Table of contents

Introduction to S3 :

- S3 is our first introduction to serverless cloud computing.
- It scales indefinately as we keep pushing the data. It takes away from operational overhead and lets us focus on other important aspects of SDLC
- S3 is object storage . To put the difference of block storage and object storage very bluntly , we can run an operating system on block storage but not on object storage i.e. EBS vs S3

Creating a Bucket

- In order to get started with S3 , we need to create a bucket . Consider buckets to be similar to drives in your computers. As in we can create folders and files inside a drive but we cannot create a drive inside a drive . Same goes for a bucket.
- Navigate to services , select S3 from the services and click on create bucket
  - Name and Region
    - Bucket name : A bucket name has to be unique globally. It has to be URL complient
    - Region : S3 as a service is global but bucket is created inside a region.
    - Copy settings from an existing bucket: This is meant for copying settings from an existing bucket
    - After first step we can directly create a bucket , it'll be created with default settings. If we want to configure , click on next
  - Configure options : Here we can select various options like versioning , server access logging ,tags and encryption
  - Set permissions : Here we can control if the objects in the bucket can be pubicly accessible (Best practice is keep default setting as block all public access unless absolutely necessary)
  - Review : It is just a run down of all the options we have selected till now . Click on create bucket to proceed

Uploading an Object

- S3 offers multiple ways to interact with the bucket (AWS console , CLI ,SDK).
- For console , go to S3 . Click on the newly created bucket and click on upload object
  - Click on add files , here we can select the files or can simply drag and drop

        - Similar to bucket creation , we can simply upload the files with default permissions. If not , click on next
    - Set permissions : These are object level permissions (Public access). Here we define the owner of the object . It can also be another account.
    - Set properties : Choosing a storage tier , we'll discuss about it going forward . AS of now we can keep it standard.
    - Review : Quick run down of already selected settings. Click on upload

Storage Tiers

- Similar to EC2 , S3 offers various payment modules one can chose from to suit their use case
- These modules are called storage tiers
- Each tier offers certain benfitis of using it along with it's cost implications
- Always consider the client use case(amount of data , how frequently it'll be accessed , SLA) before selecting a storage tier
- Most people use S3 standard .

Versioning

- Versioning enables us to push objects of same name to be kept in one bucket . Without versioning the object will be directly replaced
- In order to enable versioning
    - Click on bucket name
    - Navigate to properties
    - Select versioning and click on enable
- This feature proves quite helpful where we want to maintain multiple versions of files such as s/w releases
- Versioning also creates a safety layer where if a object is accidently deleted , we can recover it
    - When we delete an object in a bucket which has versioning enabled , it creates a older version of it and attahces a delete marker on it
    - If we want to recover that object , we just have to delete the delete marker and the object will be shown as latest version
    - Delete marker and older versions can be seen by clicking "show" in front of versions
    - In order to delete file permanantly , we have delete the older version of it .

S3 lifecycle management

- S3 lifecycle is a convenient way for us to transition our objects from one storage tier to another
- There can be use cases where client wants to move data from S3 standard to S3 IA
- Lifecycle rules can also be used to expire certain objects . This again removes the additional efforts of deleting files manually.
- In order to configure Lifecycle management rule
    - Navigate to management inside a bucket
    - Click on lifecycle and select +Add lifecycle rule
    - Name and scope : We give a prompt name to the rule and select if we want this rule to apply on all objects of the bucket. This can be controlled using prefix

  - Transitions : We select in which storage tier we want to transition the object to and after how many days of object creation . This applies on current version as well as previous versions
  - Expiration : If we want to delete the objects after above transitions , we can select the version and number of days after which they should be expired . Current versions of the objects will be transitioned into previous versions . Previous versions will be purged from the bucket
  - Review : Run though of the settings we have selected , click on save to apply the rule
- Important thing to consider while using lifecycle management , even though it is a great mechanism for saving storage cost , The transition from one storage to another will cost more if there small objects in large number . Always weigh in the pro and con before taking a decision

S3 replication

- Even though S3 replicates the data internally when it is uploaded , it always makes sense to have an additional backup of the bucket. Preferably in another region .
- S3 replication proves to be a convenient way to configure this reduntant copying operation
- This replication can be across region or within the same region as well. It can be configured across account too.
- In order to configure it , navigat to management inside a bucket
  - Select replication and click on +Add rule
  - Set source : This is your source bucket , data pushed to this bucket will ideally be replicated . This can be restricted to a certain prefix.
  - Set destination : Select the destinaton bucket in which we want to copy the data .There are options to select bucket from another account , change of storage class and the object ownership.
  - Configure rule options : Select the IAM role required (will be discussed in upcoming sessions) and the rule name.
  - Review : Quick run down of the settings configured . Click on save .
- S3 replication rule can only be used if there is versioning enabled on bucket . Also it will only copy data which has arrived post creation of the rule

Static website hosting

- S3 being serverless and with practically unlimited storage , can prove to be a viable option to host a static website
- Configuring it is easy and it can handle unlimited traffic as it scales dynamically
- We need to create 2 html pages , one can host the home page and another can be the error page in case host is not reachable .
- Once created , upload both files in S3 . Make sure both of these objects are public.
- Once uploaded , go to the properties tab inside the bucket
  - Select static buckt hosting and select "Use this bucket to host a static website"
  - Under index document , type the file name of the home page html that we have created
  - Follow the same for error document
  - Click on save , the endpoint mentioned above can be used to access the website.
- S3 website scales automatically . And this can be only used to host static website

Bucket policy

- Bucket policies govern the interaction with underlying objects inside the bucket
- This can be used to enforce encryption or allow crossaccount access.
- These are written in JSON , but also there is policy generator to help create the JSON
- In order to create the bucket policy , navigate to permissions inside a bucket and click on bucket policy
    - We will find that the buckt policy tab is blank . At the bottom of the page , we'll see policy generator option .
    - If you'll click on policy generator , it will help us define the json for any action that we want to allow or deny
        - 1. Select the policy type as S3
        - 2. Add statement is where we need to select the action i.e. allow or deny . Let us select deny for now
            - Keep principal as "*". It states for which users this policy is applicable
            - In actions , we have to select the api on the which this deny policy is applicable . As of now let us select delete object.
            - in Amazon Resource Name , copy the arn of the bucket and add "/*" in front of it . It signifies that the policy is applicable for all the objects inside that bucket .
            - Once done , click on add statement and generate policly
    - This will create a json , copy it and paste it in the bucket policy tab. And click on save .
    - This should look something like this

```
    "Version": "2012-10-17",
    "Id": "Policy1598646413917",
    "Statement": [
        {
            "Sid": "Stmt1598646412509",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:DeleteObject",
            "Resource": "arn:aws:s3:::s3bucketname/*"
        }
    ]
}
```

- We can also try adding conditions like enforcing server side encryption .
    - Go to policy generator and select the service as S3 and select Effect as deny
    - Keep principal as as "*" and in the actions select getObject
    - Give the arn of the bucket and add "/*" after it
    - Click on add conditions
        - In Conditions select "Bool". It stands for value being true or false
        - in Key select "aws:SecureTransport" . This is a key which checks if the requests are going over http ot https
        - In Value write "false "
        - Click on add condition and then add statement
        - Policy should look something like this

```
{
    "Version": "2012-10-17",
    "Id": "Policy1566073168046",
    "Statement": [
        {
            "Sid": "Stmt1566073166682",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::firstbuckettotests3/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

- After saving this policy we can observe if we are able to get the objects over http .

S3 cheat code :

- S3 has practically unlimited storage . Hence it has to be in the center is desigining your storage stratergy
- S3 versioning needs to be implemented as a standard practice
- All the important S3 bucket need to have a replication
- Lifecycle management rule should only be applied after considering the number of objects and object size