon S3 Glacier Flexible Retrieval:** Sometimes simply referred to as S3 Glacier, the S3 Glacier Flexible Retrieval class is designed for situations where

occasional, large-scale data retrievals are needed at a minimal cost alongside slightly more flexible retrieval options, with retrieval times that ranging between as little as 1-5 minutes and up to 12 hours.

For example, bulk retrievals, which is the standard flexible retrieval option can be executed for free within 5-12 hours, making it well-suited for backup and disaster recovery scenarios whilst for cases that require more swift access to the data, the service also offers the option of expedited retrieval, which can retrieve the data in a matter of minutes albeit at an additional cost. Though the retrieval speed is lower than the one provided by Instant Retrieval, the trade-off may be worth it for many organizations due to the reduced cost of storage, which can add up to significant sums quite easily, especially when dealing with large quantities of data.

3. **Amazon S3 Glacier Deep Archive:** The lowest cost storage class, S3 Glacier Deep Archive is designed for data that are to be stored for the long-term, with the expectation of being accessed infrequently. However, the low cost comes at the trade-off of retrieval time, which can take up to 12 hours. This makes it ideal for use cases where immediate access to the data is not a priority but long-term durability and cost efficiency are crucial such as compliance archives, legal records or digital media preservation.

TLDR;

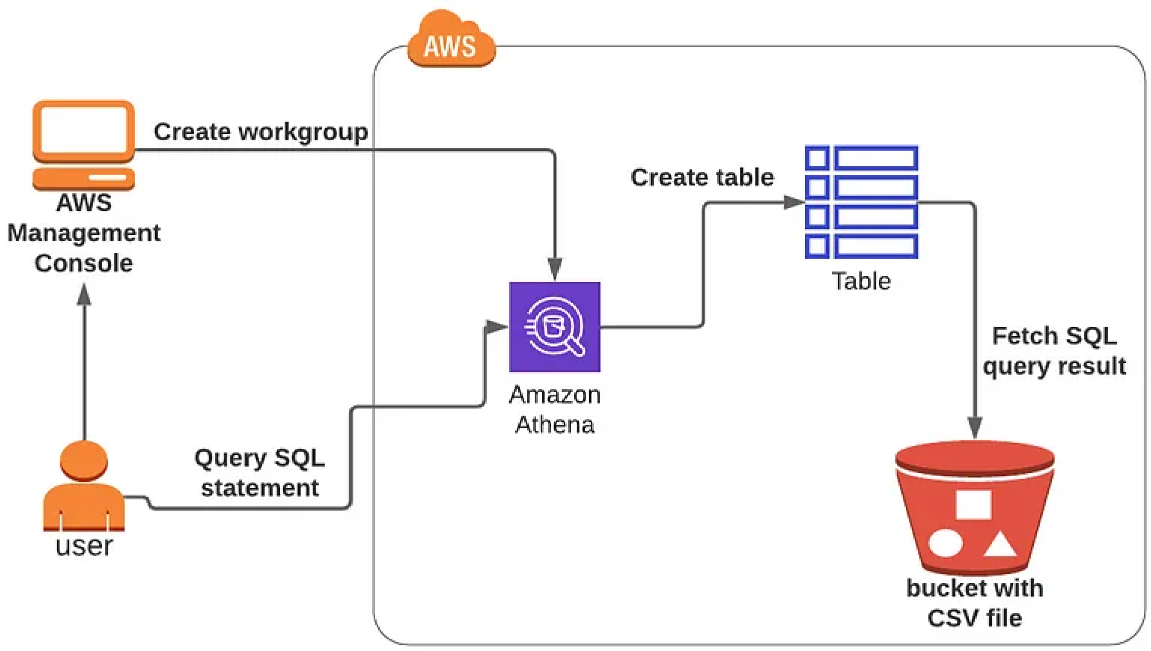
S3 Glacier is a storage service tier specializing in data archival, and offers three different storage classes each offering data access at varying retrieval speeds and price points, often trading one for the other. With the Instant retrieval class for example, being slightly more expensive but providing millisecond-level retrieval speeds, as opposed to the Deep Archive class, which has the lowest storage costs available, but data retrieval can take up to 12 hours.

Athena

Named after the Greek goddess of wisdom, Amazon Athena is a serverless, interactive query service that allows you to analyze data stored in different locations and formats as a single dataset using standard SQL, mostly sourcing the data from Amazon S3 and supports a wide variety of file formats such as CSV, JSON, ORC and Apache Paraquet right out of the box thereby eliminating the need for complex ETL processes and infrastructure setup, enabling the user to start querying their data immediately.

Though not necessarily low latency in nature, Athena usually returns the results of queries in a matter of seconds, making it rather quick to use which alongside its relative cost-effectiveness has led to its widespread integration and use with business intelligence tools such as Microsoft PowerBI and Amazon Quicksight (A service we will learn more about later). In fact, Amazon even provides its own Microsoft Power BI Connector for Amazon Athena which can be used to analyze data present in Amazon Athena from the Microsoft PowerBI desktop application directly (at least on Windows machines).

Also worth mentioning is that workloads that utilize Apache Spark can also be run using Amazon Athena making it extremely easy to run analytics applications without the need to provision or manage resources and infrastructure, due to the serverless nature of Amazon Athena. In fact, since Athena is a serverless offering, when using it to run queries on the desired S3 buckets, the user usually only has to pay for the queries they run, and nothing else, making it a cost-effective solution for ad-hoc data analysis and reporting.



Amazon AthenaSQL Architecture (Credit: Whizlabs)

Amazon Athena works through something called workgroups, where users can control and manage their different query statements as well as the resources required to run said queries. Once the workgroup is created, the user can input SQL queries into Athena through the AWS Management Console or Command Line Interface (CLI) as shown in the diagram above. Athena, a query service, interacts with data stored in Amazon S3, running SQL statements directly on the files without requiring data movement.

Next, a table is created in Athena which serves as a metadata layer over the data in the S3 bucket. The actual data, in this case, is stored as CSV files in S3, and Athena reads the table structure to know how to query it. Finally, after the query is executed, Athena fetches the results from the CSV file in the S3 bucket and presents them to the user in the form of SQL query results, which can be retrieved through the console/ CLI.

Similar architecture and methods are used when running Apache Spark applications using Amazon Athena.

TLDR;

Amazon Athena can be used to run queries on-demand on an S3 bucket with support for a wide variety of file formats such as CSV, JSON, Apache Paraquet, etc. The queries run by Amazon Athena usually return results within a matter of seconds and the service only charges for the queries and not the associated resources required to run them.

The service also supports running analytics applications that utilize Apache Spark workloads.

Transfer Acceleration

Amazon S3 Transfer Acceleration speeds up file transfers to and from Amazon S3 by using AWS's globally distributed edge locations. It leverages Amazon CloudFront's network of edge locations to route data optimally, reducing latency and improving upload and download speeds, especially for users far from the S3 bucket's region. This service is ideal for transferring large files or gathering data sets under the domain of one roof/service quickly and efficiently.

This is especially useful in cases where data stored across various points globally needs to be centrally gathered and performed operations on. However, while it can significantly improve data transfer speeds, it comes with an additional charge compared to regular S3 transfers. Therefore, it’s crucial to evaluate whether the performance improvement is worth the cost for specific use cases. For example, organizations dealing with global users or heavy media files will likely benefit, but for smaller, local transfers, the cost may outweigh the performance gains.

It is also worth noting that once transfer acceleration is enabled for a S3 bucket, it receives an accelerated endpoint with a `s3-accelerate` domain name (for example, `https://<bucketname>.s3-accelerate.amazonaws.com`). In order to benefit from the accelerated data transfers, one has to use the new accelerated endpoint as AWS will not automatically optimize transfers performed using the default standard S3 endpoint.

TLDR;

Aggregate the data from all these global sites as quickly as possible in a single Amazon S3 bucket ⇒ S3 Transfer Acceleration

Accidental Deletion

AWS has made it extremely easy to create, configure, monitor and manage your S3 objects be it through the AWS Management Console, CLI or the S3 API. However, this ease-of-use may also backfire as the barriers preventing the deletion objects are rather low by default. This can, if not managed properly lead to the accidental deletion of the data stored using Amazon S3.

As such, to prevent to accidental deletion of your data in AWS from occurring, it may be desirable to add extra layers of security to protect the data from accidental deletion. The most common way in which this can be achieved is through the combined use of **Versioning** and **Multi-Factor Authentication (MFA)**.

Versioning keeps multiple versions of an object in an S3 bucket, allowing you to recover previous versions if an object is mistakenly deleted or overwritten. It is a built-in feature of all object storage methods, though this does result in greater expenses on the user end as Amazon charges us for the total storage used, which can balloon into large amounts quite easily when storing all the different historical versions of the objects in an S3 bucket.

MFA Delete, short for Multi-Factor Authentication Delete, adds an extra layer of security by requiring additional authentication for delete operations, ensuring that only authorized users can permanently remove data, resulting in the addition of a very human level of safeguarding for your critical data against accidental or unauthorized deletion.