### **AWS S3 Keywords**

* **Storage Cost:** 
  + This cost is based on the amount of data stored in S3. Storage classes like Standard incur higher costs because the data is stored across multiple availability zones with instant access and durability. Lower-cost classes, such as Glacier or Glacier Deep Archive, store data in an archive format, optimizing for minimal storage costs by trading off instant access.
* **PUT Request Cost:** 
  + Every time you upload or write data to S3, AWS uses resources like compute and networking to process the upload. Since this involves infrastructure costs, especially for high-throughput and low-latency operations in S3 Standard, AWS charges per 1,000 PUT operations.
* **GET Request Cost:** 
  + Fetching data from S3 also consumes AWS infrastructure resources, such as compute and bandwidth. S3 Standard GET requests are cheaper per 10,000 operations because data is instantly available, while archival storage (like Glacier) GETs are more expensive due to the processing required to restore archived data before serving it.
* **Data Transfer Out Cost:** 
  + This applies when data is downloaded from S3 to the internet (e.g., for external users). The charges account for bandwidth usage, which varies by region and the amount of data transferred. This cost is higher for archival classes because data restoration happens before transfer.
* **Retrieval Charges:** 
  + In storage classes like Glacier or Deep Archive, data is moved to an archive state to minimize storage costs. When you retrieve data, it is first restored to a "ready-to-access" state using AWS compute and resources. Retrieval charges reflect the effort of moving data out of the archive. Classes like Standard or Intelligent-Tiering have no retrieval charges because the data is always in a ready state.
* **Minimum Storage Duration Charge:** 
  + This applies to storage classes like Glacier (90 days), Deep Archive (180 days), or Standard-IA (30 days). AWS designs these classes for long-term storage; deleting data before the minimum retention period forces AWS to recover costs for unused storage capacity. For example, if you delete a 10 GB object from Glacier after 10 days, you’re still charged for the remaining eighty days.
* **Lifecycle Transitions:** 
  + When you configure lifecycle rules to move data between storage classes, AWS charges for the transition process. For instance, moving data from Standard to Glacier involves indexing, archival, and metadata updates, which require resources.
* **Availability Zones:** 
  + High availability is achieved by storing copies of data across multiple zones (≥3 for most classes). Classes like Standard-IA or Glacier store data redundantly across zones, while One Zone-IA saves costs by replicating data within a single zone. This choice impacts resilience against zone failures.
* **Regions:** 
  + Data storage pricing and latency depend on the selected AWS region. Data stored closer to your users improves performance but may vary in cost due to local resource availability and infrastructure.
* **Durability:** 
  + S3 is designed for "eleven nines" (99.999999999%) durability, meaning that the likelihood of losing data is extraordinarily low. This is achieved by replicating data across multiple disks and zones, ensuring resilience even during catastrophic failures.
* **Availability:** 
  + The availability percentage (e.g., 99.9% for Standard-IA) indicates how often data can be accessed without downtime. Higher availability requires more infrastructure redundancy, contributing to higher costs for classes like Standard.
* **Metadata:** 
  + Metadata is user-defined information (e.g., tags, permissions) stored with objects. It aids in categorizing and querying data efficiently. Classes like Standard allow instant access to metadata, while retrieval for archival classes may require restoring the object.
* **First Byte Latency:** 
  + In classes like Standard or Intelligent-Tiering, the first byte of data is available within milliseconds because data is in an active state. For Glacier, latency can be minutes or hours, as objects are restored from an archive before access.
* **Minimum Eligible Object Size:** 
  + Some classes, such as Standard-IA, are optimized for objects larger than 128 KB. If smaller objects are stored, the cost is calculated as if they were 128 KB. This design ensures efficient storage utilization and cost recovery.
* **Data Redundancy:** 
  + AWS stores multiple copies of data (at least three in Standard) to ensure reliability. Classes like One Zone-IA reduce redundancy to save costs, storing data within a single availability zone.
* **Delete Marker:** 
  + When you delete an object in version-enabled buckets, AWS adds a delete marker rather than removing the object permanently. This allows you to restore the object later if needed, offering a safeguard against accidental deletions.
* **Versioning:** 
  + Versioning helps maintain all versions of an object, preventing accidental data loss. If you modify or delete a file, previous versions remain intact and can be restored, ensuring data consistency.

**Scenario**

* **Data size**: 10GB (10 files of 1GB each).
* **Team members**: 3 (each downloads all 10 files).
* **Re-uploads**: Files are re-uploaded after modifications.
* **Storage duration**: Twelve months.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| storage Class | Storage Cost Rate | Storage Cost | PUT Request Cost Rate | PUT Request Cost | GET Request Cost Rate | GET Request Cost | Data Transfer Out | Data Transfer | Retrieval | Retrieval | Total Estimated |
| (/GB/Month) | (10 GB) | (/1,000 requests) | (10 requests) | (/1,000 requests) | (100 requests) | Cost Rate (/GB) | Out Cost (10 GB) | Cost Rate (/GB) | Cost (10 GB) | Monthly Cost |
| S3 Standard | $0.025 | $0.25 | $0.005 | $0.00005 | $0.0004 | $0.00004 | $0.015 | $0.15 | N/A | N/A | $0.40009 |
| S3 Standard-IA | $0.019 | $0.19 | $0.01 | $0.0001 | $0.01 | $0.001 | $0.015 | $0.15 | $0.01 | $0.10 | $0.4411 |
| S3 One Zone-IA | $0.0152 | $0.152 | $0.01 | $0.0001 | $0.01 | $0.001 | $0.015 | $0.15 | $0.01 | $0.10 | $0.4031 |
| S3 Glacier Instant Retrieval | $0.005 | $0.05 | $0.05 | $0.0005 | $0.03 | $0.003 | $0.015 | $0.15 | $0.03 | $0.30 | $0.5035 |
| S3 Glacier Flexible Retrieval | $0.0036 | $0.036 | $0.05 | $0.0005 | $0.03 | $0.003 | $0.015 | $0.15 | $0.03 | $0.30 | $0.4895 |
| S3 Glacier Deep Archive | $0.00099 | $0.0099 | $0.05 | $0.0005 | $0.03 | $0.003 | $0.015 | $0.15 | $0.02 | $0.20 | $0.3634 |

**Note: Retrieval is multiplied by 10, which won’t be the case if put in cold storage.**

AWS S3 can handle **3500 PUT, COPY, POST, or DELETE requests per second per prefix** and **5500 GET or HEAD requests per second per prefix**.

* **A prefix** is the portion of an object key between the bucket name and the object name (e.g., bucket-name/path1/).
* This limit refers to the maximum parallel operations that S3 can process for a single prefix in **one second**.

**Scenario 1: Sequential Uploads (10,000 Large Files)**

* **Example**: 10,000 files (1 GB each) uploaded sequentially to path1/.
  + Each file upload takes ~80 seconds over a **100 Mbps network**.
  + Since uploads are sequential, the **3500 PUT requests per second limit** does not apply because only one file is being uploaded at a time.
  + **Performance**: Bottlenecked by your network speed, not S3.

**Scenario 2: Parallel Uploads (10,000 Large Files)**

* **Example**: 10,000 files (1 GB each) uploaded in parallel to path1/ using multiple threads or applications.
  + S3 can handle up to **3500 parallel uploads per second** for path1/.
  + Any requests above this limit will be **queued** or **throttled** by S3.
  + **Solution**: Split uploads across multiple prefixes (path1/, path2/, path3/) to scale throughput.
    - **Result**: Each prefix can handle 3500 PUTs per second, giving a total throughput of **10,500 PUTs per second** for 3 prefixes.

**Scenario 3: Multiple Applications Uploading Simultaneously**

* **Example**: Two applications writing to path1/:
  + App A uploads 2000 objects per second.
  + App B uploads 2000 objects per second.
  + **Total**: 4000 PUT requests per second.
  + **Result**: Exceeds the 3500 limit; some requests will be throttled or delayed.
  + **Solution**: Distribute writes to multiple prefixes or optimize request patterns.

**Key Takeaways**

1. **Sequential operations** won’t hit the limit because each upload takes time, and no parallelism is involved.
2. **Parallel operations** are where the limit becomes significant. To maximize throughput:
   * Use multiple prefixes (path1/, path2/, etc.).
   * Use multi-threaded applications to parallelize uploads.
3. S3’s limits are per prefix. Organize object keys effectively to scale operations.