🗂️ AWS S3: Advanced Concepts — SAA-C03 Exam Notes

1️⃣ Moving Between Storage Classes

Overview: Amazon S3 offers several storage classes for different use cases and cost optimization. You can **manually** or **automatically transition** objects between these classes to reduce costs based on access patterns.

2️⃣ What Are Lifecycle Rules?

Lifecycle Rules are used to **automate transitions and expiration** of objects in an S3 bucket based on a defined policy.

They consist of two major **actions**:

🌀 A) Transition Actions

These define when an object should be moved to another **storage class**.

Supported Transitions:

From	To	Use Case
STANDARD	STANDARD_IA	Infrequently accessed data (after 30 days)
STANDARD_IA	GLACIER_IR / GLACIER	Long-term archival
ANY	INTELLIGENT_TIERING	Uncertain or unpredictable access patterns
STANDARD	ONEZONE_IA	Infrequent, but not critical data (single AZ)

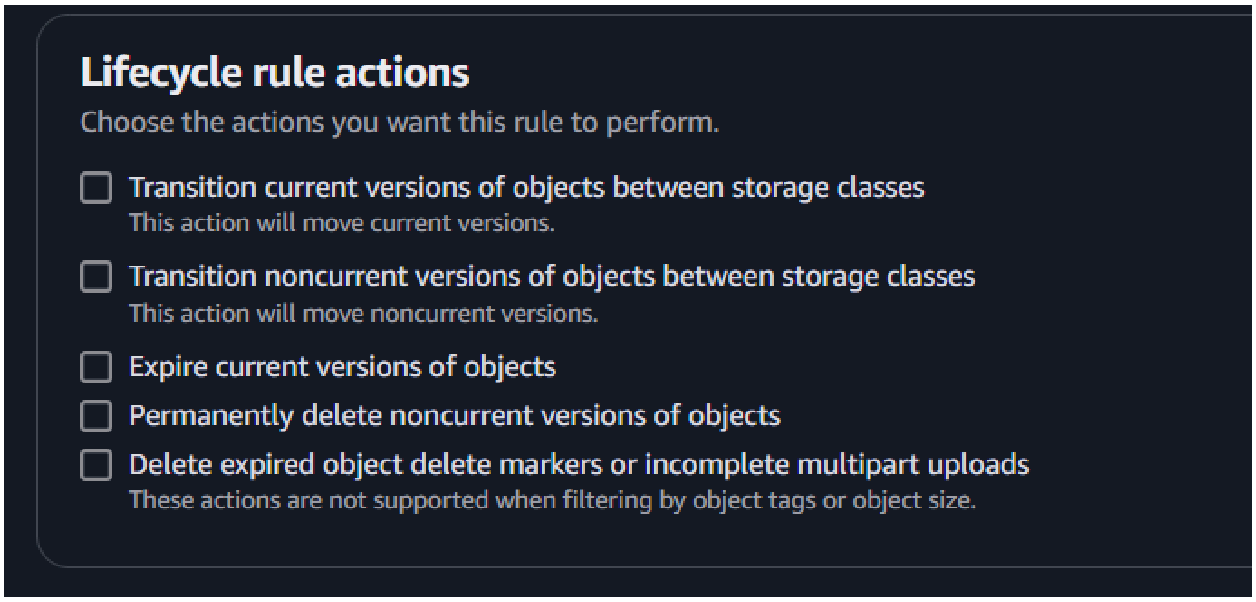
⛔ B) Expiration Actions

These specify when objects should be **deleted** automatically.

- Use Case: Delete logs, temporary files, or backups after a set period.
- Can apply to:
  - Current versions
  - Previous versions (in versioned buckets)
  - Incomplete multipart uploads (to save costs)

✅ AWS S3 Lifecycle Rule Actions (Official Categories)

Lifecycle rules in S3 define actions that Amazon S3 applies **automatically** to objects **based on age** or other conditions. There are **four official categories of actions** under S3 Lifecycle configuration:



Lifecycle rule actions

Choose the actions you want this rule to perform.

- ☐ Transition current versions of objects between storage classes
  This action will move current versions.
- ☐ Transition noncurrent versions of objects between storage classes
  This action will move noncurrent versions.
- ☐ Expire current versions of objects
- ☐ Permanently delete noncurrent versions of objects
- ☐ Delete expired object delete markers or incomplete multipart uploads
  These actions are not supported when filtering by object tags or object size.

1️⃣Transition

Automatically transition objects to another **storage class** (e.g., from STANDARD to STANDARD_IA, GLACIER, etc.).

Supported transitions:

- To **STANDARD_IA**
- To **ONEZONE_IA**
- To **INTELLIGENT_TIERING**
- To **GLACIER_IR** (Glacier Instant Retrieval)
- To **GLACIER** (formerly called Glacier – now known as **S3 Glacier Flexible Retrieval**)
- To **DEEP_ARCHIVE**

2️⃣Expiration

Permanently delete objects from the bucket after a specified number of days since creation.

💵 Amazon S3 Requester Pays (Explained)

🔹 What Is It?

By default, **the bucket owner pays** for all data transfer and request costs in Amazon S3 — even when others access/download the data.

With **Requester Pays**, you flip that model:
➡️ **The person downloading the data (the requester) pays** for the data transfer and requests, **not the bucket owner**.

🔹 When to Use It?

Useful when you're hosting **public datasets or shared content** and don't want to bear the download costs for all users.

🔹 Simple Real-Life Example:

Imagine you're a government agency that stores **public weather data** in an S3 bucket.

- Bucket: `s3://gov-weather-data`
- Data: Huge files (hundreds of GBs) updated daily
- Access: Public

🔸 Without **Requester Pays**:
Anyone can download the files, and **you (the bucket owner)** pay for **all bandwidth and requests**.

🔸 With **Requester Pays** enabled:
If someone (e.g., a company or researcher) downloads the files:

- They pay for **GET requests** and **data transfer**
- You only pay for **storage**

🔹 Important Notes for Exam:

- **Requester must use a signed request** (i.e., must authenticate) — even for public buckets.
- **Requester Pays is set at the bucket level**.

📣 AWS S3 Notifications & Events

🔹 What Are S3 Notifications?

S3 Notifications allow Amazon S3 to **automatically send event information** when specific actions occur in a bucket. These notifications help trigger **automated responses** or **further processing**.

They are configured per bucket and define:

- **Which events to monitor**
- **Optional filters** (prefix/suffix)
- **Where to send the event information**

🔹 What Are S3 Events?

S3 Events are **triggers based on object-level operations**. These occur when:

- An object is **created**, **deleted**, **replicated**, **archived**, or **restored**

Common S3 Events:

Event Name	Description
`s3:ObjectCreated:*`	Any object is created
`s3:ObjectCreated:Put`	Uploaded via `PUT`
`s3:ObjectCreated:CompleteMultipartUpload`	Multipart upload completed
`s3:ObjectRemoved:*`	Any object is deleted
`s3:ObjectRemoved:Delete`	Deleted via `DELETE` request
`s3:Replication:OperationFailedReplication`	Replication failed
`s3:ObjectRestore:Completed`	Glacier/Deep Archive restore completed

📤 Where Can S3 Send These Notifications?

You can configure S3 to send event notifications to **three types of AWS destinations**:

Destination	Use Case Example
Amazon SNS	Notify multiple subscribers (fan-out architecture)
Amazon SQS	Queue for reliable background processing or batching
AWS Lambda	Trigger custom code in real-time (e.g., image resizing)

🛠️ Key Configuration Points (for the exam):

- 🔧 You can **filter events** using object key **prefixes/suffixes**.
  - Example: Trigger only when `.jpg` files are uploaded under `images/`
- 🔐 You must grant **S3 permissions to publish** to SNS/SQS or invoke Lambda.
- ⚠️ **Only one destination per event type** can be configured **using the S3 console** (for basic setup).
- 🧩 Advanced multi-destination setups require **EventBridge or Lambda fan-out logic**.

🚀1️⃣ S3 Baseline Performance

Amazon S3 is designed for massive scalability with high throughput.

✅ **Default Performance Limits (per prefix):**

Operation Type	Throughput (per prefix)
PUT/COPY/POST/DELETE	Up to **3,500 requests per second**
GET/HEAD	Up to **5,500 requests per second**

- S3 automatically scales **horizontally** and handles increased load over time.
- If your workload needs **more throughput**, **use multiple prefixes** (e.g., `folder1/`, `folder2/`) to parallelize traffic.

🧩2️⃣ S3 Multipart Upload

📌 What is it?

Multipart Upload lets you upload a single large object as **multiple parts in parallel**, which:

- Speeds up uploads of large files (>100MB recommended, required for >5GB)
- Allows **resumable uploads** if a part fails

✅ Key Facts:

- **Minimum part size**: 5 MB (except last part)
- **Maximum parts**: 10,000
- Required for uploading files **>5 GB** (mandatory)
- Supports **parallelism** — you can upload different parts simultaneously
- After upload, use `CompleteMultipartUpload` to assemble the object

🌐3️⃣ S3 Transfer Acceleration
📌 What is it?
S3 Transfer Acceleration (TA) uses Amazon CloudFront's globally distributed edge locations to accelerate uploads and downloads.

Instead of uploading directly to S3, users upload to the nearest AWS edge location, which then routes the data to the S3 bucket over AWS's high-speed internal network.

📦4️⃣ S3 Byte-Range Fetches

📌 What is it?

Byte-Range Fetches allow clients to **download specific byte ranges** of an object. Useful for:

- **Resuming interrupted downloads**
- **Parallel downloads** (multi-threading large file downloads)
- Fetching metadata or partial contents (e.g., video streaming previews)

✅ Example Use Case:

- Large file (10GB log)
- Instead of downloading the whole file, you fetch bytes 0–10MB, 10MB–20MB, etc., concurrently

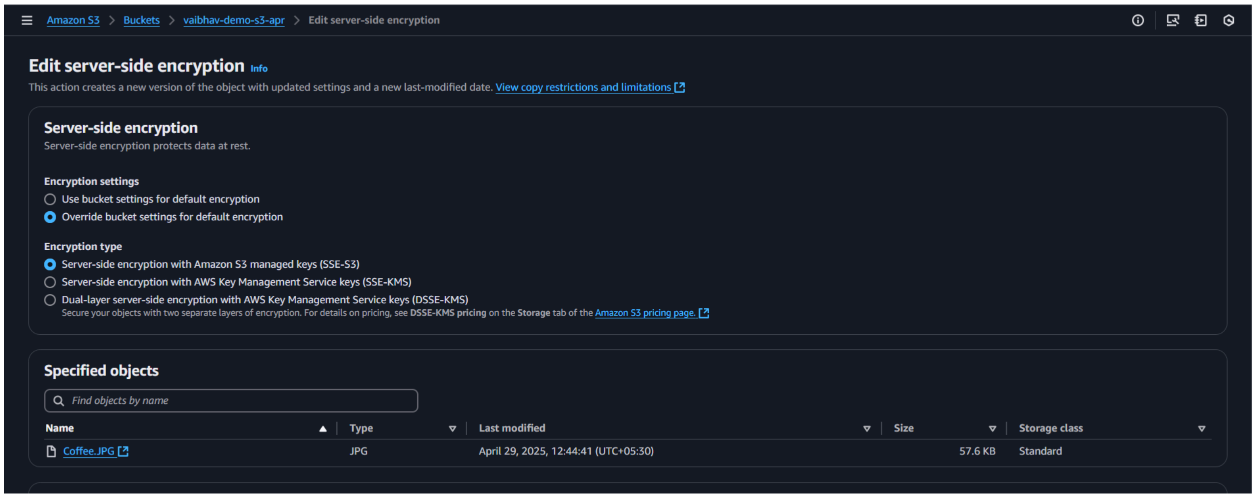
✅ AWS S3 Security Concepts

🔐 S3 Object Encryption (At-Rest)

AWS offers **4 methods** for encrypting S3 objects at rest:

1. SSE-S3 (Server-Side Encryption with Amazon S3-Managed Keys)

- **Encryption Type**: AES-256
- This is the default encryption applied on all objects of S3 if you didn't specify any.
- **Key Management**: Handled entirely by AWS
- **Setup**: No customer action needed (can be set as default bucket encryption)
- **Protocol**: HTTPS for secure transfer



Amazon S3 > Buckets > vaibhav-demo-s3-apr > Edit server-side encryption

Edit server-side encryption Info

This action creates a new version of the object with updated settings and a new last-modified date. View copy restrictions and limitations

Server-side encryption

Server-side encryption protects data at rest.

Encryption settings

- ○ Use bucket settings for default encryption
- ● Override bucket settings for default encryption

Encryption type

- ● Server-side encryption with Amazon S3 managed keys (SSE-S3)
- ○ Server-side encryption with AWS Key Management Service keys (SSE-KMS)
- ○ Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
  Secure your objects with two separate layers of encryption. For details on pricing, see **DSSE-KMS pricing** on the **Storage** tab of the Amazon S3 pricing page.

Specified objects

Name	Type	Last modified	Size	Storage class
Coffee.JPG	JPG	April 29, 2025, 12:44:41 (UTC+05:30)	57.6 KB	Standard

This action creates a new version of the object with updated settings and a new last-modified date

- **Limits**:
  - No key rotation control
  - Least granular permission

2. SSE-KMS (Server-Side Encryption with AWS Key Management Service)

- **Encryption Type**: Envelope encryption using AES-256; master key in AWS KMS
- **Key Management**: Customer-managed via AWS KMS
- **Setup**: Specify KMS key (CMK - Customer Managed Key or AWS Managed Key)
- **Protocol**: HTTPS + AWS KMS APIs

- **Limits**:

    - KMS **request quota limits** (e.g., ~5,500 requests/sec per CMK)
    - Additional **cost for KMS usage**
    - Allows fine-grained access control via IAM/KMS policies

3. SSE-C (Server-Side Encryption with Customer-Provided Keys)

- **Encryption Type**: AES-256
- **Key Management**: Customer provides and manages keys
- **Setup**: Provide encryption key in each PUT/GET request header
- **Protocol**: Must use HTTPS (key sent in headers)
- **Limits**:

    - AWS does **not store** the key – user must handle key durability
    - Not supported via AWS SDK for some services (e.g., Glacier)
    - Can only be done through CLI.

4. CSE (Client-Side Encryption)

- **Encryption Type**: Customer encrypts data *before* uploading
- **Key Management**: Fully controlled by customer
- **Setup**:
    - Use AWS SDK with a KMS key or custom encryption libraries
- **Protocol**: HTTPS (recommended) + local encryption

- **Limits**:

    - Full responsibility on customer to handle key rotation, key loss, etc.
    - No server-side support (S3 stores already-encrypted blob)

🔐 Encryption in Transit

- Protects **data in motion** using **SSL/TLS (HTTPS)**
- AWS S3 supports **TLS 1.2+**
- **Best Practice**: Always use HTTPS to access S3 buckets

🚫 Forced Encryption in Transit

- **Goal**: Enforce use of HTTPS only
- **Implementation Methods**:

Bucket Policy Condition:

```
{
  "Condition": {
    "Bool": {
      "aws:SecureTransport": "false"
    }
  },
  "Effect": "Deny",
  "Principal": "*",
  "Action": "s3:*",
  "Resource": ["arn:aws:s3:::your-bucket-name/*"]
}
```

1. **CloudFront** in front of S3:
   - Redirect all HTTP to HTTPS
   - Restrict direct S3 access

🌐 S3 CORS (Cross-Origin Resource Sharing)

❓ What is S3 CORS?

CORS (Cross-Origin Resource Sharing) is a **browser security feature** that controls how web applications running at one origin (domain) can access resources from a **different origin**.

In the context of **Amazon S3**, **CORS is used to enable cross-origin requests** to your S3 bucket — for example, allowing a web application hosted on `https://example.com` to access images or files stored in your S3 bucket.

📘 Why Use CORS in S3?

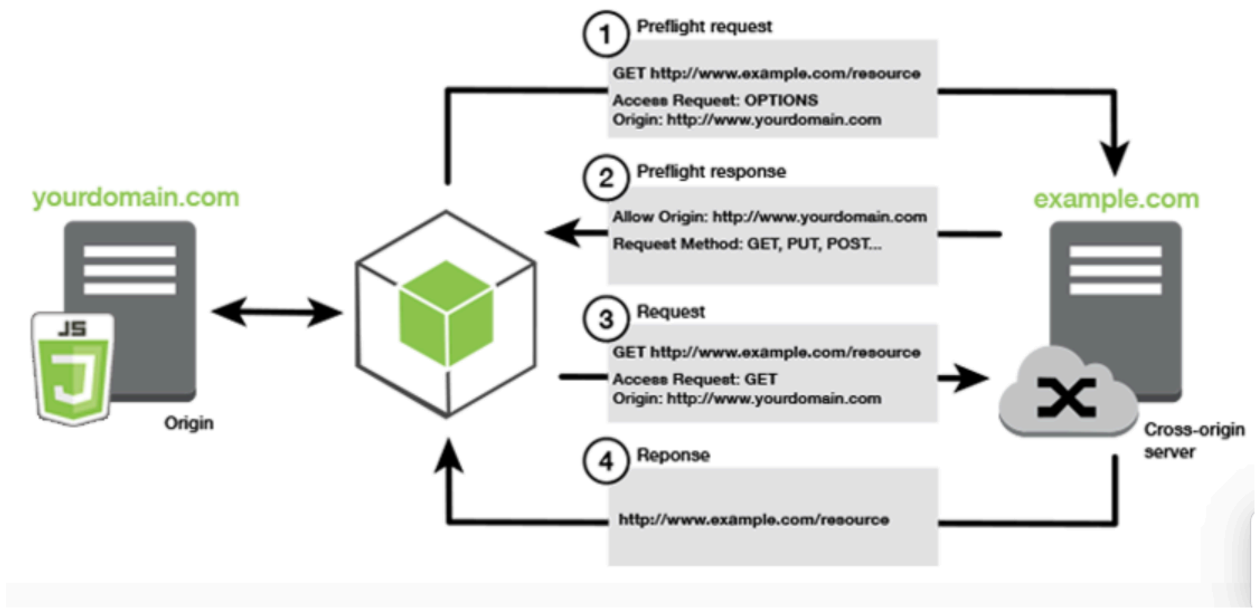
- Allow web browsers to securely fetch content (e.g., images, fonts, videos) hosted in S3 from a different domain.
- Common in scenarios like:
  - Hosting frontend on one domain and assets in S3
  - Allowing AJAX/JavaScript to call S3 directly

🔧 CORS Configuration (in Bucket Settings)

You configure CORS by adding a **CORS rule** in the S3 bucket settings.

🔒 Key Points :

- CORS is only **relevant for browsers** (client-side web apps).
- Required if the S3 bucket is **accessed directly from a browser script** in a different origin.
- You must explicitly define **AllowedOrigins**, **AllowedMethods**, and optionally **AllowedHeaders** and **ExposeHeaders**.
- If CORS is not set and a cross-origin request is made, the **browser will block** the request (S3 itself does not block it, the browser does).



1 Preflight request
GET http://www.example.com/resource
Access Request: OPTIONS
Origin: http://www.yourdomain.com
2 Preflight response
Allow Origin: http://www.yourdomain.com
Request Method: GET, PUT, POST...
3 Request
GET http://www.example.com/resource
Access Request: GET
Origin: http://www.yourdomain.com
4 Reponse
http://www.example.com/resource
yourdomain.com
JS
Origin
example.com
Cross-origin server

🔐 S3 MFA Delete

❓ What is MFA Delete?

MFA Delete is a **security feature** for **versioned S3 buckets** that **requires multi-factor authentication (MFA)** for specific high-risk operations, such as:

- **Deleting object versions**
- **Disabling versioning**

⚠️ Important Limitations

- **MFA Delete can only be enabled using the AWS CLI or SDK — not via the AWS Management Console.**
- **You must be the root user to enable or disable MFA Delete.**
- **It only works with versioning-enabled buckets.**
- **Not supported with IAM users — root credentials are required.**

🔑 Key Features

- Adds an extra layer of protection to prevent **accidental or malicious deletions**.
- Requires an MFA device (e.g., virtual or hardware MFA).

📄 AWS S3 Access Logs

❓ What are S3 Access Logs?

S3 Access Logs provide **detailed records** about requests made to your S3 bucket. These logs can help with:

- Auditing access
- Troubleshooting permissions
- Analyzing usage patterns
- Meeting compliance requirements

🛠️ Key Features

- Logs are delivered in the **Common Log Format**.
- Each log record includes:
  - Requester
  - Bucket name
  - Request time
  - Action (GET, PUT, DELETE, etc.)
  - Response status
  - Error code (if any)
  - Bytes transferred
- Delivered to a **target S3 bucket** (you must specify it)

⚠️ S3 Access Logs – Warning & Best Practices

⚠️ Access Logs Can Generate Infinite Loops

If you **send access logs to the same bucket being logged**, every log delivery becomes a **new write request**, which then gets logged — leading to:

- **Log recursion**
- **Unexpected high costs**
- **Potential storage explosion**

\

🔗 Amazon S3 Pre-Signed URL

❓ What is a Pre-Signed URL?

A **Pre-Signed URL** is a **time-limited, secure URL** that grants **temporary access** to an object in an S3 bucket without making it public.

Used when you want to:

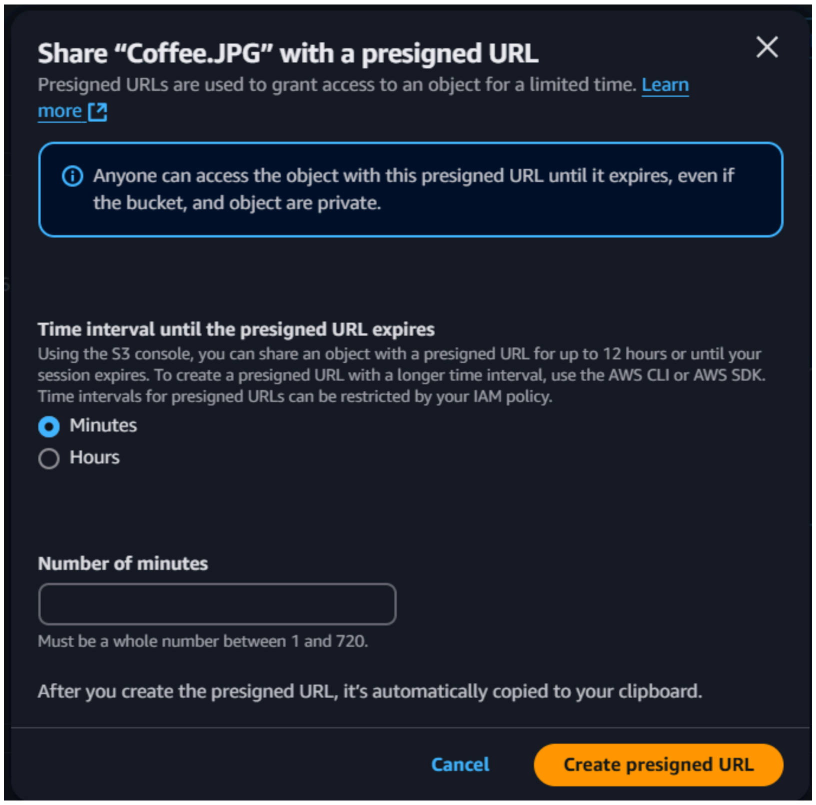
- Share private files securely
- Allow uploads/downloads without giving full S3 permissions

🛠️ How It Works

- A pre-signed URL is generated using **AWS credentials** (IAM user, assumed role, or application).
- It includes:
  - The S3 object key
  - Expiration time
  - Signature (authenticating the request)
- Can be used for:
  - **GET** requests (downloads)
  - **PUT** requests (uploads)

✅ Common Use Cases

- Letting users **download** private content (e.g., PDFs, images)
- Allowing clients to **upload** directly to S3 without needing IAM credentials
- Temporary access for limited-time events, downloads, etc.



Share "Coffee.JPG" with a presigned URL

Presigned URLs are used to grant access to an object for a limited time. Learn more

Anyone can access the object with this presigned URL until it expires, even if the bucket, and object are private.

Time interval until the presigned URL expires

Using the S3 console, you can share an object with a presigned URL for up to 12 hours or until your session expires. To create a presigned URL with a longer time interval, use the AWS CLI or AWS SDK. Time intervals for presigned URLs can be restricted by your IAM policy.

[✓ Minutes] Hours

Number of minutes

Must be a whole number between 1 and 720.

After you create the presigned URL, it's automatically copied to your clipboard.

🧊 Amazon S3 Glacier Vault Lock

❓ What is Vault Lock?

S3 Glacier Vault Lock allows you to **enforce compliance controls** on a vault, such as **WORM (Write Once, Read Many)** policies, via a **vault access policy** that becomes **immutable** after it's locked.

🔐 Key Features

- Enables **compliance retention** for archives.
- Once the Vault Lock policy is set and **locked**, it **cannot be changed**.

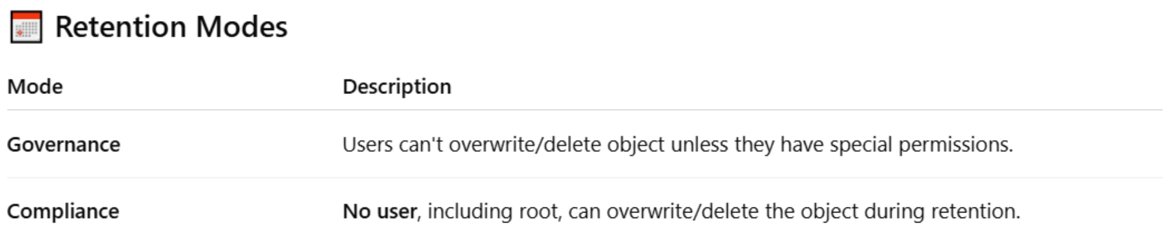
📦 Amazon S3 Object Lock

❓ What is S3 Object Lock?

S3 Object Lock prevents object versions from being **deleted or overwritten** for a specified period or indefinitely. It's used for **WORM** storage in **S3 Standard**, **S3 IA**, and **S3 Glacier**.

🔐 Key Concepts

- Requires **versioning to be enabled** on the bucket.
- Retention and protection apply **per object version**.



📅 Retention Modes

Mode	Description
Governance	Users can't overwrite/delete object unless they have special permissions.
Compliance	**No user**, including root, can overwrite/delete the object during retention.

⏳ Retention Period

- Defined in **days** or **timestamp**.
- Applies only to the **specific version** of the object.
- After the retention period ends, the object becomes modifiable/deletable.

🏷️ Legal Hold

- Acts like a "pause" on deletion, **regardless of retention period**.
- Can be applied or removed **independently of retention settings**.
- Does **not expire automatically** — must be explicitly removed.

✅ Use Cases

- Financial records retention
- Regulatory compliance (e.g., SEC, FINRA, GDPR)
- Legal evidence preservation.

🚪 Amazon S3 Access Points

❓ What Are S3 Access Points?

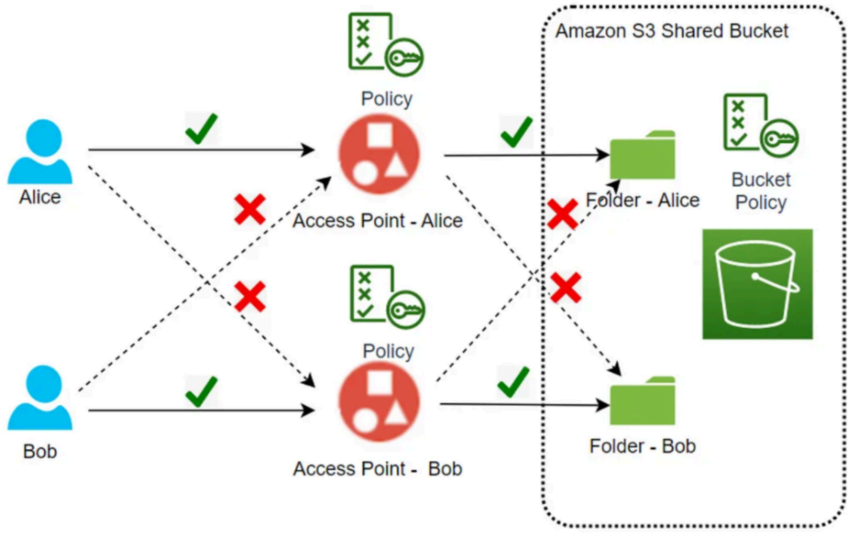
S3 Access Points are named network endpoints that simplify **managing access** to shared S3 buckets — especially in **large-scale, multi-application, or multi-team** environments.

Each access point has:

- A **unique name** within the AWS Region and account
- An **associated bucket**
- Its own **access policy**
- An optional **VPC configuration**

🔐 Security Benefits

- Reduces the complexity of managing **bucket-level policies** for multiple use cases.
- Allows the use of **VPC-only endpoints** to restrict S3 access to **private networks**.
- Helps enforce **least privilege access control**.



🧠 Amazon S3 Object Lambda

❓ What Is S3 Object Lambda?

S3 Object Lambda enables you to **transform data as it's retrieved from S3** using a custom **Lambda function** — *without modifying the underlying object in the bucket*.

You can dynamically:

- Redact sensitive data (e.g., PII)
- Reformat data (e.g., CSV → JSON)
- Filter content (e.g., return only certain rows)

✅ Common Use Cases

- **Redact sensitive information** from log files or documents
- **Dynamically watermark images**
- **Mask financial records** before exposing them externally
- **Custom filtering or transformations** for downstream systems