S3 Introduction

Uday Manchanda

August 23, 2021

1 Buckets and Objects

- Allows people to store objects (files) in buckets (directories)
- Globally unique name, defined at region level

1.1 Objects

- Objects have a key
- The key is the **full path**
- Key is composed of prefix + object name
- There is no concept of "directories" within buckets, although the UI will trick you to think otherwise
- Just keys with very long names that contain slashes
- Object values are the content of the body (max object size is 5TB)
- If uploading more than 5 GB must do multi part upload
- Metadata: list of text key/value pairs (system or user metadata)
- Versioning: you can version your files at the bucket level. Best practice to version your buckets to protect against unintended deletes
- When you delete a file with delete protection it adds a delete marker to it

2 Encryption

- 4 methods to encrypt objects:
  - SSE-S3: encrypts S3 objects using keys handled and managed by AWS
  - SSE-KMS: leverage AWS KMS to manage encryption keys

  - SSE-C: manage your own encryption keys
  - Client side encryption

2.1 SSE-S3

- using keys handled and managed by s3
- encrypted server side, AES256, must set header
- Once you apply the header, S3 knows the object must be encrypted

2.2 SSE-KMS

- Key management service, keys are handled and managed by this service
- user control + audit trail
- server side encryption, must apply a specific header

2.3 SSE-C

- S3 does not store the encryption you provide
- HTTPS must be used, encryption key must be provided in HTTP headers for every HTTP request made

2.4 Client Side Encryption

- clients must encrypt data before sending to s3 and decrypt data when retrieving from s3
- customer fully manages the keys and encryption cycle

2.5 Encryption in transit: SSL/TLS

- Amazon S3 exposes: HTTP endpoint (non encrypted) and HTTPS endpoint (encryption in flight)
- HTTPS is recommended, its usually the one by default

3 Security and Bucket Policies

3.1 Security

- user based: IAM policies (which API calls should be allowed for a specific user from IAM console)

- Resource based: bucket policies (bucket wide rules from s3 console - allows cross account). object access control list (ACL - finer grain). Bucket Access Control list (ACL - less common)
- 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
- Networking: supports VPC endpoints (for instances in VPC w/out www internet)
- Logging and audit: S3 access logs stored in other s3 bucket. API calls logged in cloudtrail

3.2 Bucket Policies

- JSON based policies
- Resources: buckets and objects
- Actions: Set of API to allow or deny
- Effect: allow/deny
- Principal: the account/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)
- Block public access to buckets and objects granted through: new ACLs, any ACLs, or new public bucket or access point policies
- Settings were created to prevent company data leaks

4 Websites

- can host static websites can have them accessible on the web
- bucketname.s3-website-awsregion.amazonaws.com
- Bucket needs to be public: must create policy for it and enable public ACLs

5 CORS

- Cross origin Resource Sharing
- Origin: scheme (protocol), host (domain), and port, eg https://www.google.com, port 443 since https
- We wan to get resources from a different origin
- Web browser based mechanism to allow requests to other origins while visiting the main origin
- Same origin: http://example.com/app1 and http://example.com/app2. we can make requests from the browser from the first to second url
- Different origin: http://www.example.com and http://other.example.com. browser will block this request without necessary CORS headers (Access-Control-Allow-Origin)

5.1 S3 CORS

- if a client does a cross origin request on our S3 bucket, we need to enable the correct S3 headers
- You can allow for a specific origin or for all
- EX: web browser makes GET request to s3 bucket that is website and CORS enabled
- browser sends these headers:
  - GET coffee.jpg
  - ORIGIN: http://urlofbucket.com
- If the bucket is configured with the right CORS headers then the web browser will be able to make the request

6 Consistency Model

- All operations are strongly consistent
- After a: new PUT, overwrite PUT or DELETE
- any: subsequent read request immediately receives the latest version of the object. subsequent list request immediately reflects changes.