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 encyption 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***