- S3 stands for Simple Storage Service.
- It provides object storage through a web service interface.
- Each object is stored as a file with its metadata included and is given an **ID number**.
- Objects uploaded to S3 are stored in containers called **Buckets**, whose names are **unique** and they organize the Amazon S3 namespace at the highest level.
- **These buckets are region specific**.
- You can assign permissions to these buckets, in order to provide access or restrict data transaction.
- Applications use this ID number to access an object.
- Developers can access an object via a REST API.
- Supports upload of objects.
- Uses the same scalable storage infrastructure that Amazon.com uses to run its global e-commerce network.
- Designed for storing online backup and archiving of data and applications on AWS.
- It's mainly designed with the minimal features in order to create web-scale computing in an easy way.
- Storage classes provided are:
    - Standard
    - Standard_IA i.e., Standard Infrequent Access
    - Intelligent_Tiering
    - OneZone_IA
    - Glacier
    - Deep_Archive
    - RRS i.e., Reduced Redundancy Storage (Not recommended by AWS)

    Short cut: GODISS
- Data access is provided through S3 console, which is a simple web-based interface.
- Data stored can be either Public or Private based on user requirement.
- Data stored can be encrypted.
- We can define life-cycle policies which can help in automation of data transfer, retention and deletion.
- Amazon Athena can be used to **query** S3 data as per demand.

S3 Use cases :

- Backup and storage
- Disaster Recovery
- Archive
- Hybrid Cloud storage
- Application hosting
- Media hosting
- Data lakes & big data analytics
- Software delivery
- Static website

Short cut : **Badhmass**

	S3 Standard	S3 Intelligent-Tiering	S3 Standard-IA	S3 One Zone-IA	S3 Glacier	S3 Glacier Deep Archive
Designed for durability	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)
Designed for availability	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99.9%	99.9%
Availability Zones	≥3	≥3	≥3	1	≥3	≥3
Minimum capacity charge per object	N/A	N/A	128KB	128KB	40KB	40KB
Minimum storage duration charge	N/A	30 days	30 days	30 days	90 days	180 days
Retrieval fee	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved

What is Versioning?

- Versioning is a means for keeping multiple variants of the same object in the bucket.
- Versioning is used to preserve, retrieve, and restore every version of every object stored in an S3 bucket.
- Versioning is done at the S3 Bucket level.
- Versioning can be enabled from the AWS Console / SDKs / API.
- Versioning, once enabled, cannot be completely disabled.

- The alternative to disabling versioning is placing the bucket in a "versioning-suspended" state.
- A drawback of having multiple versions of an object is you are billed multiple times (since the objects are getting stored in S3 each time).
- In order to avoid having multiple versions of the same object, S3 has a feature called Lifecycle Management. This allows us to decide on what to do when multiple versions of an object are piling up.
- One advantage of versioning is, we can provide permissions on versioned objects i.e., we can define which version of an object is public and which one is private.

What is a Static Website?

- These are the most basic types of websites and are the easiest to create.
- A static web page is a web page that is delivered to the user's web browser exactly as stored.
- It holds fixed content, where each page is coded in HTML and displays the same information to every visitor.
- No web programming or database design is required when working with them.
- They are a safe bet when it comes to security, since we do not have any interaction with databases or plugins.
- They are reliable, i.e., if any attack happens on the server, a redirection to the nearest safest node happens.
- Static websites are very fast, because there is no true backend to fetch information from.
- Hosting the website is cheap due to the non-existence of any other components.
- Scaling of the website is easy, and can be done by just increasing the bandwidth.

IAM Policy

1. An **IAM (Identity and access management) policy** is an entity in AWS, that enables you to manage access to AWS services and resources in a secure fashion.

2. **Policies** are stored on **AWS** in **JSON format** and are attached to resources as identity-based **policies.**
3. You can attach an **IAM policy** to different entities such as an **IAM group, user, or role.**
4. IAM policies gives us the power of restricting users or groups to only use the specific services that they need.

Policy Types

There are two important types of policies:

- Identity-Based-Policies
- Resource-Based-Policies

Identity-Based-Policy

1. Identity-based policies are policies that you can attach to an AWS identity (such as a user, group of users, or role).
2. These policies control what actions an entity can perform, which resources they can use, and the conditions in which they can use said resources.
3. Identity-based policies are further classified as:
    - AWS Managed Policies
    - Custom Managed Policies

AWS Managed Policies

1. AWS Managed policies are those policies that are **created and managed by AWS** itself.
2. If you are new to IAM policies, you can start with AWS managed policies before managing your own.

Custom Managed Policies

1. Custom managed policies are policies that are created and managed by you in your AWS account.
2. Customer managed policies **provide us with more precise control** than AWS managed policies.
3. You can create and edit an IAM policy in the visual editor or by creating the JSON policy document directly.
4. You can **create your own IAM policy** using the following link: https://awspolicygen.s3.amazonaws.com/policygen.html

Resource-Based-Policy

1. Resource-based policies are policies that **we attach to a resource** such as an Amazon S3 bucket.
2. Resource-based policies grant the specified permission to perform specific actions on particular resources and define under what conditions these policies apply to them.
3. Resource-based policies are in line with other policies.
4. There are **currently no AWS-managed resource-based** policies.
5. There is only one type of resource-based policy called a *trust policy*, which is attached to an IAM role.
6. An IAM role is both an **identity and a resource** that supports resource-based policies.

IAM Role

1. An IAM *role* is an **AWS IAM identity** (that we can create in our AWS account) that has specific permissions.
2. It is similar to an IAM user, which determines what the identity can and cannot do in AWS.
3. Instead of attaching a role to a particular user or group, **it can be attached to anyone who needs** it.
4. The advantage of having a role is that we **do not have standard long-term credentials** such as a password or access keys associated with it.
5. When resources assume a particular role, **it provides us with temporary security credentials** for our role session.
6. We can use roles to access users, applications, or services that don't have access to our AWS resources.
7. We **can attach one or more policies with roles,** depending on our requirements.
8. For example, we can create a role with s3 full access and attach it to an EC2 instance to access S3 buckets.

Simple storage service(S3)

1. Amazon S3 is a **simple storage service** that we can use to **store and retrieve** any amount of **data**, at any time, from anywhere on the web.
2. It gives developers and users access to **highly scalable, reliable, fast, inexpensive data storage** infrastructure.
3. S3 **guarantees 99.9% availability** at any point in time.
4. S3 has been designed to **store up to 5 TB of data**.

5. S3 is global, meaning you can create a bucket in any region and access it from anywhere. Due to this, the **name of the bucket should be a unique** one.
6. The S3 bucket objects, as well as the bucket, **can be deleted at any time** by the user.
7. We can limit access to our bucket by granting different permissions for different users.
8. S3 also comes with additional features such as **versioning, static website hosting, server access logging and life cycle policy** for storing objects, and many others.

Uploading and copying objects using multipart upload

- Multipart upload allows you to upload a single object as a set of parts.
- Each part is a contiguous portion of the object's data.
- You can upload these object parts independently and in any order.
- If transmission of any part fails, you can retransmit that part without affecting other parts.
- After all parts of your object are uploaded, Amazon S3 assembles these parts and creates the object.

  Note: When your object size reaches 100 MB, you should consider using multipart uploads instead of uploading the object in a single operation.

When to use multipart upload

- If you're uploading large objects over a stable high-bandwidth network, use multipart upload to maximize the use of your available bandwidth by uploading object parts in parallel for multi-threaded performance.
- If you're uploading over a spotty network, use multipart upload to increase resiliency to network errors by avoiding upload restarts.
- When using multipart upload, you need to retry uploading only parts that are interrupted during the upload. You don't need to restart uploading your object from the beginning.

Multipart upload process

- Multipart upload is a three-step process:
  - **Multipart upload initiation:** When you send a request to initiate a multipart upload, Amazon S3 returns a response with an upload ID, which is a unique identifier for your multipart upload. You must include this upload ID whenever you upload parts, list the parts, complete an upload, or stop an

upload. If you want to provide any metadata describing the object being uploaded, you must provide it in the request to initiate multipart upload.

- **Parts upload:** When uploading a part, in addition to the upload ID, you must specify a part number. You can choose any part number between 1 and 10,000. A part number uniquely identifies a part and its position in the object you are uploading.
- **Multipart upload completion:** When you complete a multipart upload, Amazon S3 creates an object by concatenating the parts in ascending order based on the part number. If any object metadata was provided in the initiate multipart upload request, Amazon S3 associates that metadata with the object. After a successful complete request, the parts no longer exist.

Summary

- Buckets vs Objects: global unique name, tied to a region
- S3 security: IAM policy, S3 Bucket Policy (public access), S3 Encryption
- S3 Websites: host a static website on Amazon S3
- S3 Versioning: multiple versions for files, prevent accidental deletes
- S3 Access Logs: log requests made within your S3 bucket
- S3 Replication: same-region or cross-region, must enable versioning
- S3 Storage Classes: Standard, IA, 1Z-IA, Intelligent, Glacier, Glacier Deep Archive
- S3 Lifecycle Rules: transition objects between classes
- S3 Glacier Vault Lock / S3 Object Lock: WORM (Write Once Read Many)
- Snow Family: import data onto S3 through a physical device, edge computing
- Ops Hub: desktop application to manage Snow Family devices
- Storage Gateway: hybrid solution to extend on-premises storage to S3