Amazon S3 (Simple Storage Service)

What is Amazon S3?

Amazon S3 (Simple Storage Service) is an object storage service provided by AWS that allows users to store and retrieve any amount of data from anywhere on the web. It is highly scalable, durable, and cost-effective, making it a popular choice for data storage, backup, and content distribution.

---

1. Key Features of Amazon S3

a) Scalability

- S3 automatically scales to handle unlimited data storage.
- No need to provision or manage storage capacity.

b) Durability & Availability

- Provides **99.999999999% (11 nines) durability**, meaning data loss is extremely rare.
- Replicates data across multiple Availability Zones (AZs) within a region.

c) Security

- Supports **IAM policies** for fine-grained access control.
- Uses **server-side encryption (SSE)** for data protection.
- Can integrate with **AWS KMS (Key Management Service)** for encryption key management.

d) Cost-Effectiveness

- **Pay-as-you-go pricing** (no upfront costs).
- Multiple storage classes (Standard, Intelligent-Tiering, Glacier) to optimize cost.

e) Performance

- High throughput and low latency.
- Supports parallelism for faster data uploads and retrievals.
-

2. Amazon S3 Architecture

S3 is a **flat storage system** (not hierarchical like file systems). The key components are:

a) Buckets

- A bucket is a **top-level container** for storing objects.
- Each bucket must have a **globally unique name**.
- Example bucket name: `my-website-bucket`

b) Objects

- Files stored in S3 are called **objects**.
- Each object consists of:
  - **Key (Filename)**: The unique identifier of the object within a bucket.
  - **Value (Data)**: The actual file content.
  - **Metadata**: Additional information about the file (e.g., content type).
  - **Version ID**: If versioning is enabled, each object has multiple versions.

c) Object Keys

- The **key** is the full path of the file inside the bucket.
- Example:

```
my-bucket/
  images/profile.jpg  → Key: "images/profile.jpg"
  documents/resume.pdf  → Key: "documents/resume.pdf"
```

---

3. Amazon S3 Storage Classes

S3 provides multiple storage classes based on access frequency and cost:

Storage Class	Use Case	Durability	Availability	Cost
S3 Standard	Frequent access, high performance	99.999999999 %	99.99%	High
S3 Intelligent-Tiering	Automatically moves data to lower-cost tiers	99.999999999 %	99.90%	Medium
S3 Standard-IA	Infrequent access (e.g., backups)	99.999999999 %	99.90%	Lower
S3 One Zone-IA	Cheaper, but data stored in 1 AZ	99.999999999 %	99.50%	Lower
S3 Glacier	Long-term archival storage (~1-5 min retrieval)	99.999999999 %	Varies	Low

Storage Class	Use Case	Durability	Availability	Cost
S3 Glacier Deep Archive	Ultra-low-cost archival (~12-48 hours retrieval)	99.999999999 %	Varies	Very Low

---

4. S3 Data Management Features

a) Versioning

- Stores multiple versions of an object.
- Helps prevent accidental deletion.
- Example:

```
my-bucket/
  report.pdf (Version 1)
  report.pdf (Version 2)
```

Steps:

1. Open **Amazon S3 Console**.
2. Click on the **bucket name**.
3. Go to the **Properties** tab.
4. Scroll to **Bucket Versioning**.
5. Click **Edit → Enable** Versioning.
6. Click **Save Changes**.

How to Use Versioning:

- **Upload a new version of an object**:
  - Go to the bucket, upload the same file again.
  - S3 keeps previous versions, accessible under the **Versions** tab.
- **Restore a previous version**:
  - Click on the object → Click **Versions** → Select an older version → **Download**.

b) Lifecycle Policies

- Automatically moves objects to a cheaper storage tier.
- Example: Move objects from **S3 Standard → S3 Glacier** after 30 days.
- Example Lifecycle Policy (JSON):

```
{
  "Rules": [
    {
      "ID": "Move to Glacier",
      "Status": "Enabled",
      "Filter": { "Prefix": "logs/" },
      "Transitions": [{ "Days": 30, "StorageClass": "GLACIER" }]
    }
  ]
}
```

`Steps:`

1. Open the **S3 Console**.
2. Select your **bucket**.
3. Go to the **Management** tab.
4. Click **Create Lifecycle Rule**.
5. Enter a **Rule Name** (e.g., `Move-to-Glacier`).
6. Choose **Apply to all objects** or filter by prefix (e.g., `logs/`).
7. **Select Actions**:
    - Move objects to **S3 Glacier** after 30 days.
    - Permanently delete objects after 365 days.
8. Click **Create Rule**.

c) Replication

- Copies data from one S3 bucket to another **in the same or different AWS regions**.
- Types:
    - **Same-Region Replication (SRR)**
    - **Cross-Region Replication (CRR)**
- Useful for disaster recovery and compliance.

Steps:

1. Open **S3 Console**.
2. Select your **source bucket**.
3. Go to the **Management** tab → Click **Create Replication Rule**.
4. Enter a **Rule Name** (e.g., `Backup-to-us-east-1`).
5. **Choose the Destination Bucket**:
    - Select **another bucket** in a different region (Cross-Region Replication).
    - Select **another bucket** in the same region (Same-Region Replication).
6. **IAM Role**:
    - Choose **Create new role** or **Use existing role**.
7. **Additional settings**:
    - Enable **Delete Marker Replication** (if needed).
    - Enable **Replica Storage Class** (e.g., S3 Standard-IA for cost savings).
8. Click **Save Changes**.

Notes:

- Versioning **must be enabled** on both source and destination buckets.
- Replication is **not retroactive** (only new objects are copied).

d) Object Lock

- Prevents deletion or modification of objects for a specified period.
- Used for compliance (WORM – Write Once, Read Many).

Steps:

1. Open **S3 Console**.
2. Click **Create Bucket**.
3. Under **Advanced Settings**, enable **Object Lock** (must be enabled at bucket creation).
4. Click **Create Bucket**.
5. Upload an object.
6. Click on the **object → Properties**.
7. Scroll to **Object Lock**.
8. Set **Retention Mode**:
    - **Governance Mode**: Allows privileged users to override retention.
    - **Compliance Mode**: Fully protects objects from deletion/modification.
9. Set **Retention Period** (e.g., 1 year).
10. Click **Save Changes**.

---

5. Access Control & Security

a) IAM Policies

- Use **AWS Identity and Access Management (IAM)** to grant/restrict access.
- Example IAM policy (Allow read-only access to `my-bucket`):

```
{
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::my-bucket/*"
}
```

b) Bucket Policies

- JSON-based rules to control access at the bucket level.
- Example (Allow public read access):

```
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::my-bucket/*"
}
```

Steps:

1. Open **S3 Console**.
2. Click on the **bucket**.
3. Go to the **Permissions** tab → Scroll to **Bucket Policy**.
4. Click **Edit Policy** and enter a JSON policy.

c) Encryption

- **Server-Side Encryption (SSE)**:
  - **SSE-S3**: AWS manages encryption keys.
  - **SSE-KMS**: Uses AWS KMS for key management.
  - **SSE-C**: You manage encryption keys.
- **Client-Side Encryption**:
  - Encrypt data before uploading.

Steps:

1. Open **S3 Console**.
2. Select your **bucket**.
3. Go to the **Properties** tab.
4. Scroll to **Default Encryption**.
5. Click **Edit** → Choose encryption type:
   - **AES-256 (SSE-S3)**
   - **AWS KMS (SSE-KMS)** (requires KMS key)
6. Click **Save Changes**.

d) Block Public Access

- Ensures that sensitive data is **not publicly accessible**.
- Can be enforced at the **bucket level**.

---

6. S3 Performance Optimization

a) Multipart Uploads

- Splits large files into smaller parts for faster uploads.
- Required for files **over 5GB**.
- Example CLI command:

```
aws s3 cp largefile.zip s3://my-bucket/ --storage-class STANDARD --multipart-chunksize 64MB
```

b) S3 Transfer Acceleration

- Uses **AWS Edge Locations** (CloudFront) for faster uploads over long distances.

c) Requester Pays

- The person downloading the file **pays the cost** instead of the bucket owner.

---

7. S3 Data Access Methods

a) AWS Console

- GUI-based file management.

b) AWS CLI

- Upload file:

```
aws s3 cp file.txt s3://my-bucket/
```

- Download file:

```
aws s3 cp s3://my-bucket/file.txt .
```

- List all objects:

```
aws s3 ls s3://my-bucket/
```

c) AWS SDKs

- Code example (Python Boto3):

```
import boto3
s3 = boto3.client('s3')
s3.upload_file("file.txt", "my-bucket", "file.txt")
```

d) Presigned URLs

- Generate temporary URLs for secure file access.
- Example (Python Boto3):

```
url = s3.generate_presigned_url('get_object', Params={'Bucket': 'my-bucket',
'Key': 'file.txt'}, ExpiresIn=3600)
print(url)
```

Conclusion

Amazon S3 is a powerful, secure, and highly scalable object storage solution used for various applications, from simple backups to large-scale data lakes. With features like **lifecycle management, versioning, encryption, and replication**, S3 ensures reliability, security, and cost optimization.