DevOps

[AWS S3 (Simple Storage Service) 2](#_Toc193630120)

[Overview 2](#_Toc193630121)

[S3 Versioning 4](#_Toc193630122)

[Storage Classes 6](#_Toc193630123)

[S3 Lifecycle Management 8](#_Toc193630124)

[S3 CORS 9](#_Toc193630125)

[S3 Replication: CRR & SRR 11](#_Toc193630126)

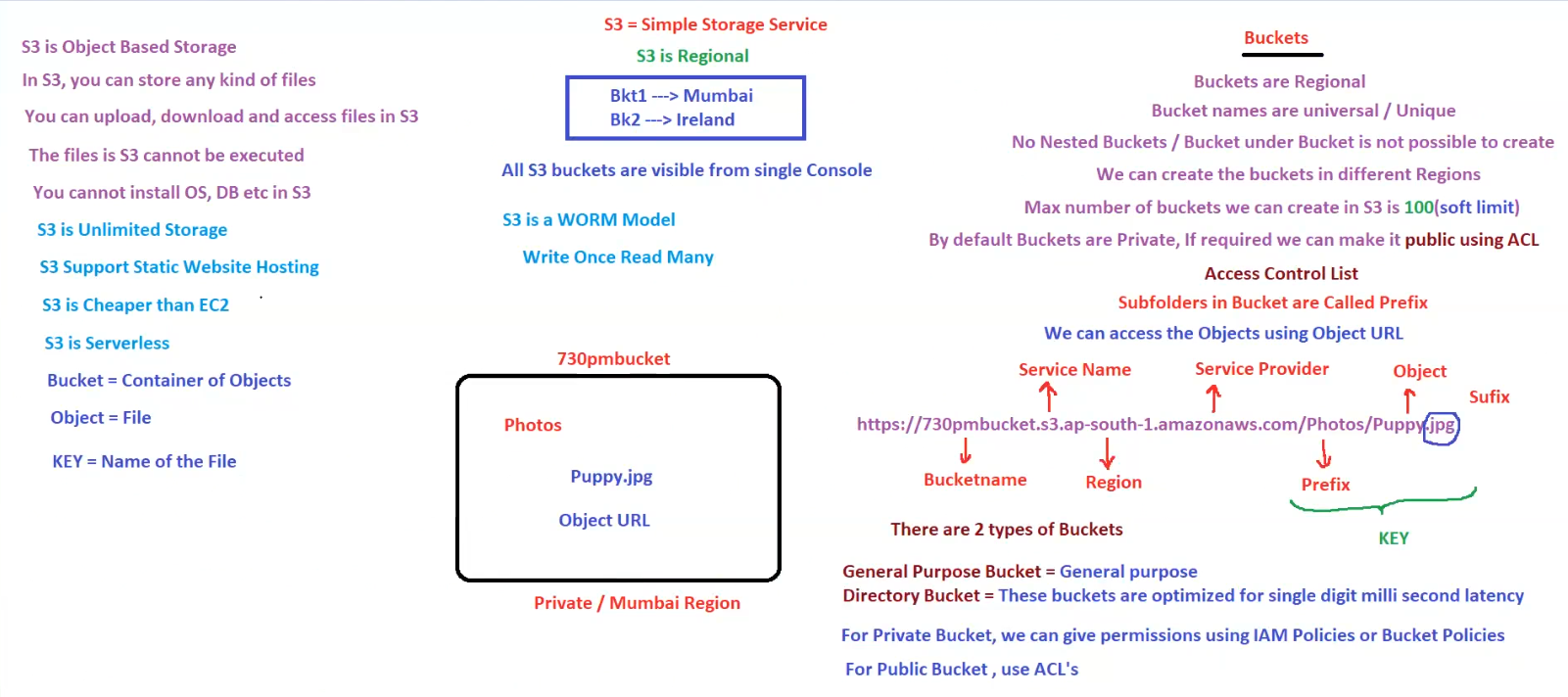
[S3 Encryption 13](#_Toc193630127)

[S3 Advanced Features 15](#_Toc193630128)

[AWS Transfer Family 17](#_Toc193630129)

# AWS S3 (Simple Storage Service)

## Overview



**1. What is S3?**

* **S3 (Simple Storage Service)** is an **Object-Based Storage** solution provided by AWS.
* It allows users to **store, upload, download, and access files**.
* **Key Characteristics:**
  + S3 provides **unlimited storage**.
  + **Supports static website hosting**.
  + **Cheaper than EC2** for storage solutions.
  + **Serverless**, meaning no infrastructure management.
  + **WORM Model (Write Once Read Many)** – Ensures data integrity.

**2. S3 Buckets**

* **Bucket**: A container for objects (files).
* **Objects**: The files stored inside a bucket.
* **Key**: The unique name of an object in a bucket.

**Bucket Characteristics**

* **Buckets are Regional** (data is stored in a specific AWS region).
* **Bucket names are globally unique** across AWS.
* **No nested buckets** (buckets cannot be created inside another bucket).
* **Buckets are visible from a single AWS console**.
* **Soft limit** of **100 buckets per AWS account** (can be increased via AWS support).
* **By default, buckets are private** – Access can be controlled using **ACLs (Access Control Lists)** or **IAM Policies**.

**Example of S3 Bucket Structure**

* **Bucket Name**: 730pmbucket
* **Region**: Mumbai (ap-south-1)
* **Object**: Puppy.jpg
* **Path**:
* https://730pmbucket.s3.ap-south-1.amazonaws.com/Photos/Puppy.jpg

**3. S3 Storage Model**

* **Buckets contain Objects (Files).**
* **Subfolders within a Bucket are called Prefixes.**
* Objects are accessed using an **Object URL**.

**URL Breakdown**

https://730pmbucket.s3.ap-south-1.amazonaws.com/Photos/Puppy.jpg

| **Component** | **Meaning** |
| --- | --- |
| 730pmbucket | Bucket Name |
| s3.ap-south-1.amazonaws.com | AWS S3 Service Provider & Region |
| Photos | Prefix (subfolder) |
| Puppy.jpg | Object Name (Key) |

**4. Types of Buckets**

1. **General Purpose Bucket** – Used for general storage purposes.
2. **Directory Bucket** – Optimized for **low-latency** access (single-digit millisecond latency).

**5. Access Control in S3**

* **Private Bucket**:
  + Permissions managed using **IAM Policies** or **Bucket Policies**.
* **Public Bucket**:
  + Can be made **public using ACLs** (not recommended for sensitive data).

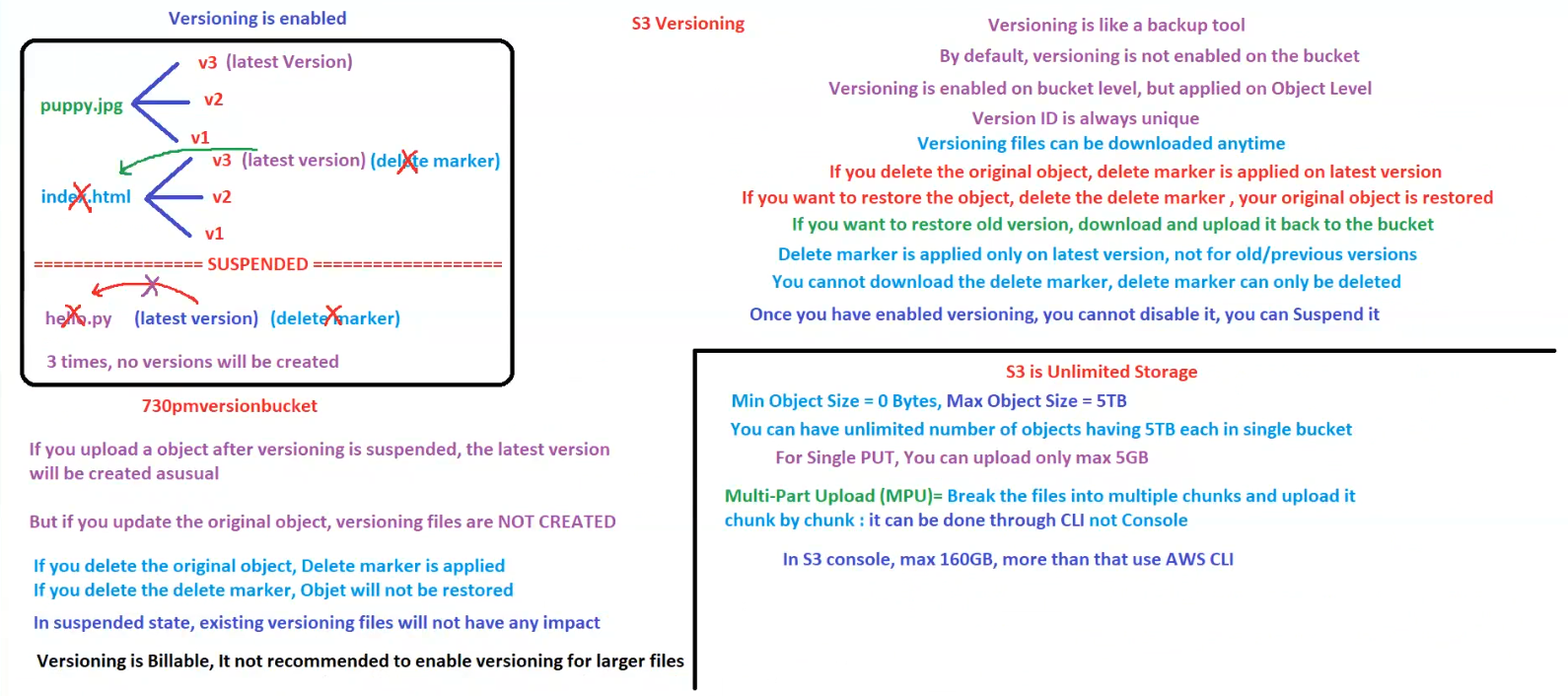
**Access Control Mechanisms**

* **IAM Policies**: Define fine-grained permissions for users.
* **Bucket Policies**: Control access to the entire bucket.
* **ACLs (Access Control Lists)**: Set permissions at the object level.

**6. Key Takeaways**

✅ **S3 is highly scalable and cost-effective**.  
✅ **Data is stored in buckets at the regional level**.  
✅ **Files are accessed via Object URLs**.  
✅ **Supports Static Website Hosting**.  
✅ **Access control is managed using IAM, ACLs, and Bucket Policies**.

## S3 Versioning



**AWS S3 Versioning & Storage Limits**

**1. S3 Versioning Overview**

* **Versioning acts as a backup tool** to retain multiple versions of an object.
* **By default, versioning is disabled** on an S3 bucket.
* **Versioning is enabled at the bucket level but applies to individual objects.**
* Each object version has a **unique Version ID**.
* **Once enabled, versioning cannot be disabled** but can be **suspended**.

**How Versioning Works**

* When an object is updated, a **new version is created**.
* If an object is **deleted**, a **delete marker** is applied to the latest version.
* To **restore a deleted object**, the delete marker must be removed.
* **Old versions can be manually restored** by downloading and re-uploading them.

**Delete Marker Behavior**

* **Applied only to the latest version** when an object is deleted.
* **Previous versions remain intact**.
* **Cannot download a delete marker**, but it can be **removed** to restore the object.

**Impact of Suspending Versioning**

* **New uploads will be stored as the latest version (without versioning).**
* **Existing versioned files remain unaffected.**
* **If an object is updated while versioning is suspended, no new versions are created.**

**Billing Considerations**

* **Versioning is billable**, meaning **storage costs increase** with multiple object versions.
* **Not recommended for large files** due to cost implications.

**2. S3 Object Storage Limits**

* **S3 provides unlimited storage**.
* **Min Object Size:** **0 Bytes**, **Max Object Size:** **5 TB**.
* **A single bucket can store an unlimited number of objects (each up to 5 TB).**
* **For a single PUT operation**, the max upload size is **5 GB**.

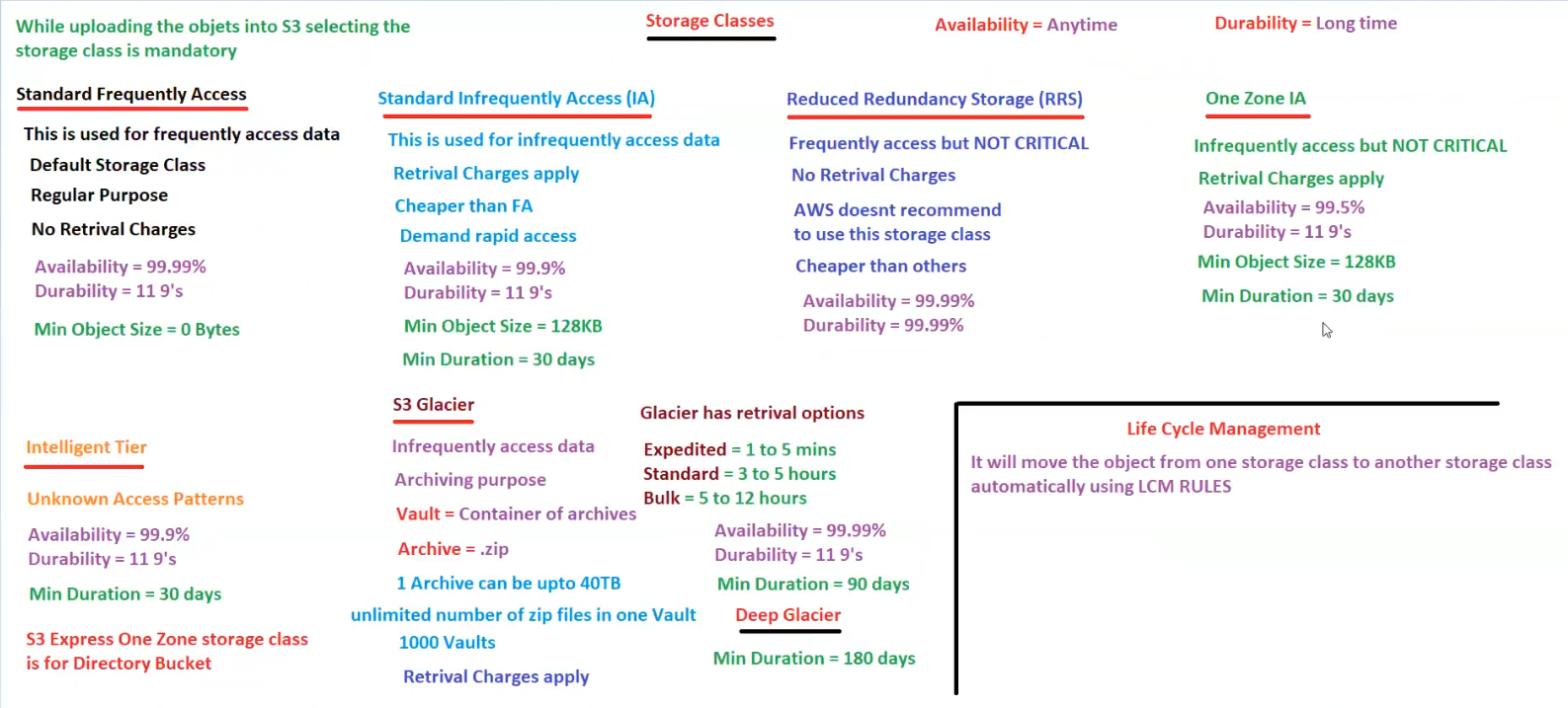
**Multi-Part Upload (MPU)**

* **Files larger than 5 GB must be uploaded in multiple parts.**
* **MPU breaks files into smaller chunks for efficient uploads.**
* **Can be done via AWS CLI but not directly through the AWS Console.**
* **AWS Console supports uploads up to 160 GB**, for larger files, **use AWS CLI**.

**3. Summary of Key Takeaways**

| **Feature** | **Description** |
| --- | --- |
| **Versioning** | Stores multiple versions of an object for backup and recovery |
| **Delete Marker** | Applied when an object is deleted, latest version only |
| **Suspending Versioning** | Stops creating new versions but does not delete existing ones |
| **Storage Limits** | Unlimited objects, max size 5 TB per object |
| **Multi-Part Upload** | Required for files larger than 5 GB, CLI required for >160 GB |

## Storage Classes



**AWS S3 Storage Classes & Lifecycle Management**

**1. Overview**

* **When uploading objects to S3, selecting a storage class is mandatory.**
* **Storage classes** determine cost, availability, durability, and retrieval speed.
* **Lifecycle Management (LCM)** automates the movement of objects between storage classes.

**2. S3 Storage Classes**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Storage Class** | **Use Case** | **Availability** | **Durability** | **Minimum Object Size** | **Min Duration** | **Retrieval Charges** |
| **Standard (Frequently Accessed)** | Default storage, used for regularly accessed data | 99.99% | 11 9's (99.999999999%) | 0 Bytes | No limit | No |
| **Standard Infrequent Access (IA)** | Used for infrequently accessed data with rapid retrieval | 99.9% | 11 9's | 128 KB | 30 days | Yes |
| **One Zone IA** | Lower-cost alternative for infrequent access (data stored in a single AZ) | 99.5% | 11 9's | 128 KB | 30 days | Yes |
| **Reduced Redundancy Storage (RRS)** | Used for frequently accessed, **non-critical** data | 99.99% | 99.99% | No limit | No limit | No |
| **Intelligent Tiering** | Used for **unknown access patterns**, automatically moves data between Standard & IA | 99.9% | 11 9's | No limit | 30 days | No |
| **S3 Glacier** | Used for **archival storage**, rarely accessed data | 99.99% | 11 9's | No limit | 90 days | Yes |
| **Deep Glacier** | Long-term archival storage with lowest cost | 99.99% | 11 9's | No limit | 180 days | Yes |

**3. S3 Glacier Retrieval Options**

| **Retrieval Type** | **Time Required** |
| --- | --- |
| **Expedited** | 1 - 5 mins |
| **Standard** | 3 - 5 hours |
| **Bulk** | 5 - 12 hours |

* **Vault**: A container for archives in Glacier.
* **Archives**: Individual files stored in a Vault (e.g., .zip files).
* **One archive can be up to 40 TB**.
* **Unlimited number of archives per vault** (up to 1,000 vaults per account).

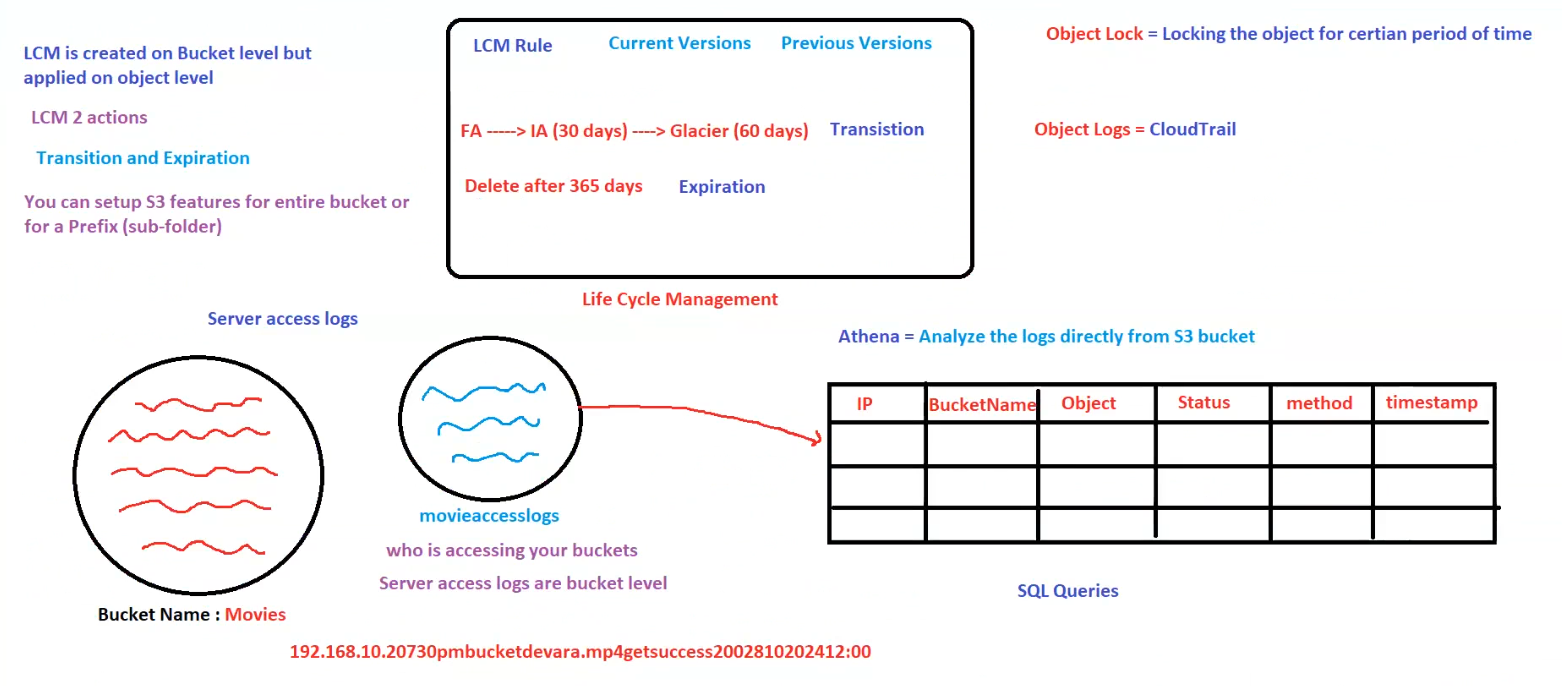
**4. Lifecycle Management (LCM)**

* **LCM automatically moves objects between storage classes** based on defined rules.
* Example:
  + **After 30 days → Move from Standard to IA**
  + **After 90 days → Move from IA to Glacier**
  + **After 180 days → Move from Glacier to Deep Glacier**
* Helps **optimize storage costs** while retaining data for long-term use.

**5. Summary of Key Takeaways**

✅ **Selecting the right storage class** is crucial for cost and performance optimization.  
✅ **Frequent access data → Use Standard class.**  
✅ **Infrequent access → Use IA or One Zone IA.**  
✅ **Archival data → Use Glacier or Deep Glacier.**  
✅ **Use Lifecycle Management (LCM) to automate storage transitions.**

## S3 Lifecycle Management



**AWS S3 Lifecycle Management & Logging**

**1. Lifecycle Management (LCM) in S3**

* **LCM is created at the bucket level but applies at the object level.**
* **Two primary actions in LCM:**
  + **Transition** → Moves objects to different storage classes over time.
  + **Expiration** → Deletes objects automatically after a specified period.

**LCM Rule Example**

| **Action** | **Time Duration** |
| --- | --- |
| **Transition** | Move from Standard to IA after **30 days**, then to Glacier after **60 days**. |
| **Expiration** | Delete object after **365 days**. |

* **LCM can be set for the entire bucket or specific prefixes (subfolders).**

**2. S3 Object Management Features**

* **Object Lock**: Prevents an object from being deleted or modified for a certain period.
* **Object Logs (CloudTrail)**: Tracks actions performed on objects for security and auditing.

**3. S3 Server Access Logs**

* **Monitors bucket activity** (who is accessing objects, from where, and what actions are taken).
* **Access logs are at the bucket level** and stored in a separate S3 bucket.
* **Example:**
  + Bucket storing files: **Movies**
  + Bucket storing logs: **movieaccesslogs**

**Server Access Log Example**

192.168.10.207 30pmbucket devara.mp4 GET success 200 2028102020412:00

| **IP Address** | **Bucket Name** | **Object** | **Method** | **Status** | **Timestamp** |
| --- | --- | --- | --- | --- | --- |
| 192.168.10.207 | 30pmbucket | devara.mp4 | GET | Success (200) | 2028-10-20 24:12:00 |

* **Athena** can be used to analyze logs directly from the S3 bucket using **SQL queries**.

**4. Summary of Key Features**

| **Feature** | **Description** |
| --- | --- |
| **Lifecycle Management** | Automates transitions and deletions for cost optimization. |
| **Object Lock** | Prevents modification/deletion for a defined period. |
| **CloudTrail Logs** | Tracks object-level actions for auditing. |
| **Server Access Logs** | Logs all bucket access activity for security analysis. |
| **Athena Querying** | Analyzes logs using SQL queries directly from S3. |

**Key Takeaways**

✅ **LCM reduces storage costs by moving objects automatically to lower-cost tiers.**  
✅ **Server access logs help track who is accessing the data.**  
✅ **Athena provides an easy way to query S3 logs.**  
✅ **Object Lock secures important data against accidental deletion.**

## S3 CORS

**AWS S3 CORS (Cross-Origin Resource Sharing)**

**1. What is CORS?**

* **CORS (Cross-Origin Resource Sharing)** allows web applications running on one domain to request resources from another domain.
* By default, **browsers block cross-origin requests** for security reasons.
* CORS settings in **S3** define which domains can access objects stored in a bucket.

**2. Understanding the CORS Flow in S3**

**Scenario in the Diagram**

1. **Bucket 1 (corsbkt1)**
   * Stores **index.html** and **puppy.jpg**.
   * The browser successfully loads **index.html** and **puppy.jpg** from **corsbkt1**.
   * **Reason:** The request originates from the same domain (CORS settings allow it).
2. **Bucket 2 (corsbkt2)**
   * Stores only **puppy.jpg**.
   * The browser **fails** to load **puppy.jpg** from **corsbkt2**.
   * **Reason:** CORS policy blocks the request because **corsbkt2 does not have proper CORS settings**.

**3. CORS Configuration in S3**

To allow cross-origin access, an S3 bucket must have a **CORS policy** in JSON format. Example:

[

{

"AllowedOrigins": ["\*"],

"AllowedMethods": ["GET", "HEAD"],

"AllowedHeaders": ["\*"],

"ExposeHeaders": []

}

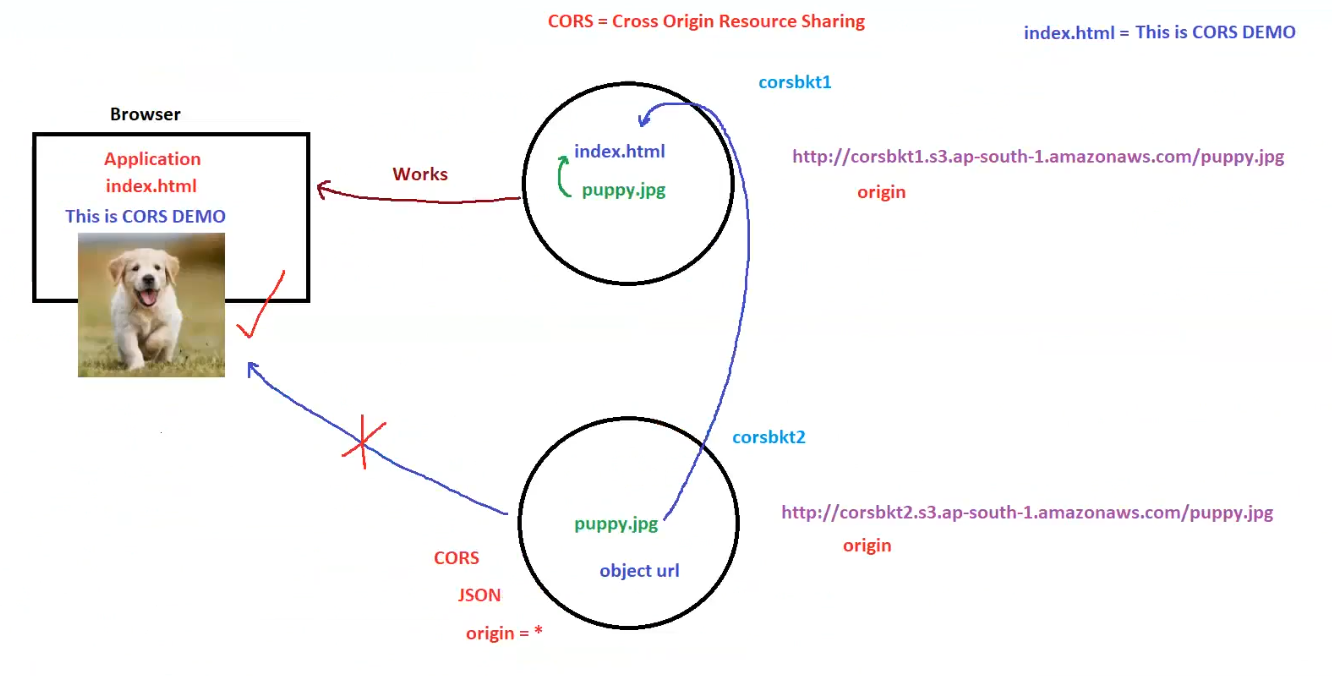
]

**Explanation of CORS Policy**

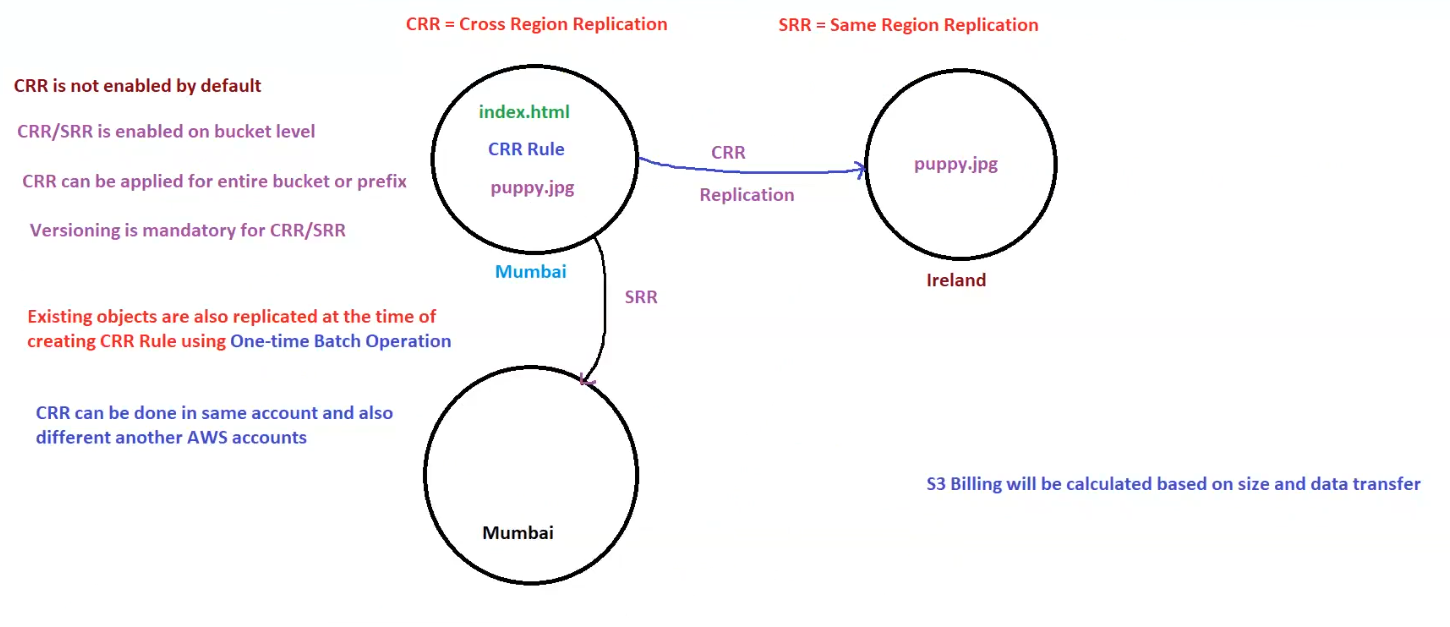
| **Parameter** | **Description** |
| --- | --- |
| "AllowedOrigins": ["\*"] | Allows requests from any domain. |
| "AllowedMethods": ["GET", "HEAD"] | Permits only **GET and HEAD** requests. |
| "AllowedHeaders": ["\*"] | Allows all request headers. |
| "ExposeHeaders": [] | No additional headers exposed in response. |

**4. Key Takeaways**

✅ **CORS is required when accessing S3 objects from a different domain.**  
✅ **Without a proper CORS policy, browsers block cross-origin requests.**  
✅ **S3 buckets must explicitly allow CORS using JSON configuration.**  
✅ **CORS applies at the bucket level, affecting all objects within the bucket.**



## S3 Replication: CRR & SRR

**AWS S3 Replication: CRR & SRR**

**1. What is S3 Replication?**

* **Replication allows automatic copying of objects from one bucket to another.**
* Two types of replication:
  1. **CRR (Cross-Region Replication)** → Replicates objects across different AWS regions.
  2. **SRR (Same-Region Replication)** → Replicates objects within the same AWS region.
* **Replication is NOT enabled by default.**
* **Versioning is mandatory** for both **CRR and SRR**.

**2. Key Features of S3 Replication**

* **Enabled at the bucket level.**
* **Can be applied to the entire bucket or specific prefixes (subfolders).**
* **Supports one-time batch operations** for existing objects.
* **CRR can be done within the same AWS account or across different AWS accounts.**
* **Billing is calculated based on data size and transfer costs.**

**3. How S3 Replication Works**

**Cross-Region Replication (CRR)**

* Source bucket: **Mumbai**
* Destination bucket: **Ireland**
* Objects (e.g., puppy.jpg) are replicated to a different AWS region.

**Same-Region Replication (SRR)**

* Source bucket: **Mumbai**
* Destination bucket: **Mumbai (another bucket)**
* Objects are replicated within the same region.

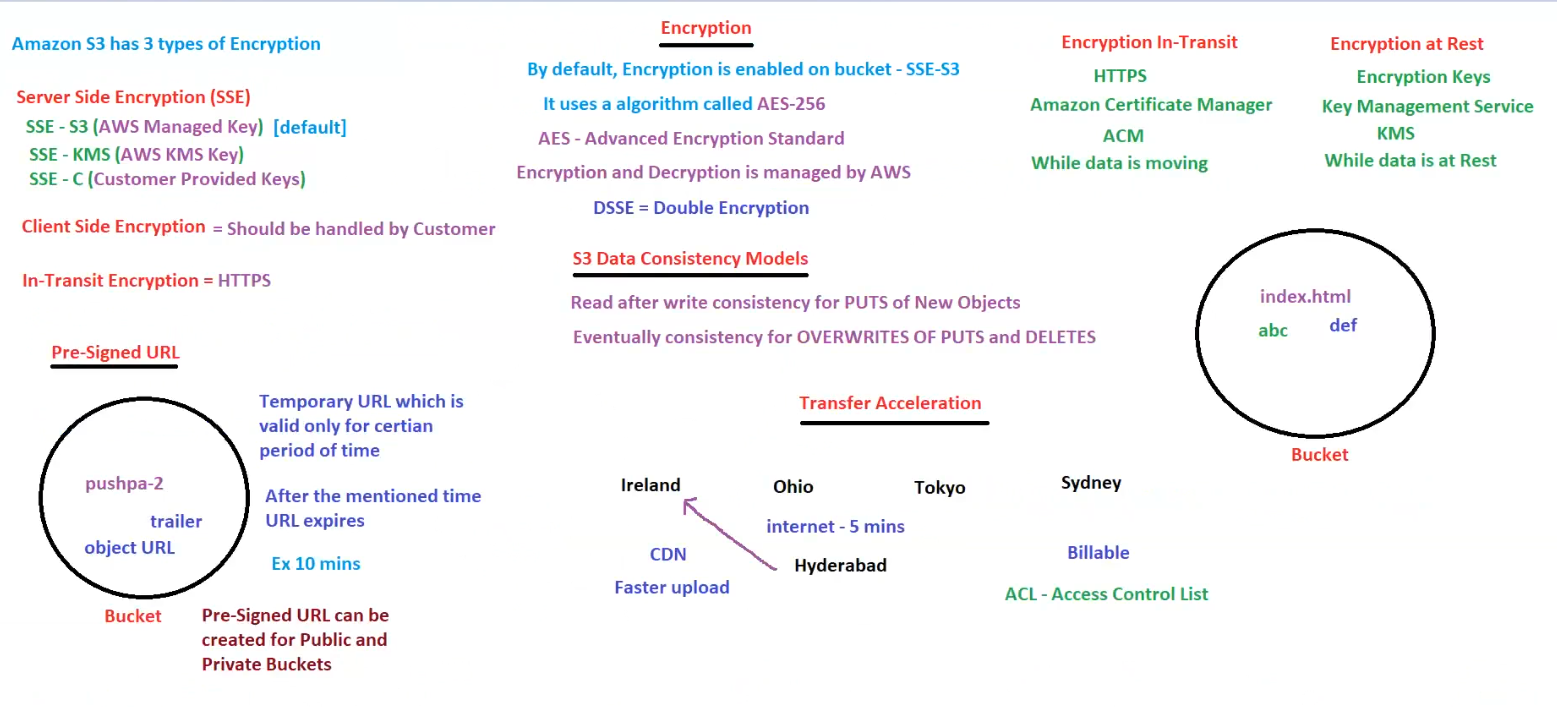
**4. Key Differences Between CRR & SRR**

| **Feature** | **Cross-Region Replication (CRR)** | **Same-Region Replication (SRR)** |
| --- | --- | --- |
| **Purpose** | Disaster recovery, compliance | Data duplication, latency reduction |
| **Regions Involved** | Different AWS regions | Same AWS region |
| **Latency** | Higher due to region transfer | Lower since within the same region |
| **Use Cases** | Multi-region backups, compliance, low-latency global access | Backup, log aggregation, same-region redundancy |
| **Billing** | Additional inter-region transfer costs | No inter-region costs, only storage costs |

**5. Summary of Key Takeaways**

✅ **CRR helps in disaster recovery and compliance by replicating across regions.**  
✅ **SRR is useful for same-region backups and data synchronization.**  
✅ **Versioning is required for both CRR and SRR.**  
✅ **Replication can be enabled for the entire bucket or specific prefixes.**  
✅ **Replication involves additional S3 billing for data transfer and storage.**

## S3 Encryption



**AWS S3 Security & Performance Features**

**1. S3 Encryption**

Amazon S3 supports **three types of encryption**:

**a) Server-Side Encryption (SSE)**

Encryption is managed by AWS at the bucket level:

* **SSE-S3** → Uses AWS-managed encryption keys (**default**).
* **SSE-KMS** → Uses AWS Key Management Service (KMS) for additional control.
* **SSE-C** → Uses customer-provided encryption keys.

**b) Client-Side Encryption**

* The **customer handles encryption** before uploading to S3.
* Requires **manual key management**.

**c) In-Transit Encryption**

* Protects data **while moving** between the client and S3.
* Uses **HTTPS (TLS)** for secure transmission.
* Managed via **Amazon Certificate Manager (ACM)**.

**d) Encryption at Rest**

* Ensures data is encrypted **when stored** in S3.
* Uses **AWS KMS (Key Management Service)**.
* Protects against unauthorized access.

**2. S3 Data Consistency Models**

| **Operation** | **Consistency Model** |
| --- | --- |
| **PUT (new objects)** | **Read-after-write consistency** |
| **PUT (overwrite existing objects)** | **Eventual consistency** |
| **DELETE** | **Eventual consistency** |

* **Read-after-write consistency**: A newly added object is **immediately available**.
* **Eventual consistency**: Updates or deletions **may take time** to propagate.

**3. Pre-Signed URLs**

* **Temporary URLs** that allow secure access to private objects.
* **Valid only for a specific time period** (e.g., 10 minutes).
* **Can be created for both public and private S3 buckets**.
* Example use case: **Providing temporary access to a video trailer** stored in S3.

**4. S3 Transfer Acceleration**

* **Speeds up file uploads to S3** by using AWS edge locations (**CloudFront CDN**).
* **Faster uploads from different geographical locations** (e.g., Hyderabad to Ireland).
* Uses AWS **global network** instead of the public internet.
* **Billing applies based on data transfer usage**.

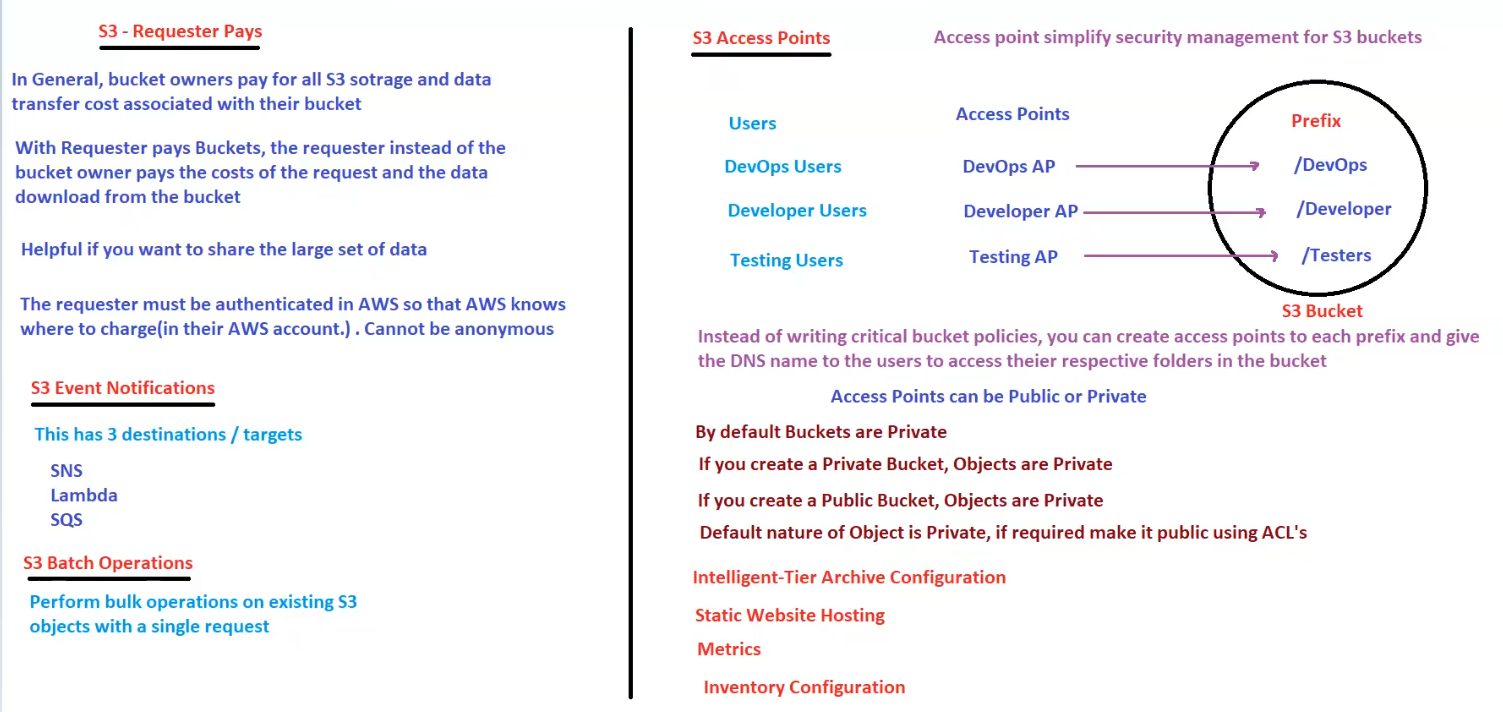
**5. Additional Features**

| **Feature** | **Description** |
| --- | --- |
| **ACL (Access Control List)** | Manages object-level permissions in S3. |
| **DSSE (Double Encryption)** | Provides additional security by applying two layers of encryption. |

**6. Key Takeaways**

✅ **S3 supports multiple encryption methods to secure data at rest and in transit.**  
✅ **S3 consistency models ensure data integrity but may cause delays for updates and deletes.**  
✅ **Pre-Signed URLs provide controlled access to private S3 objects for a limited time.**  
✅ **Transfer Acceleration improves upload speeds using AWS global infrastructure.**

## S3 Advanced Features



**AWS S3 Advanced Features**

**1. S3 Requester Pays**

* **By default, bucket owners pay for all storage and data transfer costs**.
* With **Requester Pays Buckets**, the **requester (not the bucket owner) pays** for:
  + Data transfer costs.
  + API request costs.
* **Useful for sharing large datasets** where multiple users download files.
* **Requirements:**
  + The requester **must be authenticated in AWS**.
  + **Anonymous requests are not allowed** (AWS needs to know whom to charge).

**2. S3 Event Notifications**

* **Triggers notifications when specific events occur in an S3 bucket**.
* Supports **3 destinations (targets)**:
  + **SNS (Simple Notification Service)** → Sends messages to subscribers.
  + **Lambda** → Executes serverless functions automatically.
  + **SQS (Simple Queue Service)** → Stores events in a queue for processing.

**3. S3 Batch Operations**

* **Used for performing bulk operations on existing S3 objects with a single request**.
* Examples:
  + **Copying large sets of objects**.
  + **Updating metadata**.
  + **Running AWS Lambda functions on objects**.

**4. S3 Access Points**

* **Simplifies security management** for S3 buckets by allowing multiple entry points.
* Instead of managing complex bucket policies, **access points provide individual access** for different users.
* **Example:**
  + **DevOps Users** → /DevOps Prefix
  + **Developers** → /Developer Prefix
  + **Testing Users** → /Testers Prefix
* **Access points can be Public or Private.**
* **Default behavior:**
  + **Buckets are Private by default**.
  + If a **bucket is private, objects remain private**.
  + If a **bucket is public, objects remain private** unless ACLs make them public.

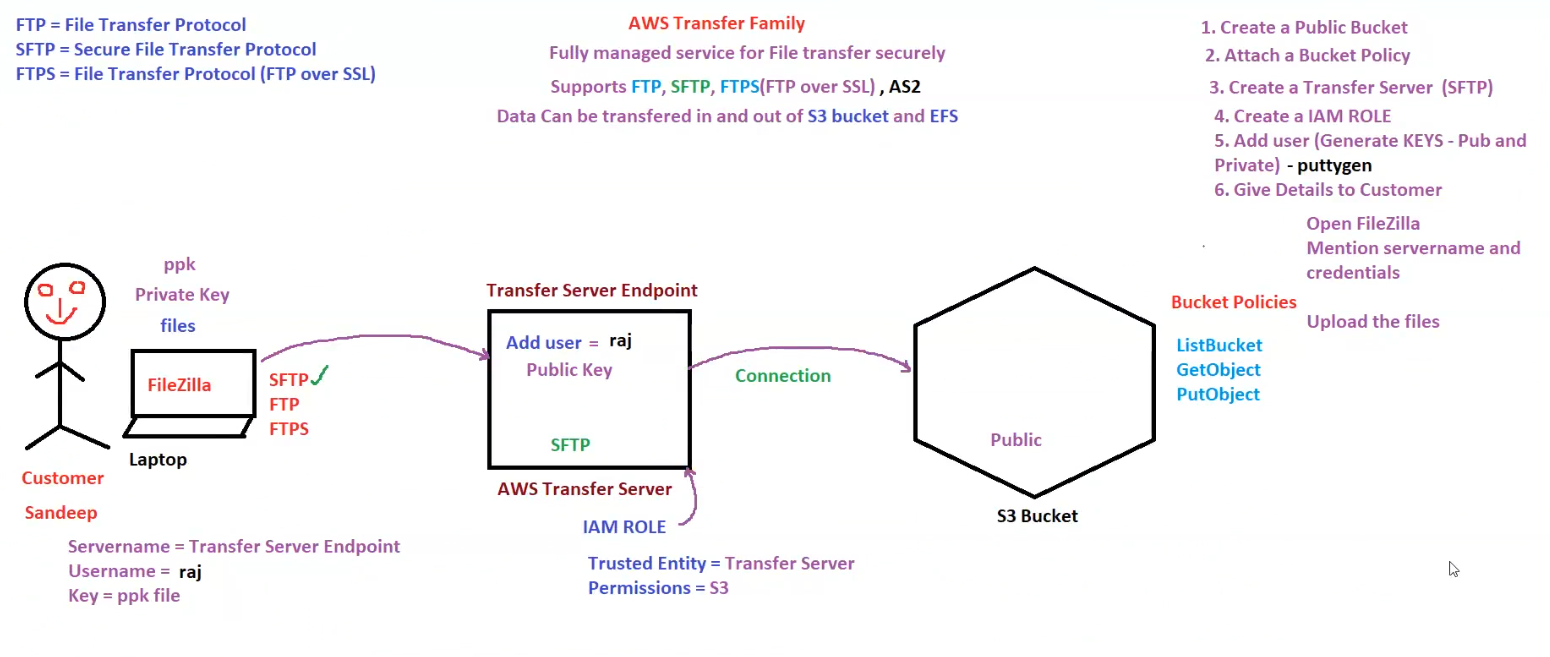
**5. Additional S3 Features**

* **Intelligent-Tier Archive Configuration** → Automates object transitions to lower-cost storage tiers.
* **Static Website Hosting** → Allows hosting websites directly from an S3 bucket.
* **Metrics** → Provides performance and usage statistics for monitoring.
* **Inventory Configuration** → Enables tracking and auditing of objects in a bucket.

**6. Summary of Key Takeaways**

✅ **Requester Pays reduces storage costs for bucket owners by shifting costs to requesters.**  
✅ **Event Notifications enable automation through SNS, Lambda, and SQS.**  
✅ **Batch Operations allow bulk processing of existing S3 objects.**  
✅ **Access Points simplify security by providing controlled access to different users.**  
✅ **By default, S3 buckets and objects are private unless explicitly made public via ACLs.**

## AWS Transfer Family



Here are structured notes based on the information in the image:

**AWS Transfer Family & S3 File Transfers**

**1. Overview of AWS Transfer Family**

* **AWS Transfer Family** is a **fully managed file transfer service** that supports:
  + **FTP (File Transfer Protocol)**
  + **SFTP (Secure File Transfer Protocol)**
  + **FTPS (FTP over SSL)**
  + **AS2 (Applicability Statement 2)**
* **Use Case:** Securely transfer files **to and from AWS S3 or EFS**.

**2. Steps to Set Up AWS Transfer for SFTP with S3**

1. **Create a Public S3 Bucket** → Where files will be stored.
2. **Attach a Bucket Policy** → Define permissions for file access.
3. **Create a Transfer Server (SFTP)** → Enables secure file transfer.
4. **Create an IAM Role** → Assign permissions to interact with the S3 bucket.
5. **Add a User (Generate Key Pair using Puttygen)**:
   * Public key is stored on AWS Transfer Server.
   * Private key (.ppk file) is given to the customer.
6. **Provide Connection Details to the Customer**:
   * Use **FileZilla** or other SFTP clients.
   * Enter **Server Name, Username (raj), and Private Key**.
   * Upload/download files securely.

**3. Components of AWS Transfer Family**

| **Component** | **Description** |
| --- | --- |
| **Transfer Server Endpoint** | The SFTP server address used to connect. |
| **IAM Role** | Provides necessary permissions to interact with S3. |
| **Trusted Entity** | AWS Transfer Server, which acts on behalf of the user. |
| **Bucket Policies** | Define access permissions for users. |

**4. S3 Bucket Policy for File Transfers**

Permissions required for AWS Transfer Family users:

* **ListBucket** → Allows users to list objects in the bucket.
* **GetObject** → Allows users to download files from S3.
* **PutObject** → Allows users to upload files to S3.

**5. Summary of Key Takeaways**

✅ **AWS Transfer Family supports secure file transfer to S3 via SFTP, FTP, and FTPS.**  
✅ **Public and private key authentication is used for secure access.**  
✅ **IAM Roles & Bucket Policies define access permissions for file transfers.**  
✅ **FileZilla or similar SFTP clients can be used to upload/download files.**