Lab 1 Report

Cost comparison notes for different storage classes

This note compares the costs and use cases of three Amazon S3 storage classes: Standard-IA, Glacier Flexible Retrieval, and Glacier Deep Archive.

S3 Standard-IA (Infrequent Access)

Storage cost: ~ $0.0125 per GB per month

Retrieval cost: ~ $0.01 per GB

Minimum duration: 30 days

Access speed: Milliseconds (immediate access)

Use case: Best for backups, disaster recovery, and infrequently accessed files that must still be retrieved instantly.

S3 Glacier Flexible Retrieval

Storage cost: ~ $0.004 per GB per month

Retrieval cost: $0.01 – $0.03 per GB (depending on retrieval tier)

Minimum duration: 90 days

Access speed:

Expedited: 1–5 minutes

Standard: 3–5 hours

Bulk: 5–12 hours

Use case: Good for data you rarely need, but may occasionally require within hours (e.g., compliance archives).

S3 Glacier Deep Archive

Storage cost: ~ $0.00099 per GB per month

Retrieval cost: $0.02 – $0.05 per GB

Minimum duration: 180 days

Access speed:

Standard: ~12 hours

Bulk: up to 48 hours

Use case: Best for long-term archival storage, like compliance data or historical records you may never retrieve but must retain for years.

Summary

Standard-IA: Low-cost with instant access.

Glacier Flexible Retrieval: Cheaper, slower retrieval (hours).

Glacier Deep Archive: Ultra-low-cost, very slow retrieval (12–48 hours).

Assessment Questions

LAB 1 ASSESSMENT

- When an object transitions from S3 Standard to S3 Standard-IA, storage costs drop considerably. S3 Standard is intended for frequently accessed data and carries a higher per-GB price, whereas Standard-IA is built for infrequently accessed data that still needs quick retrieval. The trade-off is that while storage in Standard-IA is cheaper, you incur retrieval fees whenever you access the objects.

- Objects cannot be moved directly from S3 Standard to Glacier Deep Archive with a lifecycle rule set to less than 180 days. AWS enforces minimum storage durations across infrequent access and archive classes. This means to transition an

object to Glacier Deep Archive, it must remain in its current storage class for at least 180 days from creation.

- Objects stored in Standard-IA must stay there for a minimum of 30 days. If an object is deleted or transitioned to another storage class before the 30-day period ends, AWS will still bill for the full 30 days of storage.

What worked as expected?

- Bucket creation and file uploads worked smoothly.
- Assigning different storage classes (Standard, Standard-IA, Intelligent-Tiering) was straightforward, and lifecycle rules applied correctly.
- Transitions to different classes and expiration settings appeared properly in the rule configuration.

What was challenging?

- Understanding cost trade-offs between storage classes required checking the AWS Pricing Calculator.
- Remembering the minimum duration requirement for IA classes (30 days) could cause lifecycle rule errors if not set properly.

How would you apply this in a real-world scenario?

- Lab 1: Lifecycle rules and storage classes are directly useful for cost optimization in production. For example, moving logs or backups from Standard - IA to Glacier after set periods ensures compliance while minimizing cost

Cost implications observed

- Standard storage is more expensive but has no retrieval cost.
- Standard-IA is cheaper for infrequently accessed data but incurs retrieval charges.
- Intelligent-Tiering has monitoring overhead but is best for unknown or unpredictable access patterns.
- Glacier and Deep Archive are the cheapest long-term storage but have high retrieval latency and costs.

LAB 2 REPORT

MFA Delete requirements and limitations

Requirements for MFA Delete

- S3 Versioning: The bucket must have versioning enabled, since MFA Delete relies on versioning to protect all object versions and delete markers.
- Root User Credentials: Only the AWS account root user can enable or disable MFA Delete. IAM users and roles, even with full admin rights, cannot configure it.
- MFA Device: The root user must have a registered and active MFA device. Both the device's ARN (serial number) and a valid MFA code are required to carry out protected actions.
- CLI or API Setup: MFA Delete cannot be managed through the AWS Management Console. It must be configured via the AWS CLI or Amazon S3 REST API.

Limitations of MFA Delete

- Root User Dependency: Only the root user can enable, disable, or perform protected actions with MFA Delete. This creates operational challenges, since relying on the root account goes against security best practices.

- Incompatibility with Lifecycle Policies: MFA Delete cannot be enabled on buckets that have lifecycle configurations. Because lifecycle rules are commonly used to manage object versions and control storage costs, you must remove them before turning on MFA Delete.

- No IAM User Support: MFA Delete is tied exclusively to the root account. Even IAM users with MFA enabled cannot permanently delete objects or suspend versioning in an MFA-protected bucket.

- No Console Configuration: MFA Delete cannot be enabled or disabled from the AWS Management Console. Setup and changes must be done via the AWS CLI or API, which may be inconvenient.

Assessment Questions

LAB 2 ASSESSMENT

- When you upload a new version of an object to an S3 bucket with versioning enabled, the previous version is not overwritten or removed. Instead, Amazon S3 marks the new file as the current version while keeping the older one as a non-current version. Each version has its own unique version ID, allowing you to retrieve, restore, or delete specific versions whenever needed.
- To permanently delete a particular version of an object, you must reference the object's key and version ID in your delete request. If you delete an object without specifying a version ID, S3 simply adds a delete marker to the current version. This makes the object appear deleted and inaccessible through normal requests, but all previous versions including the one with the delete marker remain in the bucket. To truly remove a version, you must explicitly target its version ID using the AWS CLI or API.
- Enabling MFA Delete is restricted to the AWS root user because it serves as a last line of defense against accidental or malicious data loss. MFA Delete locks down critical actions such as disabling versioning or permanently deleting object versions.

What worked as expected?

- Versioning behaved as expected: new uploads overwrote the "latest" view but preserved older versions.
- Delete markers worked as expected, hiding objects but keeping versions intact.
- Restoring by deleting the delete marker correctly brought objects back.

What was challenging?

- The delete marker behavior was slightly confusing at first, since the object seemed "gone" but was still retained.

- MFA Delete could not be tested because the sandbox environment restricted enabling MFA at the root user level. This made it impossible to fully validate that feature.

How would you apply this in a real-world scenario?

Versioning provides accidental deletion protection and supports compliance/audit requirements by maintaining object history. MFA Delete (though untested in this lab) would be applied in highly sensitive environments to prevent malicious or accidental permanent deletions.

Cost implications observed

- Versioning increases storage costs since all versions are retained until explicitly deleted. Storing multiple versions of large files can significantly increase costs.

- Delete markers themselves don’t add much cost, but keeping old versions indefinitely can.

LAB 3 REPORT

Replication Timing

- Initial Replication:
  After configuring and saving the replication rule, the first test objects (crr-test-1.txt and crr-test-2.txt) were uploaded to the source bucket. Replication was not immediate—objects appeared in the destination bucket after a delay of about 2–3 minutes. This is expected since S3 replication is asynchronous and eventually consistent.

- Subsequent Replication:
  The final test object (crr-test-3.txt) also replicated with a similar delay. The timing was consistent with the initial replication, confirming predictable replication behavior.

Replication Behavior and Key Differences

- Version IDs:
  Replicated objects in the destination bucket had different version IDs than their source counterparts. This is by design—S3 assigns each replicated object a new version ID within the destination bucket.

- Storage Class Transition:
  The replication rule correctly transitioned storage classes. Objects uploaded to the source bucket in Standard were replicated to the destination bucket in Standard-IA, as specified in the rule.

- Delete Marker Behavior:
  Delete markers were not replicated, as per the rule configuration.

    - Example: When crr-test-1.txt was deleted in the source bucket, a delete marker was created there.

    - In the destination bucket, the replicated copy of crr-test-1.txt remained intact and accessible.

- Object Modification:
  When crr-test-2.txt was modified and re-uploaded, S3 created a new version in the source bucket. This updated version was replicated to the destination bucket, where it also appeared as a new version. This confirms that replication applies not just to initial uploads, but also to subsequent modifications.

Assessment Questions

LAB 3 ASSESSMENT

To configure Cross-Region Replication (CRR), the following requirements must be met:

- S3 Versioning: Both the source and destination buckets must have versioning enabled. Replication relies on object versions rather than just the latest copy of an object.

- IAM Role: An IAM role must be created with a trust policy that allows Amazon S3 to assume it. The role's permissions must include:

    - s3:GetObject on the source bucket

    - s3:ReplicateObject and s3:ReplicateDelete on the destination bucket
      Without these permissions, replication will not succeed.

By default, delete markers are not replicated in Amazon S3. This is a security feature to prevent accidental deletion of replicated data. If an object is "soft-deleted" by adding a delete marker in the source bucket, the replicated object in the destination bucket remains untouched. To enable delete marker replication, you must explicitly configure the replication rule to do so. This is often done for compliance or to ensure that the source and destination buckets are kept in sync.

Yes, you can replicate a bucket in the same region. This feature is called Same-Region Replication (SRR). While CRR is for cross-regional disaster recovery and latency

reduction, SRR is used for purposes like log aggregation from multiple buckets, replicating data between development and test environments, or meeting specific data sovereignty requirements by keeping a replicated copy in a different account within the same region.

What worked as expected?

- Source and destination bucket creation with versioning enabled worked without issue.
- IAM role creation and assignment to replication rule succeeded.
- Replication worked after a short delay, with objects and versions appearing in the destination bucket as expected.

What was challenging ?

- Replication delays required patience; testing too quickly could make it seem like replication failed.
- Understanding that delete markers don’t replicate by default could cause confusion in real-world disaster recovery scenarios.

How would you apply this in a real-world scenario ?
Cross-region replication is critical for disaster recovery and regulatory compliance (e.g., HIPAA, GDPR). It ensures data is stored redundantly across geographically separate regions.

Cost implications observed

- Cross-region replication doubles storage cost since every object is duplicated in another region.

- Replication traffic between regions also incurs inter-region data transfer charges, which can add up at scale.

- Useful for mission-critical data, but cost-benefit analysis is required for less sensitive workloads.