3 S3

S3 security

S3 is a private service by default, meaning that all newly created buckets are private and can only be accessed by the owner(root user). To give access to other users you need to create policies (bucket policies, IAM policies, ACLs).

Bucket policies, IAM policies, ACLs

- ***Bucket policies***:
  - JSON-based policies applied at the bucket level
  - Allow cross-account access from other AWS accounts or anonymous access
  - recognizable because always has a `Principal` element
- ***IAM policies***:
  - JSON-based policies applied at the user/group/role level
  - Allow cross-account access but only for IAM users, not anonymous. Because you need an IAM user to attach policies to.
- ***ACLs***: Legacy access control method, use bucket policies or IAM policies instead (no longer used because limited)

Static website hosting

- S3 can host static websites and have them accessible on the internet

- The website URL will be `http://<bucket-name>.s3-website-<region>.amazonaws.com` or you can use a custom domain with Route 53

- To enable static website hosting:

  1. go to the bucket properties and enable it
  2. upload the website files to the bucket (index.html, error.html, etc.)
  3. add a bucket policy to allow public read access to the bucket

- use cases:

  - off-loading of images, videos, etc. cheaper than using EC2 with EBS or Database
  - out-of-band-pages: error pages, maintenance pages, etc. because actual server might be down

S3 versioning and MFA delete

Object versioning is a feature which can be enabled on an S3 bucket - allowing the bucket to store multi versions of objects.

- These objects can be referenced by their version ID to interact directly - or omit this to reference the latest version of an object.

- Objects aren't deleted - object deletion markers are put in place to hide objects. Versioning is an essential feature to understand for the exam.
- S3 versioning:
  - great backup tool
  - stores all versions of an object (also versions with delete markers) so all versions use storage
  - by default, versioning is disabled. id of object is null if versioning is disabled.
  - once enabled, id of object is no longer null.
  - once enabled, versioning cannot be disabled, only suspended. When suspended it can be enabled again.
  - you can delete a version of an object, but it will still be stored in the bucket with a delete marker. If you remove the delete marker, the object will be restored.
  - you can permanently delete a version of an object. Then the object and id will be removed from the bucket.
  - integrates with lifecycle rules

MFA delete

- MFA delete is a feature that uses multi-factor authentication to delete objects in a bucket.
- It can be enabled on a bucket to require MFA authentication to delete objects or change versioning state (suspended or enabled).

S3 Perfomance optimization

Uploading data

- ***Single PUT upload (default)***:
  - Object is uploaded as a single blob of data in a single stream.
  - if stream fails, you need to start over
  - Limited to 5GB (but don't use it for files larger than 100MB = minimum size for multipart upload)
  - API call: s3:PutObject
  - speed and reliabilty is limited to the speed of the network (lowest speed of the network)
- ***Multipart upload***:
  - Object is uploaded in parts (5MB - 5GB each) and parts can be uploaded in parallel. (only last part can be smaller than 5MB)
  - **minimum data size is 100MB**
  - maximum of 10,000 parts per upload
  - if part fails, only that part needs to be re-uploaded
  - improves speed and reliability of uploads
  - **recommended for files larger than 100MB**
  - API call: s3:CreateMultipartUpload, s3:UploadPart, s3:CompleteMultipartUpload
  - Multipart upload can be done by:
    - AWS SDK
    - AWS CLI
    - AWS console
    - Lifecycle rules
    - Multipart upload API

S3 Transfer Acceleration

- Transfer Acceleration utilizes the CloudFront Edge Network to accelerate uploads to S3.
- You can't have a bucket name with a period (.) in it when using Transfer Acceleration. (because it will be interpreted as a subdomain)
- Instead of uploading directly to S3, you can use a distinct URL to upload directly to an edge location which will then transfer the data to S3 using Amazon's optimized network paths. Instead of using the public internet. The URL will be `http://<bucket-name>.s3-accelerate.amazonaws.com` and is given when you enable Transfer Acceleration on the bucket.

Key Management Service (KMS)

KMS

- ***Regional and public service***
- ***FIPS 140-2(Level 2) compliant***
- KMS is integrated with many AWS services
- KMS is a managed service that allows you to create and control the encryption keys used to encrypt your data
- KMS can perform cryptographic operations on your behalf (encryption, decryption, signing, etc.)
- KMS keys can be used to encrypt data up to 4KB in size (otherwise use Data Encryption Keys)
- Supports both symmetric and asymmetric keys:
  - ***Symmetric keys***:
    - single key used to encrypt and decrypt data
    - used for encryption and decryption
    - key material never leaves KMS
    - key material is never exposed
    - key material is never stored on disk
    - key material is never transmitted outside KMS
  - ***Asymmetric keys***:
    - public and private key pair
    - used for encryption and decryption
    - public key can be shared
    - private key never leaves KMS
    - private key is never exposed
    - private key is never stored on disk

KMS keys

- KMS keys are logical - id, date, policy, description, state
- KMS keys are regional resources
- Are generated by KMS or imported by you
- Are backed by physical key material (HSMs)

KMS key policies

- KMS key policies are JSON-based policies that allow you to control who can use your keys
- KMS key policies are separate from IAM policies

- KMS key policies are attached to the key itself (kind of like a bucket policy)
- you have to explicitly allow an IAM user of the account to use the KMS key
- use Key policy + IAM policy to control access to KMS keys

Data encryption Keys (DEK)

- Data encryption keys are used to encrypt and decrypt data larger than 4KB
- Data encryption keys are generated by KMS and are used to encrypt data
- Data encryption keys are encrypted by a KMS key
- Data encryption keys are stored with the encrypted data

S3 encryption - SSE and CSE

S3 encryption at rest

Encryption can be done by the following methods:

- ***Server Side Encryption*** with S3-managed keys (SSE-S3) is a server-side encryption is mandatory for S3 buckets. It encrypts the object data with a key that is managed by S3. Data is encrypted server-side before being written to disk.
- ***Client Side Encryption (CSE)*** is a client-side encryption option that allows you to encrypt data client-side before uploading it to S3. Data is encrypted client-side before being uploaded to S3. The key is managed client-side.

Buckets aren't encrypted but objects are! Bucket encryption is something different, it's used to encrypt the bucket itself, not the objects in it. SSE and CSE are used to encrypt the objects in the bucket.

SSE-S3

There are 3 types of server-side encryption:

- SSE-C: Server-side encryption with customer-provided keys:
  - you provide the key to encrypt the object (diffrent than CSE!)
  - when you supply an object to S3, you must include the encryption key in the request
  - one way hash of key is stored with the object and then the key is destroyed
  - to decrypt the object, you must provide the same key
- SSE-S3: Server-side encryption with S3-managed keys (default):
  - S3 manages the keys for you, you have no control over the keys
  - uses ***AES-256 encryption***
  - not suitable for sensitive data or highly regulated industries (finance, healthcare, etc.) because s3 admin can see the decrypted data
- SSE-KMS: Server-side encryption with KMS keys:
  - you manage the keys using KMS
  - you can make sure that admins can only see encrypted data through role seperation by giving no KMS decryption permissions to the admin role
  - KMS gives Data encryption key (DEK) to encrypt the data and KMS key to encrypt the DEK. The KMS key is then discarded in S3 but still present in the KMS service.

Bucket encryption

Amazon S3 Bucket Keys reduce the cost of Amazon S3 server-side encryption using AWS Key Management Service (SSE-KMS). Bucket-level keys for SSE benefits over SSE-KMS:

- decrease the request traffic from Amazon S3 to AWS KMS -> each object stored in S3 requires sends an API call to KMS to encrypt the data key (DEK) using a KMS key (is often 1 and the same key)
- can reduce AWS KMS request costs by up to 99 percent (calls to KMS have a cost)
- KMS has 5,500 /10000 /50 000 requests per second limit (depends on region) (and 1000 requests per second limit for each KMS key (a verifier))

Bucket encryption

KMS generates time limited bucket key for SSE-S3 and the key is given to the bucket. The bucket key is then used inside S3 to generate any data encryption keys (DEK) inside the bucket for individual object encryption operations.

- offloads the encryption and decryption operations from the KMS service to the S3 service and thus reduces the number of requests to the KMS service. (less cost and improves scalability)
- not retroactive, only new objects are encrypted with the bucket key

Attention points for bucket keys:

- CloudTrail shows bucket ARN instead of Object ARN in KMS logs
- Works with replication:
  - the replicated object is encrypted uses the same settings as the source object, the replicated object is encrypted in the destination bucket with the same bucket key.
  - If the source is unencrypted, but the destination bucket has a bucket key, the object will be encrypted with the bucket key in the destination bucket. This results in the ETag of the source object being different from the ETag of the replica object.

S3 storage classes

S3 Standard

- ***99.999999999% (11 nines) availability*** -> if you store 10,000,000 objects you can expect to lose 1 object every 10,000 years
- because replication of objects in at least 3 AZs
- checks of data corruption using content MD5 checksums and CRCs (Cyclic Redundancy Checks) are used to detect and fix data corruption
- when object is successfully stored, you get a 200 OK response
- pricing:
  - storage cost: per GB per month
  - request cost: per 1000 requests
  - data transfer cost: IN free
  - ***no retrieval cost, no minimum duration and no minimum size*** (not the case for other classes)
- retrieval time: milliseconds first byte latency
- objects can be publicly available
- ***uses cases exam: frequently accessed data that is important and non replaceable***

S3 Standard-IA

- architecture is very similar to S3 Standard
- differences in pricing, retrieval cost, minimum duration and minimum size
- pricing:
  - storage cost: per GB per month (cheaper than S3 Standard)
  - request cost: per 1000 requests (cheaper than S3 Standard)
  - data transfer cost: IN free
  - ***retrieval cost: per GB retrieved***
  - ***minimum duration: 30 days***
  - ***minimum size: 128KB***
- ***use cases exam: used for long lived data, which is important but where access is infrequent***

S3 One Zone-IA

- Instead of 3 AZs, data is stored in a single AZ
- but still has 99.999999999% (11 nines) durability
- same limits as S3 Standard-IA (minimum duration, minimum size, retrieval cost)
- can be used with cross-region replication
- ***use cases exam: used for long lived data, which is NON critical and REPLACEABLE and where access is infrequent***

S3 Glacier-Instant

Like S3 Standard-IA but with cheaper storage, more expensive retrieval and longer minimum storage but still instant retrieval

- Object is stored for minimum 90 days. Objects can be stored for less but minimum billing always applies.
- you still have instant access to the data like S3 Standard and S3 Standard-IA it just costs more to retrieve the data. But cost less to store the data.

S3 Glacier-flexible

- Conceptually think of it as cold objects, they're not warm so they take longer to retrieve.
- Objects cannot be made public, any data retrieval process (beyond metadata) requires a retrieval process. During this process, the object is temporarily restored to S3 Standard-IA for retrieval:
  - ***Expedited***: 1-5 minutes
  - ***Standard***: 3-5 hours
  - ***Bulk***: 5-12 hours
  - -> faster is more expensive: ***first byte latency*** from minutes to hours (Need to know for exam)
- pricing:
  - 1/6th the cost of S3 Standard
  - 40kb minimum object size (minimum billing always applies)
  - minimum storage duration of 90 days (minimum billing always applies)

S3 Glacier-Deep Archive

- ***Cheapest storage class***, but ***longest retrieval time*** (12 hours)

- Conceptually think of it as deep frozen objects, they're not warm so they take longer to retrieve.
- Objects cannot be made public, any data retrieval process (beyond metadata) requires a retrieval process. During this process, the object is temporarily restored to S3 Standard-IA for retrieval:
    - ***Standard***: 12 hours
    - ***Bulk***: up to 48 hours
    - -> faster is more expensive: ***first byte latency*** between hours and days (Need to know for exam)
- pricing:
    - 40kb minimum object size (minimum billing always applies)
    - minimum storage duration of 180 days (minimum billing always applies)
    - ***use cases exam: use for archive data that rarely if ever needs to be accessed with retrieval between hours and days. eg Legal or Regulation data storage ***

S3 Intelligent-Tiering

- Storage class that contains 5 different tiers:
    - ***Frequent Access***: like S3 Standard
    - ***Infrequent Access***: like S3 Standard-IA
    - ***Archive Instant Access***: like S3 Glacier-Instant when not accessed for 90 days
    - ***Archive Access***: like S3 Glacier-flexible:
        - optional
        - when object not accessed between 90 and 270 days
        - no Instant retrieval -> app needs to be able to handle asynchronous retrieval
    - ***Deep Archive Access***: like S3 Glacier-Deep Archive:
        - optional
        - when object not accessed for 180 days or 730 days
        - (no Instant retrieval -> app needs to be able to handle asynchronous retrieval)
- pricing:
    - monitoring and automation cost per 1000 objects monitored
    - costs are similar to S3 equivalent base storage classes
- ***use cases exam: use for long lived data where usage pattern is changing or unknown***

S3 Lifecycle Configuration

Allows objects storage classes to be changed and objects deleted automatically. Step through S3 lifecycle management/configuration/rules both in theory and via an S3 console example

- You can create lifecycle rules on S3 buckets which can auto-transition or expire objects in a bucket. Great for objects that have a life cycle (activity on object is predictable)
- Lifecycle Configuration is a set of RULES that consist of ACTIONS. Rules can be applied to a whole Bucket or groups of objects in a bucket if defined by prefixes/tags
- Two types of Actions applied (both work on versions if Object Versions are enabled on bucket): -- 1. Transition Actions: Change storage class of affected objects -- 2. Expiration Actions: Delete objects/object versions EXAM: S3 ONE ZONE-IA CANNOT transition into S3 Glacier-Instant Retrieval. Transition can't go up the waterfall of classes, only down. See waterfall visual EXAM: Smaller objects can cost MORE (minimum size) EXAM: If object initially placed in Standard class, there is 30 day minimum period where an object needs to remain on S3 Standard before transition down the waterfall

EXAM: A SINGLE rule CANNOT transition to Standard-IA or One Zone-IA and THEN to glacier classes within 30 days (duration minimum); must remain in IA class for min 30 days before moving to Glacier

S3 Replication

S3 has two replication features which allow objects to be replicated between SOURCE and DESTINATION buckets in the same or different AWS accounts

S3 Replication - Two Types:

- Cross-Region Replication (CRR) is the process used when Source and Destination are in different AWS regions
- Same-Region Replication (SRR) is used when the buckets are in the same region.

S3 Replication - The Difference:

With Cross-Region Replication, The Destination Bucket, b/c it's in a different AWS account, doesn't trust the Source account or its IAM Role that gets created for Replication purposes by Source Bucket. If configuring replication between different accounts, you have to create a Bucket Policy in the Destination Bucket which allows the Source IAM role to write/replicate into the Destination bucket.

S3 Replication - Options:

1st option: What you replicate. Default is entire Source bucket to Destination bucket (everything in it), OR subset.

- For subset, create a rule that adds a filter to filter objects by prefix or tags 2nd: Storage Class to be used. The Default is to use the same class, but you can choose a cheaper class if this data is secondary. EXAM: Default Storage Class to use for S3 Replication is to maintain the same storage class as found on the Source Bucket, but you can override that in the replcation configuration (good if Destination Bucket will contain secondary data) 3rd: Ownership. Define ownership of objects in the Destination Bucket. Default is that dest buckets are owned by same entity that owns Source - This may not work if Source/Dest buckets are in different accounts (Dest bucket may not be able to read objects if objects are owned by Source). 4th: Replication Time Control (RTC): Adds guaranteed 15 minute SLA onto this process. This is used to make sure Source/Destination are in sync as closely as possible. EXAM: 15 minute replication mentioned on exam? Then you know you need Replication Time Control (RTC)

S3 Replication - Considerations:

EXAM:

- By Default, Replication is NOT retroactive and Versioning needs to be ON. Eg. If you enable Replication on a bucket that already has objects, those objects will not be replicated.
- Both Source AND Destination bucket need Versioning enabled
- You can use S3 Batch Replication to replicate pre-existing objects (since Replication is NOT retroactive by default)
- One-way replication: If you manually add objects to Destination Bucket, they will NOT be replicated to Source -- Bi-directional replication is a recently added feature, but is an additional setting that needs

to be configured
- Replication Capability: can handle unencrypted, SSE-S3 & SSE-KMS (with extra config), SSE-C
- Source Bucket Owner needs Permissions on the objects which will replicate
- Will NOT replicate: System events, Glacier, or Glacier Deep Archive
- NO DELETE markers are NOT replicated by Default. You can enable with DeleteMarkerReplication

S3 Replication - Why Use It?

- For SRR (same region): For log aggregation, synchronize PROD and TEST accounts, resilience with strict sovereignty requirements (some data cannot leave specific regions)
- For CRR (cross region): Global resilience improvements, latency reduction

S3 Replication - Resources:

- https://docs.aws.amazon.com/AmazonS3/latest/dev/replication.html
- https://aws.amazon.com/about-aws/whats-new/2019/11/amazon-s3-replication-time-control-for-predictable-replication-time-backed-by-sla/

DEMO Cross-Region Replication of an S3 Static Website

Create 2 S3 buckets - one in N. Virginia, the other in N. California and configure Cross-Region Replication (CRR) between the two.

1. iamadmin account, N.VA region selected
2. Nav to S3 > Create Source Bucket > Create Bucket > name of source "sourcebucketta[random number]" > region us-east-1 > Create Bucket
3. Enable Static Website on Source > Properties Tab > Static website hosting: edit: Enable > type: static website > index doc "index.html"/error doc "index.html" > Save changes
4. Edit Source bucket Permissions for Public Access > Permissions tab > uncheck Block All Public Access > confirm > Save changes
5. To made Source Bucket public, add policy > Permissions tab > Bucket Policy "edit" > from lesson docs, paste JSON, replace Resource arn before the "/*" > Save
6. Create Destination Bucket & Set Permissions/Policy > Create Bucket > name of dest "destinationbucketta[random number]" > AWS Region: us-west-1 > uncheck Block All Public Access > Properties tab, Enable Static website hosting > Hosting type: static > index/error docs "index.html" > Save changes > Permissions tab, edit Bucket policy, paste JSON, update ARN, Save
7. Enable Cross-Region Replication (CRR): Source Bucket Management tab > Create replication rule > Enable Versioning > Replication rule name "staticwebsiteDR" > Status "Enabled" > Choose Rule Scope: "Apply to All objects in the bucket" > Destination: Browse S3, find destination bucket, enable versioning > IAM Role: dropdown "create new role" > Create replication rule > Replicate Existing Obejcts? No (as we have no pre-existing objects)
8. Clean Up: Empty/Delete Destination Bucket > Empty/Delete Source Bucket > IAM: locate role staring with "s3crr_role[...]"

S3 PreSigned URLs

Presigned URL's are a feature of S3 which allows the system to generate a URL with access permissions encoded into it, for a specific bucket and object, valid for a certain time period

- PreSigned URLs can be used for downloads (GET) or Uploads (PUT)

Eg. S3 bucket with NO public access configured. Currently, a user would have to authenticate in IAM and be authorized to access resource. If you need to give an unauthenticated user access to the bucket, there are 3 [un-ideal] solutions 1) give mystery user AWS ID 2) give myst.user credentials 3) make object public.

- Better solution than all this is PreSigned URLs

NOTE: The PreSigned URL is used, the holder of the URL is interacting with S3 as the person who GENERATED the URL. In the example above, the mystery user would be iamadmin

EXAM: You can create a URL for an object that you have NO ACCESS TO. But, since you have no access, the PreSigned URL won't either. Not an applicable use case, but possible EXAM: When using a PreSigned URL, the permissions are the same as the CURRENT permissions of Identity that generated the PreSigned URL. (So Access Denied could mean the generating ID never had access, or doesn't now) EXAM: Don't generate PreSigned URLs using an IAM Role (temp credentials of a Role can expire before temp access to PreSigned URL)

DEMO S3 Creating and using PreSigned URLs

Create a bucket, upload an object and generate a presignedURL allowing access for any unauthenticated identities.

1. Create bucket > name "animals4lifemedia[randomnumber]" > Create bucket
2. Upload object > all5.jpv to new bucket
3. Generate PreSigned ULR > Cloud Shell (terminal icon top right of AWS) > shell command "aws s3 presign s3://animals4lifemedia745675/all5.jpg --expires-in 180"

- 180 is in seconds, 3 mins

4. Clean up: Empty/delete bucket

NOTE: Interesting Aspects...

1. Try "aws s3 presign s3://animals4lifemedia745675/all5.jpg --expires-in 604,800"
2. Nav to IAM > Users > iamadmin > Permissions: Add inline policy, copy JSON from lesson > save
3. In CloudShell, try "aws s3 ls", you'll see Access Denied. This Explicit Deny overrules S3 permissions. Refresh PreSigned URL and see that access is now denied as the current permissions of iamadmin are restricted from S3
4. iamadmin currently restricted from S3, but can still generated a PreSigned URL for it, even with no access.

- You can also generate a PreSigned URL on a non-existent object
- If you generate a PreSigned URL with an assumed Role, the URL will stop working with the temporary creds for the Role stop working
- Can now create PreSigned URL from AWS Dashbord s3 > bucket > object > Object Actions dropdown "Share with a presigned URL"

S3 Select and Glacier Select

EXAM: Understand this architecture S3 and Glacier Select allow you to use a SQL-Like statements to retrieve partial objects from S3 and Glacier.

- Ways you can retrieve PARTS of objects rather than the entire object -- Both S3 and Glacier super scalable. S3 can store objects up to 5TB and store infinite number of these objects. Often, you don't need to retrieve the entire 5TB object
- With S3 Select, filtering occurs on the service/the source (S3); The data delivered by S3 is pre-filtered WITHIN S3, making it quicker and reducing costs

S3 Events

The Amazon S3 notification feature enables you to receive notifications when certain events happen in your bucket. To enable notifications, you must first add a notification configuration that identifies the events you want Amazon S3 to publish and the destinations where you want Amazon S3 to send the notifications. You store this configuration in the notification subresource that is associated with a bucket

- Notifications generated when events occur in a bucket
- Can be delivered to SNS Topics, SQS Queues and Lambda Functions
- Types of events: object Created (Put, Post, Copy, CompleteMultiPartUpload), object Delete (*, Delete, DeleteMarkerCreated), object Restore (Post (Initiated), Completed), Replication
- Events are generated by S3 Service, known as S3 Principal, so we also need to add Resource Policies to the destination services to allow S3 svc to interact
- Events are JSON objects
- S3 Event notifications are dated and limited features. Can also use EventBridge which supports more events and wider svc integration. DEFAULT: use EventBridge as Default for S3 event notifications

S3 Access Logs

Server access logging provides detailed records for the requests that are made to a bucket. Server access logs are useful for many applications. For example, access log information can be useful in security and access audits. It can also help you learn about your customer base and understand your Amazon S3 bill.

- You have a Source bucket where you want to track access. You have a Target bucket where you'll store tracked events. Can be enabled via Console UI or via PUT Bucket Logging
- S3 Logging managed by S3 Log Delivery Group
- Logs delivered as Log Files, consisting of Log Records -- Each attribute within a record is space-delimited, and Records within in a file are newline-delimited
- Single Target bucket can be used for many Source buckets, separate logs using prefixes in Target bucket

S3 Object Lock

You can use S3 Object Lock to store objects using a write-once-read-many (WORM) model. It can help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. You can use S3 Object Lock to meet regulatory requirements that require WORM storage, or add an extra layer of protection against object changes and deletion.

- WORM Model - wrote-once-read-many. Once Object Versions are created, they can't be overwritten or deleted (object versions are locked)
- User can enable on 'new' buckets, support ticket required for adding to existing bucket
- Object Lock required Versioning to be enabled. Once you create/enable Object Lock, you can't disable/delete OL or Versioning
- Can define Object Lock on objects, or a Bucket can have a default Object Lock Settings
- You can overlap the effects, Eg. Having a Legal Hold and Retention: Governance enabled on same object

Object Lock handles retention in TWO ways:

1. Retention Periods
2. Legal Holds

- Can have both, either, or none

S3 Object Lock - Retention Period Methods - Retention Periods and Legal Holds

Retention Period: When you create lock, you specify retention period for DAYS / YEARS

Two Modes of Retention Period (EXAM):

- 1. Compliance Mode. Object version Can't be adjusted, deleted, overwritten for duration of period. Retention period duration can't be reduced. Retention Mode cannot be adjusted during period (EXAM: not even by Account Root User). This is the most strict Object Lock: Retention Period w/ Compliance Mode.
- 2. Governance Mode. Set a retention period, but special permissions can be granted allowing Lock Settings to be adjusted during the retention period -- EXAM: s3:BypassGovernanceRetention in order to allow settings adjustment, along with header along with request "x-amz-bypass-governance-retention:true" (console ui default)

S3 Object Lock - Lock Legal Hold

With Legal Hold, you don't set a Retention Period at all. Instead, for an Object Version, you set Legal Hold ON or OFF; binary. NO RETENTION.

- While Legal Hold flag is on, no deletes and no changes until Legal Hold is removed -- s3:PutObjectLegalHold required to add or remove
- Legal Hold can be used to prevent accidental deletion of critical object versions. Or if you need to flag an object version for a legal case/project etc

S3 Access Points

Amazon S3 Access Points, a feature of S3, simplifies managing data access at scale for applications using shared data sets on S3. Access points are unique hostnames that customers create to enforce distinct permissions and network controls for any request made through the access point.

- You can have 1 bucket with many access points, each access point can have different policies
- Access Points can be limited in terms of WHERE they can be accessed from. Eg. VPC or internet
- Every Access Point has its own endpoint address

- EXAM: Access Point created via Console or with this shell command: -- aws s3control create-access-point --name [name] --account-id [account-id] --bucket [bucket-name]
- Access Points can be thought of as mini buckets or different views of the bucket
- Each Access Point has a unique DNS that would be given to the users using it
- Access Point Policies control permissions for access via Access Point and are functionally equivalent to a Bucket Policy. Access Point Policy can restruct identities to certain prefixes, tags, or actions based on need
- IMPORTANT: Any permissions defined on an Access Point need to also be defined on the Bucket Policy

S3 Access Points Resource:

https://docs.aws.amazon.com/AmazonS3/latest/dev/creating-access-points.html#access-points-policies

Demo S3 Multi-Region Access Points (MRAP)

Gain practical experience working with Multi-region access points

Amazon Simple Storage Service (S3) Multi-Region Access Points provide a global endpoint for routing Amazon S3 request traffic between AWS Regions. Each global endpoint routes Amazon S3 data request traffic from multiple sources, including traffic originating in Amazon Virtual Private Clouds (VPCs), from on-premises data centers over AWS PrivateLink, and from the public internet without building complex networking configurations with separate endpoints.

1. Create two buckets: S3 > Create bucket "multi-region-demo-sydney-[random-number]", region ap-southeast-2, Enable Bucket Versioning > Create > Create 2nd bucket "multi-region-demo-canada-[random-number]", region ca-central-1, Enable Bucket Versioning > Create
2. Create Multi-Region Access Point: S3 > Multi-Region Access Points > Create Multi-Region Access Points > name "reallyreallycriticalcatdata", add the 2 created buckets > Create (MRAP ARN arn:aws:s3::704310952405:accesspoint/m6s3xgw995fsb.mrap) *NOTE: You CANNOT add or remove buckets to Multi-Region Access Point after it's cerated
3. Configure replication between the buckets: Access new MRAP > Replication & Failover tab > template:replicate among all specified buckets, Buckets: check both, Scope: Apply to all objects in bucket > Create replication rules
4. Test with CloudShell > Switch region to Tokyo > open CloudShell > enter "dd if=/dev/urandom of=test1.file bs=1M count=10" > copy ARN from MRAP dashboard, enter "aws s3 cp test1.file s3://[your-copied-mrap-arn]"

- Steps 3 and 4 generate a test file for upload, then uploads to the Sydney bucket. Replication of file to Canada bucket may take a few minutes

5. See remaining steps in the demo file in this repo
6. Clean up: s3 MRAP > select MRAP, delete > s3 buckets Empty and Delete