SIMPLE STORAGE SERVICE

- It is highly scalable, securable, durable to store the data.
- Easy to store and retrieve the data.
- Object Based storage & data is spread multiple regions.
- Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics.

BUCKETS:

- If you create a root level folder then it is called as Buckets.
- Used to store Objects.
- Bucket ownership is not transferable to another account.
- After you create a bucket, you can't change its name or Region.
- We cannot delete the bucket directly we need to make empty it first.
- After a bucket is deleted, the name becomes available for reuse
- The files which are stored in S3 can be from 0 Bytes to 5 TB.
- By default, you can create up to 100 buckets in each of your AWS accounts.
- If you need additional buckets, you can increase your account bucket limit to a maximum of 1,000 buckets by submitting a service limit increase.

OBJECTS:

- Files that you stored in a bucket is called as Objects.
- To store your data in Amazon S3, you work with resources known as buckets and objects.
- A *bucket* is a container for objects.
- An *object* is a file and any metadata that describes that file.
- With Amazon S3, you pay only for what you use

Depending on the size of the data you are uploading, Amazon S3 offers the following options:

- **Upload an object in a single operation using the AWS SDKs, REST API, or AWS CLI—**With a single PUT operation, you can upload a single object up to 5 GB in size.
- **Upload a single object using the Amazon S3 Console—**With the Amazon S3 Console, you can upload a single object up to 160 GB in size.
- **Upload an object in parts using the AWS SDKs, REST API, or AWS CLI—**Using the multipart upload API, you can upload a single large object, up to 5 TB in size.

MAJOR COMPONENTS OF OBJECT:

- KEY : NAME OF OBJECT
- VALUE : DATA IN BYTES
- VERSION ID : SHOWS VERSIONING ID FOR UNIQUNESS OF OBJECT
- META DATA : DATA ABOUT DATA WE ARE STORING

RULES TO CRAETE A BUCKET

- ➢ UNIQUE NAME
- ➢ NAME SHOULD BE 3 TO 63 CHARACTERS
- ➢ ONLY LOWER CASE IS VALID
- ➢ SHOULD NOT BE IN IP FORMAT
- ➢ MUST BEGIN AND END WITH NUMBER OR LETTER.

Note
Before March 1, 2018, buckets created in the US East (N. Virginia) Region could have names that were up to 255 characters long and included uppercase letters and underscores. Beginning March 1, 2018, new buckets in US East (N. Virginia) must conform to the same rules applied in all other Regions.

S3 VERSIONING:

- ➢ It will enable you the multiple versions of the same file.
- ➢ It will helps accidental termination.
- ➢ It can retrieve the Previous versions of the Objects.
- ➢ By default it will not be enabled we need to enable it.
- ➢ We cant disable the Versioning only we can able to suspend it.

STORAGE CLASSES IN S3:

1. STANDARD:

- Data is stored in multiple locations
- Frequently accessed data
- This is the default storage classes in s3
- Delivers low latency and high throughput performance
- Designed for 99.99% availability and 99.999999999% (11 9's) durability.
- Cost is High
- Ex: We can store the data, which we use this regularly like websites, mobile applications, gaming applications and big data analysis

2. S3 Standard-Infrequent Access:

- Access less Frequently but requires rapid access when needed
- It offers the same durability and availability but Low cost, compared to Standard
- With this we can pay when we retrieve the data.
- Ex: back-up, long-term data & Disaster recovery

3. S3 One Zone-Infrequent Access:

- It is similar to S3 Standard-Infrequent Access but stores data in a single availability zone.
- If the availability zone experiences any outage then we lost our data
- Cost will be low compared to S3 standard-Infrequent access
- Ex: Secondary backups & Storages that can easily be recreated

4. S3 Intelligent-Tiering:

- It is a storage class which movies data between two access tiers based on changing access patterns.
- This storage class offers frequent and infrequent access tiers, with the option to move data to S3 Glacier or S3 Glacier Deep Archive for long-term archival
- Ex: Data analytic, Multimedia & General-purpose storage.

5. S3 Glacier:

- It is a low-cost storage class that is designed for long-term data archiving.
- It offers very low storage costs but with a retrieval time of several hours.
- S3 Glacier is ideal for data that is rarely accessed but needs to be stored for compliance or regulatory purposes.
- Ex: Long-term archival & Compliance and regulatory requirements

6. S3 Glacier Deep Archive:

- It is the most cost-effective storage class for long-term data archiving.
- It offers the lowest storage costs but with a retrieval time of up to 12 hours.
- S3 Glacier Deep Archive is ideal for data that is required to be stored for long periods and accessed very rarely.
- Ex: Long-term archival & Compliance and regulatory requirements for data retention

S3 INTEGRATION WITH JENKINS

1. Install s3 publisher plugin in jenkins dashboard
2. Create IAM user with s3 permissions
3. Fo to manage jenkins >> configure system >> **Amazon S3 profiles >>**
4. Profile name: our-wish
5. Access-key :
6. Secret-key :
7. Test connection
8. Create a job >>
9. Go to post build actions >> **Publish artifacts to S3 Bucket >>**
10. Files to upload:
11. Source: **/*.war
12. Destination bucket: our bucket name
13. bucket_region: same-region (bucket and ec2)
14. Tick on no upload on build failure

SAVE & BUILD

CLI FOR S3:

TO CREATE A BUCKET : aws s3 mb s3://bucketname (or) aws s3api create-bucket –bucket test-bucket –region us-east-1

S3 CODE FOR TERRAFOM:

```
resource "aws_s3_bucket" "one" {
  bucket = "my-bucket-name"
}

resource "aws_s3_bucket_ownership_controls" "two" {
  bucket = aws_s3_bucket.one.id
  rule {
    object_ownership = "BucketOwnerPreferred"
  }
}

resource "aws_s3_bucket_acl" "three" {
  depends_on = [aws_s3_bucket_ownership_controls.two]

  bucket = aws_s3_bucket.one.id
  acl    = "private"
}

resource "aws_s3_bucket_versioning" "three" {
bucket = aws_s3_bucket.one.id
versioning_configuration {
status = "Enabled"
}
}
```