JIDNYASA CORPORATE TRAINING CENTRE

AWS/DEVOPS STUDY NOTES

AWS S3

SIMPLE STORAGE SERVICE

S3- Simple Storage Service:

- Amazon S3 is an **object storage service that stores data as objects within buckets**.
- An *object* is a file and any metadata that describes the file.
- A *bucket* is a container for objects.
- S3 is **highly scalable, highly available, durable, secure** object storage service by aws.
- We can store **any amount of data** on S3, there is **no limit** to how much data we can store in S3
- Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics.
- Amazon S3 provides management features so that we can **optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements.**
- S3 stores files of different types like Photos, Audio, and Videos as Objects providing more scalability and security to. It allows the users to store and retrieve any amount of data at any point in time from anywhere on the web. It facilitates features such as extremely high availability, security, and simple connection to other AWS Services.

Amazon S3 Objects:

- Fundamental entity type stored in AWS S3.
- You can store as many objects as you want to store.
- The maximum size of a single object is 5TB but you can store any number of objects in s3.
- We can say that an individual Amazon S3 objects can range in size from a minimum of 0 bytes to a maximum of 5 TB.
- Object consists following information:
    - Key
    - Version ID
    - Value
    - Metadata
    - Sub resources
    - Access control information
    - Tags

Features of S3:

1. Storage classes

One of the best features of S3 is storage We can save data in many different storage classes based on the type of frequency of data access. For example, if you know data is required once in a year then you can store data in most cheaper storage class which is glacier to cut down the storage cost. S3 offers variety of range of storage classes which can be used for cost optimisation.

There is total 8 storage classes:

1. Standard

- **Description:** Default storage class, optimized for frequently accessed data (more than once a month) with milliseconds access
- **Use Case:** General-purpose storage for any type of data, frequently accessed files.

2. Intelligent-Tiering

- **Description:** Automatically moves data to the most cost-effective access tier based on changing access patterns.
- **Use Case:** Data with unknown or unpredictable access patterns.

3. Standard-IA (Infrequent Access)

- **Description:** For data that is accessed less frequently but requires rapid access when needed.
- **Use Case:** Long-term storage, backups, and disaster recovery. Infrequently accessed data (once a month) with milliseconds access.

4. One Zone-IA

- **Description:** Similar to Standard-IA, but data is stored in a single Availability Zone (AZ), reducing cost. Recreatable, infrequently accessed data (once a month) stored in a single Availability Zone with milliseconds access.
- **Use Case:** Non-critical, reproducible data that doesn't require multi-AZ resilience.

5. Glacier Instant Retrieval

- **Description:** Low-cost storage with millisecond retrieval for archive data that needs immediate access. Long-lived archive data accessed once a quarter with instant retrieval in milliseconds.
- **Use Case:** Archive data that is infrequently accessed but requires quick retrieval.

6. Glacier Flexible Retrieval

- **Description:** Low-cost storage for data that is rarely accessed, with retrieval times ranging from minutes to hours.
- **Use Case:** Long-term archives, backups with flexible retrieval needs.

7. Glacier Deep Archive

- **Description:** Lowest-cost storage class, designed for data that is rarely accessed, with retrieval times ranging from hours to days.
- **Use Case:** Long-term archive of data that is rarely accessed, such as regulatory archives.

8. RRR - Reduced Redundancy

- **Description:** A lower-cost storage option that provides less redundancy compared to the S3 Standard storage class.
- **Use Case:** Designed for storing non-critical data that can be easily recreated if lost, such as thumbnails or transcoded media.
- **Status:** Deprecated; AWS recommends using other storage classes for better cost-efficiency and higher durability. AWS now provides better alternatives like S3 One Zone-IA, which offers similar cost benefits but with higher durability.

2. Storage management

Amazon S3 has storage management features that you can use to manage costs, meet regulatory requirements, reduce latency, and save multiple distinct copies of your data for compliance.

- S3 Lifecycle –

- It’s an action we want S3 to take during an object's lifetime.
- Actions –
    - Transitioning objects to another storage class
    - Archiving them, or deleting them after a specified period of time.

  - Transition objects to other S3 storage classes or expire objects.
- S3 Object Lock –
  - To prevent deletion of objects.

- S3 Replication –
  - Automatically copy objects from one bucket to another within the same or different AWS Regions.
  - Enhanced data availability, disaster recovery, and compliance support.

- S3 Batch Operations –
  - S3 Batch Operations allow you to perform large-scale operations like copying, tagging, or modifying access controls on millions of S3 objects with a single job.
  - Can be used to perform operations such as Copy, Replicate, and Restore on millions or billions of objects in a batch.

3. Access management and security:

- By default, S3 buckets and the objects in them are private.
- You have access only to the S3 resources that you create.
- To grant granular resource permissions that support your specific use case or to audit the permissions of your Amazon S3 resources, you can use the following features.

  - S3 Block Public Access –
    - security feature that enables you to restrict public access to your S3 buckets and objects, ensuring they remain private and secure.
    - By default, Block Public Access settings are turned on at the bucket level.
    - AWS recommend that we keep all Block Public Access settings enabled unless we need to share objects.

  - AWS Identity and Access Management (IAM) –
    - With IAM, you can centrally manage permissions that control which AWS resources users can access.
    - You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

  - Bucket policies – (Resource based policy)
    - A bucket policy is an access control policy [Json formatted], attached to an Amazon S3 bucket that defines permissions for actions (like reading, writing, or deleting objects) on the bucket and its contents, specifying who can access the bucket and under what conditions.
    - **Read** – list access, **Write** – Put access, **Delete** – Delete access
    - Policies are a simplified and more flexible access control option. With bucket policies, you can define rules that apply broadly across all requests to your Amazon S3 resources.

  - Amazon S3 access points –
    - Customed network endpoints with dedicated access policies to manage data access at scale for shared datasets in Amazon S3.
    - Access bucket over the private network of AWS.

  - Access control lists (ACLs) –

  - Grant read and write permissions for individual buckets and objects to authorized users.
  - AWS recommend using S3 resource-based policies (bucket policies) or IAM user policies for access control instead of ACLs.

- S3 Object Ownership –
  - To control how ownership of objects is managed within your S3 buckets.
  - S3 bucket-level setting that can be used to disable or enable ACLs.
  - By default, ACLs are disabled. With ACLs disabled, the bucket owner owns all the objects in the bucket and manages access to data exclusively by using access-management policies.

How to work in S3

- To start storing your data in S3 you need to first create Bucket by giving bucket name and specifying region. Then, you upload your data to that bucket as objects.

Bucket:

- A bucket is a container for objects stored in Amazon S3.
- You can store any number of objects in a bucket.
- There is no cost for creating buckets, you will be only charged for object storage and accessing objects.
- By default, you can create 100 buckets per aws account.
- You can increase the limit by contacting to AWS support.
- When you create a bucket, you enter a bucket name and choose the AWS Region where the bucket will reside.
- Once bucket is created you cannot change its name or region.
- **Bucket naming rules:** Bucket names are similar to website names, like website you need to follow some rules to name the bucket.
  - *Bucket names must be between 3 (min) and 63 (max) characters long.*
  - *Bucket names can consist only of lowercase letters, numbers, dots (.), and hyphens (-) no special characters such as @, #, $, % etc are allowed.*
  - *Bucket names must be globally unique. You cannot create another bucket with same name until and unless first one is deleted.*
  - *Bucket names must begin and end with a letter or number.*
  - *Bucket names must not contain two adjacent periods.*
  - *Bucket names must not be formatted as an IP address (for example, 192.168.5.4).*

Objects

- Objects are the fundamental entities stored in Amazon S3. that represents a file or data stored in the S3 bucket
- Objects consist of object data, metadata, key and Bucket where object is stored.
- The metadata is a set of name-value pairs that describe the object.

Keys

- An object key (or key name) is the unique identifier for an object within a bucket.
- Every object in a bucket has exactly one key.

- The combination of a bucket, object key, and optionally, version ID (if S3 Versioning is enabled for the bucket) uniquely identify each object.
- Every object in Amazon S3 can be uniquely addressed through the combination of the web service endpoint, bucket name, key, and optionally, a version. For example, in the URL https://*DOC-EXAMPLE-BUCKET*.s3.us-west-2.amazonaws.com/photos/puppy.jpg, *DOC-EXAMPLE-BUCKET* is the name of the bucket and photos/puppy.jpg is the key.

S3 Versioning

- To keep multiple versions of an object in the same bucket.
- Preserve, retrieve, and restore every version of every object stored in bucket.
- Helps to recover object from both unintended user actions and application failures.
- Versioning once enabled cannot be disabled, can be suspend only.
- When you enable S3 Versioning in a bucket, Amazon S3 generates a unique version ID for each object added to the bucket.
- Bucket versioning can be enabled two ways-
    - *During the time of creation of the bucket.*
    - *After creating the bucket, from bucket properties.*
- Once bucket versioning is enabled objects will be assigned with unique *version ID*, if objects is assigned before enabling the versioning then already uploaded object will have version id as *null*.
- Click here to see *Practical Steps for bucket versioning*
- You can use S3 Versioning to keep multiple versions of an object in one bucket so that you can restore objects that are accidentally deleted or overwritten
- Delete Marker-
    - In case of versioning is enabled then if you delete of an object, instead of removing the object permanently, Amazon S3 inserts a delete marker, which becomes the current object version. You can then restore the previous version. For more information.
    - If you overwrite an object, Amazon S3 adds a new object version in the bucket. The previous version remains in the bucket and becomes a noncurrent version. You can restore the previous version.

Bucket policy: (Resource based policy)

- Here we can also create policies to control the access to bucket and objects stored in bucket, Called as Bucket Policy.
- A bucket policy is a **resource-based** AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it.
- Only the bucket owner can associate a policy with a bucket.
- The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner.
- Bucket policies are limited to 20 KB in size.
- Bucket policies use JSON-based access policy language that is standard across AWS.
- You can use bucket policies to add or deny permissions for the objects in a bucket.
- Bucket policies allow or deny requests based on the elements in the policy, including the requester, S3 actions, resources, and aspects or conditions of the request (for example, the IP address used to make the request). For example, you can create a bucket policy that grants cross-account permissions to upload objects to an S3 bucket while ensuring that the bucket

owner has full control of the uploaded objects. For more information, see Examples of Amazon S3 bucket policies.

Example bucket policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutDeleteObject",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/UserName"
            },
            "Action": [
                "s3:PutObject",
                "s3:DeleteObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::your-bucket-name",
                "arn:aws:s3:::your-bucket-name/*"
            ]
        }
    ]
}
```

What is Bucket URI, Object URL and URI:

Bucket URI

The **Bucket URI** is a unique identifier for an entire S3 bucket within AWS. It's used to reference the bucket programmatically in scripts, configurations, or when interacting with AWS services via the AWS CLI, SDKs, or other tools.

Bucket URI Format:

```
s3://<bucket-name>
```

Example:
If you have a bucket named my-bucket, the Bucket URI would be:

```
s3://my-bucket
```

In S3, when you upload an object to a bucket, it is accessible via a unique URL and URI. Both of these serve as ways to identify and access the object, but they are used in different contexts.

Object URL:

The **Object URL** is a direct web link to the object stored in the S3 bucket. This URL can be used to access the object over the internet if the object's permissions allow public access.

Format:

```
https://<bucket-name>.s3.<region>.amazonaws.com/<key>
```

Example:

```
https://my-bucket.s3.us-west-2.amazonaws.com/my-folder/my-file.txt
```

In this example:

- my-bucket is the name of the S3 bucket.
- us-west-2 is the AWS region.
- my-folder/my-file.txt is the object key (path and file name).

Object URI:

The **Object URI** is an identifier for the object within the S3 service, often used programmatically in scripts or within AWS services. It's not directly used to access the object over the internet but to reference the object within AWS operations.

Format:

```
s3://<bucket-name>/<key>
```

Example:

```
s3://my-bucket/my-folder/my-file.txt
```

In this example:

- my-bucket is the name of the S3 bucket.
- my-folder/my-file.txt is the object key.

Key Differences:

- **Object URL** is used for accessing the object via HTTP/HTTPS from a browser or API, assuming the necessary permissions are in place.
- **Object URI** is used for identifying the object within the AWS ecosystem, typically in configurations, scripts, or when using AWS CLI/SDKs.

-: S3 PRACTICAL STEPS :-

Create S3 Bucket and Upload First Object:

➔ Login to AWS account and go to S3 console → Click on the 'Create bucket' button → Enter a 'Bucket name' (this must be unique across all of AWS) → Choose the 'AWS Region' where the bucket should be created (e.g., US East, US West) → Leave the remaining settings as default unless you have specific requirements → Click on 'Create bucket'

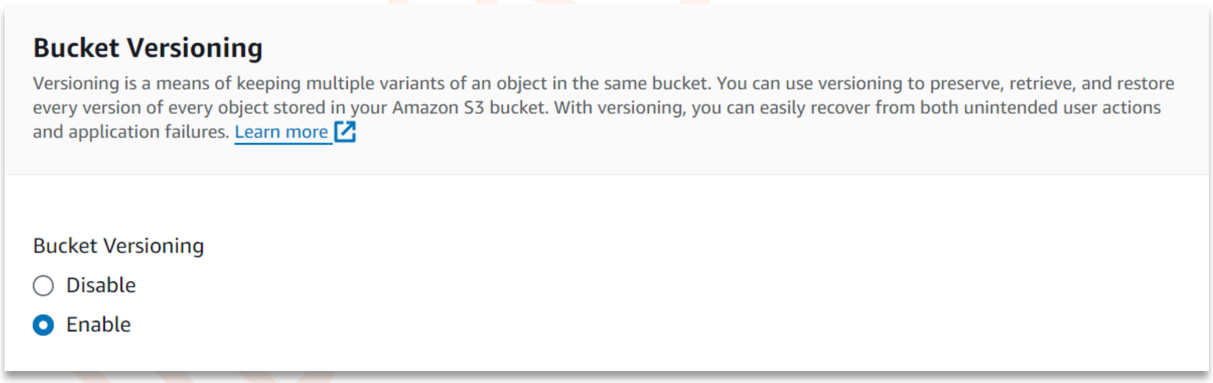
➔ Let's upload an Object to the S3 Bucket Using the AWS Management Console: → From the S3 dashboard, click on the bucket name you just created to open it → Click on the 'Upload' button → In the upload window, click Add files to select the file(s) you want to upload from your computer → You can also drag and drop files directly into the upload window → Once your file is selected, click Upload.

Enable/Suspend Bucket Versioning:

Follow these steps to use the AWS Management Console to enable/suspend versioning on already created S3 bucket.

➔ Go to Amazon S3 console → In the Buckets list, choose the bucket that to which you want to enable/suspend versioning for → Click on 'Properties' tab → Under Bucket Versioning, click on 'Edit' option → Choose 'Enable' or 'suspend' → click on 'Save changes'

You can also enable Bucket versioning while bucket creation as show in below screenshot:



Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning

Disable [✓ Enable]

Object Lock:

S3 Object Lock is an Amazon S3 feature that allows you to store objects using a WORM (Write Once, Read Many) model. It can be used to prevent an object from being deleted or overwritten for a fixed amount of time or indefinitely.

Note: Once enabled, Object Lock cannot be disabled on the bucket.

➔ In the AWS Management Console, navigate to the S3 service → click "Create bucket" → Enter a bucket name and select the appropriate region → Under "Object Lock" check the box labelled → "Enable Object Lock" → Configure other bucket settings as needed (versioning, encryption, etc.). → Click "Create bucket" → Navigate to the Bucket Properties, In the S3 console, go to the

"Properties" tab of the newly created bucket → Under "Object Lock" click "Edit" → now select either Governance Mode or Compliance Mode

- Governance Mode: Users with special permissions can still delete or alter the objects.
- Compliance Mode: Even the root account cannot alter or delete the objects during the retention period.

→ Now Navigate to your bucket and upload an object as usual. → Go to the "Properties" tab of the object → Under "Object Lock" you can set the retention date and choose between governance and compliance modes.

S3 Bucket for Static Website hosting:

To host a static website using an Amazon S3 bucket, follow these steps:

Step 1: Create an S3 Bucket

1. **Navigate to S3**:
   - Open the AWS Management Console and navigate to the S3 service.
2. **Create a New Bucket**:
   - Click "Create bucket."
   - Enter a unique bucket name (it should match your website domain name if you want to use a custom domain, e.g., example.com).
   - Select a region for the bucket.
3. **Set Bucket Configurations**:
   - For "Block Public Access settings," uncheck "Block all public access" to make your bucket publicly accessible. Confirm by checking the acknowledgment box.
   - You can enable versioning and other settings as needed.
4. **Create the Bucket**:
   - Review your settings and click "Create bucket."

Step 2: Enable Static Website Hosting

1. **Open Bucket Properties**:
   - In the S3 console, select your newly created bucket.
   - Go to the "Properties" tab.
2. **Configure Static Website Hosting**:
   - Scroll down to the "Static website hosting" section and click "Edit."
   - Select "Enable" to turn on static website hosting.
   - Specify the index document (e.g., index.html). This is the default page that will load when someone accesses your website.
   - Optionally, specify an error document (e.g., error.html) that users will see if they try to access a page that doesn't exist.
3. **Save Changes**:
   - Click "Save changes" to enable static website hosting.

Step 3: Upload Your Website Files

1. **Upload Files**:
   - Go to the "Objects" tab in your bucket.
   - Click "Upload" to add your website files (e.g., index.html, style.css, images, etc.).
   - You can drag and drop files or use the file selector.
2. **Set Permissions**:
   - After uploading, ensure that your files are publicly accessible. You can do this by selecting the files, clicking "Actions," and choosing "Make public."

    - Alternatively, set the public-read ACL (Access Control List) during the upload process.

Step 4: Set Bucket Policy for Public Access

1. **Bucket Policy**:
    - To allow public access to all objects in your bucket, you need to add a bucket policy.
    - Go to the "Permissions" tab of your bucket.
    - Under "Bucket policy," click "Edit."
2. **Add a Public Access Policy**:
    - Paste the following JSON policy, replacing YOUR-BUCKET-NAME with your actual bucket name:

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

Replace 'YOUR-BUCKET-NAME' part with your bucket name:

- Click "Save changes."

Step 5: Access Your Website

1. **Find the Website URL**:
    - Go to the "Properties" tab in your bucket.
    - Under "Static website hosting," you'll see the "Bucket website endpoint." This is the URL for your static website (e.g., http://YOUR-BUCKET-NAME.s3-website-REGION.amazonaws.com).
2. **Test the Website**:
    - Open the URL in your web browser to see your static website.

S3 bucket replication within the same AWS account

Step 1: Create the Source and Destination Buckets

1. **Create the Source Bucket**:
    - Go to the S3 console in the AWS Management Console.
    - Click "Create bucket."
    - Name your bucket (e.g., source-bucket).
    - Choose the region where you want the source bucket to reside.
    - Click "Create bucket."
2. **Create the Destination Bucket**:
    - Repeat the same process to create the destination bucket (e.g., destination-bucket).
    - You can choose the same or a different region depending on your replication needs.

Step 2: Enable Versioning on Both Buckets

1. **Enable Versioning on the Source Bucket**:
    - Select the source bucket in the S3 console.
    - Go to the "Properties" tab.
    - Under "Bucket Versioning," click "Edit," and then "Enable versioning."
    - Click "Save changes."
2. **Enable Versioning on the Destination Bucket**:
    - Repeat the same steps for the destination bucket.

Step 3: Set Up an IAM Role for Replication (s3_replicator_role)

1. **Create an IAM Role**:
    - Go to the IAM console in the AWS Management Console.
    - Click "Roles" and then "Create role" name – s3_replicator_role
2. **Select the Trusted Entity**:
    - Choose "S3" as the trusted entity.
    - Click "Next."
3. **Attach Permissions**:
    - Attach the "AmazonS3FullAccess" policy, or create a custom policy with the necessary permissions.
    - Click "Next" and then "Create role."
4. **Note the Role ARN**:
    - After creating the role, note the Role ARN for later use.

Step 4: Set Up the Replication Rule

1. **Go to the Source Bucket**:
    - In the S3 console, select your source bucket.
    - Go to the "Management" tab.
2. **Create a Replication Rule**:
    - Click "Create replication rule."
    - Provide a name for the rule (e.g., Replicate-to-Destination).
3. **Set Source and Destination**:
    - Under "Source," select “All objects”
    - Under "Destination," choose the destination bucket you created earlier.
4. **Choose the IAM Role**:
    - Select "Choose an existing role" and pick the role you created earlier from the dropdown “s3_replicator_role”.
5. **Additional Options**:
    - Cancel replicate existing object option
6. **Review and Save**:
    - Review your settings and click "Save."

Step 5: Verify Replication

1. **Upload Objects to the Source Bucket**:
    - Upload some test objects to the source bucket.
2. **Check the Destination Bucket**:
    - After a few moments, check the destination bucket to see if the objects have been replicated.

Important: for the replication to happen destination bucket must have below policy attached:
Destination Bucket Policy for Replication

```
{
  "Version": "2012-10-17",
  "Id": "Policy1723050132931",
  "Statement": [
    {
      "Sid": "Stmt1723050123902",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::ACCOUNT-ID:role/s3_replicator_role"
      },
      "Action": [
        "s3:GetBucketVersioning",
        "s3:ObjectOwnerOverrideToBucketOwner",
        "s3:PutBucketVersioning",
        "s3:ReplicateDelete",
        "s3:ReplicateObject"
      ],
      "Resource": [
        "arn:aws:s3:::DESTINATION-BUCKET-NAME",
        "arn:aws:s3:::DESTINATION-BUCKET-NAME/*"
      ]
    }
  ]
}
```

Replace ACCOUNT-ID with your account ID and DESTINATION-BUCKET-NAME with your destination bucket.

S3 Cross Account Replication:

Setting up cross-account replication in Amazon S3 involves configuring two S3 buckets (one in the source account and one in the destination account) and setting up the necessary IAM roles and bucket policies to allow replication. Below are the steps to achieve this:

Step 1: Set Up the Source and Destination Buckets

In the Source Account:

1. **Create the Source Bucket**:
   - In the AWS Management Console, go to the S3 service.
   - Click "Create bucket."
   - Name the bucket (e.g., source-bucket).
   - Choose the region where you want the bucket to reside.
   - Enable versioning under the "Bucket Versioning" section.
   - Click "Create bucket."

In the Destination Account:

1. **Create the Destination Bucket**:

  - Log into the AWS Management Console with the destination account credentials.
  - Go to the S3 service and create a bucket (e.g., destination-bucket).
  - Choose the region where you want the bucket to reside.
  - Enable versioning under the "Bucket Versioning" section.
  - Click "Create bucket."

Step 2: Set Up IAM Roles and Policies
In the source Account:

1. **Create an IAM Role**: (s3_replicator_role)
   - Go to the IAM console.
   - Click on "Roles" and then "Create role" with S3FullAccess policy attached named as-s3_replicator_role
2. **Note the Role ARN**:
   - After the role is created, note down the Role ARN as it will be needed in the source account.

Step 3: Configure Bucket Policies
In the Destination Account:

1. **Add a Bucket Policy to the Destination Bucket**:
   - Go to the S3 console, select the destination bucket.
   - Under the "Permissions" tab, click "Bucket policy" and edit it with the following:

```
{
  "Version": "2012-10-17",
  "Id": "Policy1723050132931",
  "Statement": [
    {
      "Sid": "s3 cross account replication rule",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::SOURCE-ACCOUNT-ID:role/s3_replicator_role"
      },
      "Action": [
        "s3:GetBucketVersioning",
        "s3:ObjectOwnerOverrideToBucketOwner",
        "s3:PutBucketVersioning",
        "s3:ReplicateDelete",
        "s3:ReplicateObject"
      ],
      "Resource": [
        "arn:aws:s3:::destination-bucket ",
        "arn:aws:s3:::destination-bucket/*"
      ]
    }
  ]
}
```

Replace SOURCE-ACCOUNT-ID with the AWS account ID of the source account, YOUR-ROLE-NAME with the role name used in the source account, and destination-bucket with the actual name of your destination bucket.

    - Save the changes.

Step 4: Set Up the Replication Rule

In the Source Account:

1. **Go to the Source Bucket**:
    - In the S3 console, select your source bucket.
    - Go to the "Management" tab.
2. **Create a Replication Rule**:
    - Click "Create replication rule."
    - Provide a name for the rule (e.g., CrossAccountReplication).
    - Choose the IAM role that was created in the source account (s3_replicator_role)
3. **Set Source and Destination**:
    - Under "Source," select all objects in the bucket option
    - Under "Destination," enter the destination bucket name and the destination Account ID
4. **Choose the IAM Role**:
    - Select "Choose an existing role" and select the IAM role from the destination account that was created for replication.
5. **Save the Rule**:
    - Review your settings and save the rule.

Step 5: Test and Verify the Setup

1. **Upload Objects to the Source Bucket**:
    - Upload a few test objects to the source bucket.
2. **Check the Destination Bucket**:
    - After a short while, check the destination bucket in the destination account to ensure that the objects have been replicated.

S3 Batch Operations Job to Copy Existing Objects:

Let's see how we can copy existing objects using S3 batch Job operations:

STEP 1: Create two buckets, one source bucket and one destination bucket, enable bucket versioning on both the buckets.
STEP 2: Upload few objects into Source Bucket
STEP 3: Now create Replication Rule for Source bucket to replicate objects into Destination bucket (*Do not replicate existing object while creating Replication Rule)
STEP 4: Upload one object into source bucket and confirm if object is being replicate successfully, (We are going to use this replication configuration for Batch Operations Job for replicating existing objects of source bucket)
STEP 5: Now lets create Batch operations, for that login on S3 console and go to left sides navigation pane

➔ Click on 'Batch Operations' → click on 'Create Job' → Inside manifest option select 'Create manifest using S3 Replication configuration' → For source select 'Bucket in this AWS account' → and then select the source bucket name by using 'Brows' button → Click on 'Next' → For operation select 'Replicate' → Add description → set priority to '1' → Uncheck 'Generate completion report' → For permissions Select 'Choose from existing IAM roles' that you want to use for Batch Operation Job → Click on 'Next' → Review the Job configuration and click on 'Create Job'

➔ You can see job is created wait for a Job to go in 'Waiting for Approval' state and then Select the job and click on 'Run job' → Review job configuration and then click to "Run Job".

==> Once the Job completes you can check if existing objects are replicated.

S3 Lifecycle Configuration:

allows you to automate the management of your objects by setting rules that define actions to transition objects to different storage classes or to delete them after a certain period.

Scenario

Let's assume you have a bucket that stores logs, and you want to:

1. Transition logs to the **Glacier** storage class 30 days after they are created.
2. Permanently delete logs 365 days after they are created.
3. Move incomplete multipart uploads older than 7 days to be automatically deleted.

Step 1: Create an S3 Bucket (if not already created)

1. **Navigate to S3 in AWS Management Console**:
    - Open the AWS Management Console and go to the S3 service.
2. **Create a Bucket**:
    - Click on "Create bucket."
    - Provide a unique bucket name (e.g., my-log-bucket).
    - Leave other settings as default, unless specific changes are required.
    - Click "Create bucket."

Step 2: Configure Lifecycle Rules

1. **Go to the Bucket**:
    - In the S3 console, click on your bucket name (e.g., my-log-bucket).
2. **Navigate to the "Management" Tab**:
    - Click on the "Management" tab.
    - Scroll down to the "Lifecycle rules" section.
3. **Create a New Lifecycle Rule**:
    - Click "Create lifecycle rule."
    - **Name the Rule**: Enter a name for your rule (e.g., ManageLogs).
4. **Choose Rule Scope**:
    - **Apply to all objects in the bucket**: If you want the rule to apply to all objects.
5. **Add Transitions**:
    - Under "Lifecycle rule actions," click "Transition current versions of objects between storage classes."
    - Click "Add transition."
    - **Transition to Glacier**:
        - Select the **Glacier** storage class.
        - Set the transition to occur **30 days** after the object's creation.
        - Click "Save."
6. **Add Expiration**:
    - Under "Lifecycle rule actions," click "Expire current versions of objects."
    - Set the expiration to **365 days** after the object's creation.
    - This will permanently delete objects after one year.

    - Click "Save."
7. **Manage Incomplete Multipart Uploads**:
    - Scroll down to "Delete incomplete multipart uploads."
    - Set the rule to delete incomplete uploads **7 days** after the initiation of the multipart upload.
8. **Review and Save the Rule**:
    - Review the configuration and click "Create rule."

Step 3: Verify the Lifecycle Rule

1. **Check the Rule**:
    - Once the rule is created, it will appear under the "Lifecycle rules" section of the "Management" tab.
    - You can view, edit, or delete the rule as needed.