Amazon S3

→ object storing service that offers scalability, data availability security & performance

→ store & retrieve any amount of data.

→ doesn't have directories, everything is "Key"

→ stores objects (files) in buckets (directories)

→ bucket name must be GLOBALLY UNIQUE. (looks like global service but is not :))

→ bucket is @ region level.

→ Naming conventions
- No uppercase,
- No underscore
- Not an IP
- Must start with lowercase letter or number
- prefix MUST NOT xn--
- suffix MUST NOT -s3alias

→ Key / value pair
- Key: full path i.e. prefix + object name
- value: content of the object!

{ Max obj size = 5000 GB (5TB)
if obj size > 5GB then "multipart upload"

Metadata
tags
Version ID

extra layer of security by AWS to prevent data leaks.

→ while making a bucket if we block all the public acces then opening a bucket object with its public URL is not possible, eventhough we can directly open it but not with public URL, this because it is a pre-signed URL which has the users credentials & hence ever can open it but not the public

Amazon S3 – Security

- User-Based
  - IAM Policies – which API calls should be allowed for a specific user from IAM
- Resource-Based
  - Bucket Policies – bucket wide rules from the S3 console - allows cross account
  - Object Access Control List (ACL) – finer grain (can be disabled)
  - Bucket Access Control List (ACL) – less common (can be disabled)
- Note: an IAM principal can access an S3 object if
  - The user IAM permissions ALLOW it OR the resource policy ALLOWS it
  - AND there's no explicit DENY
- Encryption: encrypt objects in Amazon S3 using encryption keys

S3 Bucket Policies

- JSON based policies
  - Resources: buckets and objects
  - Effect: Allow / Deny
  - Actions: Set of API to Allow or Deny
  - Principal: The account or user to apply the policy to
- Use S3 bucket for policy to:
  - Grant public access to the bucket
  - Force objects to be encrypted at upload
  - Grant access to another account (Cross Account)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicRead",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::examplebucket/*"
            ]
        }
    ]
}
```

"Principal": "*" → To all

"Action": "s3:GetObject" → allow all to get obj. from S3 bucket with name examplebucket/* & all files!

bucket policy or IAM policies —attached→ bucket

bucket ACLs —attached→ bucket + objects

→ So for allowing public access we will have to attach bucket policy allowing public access to our S3 bucket!

→ If its an IAM user then there must be an IAM policy allowing the user to access S3 buckets. If there is an explicity deny then it will be more effective than bucket policy

→ If its an EC2 instance then we will have to create an IAM role which gives correct IAM permissions to access S3 bucket.

→ If cross account access then we need an IAM policy which allows a user from other account to make API calls

"use policy generator to create a bucket policy!"

* To make an object public
  1. turn off Block all public access
  2. Add a policy to allow public access to the bucket

Amazon S3 – Static Website Hosting

- S3 can host static websites and have them accessible on the Internet
- The website URL will be (depending on the region)
  - http://*bucket-name*.s3-website-*aws-region*.amazonaws.com
  OR
  - http://*bucket-name*.s3-website.*aws-region*.amazonaws.com
- If you get a 403 Forbidden error, make sure the bucket policy allows public reads!

→ after enabling S3 bucket for reads then

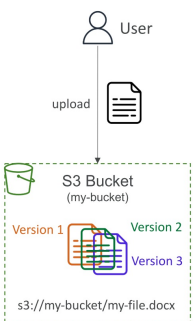
→ bucket properties में जाके Turn on static website hosting.
define { index.html }

{ for customers to access content @ website endpoint, you must make all the contents publicly readable }

→ after uploading index.html
go to properties & open the bucket website endpoint

Amazon S3 - Versioning

- You can version your files in Amazon S3
- It is enabled at the bucket level
- Same key overwrite will change the "version": 1, 2, 3....
- It is best practice to version your buckets
  - Protect against unintended deletes (ability to restore a version)
  - Easy roll back to previous version
- Notes:
  - Any file that is not versioned prior to enabling versioning will have version "null"
  - Suspending versioning does not delete the previous versions

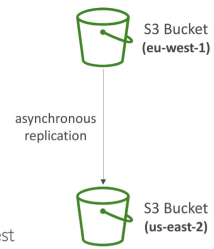


show versions toggle on करके delete करोगे तो permanently delete होगी i.e. deletion with version ID!
toggle off करके delete करने से update होगी with a delete marker & not actually delete
so deleting the delete marker will restore the file

Amazon S3 – Replication (CRR & SRR)

replication only works if versioning enabled

- Must enable Versioning in source and destination buckets
- Cross-Region Replication (CRR)
- Same-Region Replication (SRR)
- Buckets can be in different AWS accounts
- Copying is asynchronous
- Must give proper IAM permissions to S3



→ asynchronous replication ! i.e. eventually consistent.

- Use cases:
  - CRR – compliance, lower latency access, replication across accounts
  - SRR – log aggregation, live replication between production and test accounts

→ After replication is enabled ONLY NEW objects are replicated

→ If we want to upload older objects then use S3 batch replication.

→ The replicated objects will have the same version ID)

Amazon S3 – Replication (Notes)

- After you enable Replication, only new objects are replicated
- Optionally, you can replicate existing objects using S3 Batch Replication
  - Replicates existing objects and objects that failed replication
- For DELETE operations → by default not replicated (can be in replication rule)
  - Can replicate delete markers from source to target (optional setting)
  - Deletions with a version ID are not replicated (to avoid malicious deletes)
- 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

} not transitive !

1) need to create a SRC bucket (origin) & a target bucket (replica) with versioning.

2) In the origin bucket > management (down) > create a replication rule { scope, this acct. or another acct., target bkt., create a new IAM role }

Storage classes in S3

. Data transfer into S3 is free!
. All storage class are highly durable

(1) Standard - General Purpose

* min retrieval time
* max cost
* No retrieval fee

→ high availability → high throughput
→ low latency
→ for frequently accessed data → big data analytics, mobile & gaming applic^n, content distro.
→ less retrieval time
→ most expensive among the classes

(2) Infrequent Acces 30 days

→ Pay for retrieval
→ cheaper than standard → disaster recovery backups
for additional cost saving
→ One Zone - IA in single AZ
→ for storing backups of on premesis data or data which can be created!

(3) Glacier storage classes

→ Large retrieval time for archiving & backup
→ cheaper { pay for storage + retrieval of obj }

3a] Glacier instant retrieval ≥ 90 days

→ m.sec retrieval good for quaterly retrieval

3b] Glacier flexible retrieval ≥ 90 days

→ expedited (1-5 min), standard (3-5 hr), bulk (5-12) hr (fee)

3c] Glacier Deep Archives ≥ 180 days

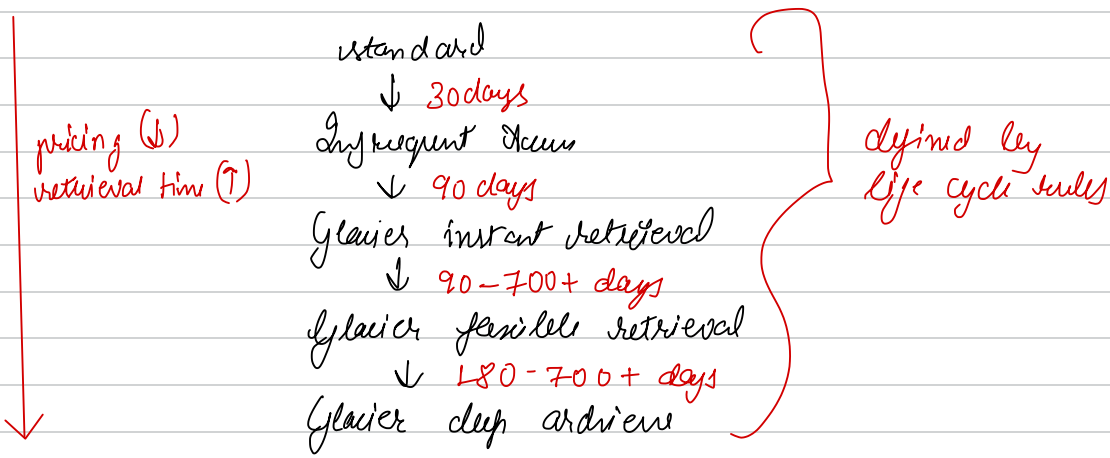
→ standard (12 hr), bulk (48 hr)

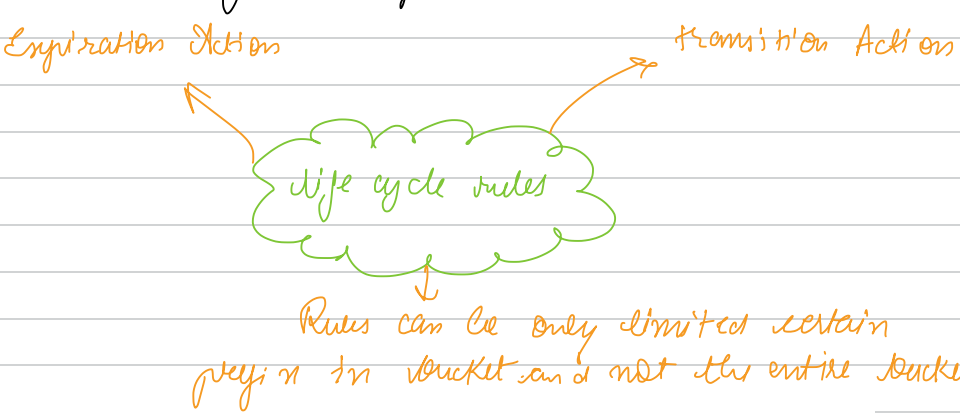
④ Intelligent tiering

* No retrieval fee
* Auto tiering fee

→ Automatically moves objects between storage classes based on usage

→ Fee on Auto tiering, monitored monthly





Amazon S3 – Lifecycle Rules (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 60 days. The source images should be able to be immediately retrieved for these 60 days, and afterwards, the user can 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 60 days
- S3 thumbnails can be on **One-Zone IA**, with a lifecycle configuration to expire them (delete them) after 60 days

Amazon S3 – Lifecycle Rules (Scenario 2)

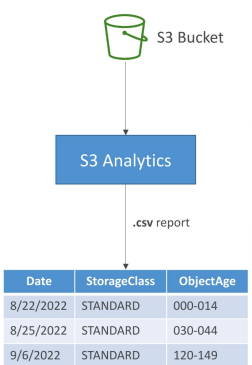
- A rule in your company states that you should be able to recover your deleted S3 objects immediately for 30 days, although this may happen rarely. After this time, and for up to 365 days, deleted objects should be recoverable within 48 hours.

- **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
- Transition the "noncurrent versions" of the object to **Standard IA**
- Transition afterwards the "noncurrent versions" to **Glacier Deep Archive**

Amazon S3 Analytics – Storage Class Analysis

- Help you decide when to transition objects to the right storage class
- Recommendations for Standard and Standard IA
  - Does NOT work for One-Zone IA or Glacier
- Report is updated daily
- 24 to 48 hours to start seeing data analysis

- Good first step to put together Lifecycle Rules (or improve them)!

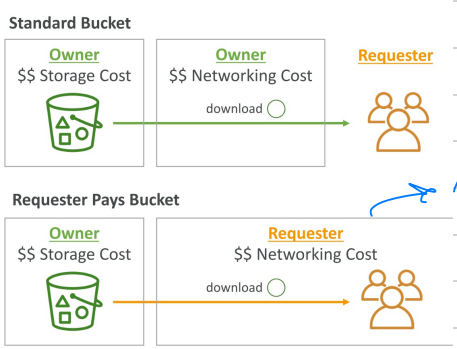


Date	StorageClass	ObjectAge
8/22/2022	STANDARD	000-014
8/25/2022	STANDARD	030-044
9/6/2022	STANDARD	120-149

helps to decide lifecycle rule.

S3 – Requester Pays

- In general, bucket owners pay for all Amazon S3 storage and data transfer costs associated with their bucket
- With Requester Pays buckets, the requester instead of the bucket owner pays the cost of the request and the data download from the bucket
- Helpful when you want to share large datasets with other accounts
- The requester must be authenticated in AWS (cannot be anonymous)



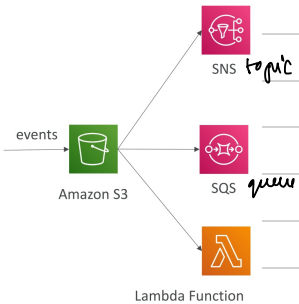
→ in general, pay for storage + networking

→ the requesters will pay!

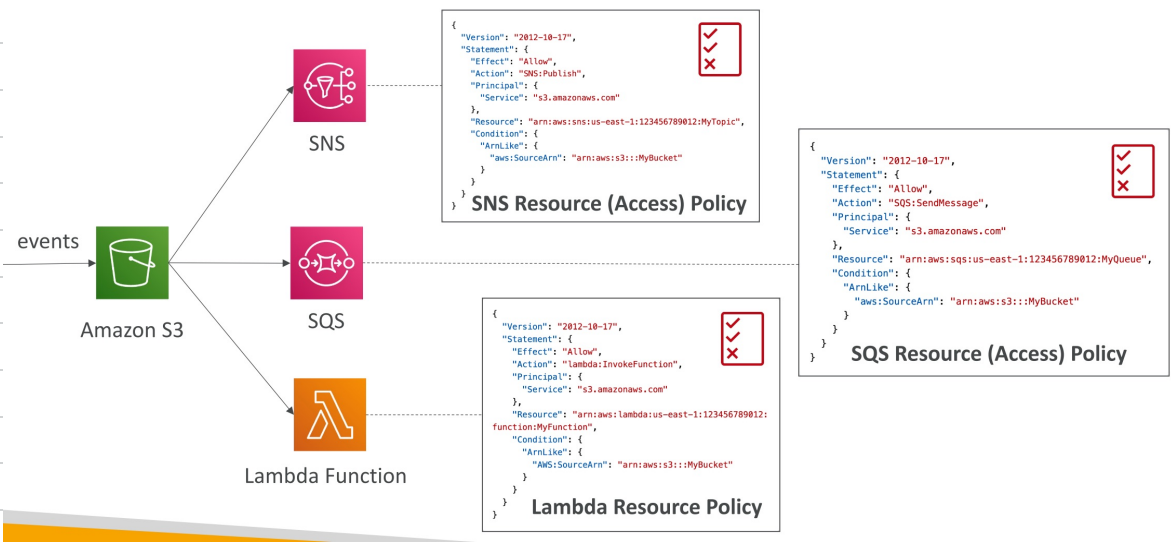
↳ that's when AWS will be able to bill them

S3 Event Notifications

- 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

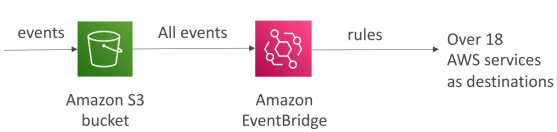


S3 Event Notifications – IAM Permissions



event notification target → SNS topics, SQS queues, 2 functions

Event bridge → 18 AWS services as destinations.



- **Advanced filtering** options with JSON rules (metadata, object size, name...)
- **Multiple Destinations** – ex Step Functions, Kinesis Streams / Firehose…
- **EventBridge Capabilities** – Archive, Replay Events, Reliable delivery

to setup event notification !

create bucket > bucket properties > Event notification {create}
+
Amazon Event Bridge {integration} turn on !

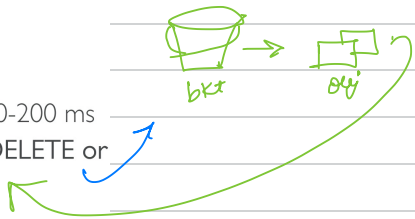
Create a event-notification:

(1) event name
(2) event types → object creation
→ removal
→ restore
→ object ACL
→ obj-tagging
→ Reduced redundancy use.
→ replication
→ life cycle
→ intelligent tiering
(3) destination → SQS → create SQS queue & change the access policy to enable S3 bucket to write in the SQS queue
SNS

policy generator. Add a policy for SQS queue using → very permissive
principal = *
aws service = SQS
Action = Send Message
ARN = {* of the SQS queue *}

(4) Create event.

S3 – Baseline Performance

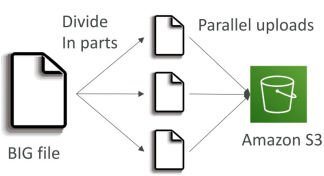


- Amazon S3 automatically scales to high request rates, latency 100-200 ms
- Your application can achieve at least 3,500 PUT/COPY/POST/DELETE or 5,500 GET/HEAD requests per second per prefix in a bucket.
- There are no limits to the number of prefixes in a bucket.
- Example (object path => prefix):
  - 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

5500 x 4

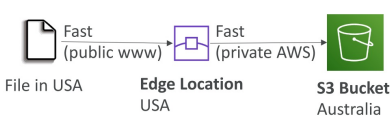
- Multi-Part upload:
  - recommended for files > 100MB, must use for files > 5GB
  - Can help parallelize uploads (speed up transfers)

only uploads !



- S3 Transfer Acceleration
  - Increase transfer speed by transferring file to an AWS edge location which will forward the data to the S3 bucket in the target region using AWS private network
  - Compatible with multi-part upload

uploads + downloads !

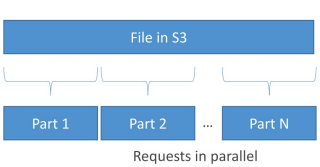


S3 Performance – S3 Byte-Range Fetches

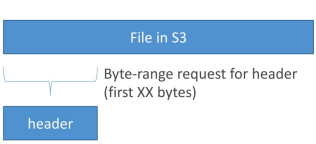
for reading file!

- Parallelize GETs by requesting specific byte ranges
- Better resilience in case of failures

Can be used to speed up downloads



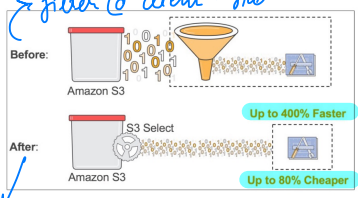
Can be used to retrieve only partial data (for example the head of a file)



S3 Select & Glacier Select

- Retrieve less data using SQL by performing server-side filtering
- Can filter by rows & columns (simple SQL statements)
- Less network transfer, less CPU cost client-side

filter @ client - side



https://aws.amazon.com/blogs/aws/s3-glacier-select/

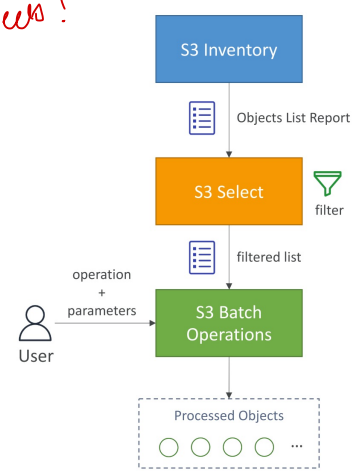
filter @ server side

so that we have a small file to download

S3 Batch Operations

on a list of objects!

- Perform bulk operations on existing S3 objects with a single request, example:
  - Modify object metadata & properties
  - Copy objects between S3 buckets
  - Encrypt un-encrypted objects
  - Modify ACLs, tags
  - Restore objects from S3 Glacier
  - Invoke Lambda function to perform custom action on each object
- A job consists of a list of objects, the action to perform, and optional parameters
- S3 Batch Operations manages retries, tracks progress, sends completion notifications, generate reports …
- You can use S3 Inventory to get object list and use S3 Select to filter your objects

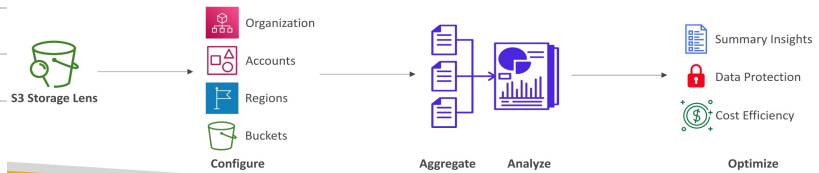


{ S3 Inventory + S3 Select }

to get the list of objects to perform job on!

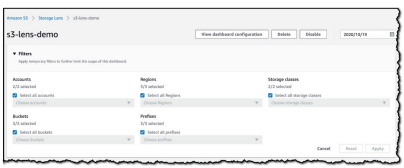
S3 – Storage Lens

- Understand, analyze, and optimize storage across entire AWS Organization
- Discover anomalies, identify cost efficiencies, and apply data protection best practices across entire AWS Organization (30 days usage & activity metrics)
- Aggregate data for Organization, specific accounts, regions, buckets, or prefixes
- Default dashboard or create your own dashboards
- Can be configured to export metrics daily to an S3 bucket (CSV, Parquet)

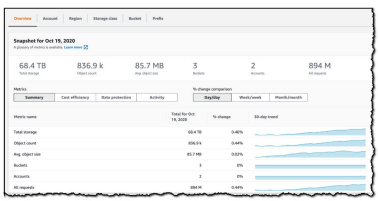


Storage Lens – Default Dashboard

- Visualize summarized insights and trends for both free and advanced metrics
- Default dashboard shows Multi-Region and Multi-Account data
- Preconfigured by Amazon S3
- Can't be deleted, but can be disabled



https://aws.amazon.com/blogs/aws/s3-storage-lens/



https://aws.amazon.com/blogs/aws/s3-storage-lens/

Storage Lens – Metrics

(1) Summary metrics

→ general insights {Storage Bytes, Obj count}
→ identify fastest growing buckets & prefixes.

(2) Cost – Optimization Metrics

→ Insights to manage & optimize storage cost
{Non Current Version Storage Bytes, Incomplete Multipart upload Storage Bytes}
→ identify buckets with incomplete multipart upload which is older than 7 days
→ identifying which objects could be transitioned to lower storage tiers

(3) Data – Protection Metrics

→ insights for data protections {MFA Delete, Versioning Enabled SSE KMS Enable Bucket Count}
→ identify buckets that aren't following data protection best practices.

(4) Access - management metrics

→ insights on S3 obj. ownership.
→ identify which obj. ownership setting your buckets use.

(5) Event Metrics

→ insights on S3 event Notification
{ Event Notification Enabled Bucket Count }
→ identify which buckets have S3 Event Notification config's.

(6) Performance metrics

→ insights for S3 transfer Acceleration
→ identifies which buckets have S3 transfer acceleration enabled

(7) Activity Metrics

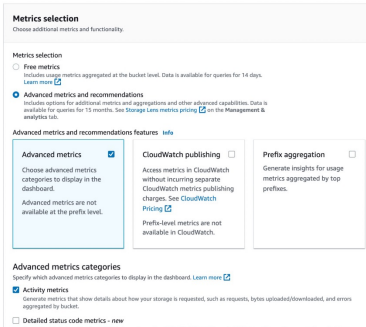
→ insights on how storage is requested
→ All Requests, Get Requests, Put Requests, List Requests, Bytes Downloaded

(8) Detailed status code Metrics

→ Insights for HTTP status code
→ 200 OK Status Count, 403 Forbidden Error Count, 404 Not Found Error Count

Storage Lens – Free vs. Paid

- Free Metrics
  - Automatically available for all customers
  - Contains around 28 usage metrics
  - Data is available for queries for 14 days
- Advanced Metrics and Recommendations
  - Additional paid metrics and features
  - **Advanced Metrics** – Activity, Advanced Cost Optimization, Advanced Data Protection, Status Code
  - **CloudWatch Publishing** – Access metrics in CloudWatch without additional charges
  - **Prefix Aggregation** – Collect metrics at the prefix level
  - Data is available for queries for 15 months



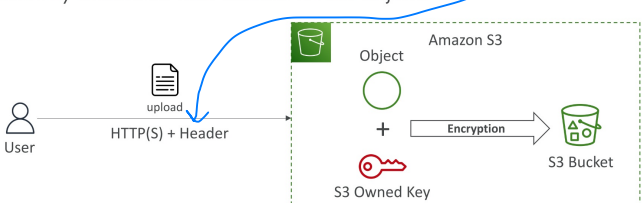
#S3 Security !

Amazon S3 – Object Encryption

- You can encrypt objects in S3 buckets using one of 4 methods
- Server-Side Encryption (SSE)
  - Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3) – Enabled by Default
    - Encrypts S3 objects using keys handled, managed, and owned by AWS
  - Server-Side Encryption with KMS Keys stored in AWS KMS (SSE-KMS)
    - Leverage AWS Key Management Service (AWS KMS) to manage encryption keys
  - Server-Side Encryption with Customer-Provided Keys (SSE-C)
    - When you want to manage your own encryption keys
- Client-Side Encryption
- It's important to understand which ones are for which situation for the exam

Amazon S3 Encryption – SSE-S3

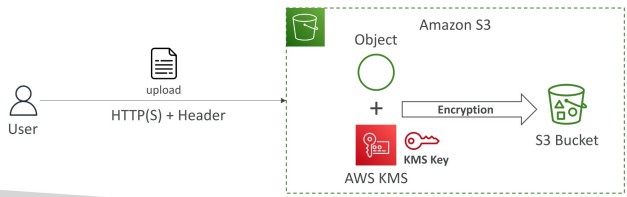
- Encryption using keys handled, managed, and owned by AWS
- Object is encrypted server-side
- Encryption type is AES-256
- Must set header "x-amz-server-side-encryption": "AES256"
- Enabled by default for new buckets & new objects



Amazon S3 Encryption – SSE-KMS

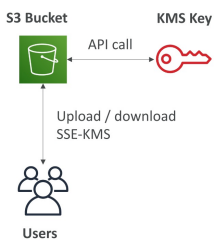
managing your own key using AWS KMS

- Encryption using keys handled and managed by AWS KMS (Key Management Service)
- KMS advantages: user control + audit key usage using CloudTrail
- Object is encrypted server side
- Must set header "x-amz-server-side-encryption": "aws:kms"



SSE-KMS Limitation

- If you use SSE-KMS, you may be impacted by the KMS limits
- When you upload, it calls the GenerateDataKey KMS API
- When you download, it calls the Decrypt KMS API
- Count towards the KMS quota per second (5500, 10000, 30000 req/s based on region)
- You can request a quota increase using the Service Quotas Console

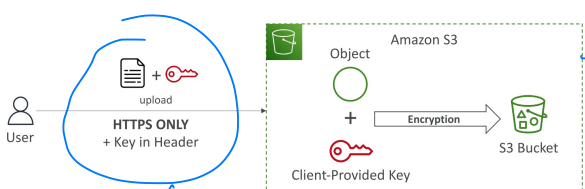


If S3 bucket in a high throughput work environment this will exceed the KMS quotas & cause a throttling kind of use case.

Amazon S3 Encryption – SSE-C

user manages keys outside AWS

- Server-Side Encryption using keys fully managed by the customer outside of AWS
- Amazon S3 does NOT store the encryption key you provide
- HTTPS must be used
- Encryption key must provided in HTTP headers, for every HTTP request made

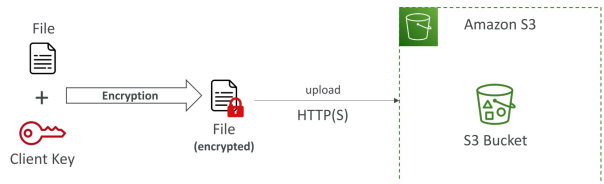


for encrypt & decrypt user must provide the key.

Because these keys are being transmitted hence in-flight encryption is necessary.

Amazon S3 Encryption – Client-Side Encryption

- Use client libraries such as Amazon S3 Client-Side Encryption Library
- Clients must encrypt data themselves before sending to Amazon S3
- Clients must decrypt data themselves when retrieving from Amazon S3
- Customer fully manages the keys and encryption cycle



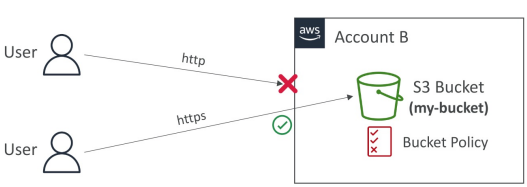
Amazon S3 – Encryption in transit (SSL/TLS)

- Encryption in flight is also called SSL/TLS

- Amazon S3 exposes two endpoints:
  - HTTP Endpoint – non encrypted
  - HTTPS Endpoint – encryption in flight

- HTTPS is recommended
- HTTPS is mandatory for SSE-C
- Most clients would use the HTTPS endpoint by default

Amazon S3 – Force Encryption in Transit aws:SecureTransport




```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-bucket/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

Amazon S3 – Default Encryption vs. Bucket Policies

- SSE-S3 encryption is automatically applied to new objects stored in S3 bucket
- Optionally, you can "force encryption" using a bucket policy and refuse any API call to PUT an S3 object without encryption headers (SSE-KMS or SSE-C)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "s3:PutObject",
            "Principal": "*",
            "Resource": "arn:aws:s3:::my-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "s3:PutObject",
            "Principal": "*",
            "Resource": "arn:aws:s3:::my-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "false"
                }
            }
        }
    ]
}
```

- Note: Bucket Policies are evaluated before "Default Encryption"

Q What is CORS ?

→ CROSS Origin Resource Sharing

Origin = scheme (protocol) + host (domain) + port (implied port)

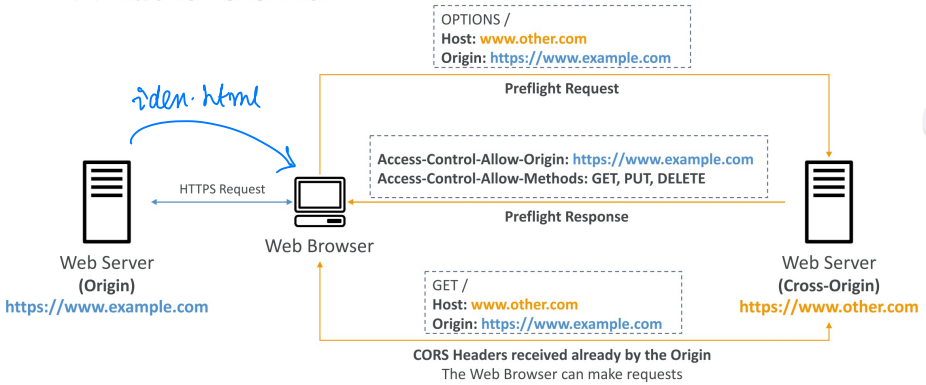
https://www.example.com { port = 443 }

CORS is a web browser based mechanism to allow requests to other origin while visiting the main origin.

Same origin	different origin
http://www.example.com/app1	http://www.example.com/
http://www.example.com/app2	http://other.example.com

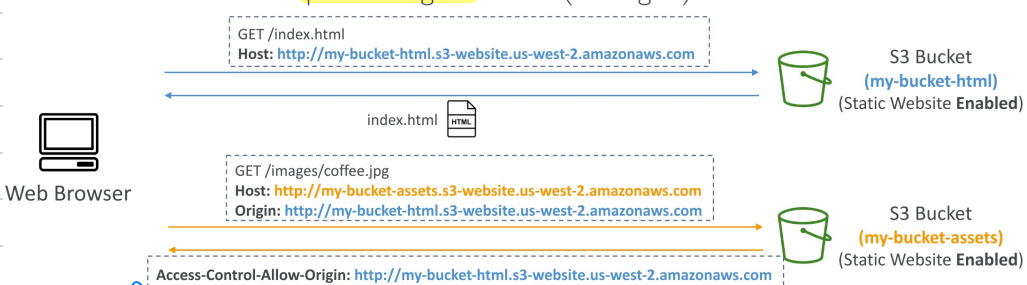
Request won't be fulfilled unless the other origin allows for the requests using CORS headers {Access-Control-Allow-Origin}

What is CORS?



index.html ways get an image from other web servers.

- If a client makes a cross-origin request on our S3 bucket, we need to enable the correct CORS headers
- It's a popular exam question
- You can allow for a specific origin or for * (all origins)



↳ if allowed then the assets will be sent

bucket > permissions > bucket policy to add bucket policies

also

permissions > Cross-Origin resource sharing (CORS)

add a JSON document

CORS_CONFIG.json

```
[
    {
        "AllowedHeaders": [
            "Authorization"
        ],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
            "<url of first bucket with http://...without slash at the end>"
        ],
        "ExposeHeaders": [],
        "MaxAgeSeconds": 3000
    }
]
```

Access control allow method

Access control Allow orig's

Amazon S3 – MFA Delete

- MFA (Multi-Factor Authentication) – force users to generate a code on a device (usually a mobile phone or hardware) before doing important operations on S3
- MFA will be required to:
  - Permanently delete an object version
  - Suspend Versioning on the bucket
- MFA won't be required to:
  - Enable Versioning
  - List deleted versions
- To use MFA Delete, Versioning must be enabled on the bucket
- Only the bucket owner (root account) can enable/disable MFA Delete

destructive operations.

Google Authenticator

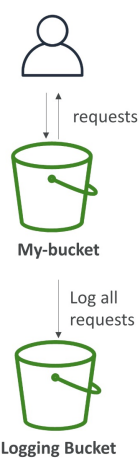
MFA Hardware Device

S3 Access Logs

- For audit purpose, you may want to log all access to S3 buckets
- 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…
- The target logging bucket must be in the same AWS region

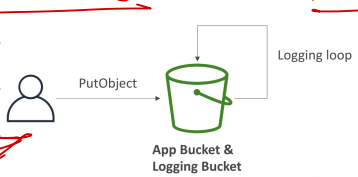
but never the same bucket

- The log format is at: https://docs.aws.amazon.com/AmazonS3/latest/dev/LogFormat.html



→ never ever set the logging bucket to be monitored bkt

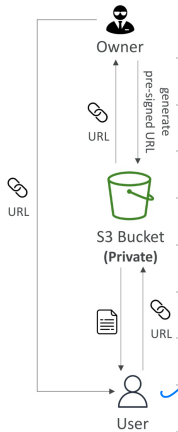
bucket will grow exponentially



Amazon S3 – Pre-Signed URLs

- Generate pre-signed URLs using the S3 Console, AWS CLI or SDK
- URL Expiration
  - S3 Console – 1 min up to 720 mins (12 hours)
  - AWS CLI – configure expiration with *--expires-in* parameter in seconds (default 3600 secs, max. 604800 secs ~ 168 hours)
- Users given a pre-signed URL inherit the permissions of the user that generated the URL for GET / PUT

- Examples:
  - Allow only logged-in users to download a premium video from your S3 bucket
  - Allow an ever-changing list of users to download files by generating URLs dynamically
  - Allow temporarily a user to upload a file to a precise location in your S3 bucket



→ no need to make a bucket public!

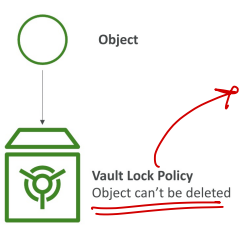
can be any user outside the AWS

console ka action hai "share the pre-signed URL"

& time kitna valid for.

S3 Glacier Vault Lock

- Adopt a WORM (Write Once Read Many) model
- Create a Vault Lock Policy
- Lock the policy for future edits (can no longer be changed or deleted)
- Helpful for compliance and data retention



on top of glacier BUCKET.

can never be deleted by administration or AWS.

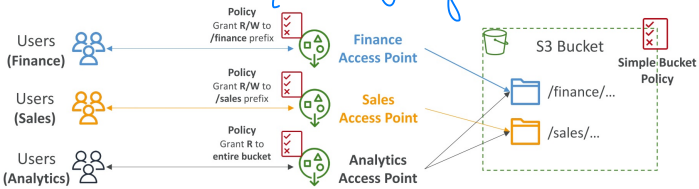
S3 Object Lock (versioning must be enabled)

lock not @ bkt level but @ the object level. i.e. object lock

- Adopt a WORM (Write Once Read Many) model
- Block an object version deletion for a specified amount of time
- Retention mode - Compliance: strict
  - Object versions can't be overwritten or deleted by any user, including the root user
  - Objects retention modes can't be changed, and retention periods can't be shortened
- Retention mode - Governance: lenient
  - Most users can't overwrite or delete an object version or alter its lock settings
  - Some users have special permissions to change the retention or delete the object
- Retention Period: protect the object for a fixed period, it can be extended
- Legal Hold:
  - protect the object indefinitely, independent from retention period
  - can be freely placed and removed using the *s3:PutObjectLegalHold* IAM permission

can put or release lock

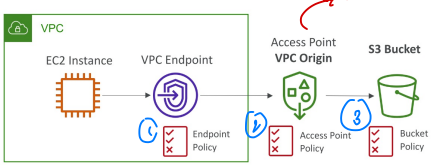
S3 – Access Points



- Access Points simplify security management for S3 Buckets
- Each Access Point has:
  - its own DNS name (Internet Origin or VPC Origin)
  - an access point policy (similar to bucket policy) – manage security at scale

S3 – Access Points – VPC Origin

- We can define the access point to be accessible only from within the VPC
- You must create a VPC Endpoint to access the Access Point (Gateway or Interface Endpoint)
- The VPC Endpoint Policy must allow access to the target bucket and Access Point




```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::awsexamplebucket1/*",
        "arn:aws:s3:us-west-2:123456789012:accesspoint/example-vpc-ap/object/*"
      ]
    }
  ]
}
```

S3 Object Lambda

- Use AWS Lambda Functions to change the object before it is retrieved by the caller application
- Only one S3 bucket is needed, on top of which we create S3 Access Point and S3 Object Lambda Access Points.
- Use Cases:
  - Redacting personally identifiable information for analytics or non-production environments.
  - Converting across data formats, such as converting XML to JSON.
  - Resizing and watermarking images on the fly using caller-specific details, such as the user who requested the object.