**S3 MFA – DELETE**

* MFA (multi factor authentication) forces user to generate a code on a device (usually a mobile phone or hardware) before deleting object in S3. Hence, add extra security level for accidental deletes
* To use MFA-Delete, enable Versioning on the S3 bucket
* You will need MFA to
  + permanently delete an object version
  + suspend versioning on the bucket
* You won’t need MFA for
  + enabling versioning
  + listing deleted versions
* Only the bucket owner (root account) can enable/disable MFA-Delete
* MFA-Delete currently can only be enabled using the CLI

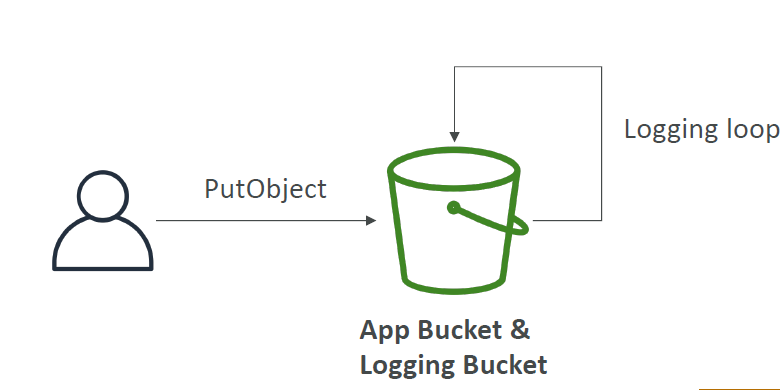
*\*\* Check hands on, how it is configured through CLI*

**S3 ACCESS LOGS**

* For audit purpose, Any request made to S3, from any account, authorized or denied, will be logged into another S3 bucket
* That data can be analyzed using data analysis tools…
* Or Amazon Athena as we’ll see later in this section
* The log format is at: <https://docs.aws.amazon.com/AmazonS3/latest/dev/LogFormat.html>

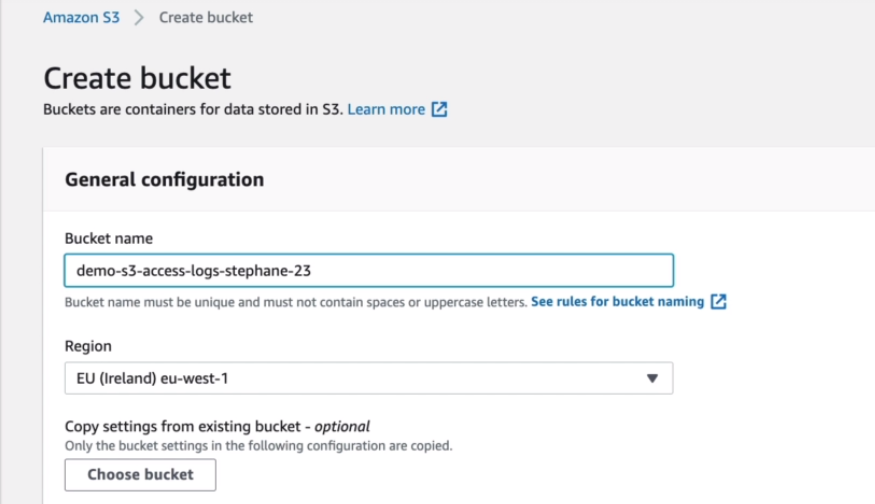
© S**S3 Access Logs: Warning**

* Do not set your logging bucket to be the monitored bucket
* It will create a logging loop, and your bucket will grow in size exponentially

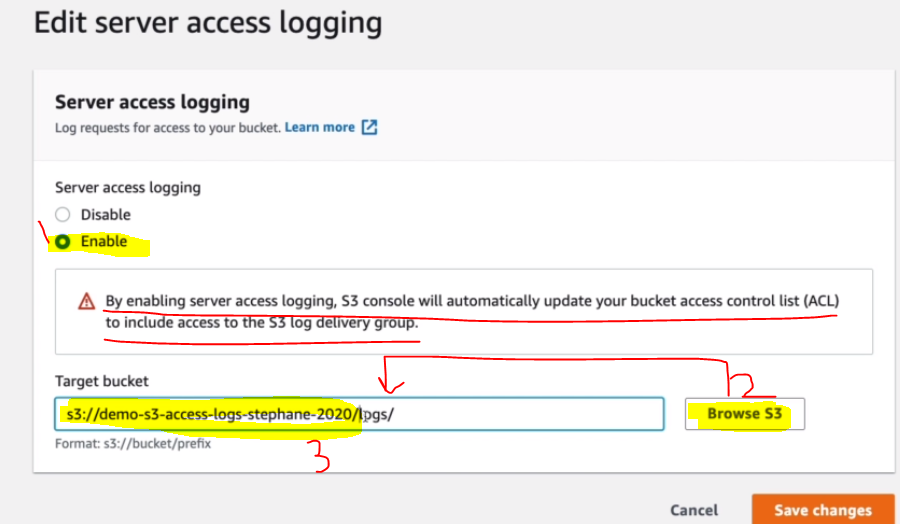


**S3 Access Logs hands on**

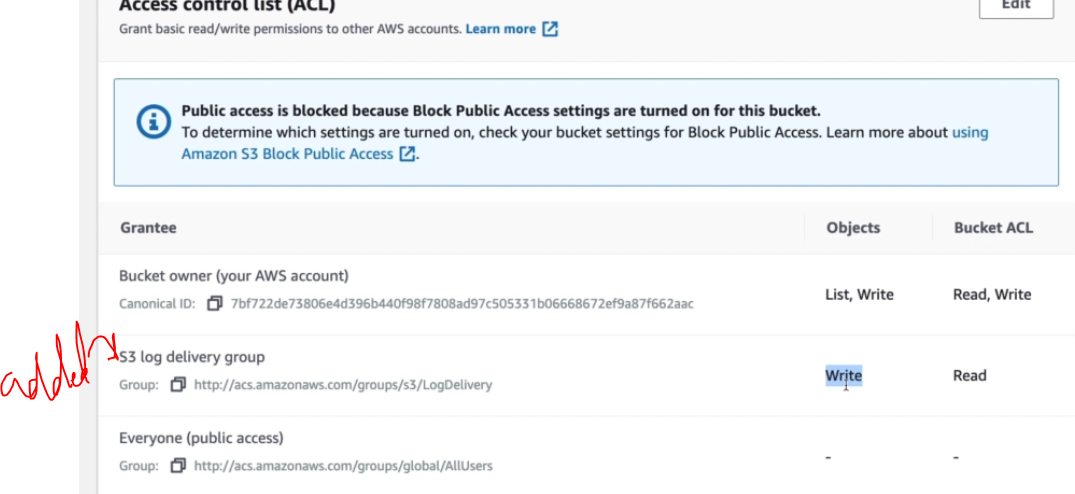
1 .Create a bucket for logging and keep other settings as is.



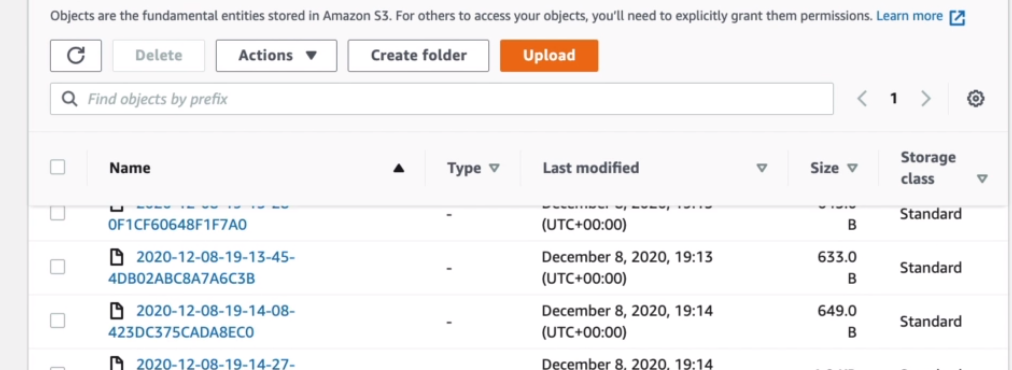
2. Go to Bucket (on which logging needs to be enabled) -> Properties -> Server access Logging



*\*\* Log bucket will grant write permission explicity to bucket and can be verified from permission of log bucket as:*



3. Access object of Bucket and see its access log in log bucket

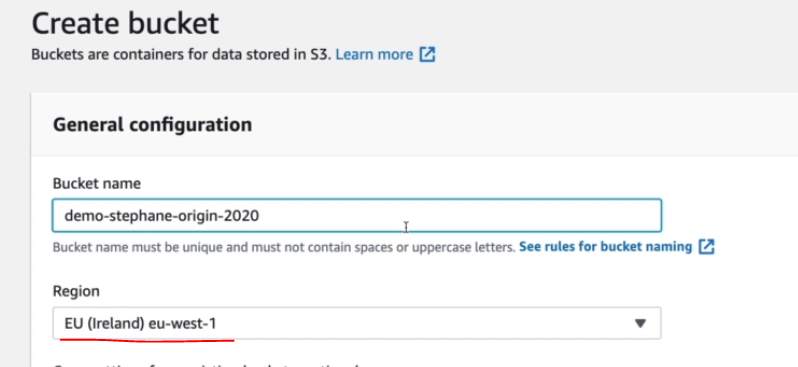


**S3 REPLICATION (CRR & SRR)**

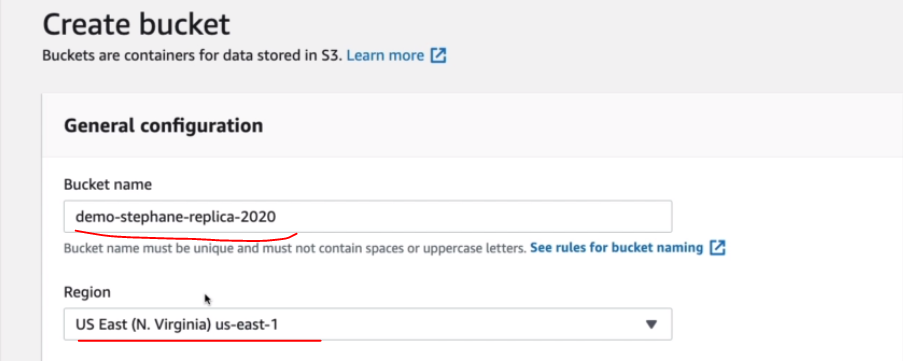
* S3 replication – asynchronous replication between buckets in different regions is called *Cross Region Replication (CRR)* and in same region called *Same Region Replication (SRR)*
* CRR - Use cases: compliance, lower latency access, replication across accounts
* SRR – Use cases: log aggregation, live replication between production and test accounts
* For S3 replication - Must enable versioning in source and Destination
* Buckets can be in different account and we can replicate all objects or limit the object fron source while creating replicating rule
* Must give proper IAM permissions to S3 Buckets
* After activating S3 replications, only new objects are replicated (not retroactive)
* For DELETE operations:
  + If you delete without a version ID, it adds a delete marker, not replicated
  + If you delete with a version ID, it deletes in the source, not replicated
* There is no “chaining” of replication
  + If bucket 1 has replication into bucket 2, which has replication into bucket 3
  + Then objects created in bucket 1 are not replicated to bucket 3

**S3 replication Hands on**

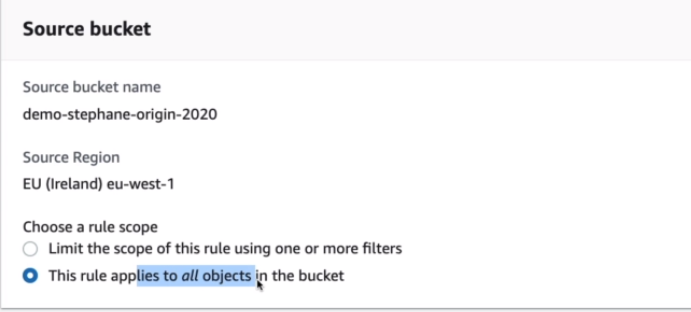
1. Create Source Bucket and enable versioning

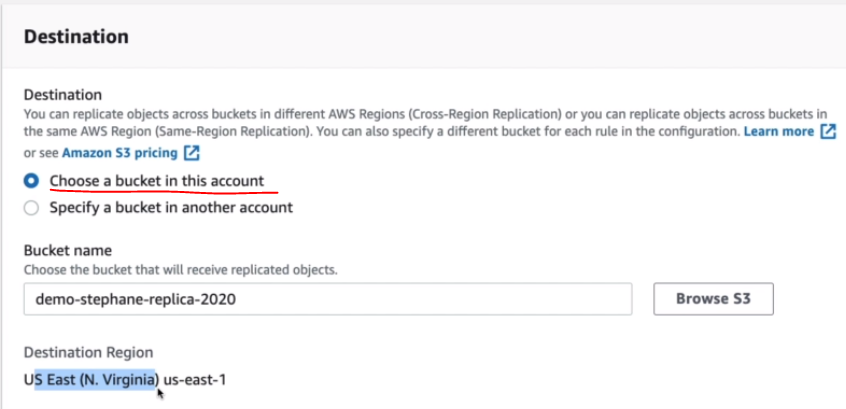


1. Create Target Bucket (replica) and enable versioning

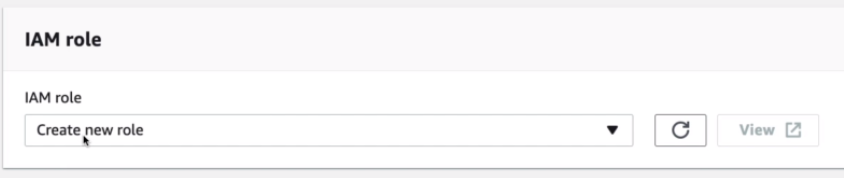


1. For Enabling replication , Go to Source Bucket -> Management -> Replication rules -> Create replication Rule
2. Configure source and destination details





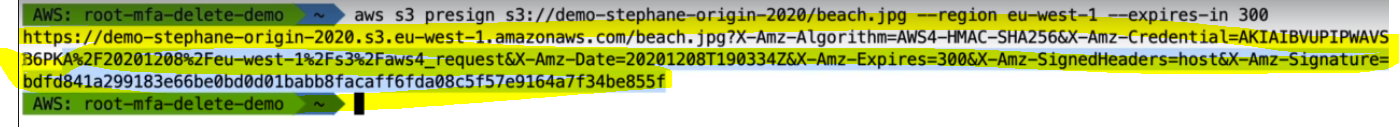
1. Create IAM role for performing replication , we can leverage the S3 feature of creating role themselves



1. Save

**S3 PRE-SIGNED URL**

* Can generate pre-signed URLs using SDK or CLI
* For downloads (easy, can use the CLI)
* For uploads (harder, must use the SDK)
* Valid for a default of 3600 seconds, can change timeout with --expires-in [TIME\_BY\_SECONDS] argument
* Users given a pre-signed URL inherit the permissions of the person who generated the URL for GET / PUT
* Examples :
  + Allow only logged-in users to download a premium video on your S3 bucket
  + Allow an ever changing list of users to download files by generating URLs dynamically
  + Allow temporarily a user to upload/download a file to a precise location in our bucket
* Create pre-signed URL

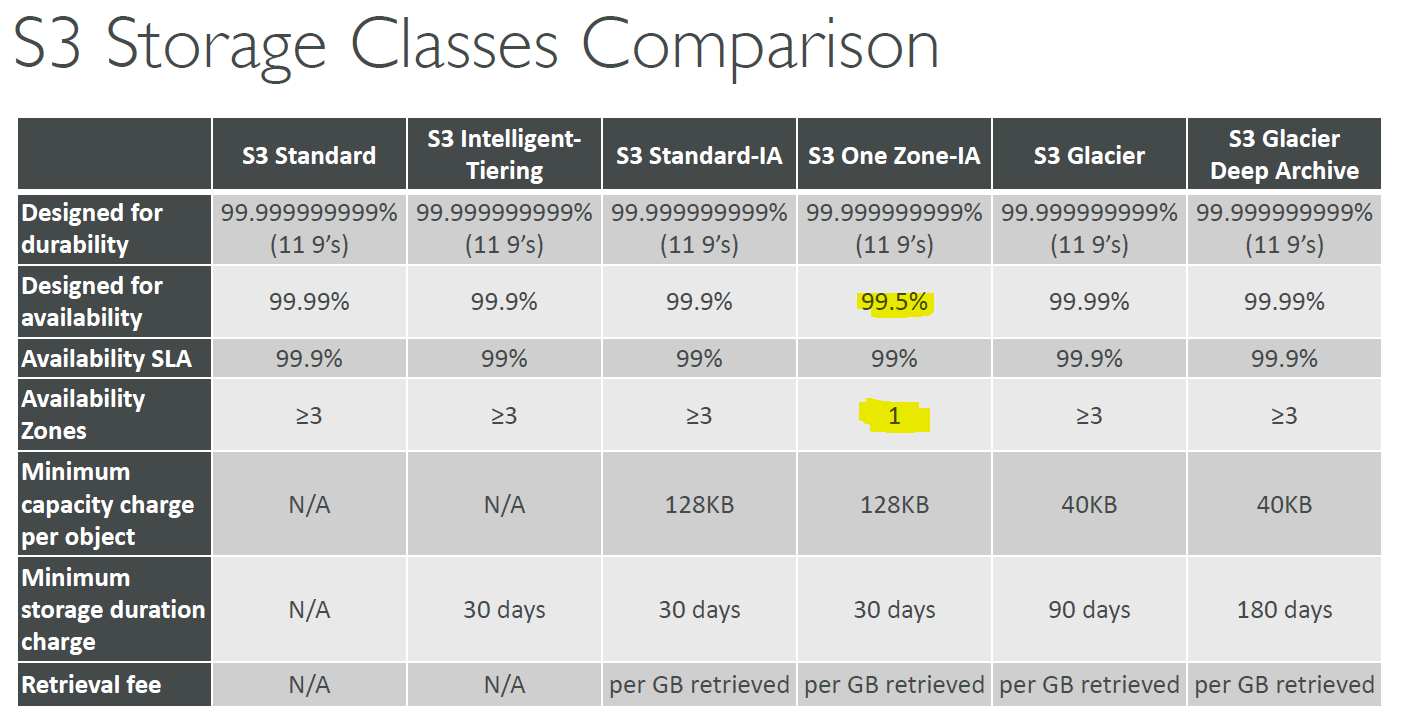


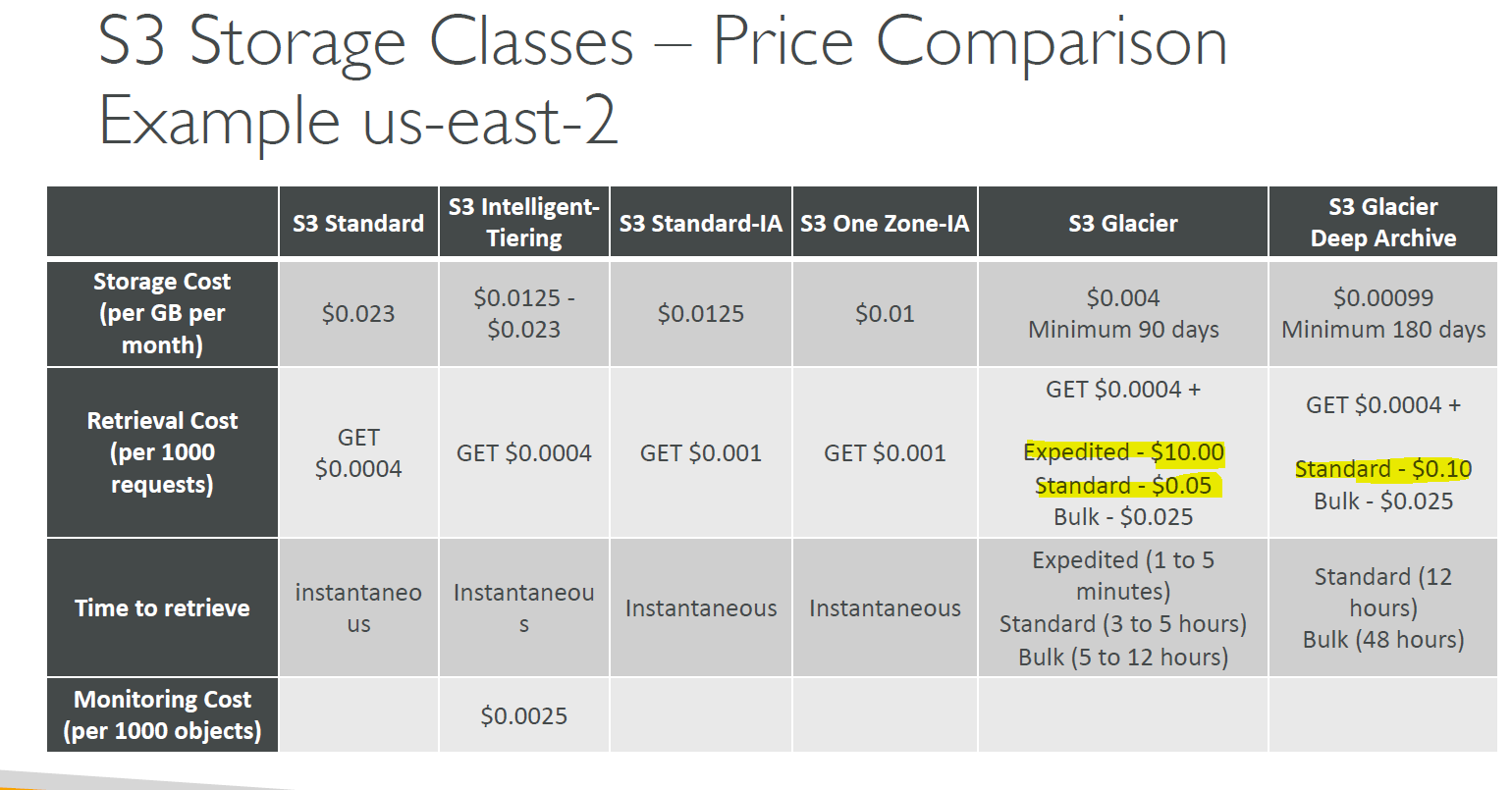
**S3 STORGAE CLASSES**

1. **Amazon S3 Standard - General Purpose**
   * High durability (99.999999999%) of objects across multiple AZ
   * 99.99% Availability over a given year
   * Sustain 2 concurrent facility failures
   * Use Cases: Big Data analytics, mobile & gaming applications, content distribution…
2. **Amazon S3 Standard-Infrequent Access (IA)**
   * Suitable for data that is less frequently accessed, but requires rapid access when needed
   * High durability (99.999999999%) of objects across multiple AZs
   * 99.9% Availability
   * Low cost compared to Amazon S3 Standard -General purpose
   * Sustain 2 concurrent facility failures
   * Use Cases: As a data store for disaster recovery, backups…
3. **Amazon S3 One Zone-Infrequent Access**
   * Same as IA but data is stored in a single AZ
   * High durability (99.999999999%) of objects in a single AZ; data lost when AZ is destroyed
   * 99.5% Availability
   * Low latency and high throughput performance
   * Supports SSL for data at transit and encryption at rest
   * Low cost compared to IA (by 20%)
   * Use Cases: Storing secondary backup copies of on-premise data, or storing data you can recreate
4. **Amazon S3 Intelligent Tiering**
   * Same low latency and high throughput performance of S3 Standard
   * Small monthly monitoring and auto-tiering fee
   * Automatically moves objects between two access tiers based on changing access patterns
   * Designed for durability of 99.999999999% of objects across multiple Availability Zones
   * Resilient against events that impact an entire Availability Zone
   * Designed for 99.9% availability over a given year
5. **Amazon Glacier**
   * Low cost object storage meant for archiving / backup
   * Data is retained for the longer term (10s of years)
   * Alternative to on-premise magnetic tape storage and keep tape somewhere
   * Average annual durability is 99.999999999%
   * Cost per storage per month ($0.004 / GB) + retrieval cost
   * Each item in Glacier is called “Archive” (not object) and size can be up to 40TB
   * Archives are stored in ”Vaults” (not buckets)
6. **Amazon Glacier Deep Archive**
   * Similar to Glacier, but it is used for very long term storage and at very cheap price

**Retrieval option in Glacier and Glacier Deep Archive**

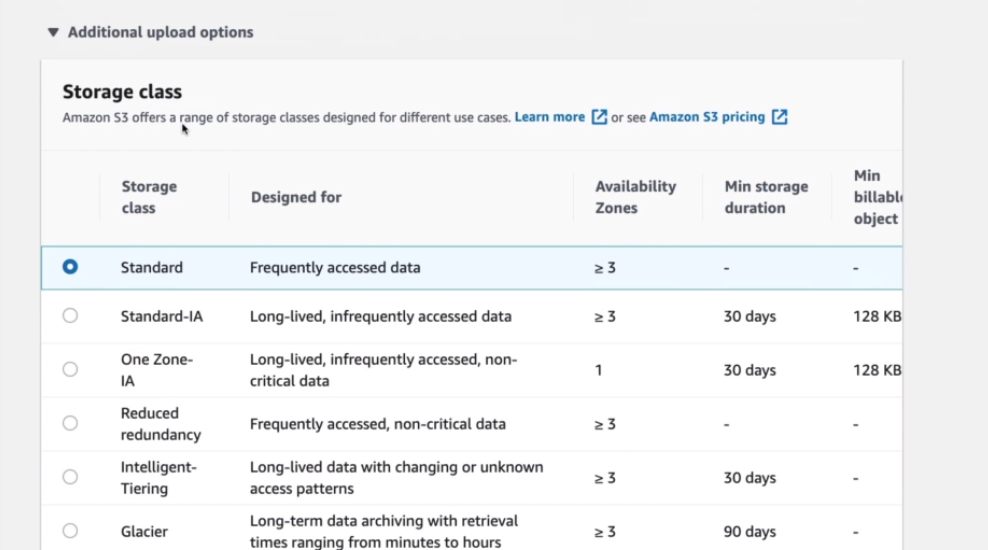
* + Amazon Glacier – 3 retrieval options:
    - Expedited (1 to 5 minutes) -- expensive
    - Standard (3 to 5 hours)
    - Bulk (5 to 12 hours)
    - Minimum storage duration of 90 days
  + Amazon Glacier Deep Archive – for long term storage – cheaper:
    - Standard (12 hours)
    - Bulk (48 hours)
    - Minimum storage duration of 180 days



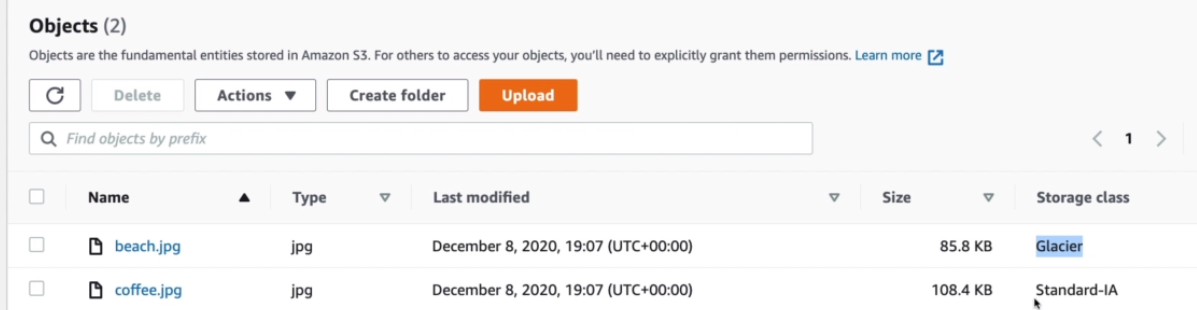


**S3 Storage Class Hands on**

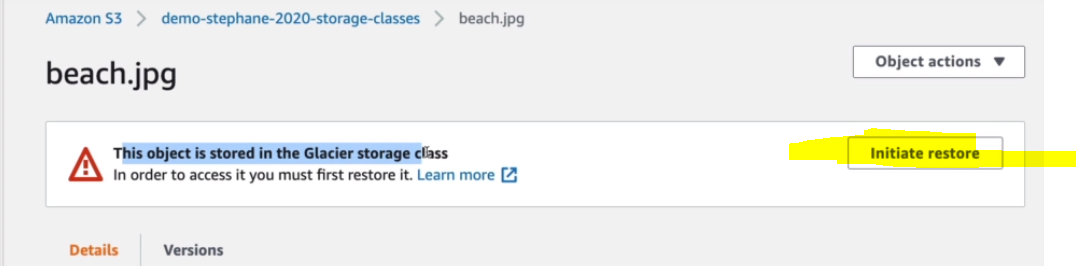
1. Go to Bucket and click on upload and choose storage class in *additional upload options*



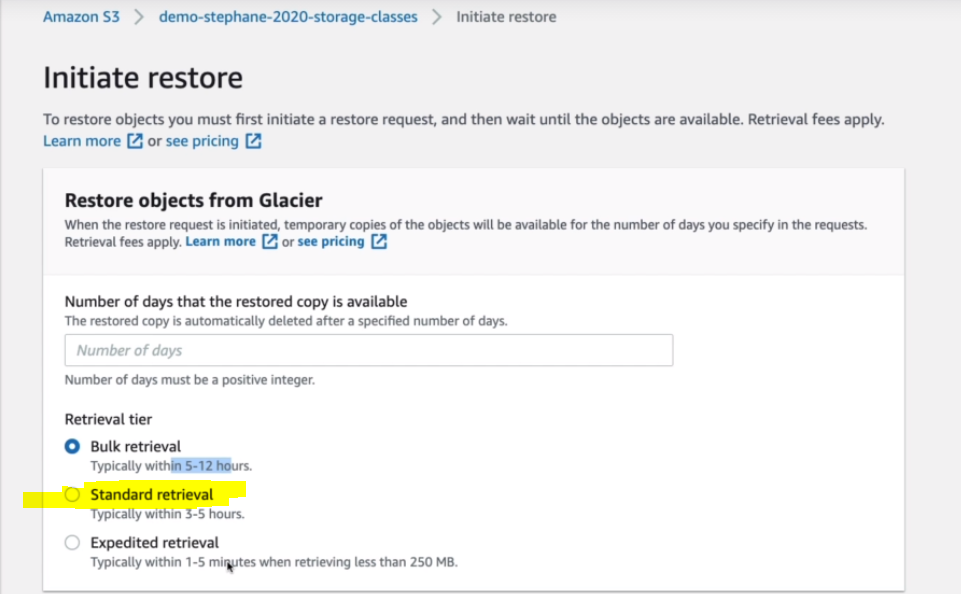
1. Suppose, you have store object in Glacier class, then you cannot access it directly



1. if you click on beach.jpg – will ask to restore first

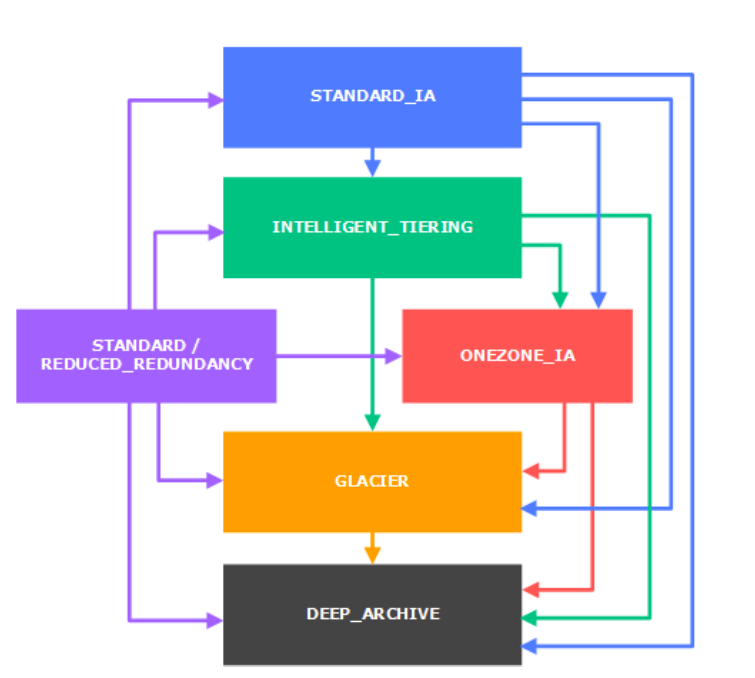


1. Restore file from Glacier : Initiate restore -> enter retrieval option (Accordingly will take time to restore)



**S3 LIFECYCLE RULES**

* Allows moving objects b/w storage classes



* **Transition actions**: It defines when objects are transitioned to another storage class.
  + Move objects to Standard IA class 60 days after creation
  + Move to Glacier for archiving after 6 months
* **Expiration actions**: configure objects to expire (delete) after some time
  + Access log files can be set to delete after a 365 days
  + Can be used to delete old versions of files (if versioning is enabled)
  + Can be used to delete incomplete multi-part uploads
* Rules can be created for a certain prefix (ex - s3://mybucket/mp3/\*)
* Rules can be created for certain objects tags (ex - Department: Finance)

**Scenario 1**

**Your application on EC2 creates images thumbnails after profile photos are uploaded to Amazon S3. These thumbnails can be easily recreated, and only need to be kept for 45 days. The source images should be able to be immediately retrieved for these 45 days, and afterwards, the usercan wait up to 6 hours. How would you design this?**

* S3 source images can be on STANDARD, with a lifecycle configuration to transition them to GLACIER after 45 days.
* S3 thumbnails can be on ONEZONE\_IA, with a lifecycle configuration to expire them (delete them) after 45 days.

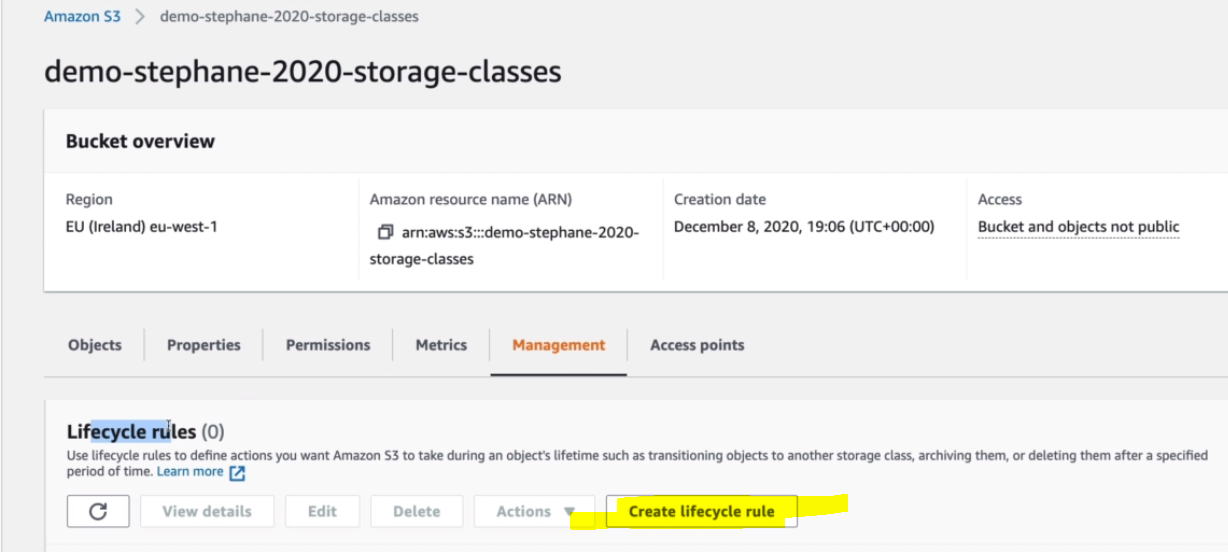
**Scenario 2**

**A rule in your company states that you should be able to recover your Deleted S3 objects immediately for 15 days, although this may happen rarely. After this time, and for up to 365 days, deleted objects should be recoverable within 48 hours.**

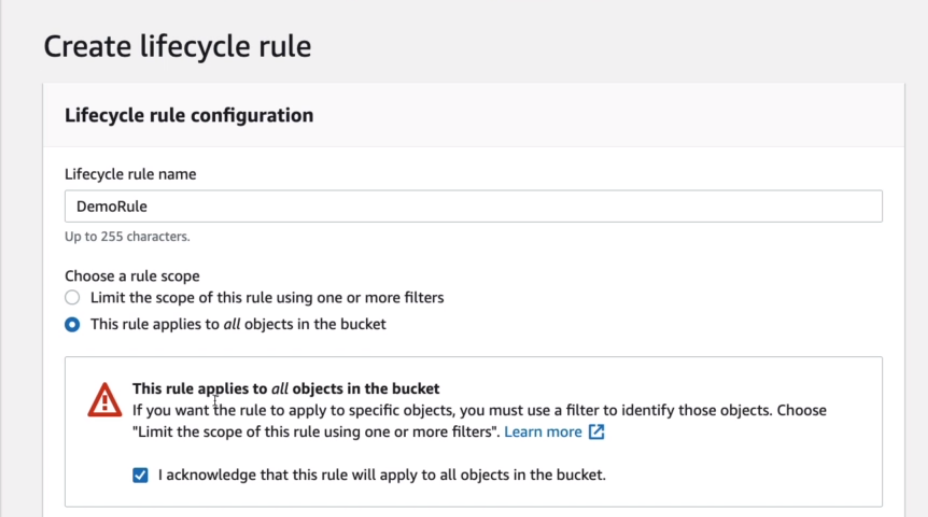
* You need to enable S3 versioning in order to have object versions, so that “deleted objects are in fact hidden by a “delete marker” and can be recovered
* You can transition these “noncurrent versions” of the object to S3\_IA
* You can transition afterwards these “noncurrent versions” to DEEP\_ARCHIVE

**S3 Lifecycle Rules Hands on**

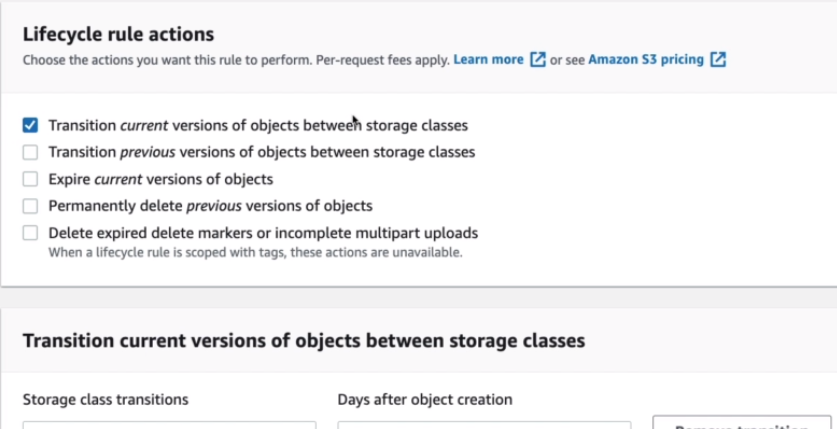
1. Go to Bucket -> Management ->Create lifecycle rule



1. Choose rule name and rule scope

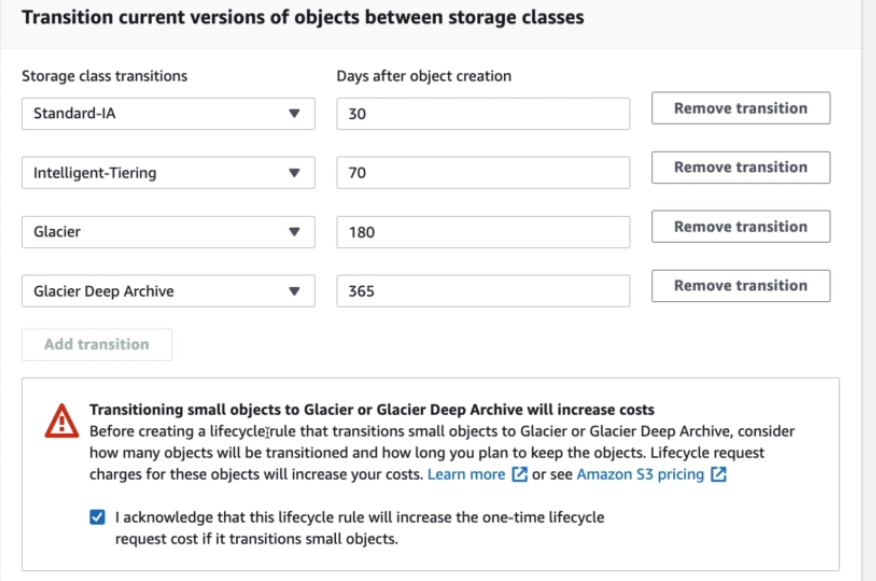


1. Choose lifecycle rule action



\*\* We can select multiple rule actions and configure them in next step

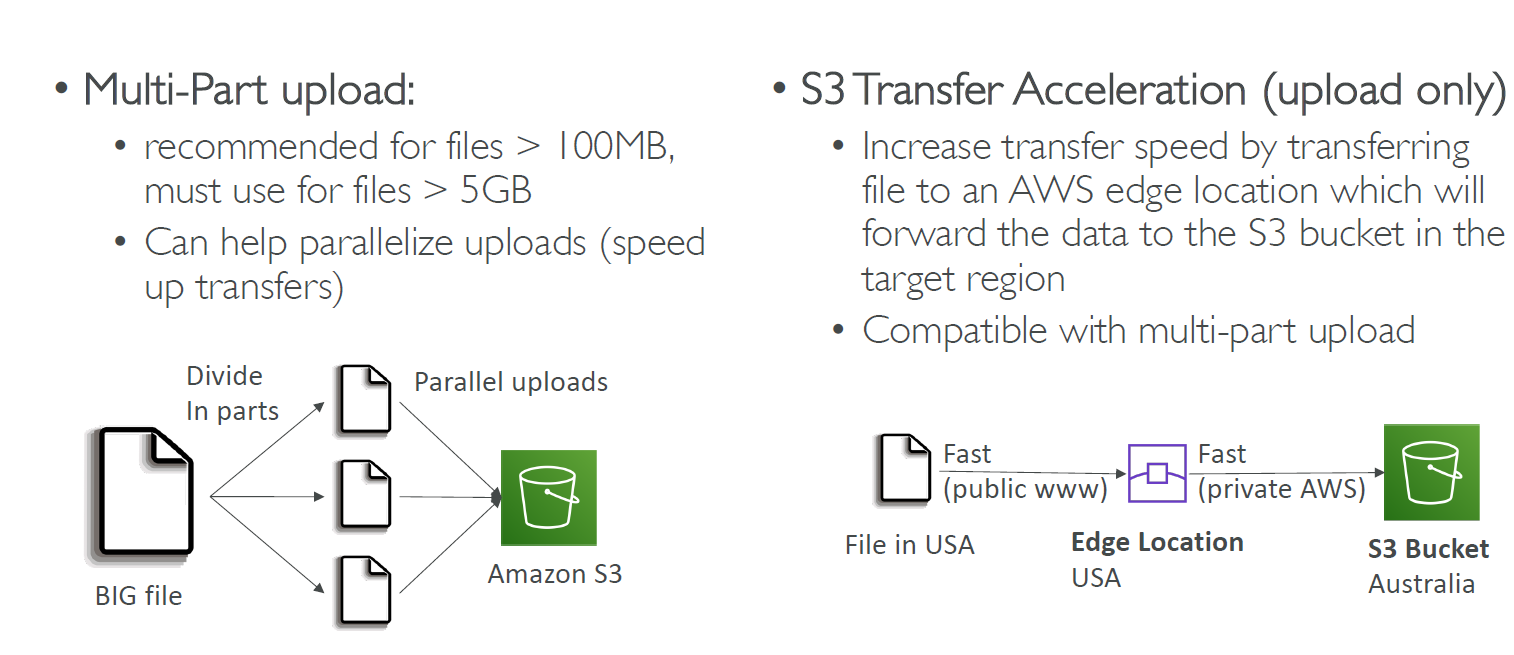
1. Configure details

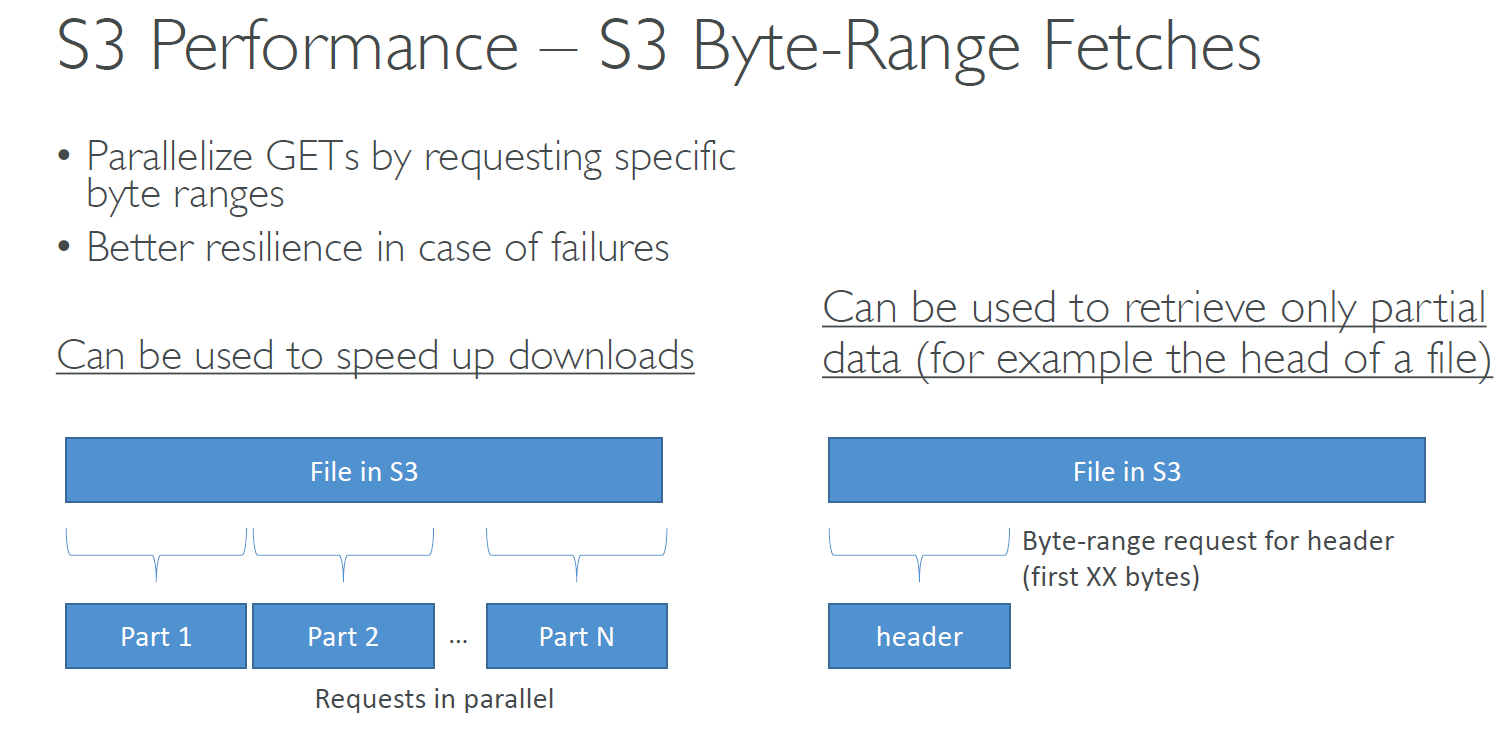


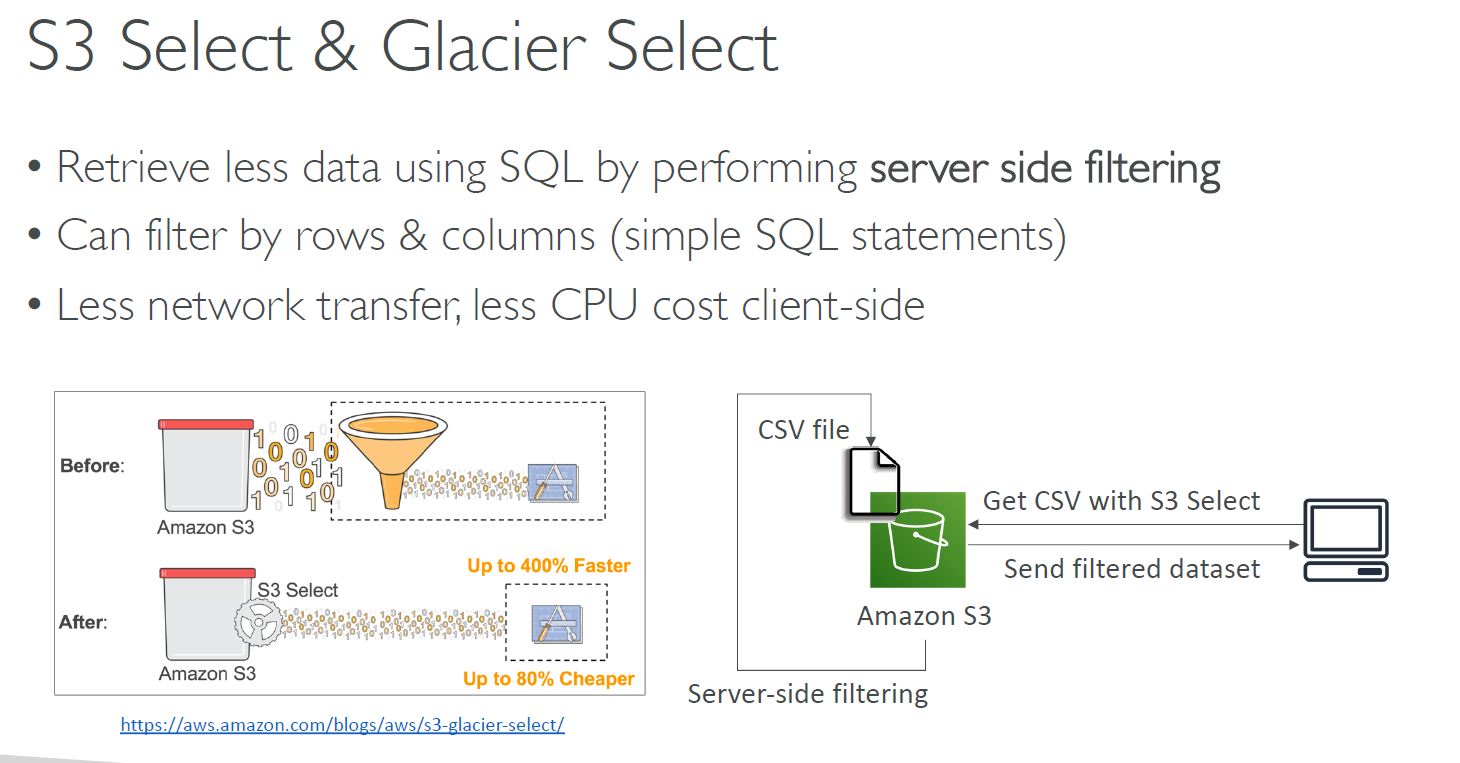
1. Save rule

**S3 – Baseline Performance**

* Amazon S3 automatically scales to high request rates with low latency 100-200 ms.
* Your application can achieve at least 3,500 PUT/COPY/POST/DELETE and 5,500 GET/HEAD requests per second per prefix in a bucket.
* There are no limits to the number of prefixes in a bucket.
* Example (object path => prefix):
  + bucket/folder1/sub1/file => /folder1/sub1/
  + bucket/folder1/sub2/file => /folder1/sub2/
  + bucket/1/file => /1/
  + bucket/2/file => /2/
* If you spread reads across all four prefixes evenly, you can achieve 22,000 requests per second for GET and HEAD

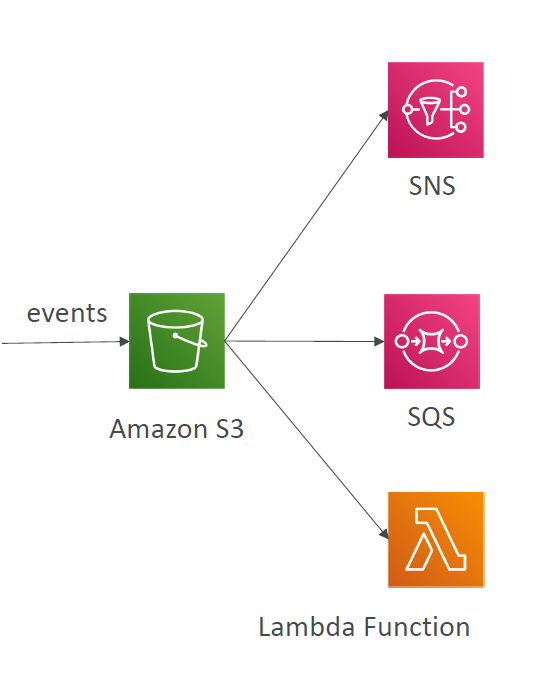






**S3 EVENT NOTIFICATION**

* S3:ObjectCreated, S3:ObjectRemoved, S3:ObjectRestore, S3:Replication…
* Object name filtering possible (\*.jpg)
* Use case: generate thumbnails of images uploaded to S3
* Can create as many “S3 events” as desired
* S3 event notifications typically deliver events in seconds but can sometimes take a minute or longer
* If two writes are made to a single non-versioned object at the same time, it is possible that only a single event notification will be sent
* If you want to ensure that an event notification is sent for every successful write, you can enable versioning on your bucket.

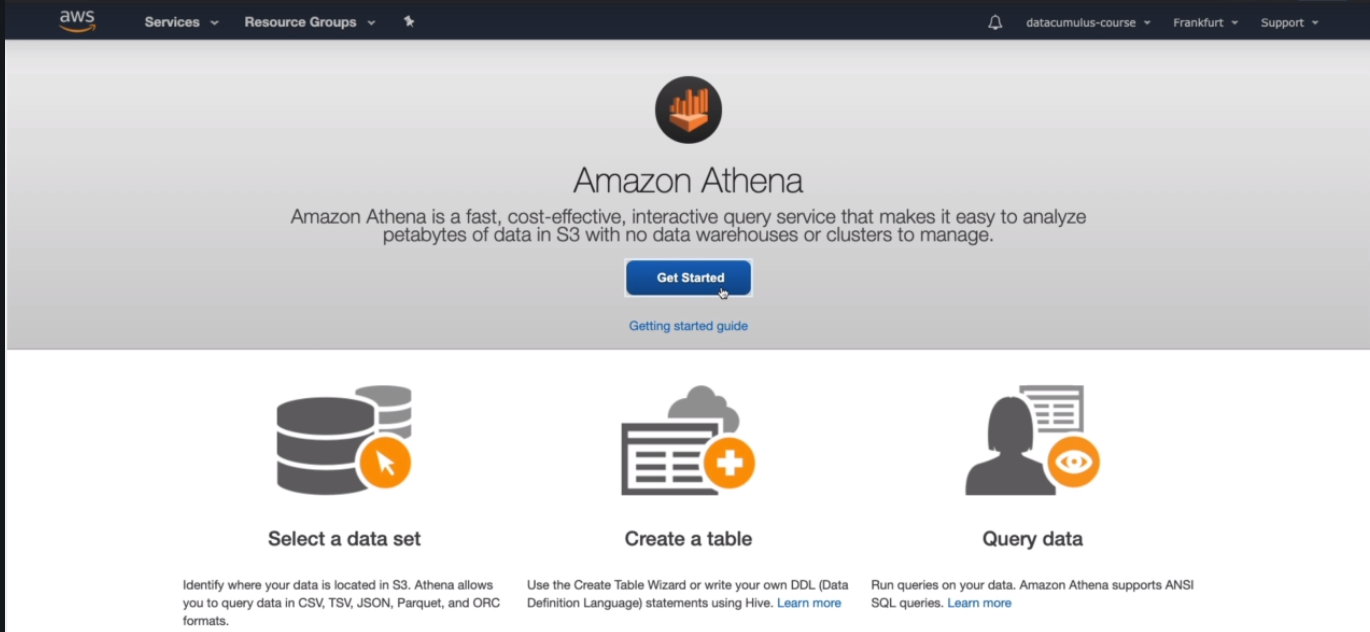


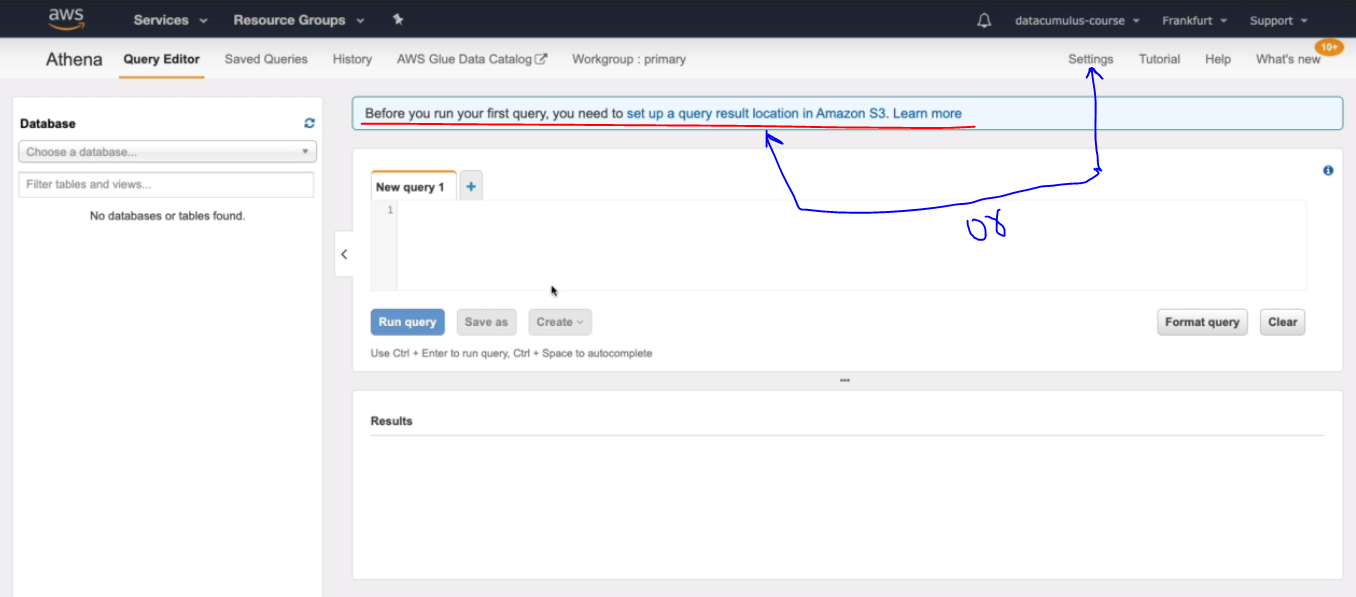
\*\* See its hand on

**AWS ATHENA**

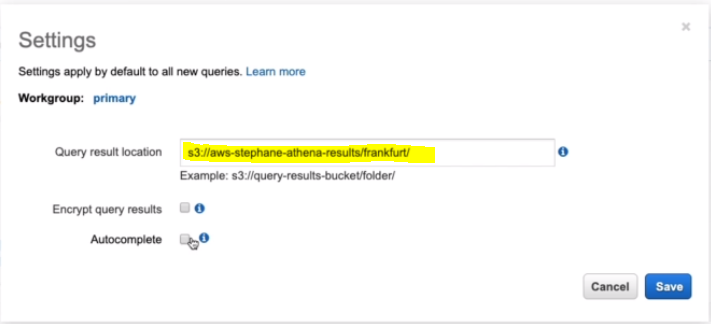
* ***Interactive query service*** to analyze peta bytes of data in S3 without loading into data warehouse or clusters
* Uses SQL language to query the files
* Has a JDBC / ODBC driver
* *Charged per query and amount of data scanned*
* Supports CSV, JSON, ORC, Avro, and Parquet (built on Presto)
* Use cases: Business intelligence / analytics / reporting, analyze & query ,VPC Flow Logs, ELB Logs, CloudTrail trails, etc...
* *Exam Tip: Analyze data directly on S3 => use Athena*

**ATHENA hands on**

1. Search for Athena in service 
2. Click on Get started and it will open Query editor



1. Before using query editor for first time, Aws gives warning to setup S3 location and it can be setup :
   1. Directly click on warning
   2. Setting



1. Create a database and table

create database s3\_access\_logs\_db;

CREATE EXTERNAL TABLE IF NOT EXISTS s3\_access\_logs\_db.mybucket\_logs(

BucketOwner STRING,

Bucket STRING,

RequestDateTime STRING,

RemoteIP STRING,

Requester STRING,

RequestID STRING,

Operation STRING,

Key STRING,

RequestURI\_operation STRING,

RequestURI\_key STRING,

RequestURI\_httpProtoversion STRING,

HTTPstatus STRING,

ErrorCode STRING,

BytesSent BIGINT,

ObjectSize BIGINT,

TotalTime STRING,

TurnAroundTime STRING,

Referrer STRING,

UserAgent STRING,

VersionId STRING,

HostId STRING,

SigV STRING,

CipherSuite STRING,

AuthType STRING,

EndPoint STRING,

TLSVersion STRING

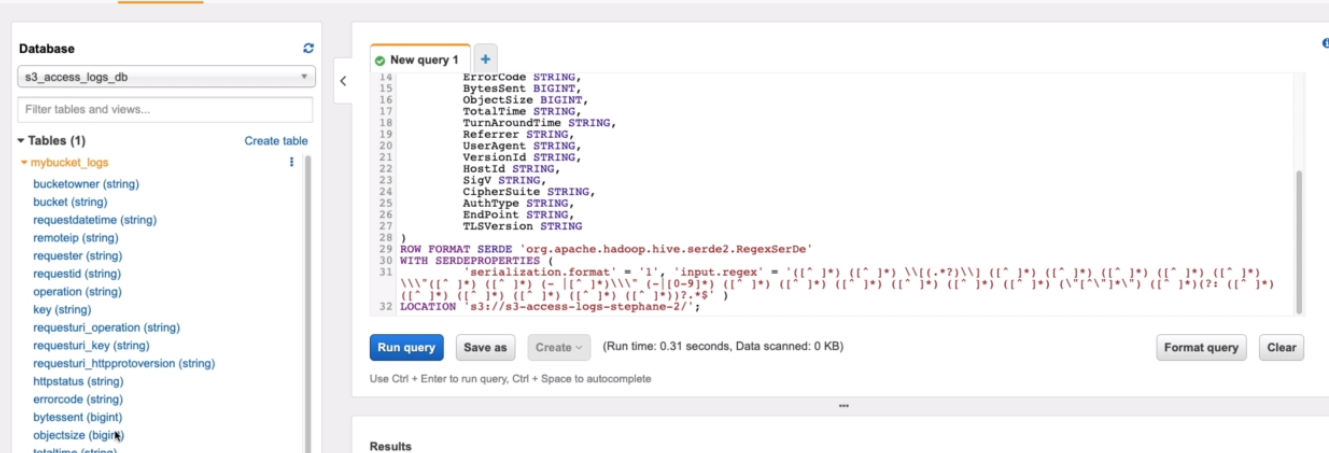
)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'

WITH SERDEPROPERTIES (

'serialization.format' = '1', 'input.regex' = '([^ ]\*) ([^ ]\*) \\[(.\*?)\\] ([^ ]\*) ([^ ]\*) ([^ ]\*) ([^ ]\*) ([^ ]\*) \\\"([^ ]\*) ([^ ]\*) (- |[^ ]\*)\\\" (-|[0-9]\*) ([^ ]\*) ([^ ]\*) ([^ ]\*) ([^ ]\*) ([^ ]\*) ([^ ]\*) (\"[^\"]\*\") ([^ ]\*)(?: ([^ ]\*) ([^ ]\*) ([^ ]\*) ([^ ]\*) ([^ ]\*) ([^ ]\*))?.\*$' )

LOCATION 's3://s3-access-logs-stephane-2/';



1. Now Run the SQl query

