Replicating objects within and across Regions

We can use replication to enable automatic, asynchronous copying of objects across Amazon S3 buckets. Buckets that are configured for object replication can be owned by the same AWS account or by different accounts. We can replicate objects to a single destination bucket or to multiple destination buckets. The destination buckets can be in different AWS Regions or within the same Region as the source bucket.

There are two types of replication: *live replication* and *on-demand replication*.

- **Live replication** – To automatically replicate new and updated objects as they are written to the source bucket, use live replication. Live replication doesn't replicate any objects that existed in the bucket before we set up replication. To replicate objects that existed before we set up replication, use on-demand replication.
- **On-demand replication** – To replicate existing objects from the source bucket to one or more destination buckets on demand, use S3 Batch Replication. For more information about replicating existing objects, see When to use S3 Batch Replication.

There are two forms of live replication: *Cross-Region Replication (CRR)* and *Same-Region Replication (SRR)*.

- **Cross-Region Replication (CRR)** – We use CRR to replicate objects across Amazon S3 buckets in different AWS Regions. For more information about CRR, see When to use Cross-Region Replication.
- **Same-Region Replication (SRR)** – We use SRR to copy objects across Amazon S3 buckets in the same AWS Region. For more information about SRR, see When to use Same-Region Replication.

1. S3 Replication

S3 Replication is an automated and managed feature that asynchronously copies objects between buckets. This is an excellent option for continuous data synchronization, disaster recovery, or to bring data closer to users in different regions.

Key Features:

Automatic Copying: Once configured, new objects uploaded to the source bucket are automatically replicated to the destination bucket in another account.

Existing Objects: You can also replicate existing objects by creating a one-time S3 Batch Replication job.

Versioning: Both source and destination buckets must have versioning enabled.

Permissions: This method requires setting up an IAM role in the source account that has permissions to replicate objects to the destination bucket. The destination bucket's policy must grant the source account's IAM role permission to replicate objects.

High-Level Steps:

Enable Versioning: Ensure versioning is turned on for both the source and destination S3 buckets.

Create an IAM Role: In the source account, create an IAM role that S3 can assume to replicate objects on your behalf. This role's policy will grant permissions to read from the source bucket and replicate to the destination bucket.

Configure Replication Rule: In the source bucket's management settings, create a replication rule.

Specify the destination bucket in the other AWS account.

Point to the IAM role you created.

You can choose to replicate the entire bucket or filter by prefix or tags.

Update Destination Bucket Policy: The destination bucket policy must be updated to allow the IAM role from the source account to replicate objects.

Replicate Existing Objects (Optional): If you need to copy objects that existed before the replication rule was set up, you can use S3 Batch Replication.

2. S3 Batch Operations

For large-scale, one-time data transfers, S3 Batch Operations provide a managed and auditable way to copy billions of objects.

Key Features:

Bulk Operations: Designed to perform actions on a massive number of objects.

Manifest File: We provide a manifest file (either a CSV or an S3 Inventory report) that lists the objects you want to copy.

Managed Process: S3 Batch Operations manages the copying process, including retries and completion tracking.

High-Level Steps:

Create a Manifest: Generate a list of the objects to be copied. The easiest way is to use an S3 Inventory report for the source bucket.

Set up Permissions:

Create an IAM role in the account that will run the Batch Operations job (this can be the source or destination account). This role needs permission to read the manifest file and the source bucket, and write to the destination bucket.

The source bucket policy must grant the IAM role permission to read the objects.

The destination bucket policy must grant the IAM role permission to write the objects.

Create and Run the Job:

In the S3 console, go to "Batch Operations" and create a new job.

Select the manifest file.

Choose the "Copy" operation and specify the destination bucket.

Assign the IAM role you created.

The job will then process the manifest and copy the files. You can monitor its progress in the console.

3. AWS Command Line Interface (CLI)

Using the AWS CLI with the aws s3 sync command is a flexible, hands-on method for moving objects between accounts. This approach is well-suited for both one-time transfers and recurring syncs that can be scripted and automated.

Key Features:

Granular Control: Provides a high degree of control over the transfer process.

Scriptable: Can be easily integrated into scripts for automation.

Efficient Syncing: The sync command compares the source and destination and only copies new or updated files.

High-Level Steps:

Configure AWS CLI: We need to have the AWS CLI installed and configured with credentials for both the source and destination AWS accounts. You can set up named profiles for each account in your AWS config file.

Set up Cross-Account Permissions:

IAM User/Role in Destination Account: Create an IAM user or role in the destination account with permissions to write to the destination bucket.

Source Bucket Policy: The bucket policy on the source bucket must grant the IAM user or role from the destination account permission to read objects.

Run the sync Command: From the terminal, we can then run the aws s3 sync command, specifying the source bucket and the destination bucket, along with the appropriate profile for each account.

For example:

```
aws s3 sync s3://source-bucket-name s3://destination-bucket-name --source-profile source_account_profile --profile destination_account_profile
```

4. AWS DataSync

AWS DataSync is a data transfer service that simplifies, automates, and accelerates moving large amounts of data between on-premises storage, edge locations, and AWS storage services like S3. It can also be used for cross-account S3 transfers.

Key Features:

Accelerated Transfers: DataSync uses a purpose-built protocol to speed up data transfer.

Fully Managed: It handles many of the tasks involved in data transfer, such as encryption, validation, and optimization.

High-Level Steps:

Create an IAM Role: In the source account, create an IAM role that gives DataSync permissions to access the destination bucket.

Update Destination Bucket Policy: In the destination account, update the S3 bucket policy to allow the DataSync IAM role from the source account to write data.

Create DataSync Locations: In the source account's DataSync console, create locations for both the source and destination S3 buckets.

Create and Start a Task: Create a DataSync task that specifies the source and destination locations and configure the transfer settings. Then, start the task to begin the data movement.

Choosing the Right Option

For ongoing, automatic synchronization of new objects, S3 Replication is the ideal choice.

For a large, one-time migration of existing objects, S3 Batch Operations is the most robust and manageable solution.

For flexible, scriptable transfers or when you want more direct control, the AWS CLI is a powerful tool.

For highly optimized and accelerated transfers of large datasets, AWS DataSync is a strong contender.

AWS Datasync:
https://medium.com/@nayanarora55/cross-account-s3-migration-aws-datasync-does-it-so-you-dont-have-to-9b85a23d1464

Perspective guidance on S3 migration:
https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/copy-data-from-an-s3-bucket-to-another-account-and-region-by-using-the-aws-cli.html

Replication Summary:

Methodology	Best use case	Tentative speed and its factors	Cost factors
AWS CLI	Smaller size of bucket (tens of GBs), one time replication	Depends upon the speed of the machine running S3 sync command	API calls for S3 GET, PUT and LIST
S3 Bucket Replication	Synchronous replications for S3 bucket but also can be used for one time job too	Data transfer rate is **high** since it is a managed service by AWS. Data Stays entirely within the secure AWS global network.	Data transfer + API requests for replication
S3 Batch Operations	One-time, large-scale copy of **existing** objects	**Very High**. Designed for massive, parallel bulk operations. Data Stays entirely within the secure AWS global network.	Per-job fee + per-object fee + API requests
AWS DataSync	Massive (TB-PB scale), high-speed migrations	**Highest**. Uses a custom protocol for maximum throughput. Data Stays entirely within the secure AWS global network.	Per-GB transferred fee

Global Accelerator optimizes routing for the TCP/UDP traffic so it is not relevant in our context.

AWS Transfer Acceleration utilizes the edge resources of AWS to speed up the data transfer from outside of AWS network to and fro. It seems a little unnecessary unless the buckets are a large region apart.

When we do `aws s3 cp` OR `aws s3 sync`
These commands automatically use **multipart upload** for files over a certain size (**default 8 MB parts**). So we don't need to manage it, it happens behind the scenes.

Recommended method for 10s of GBs: AWS Cloudshell / CLI

AWS CLI Allows for a quick replication between the S3 buckets with the condition that the hardware that is running the CloudShell or CLI is intermediary for the data transfer. More clearly, this is the flow of data when using CloudShell.

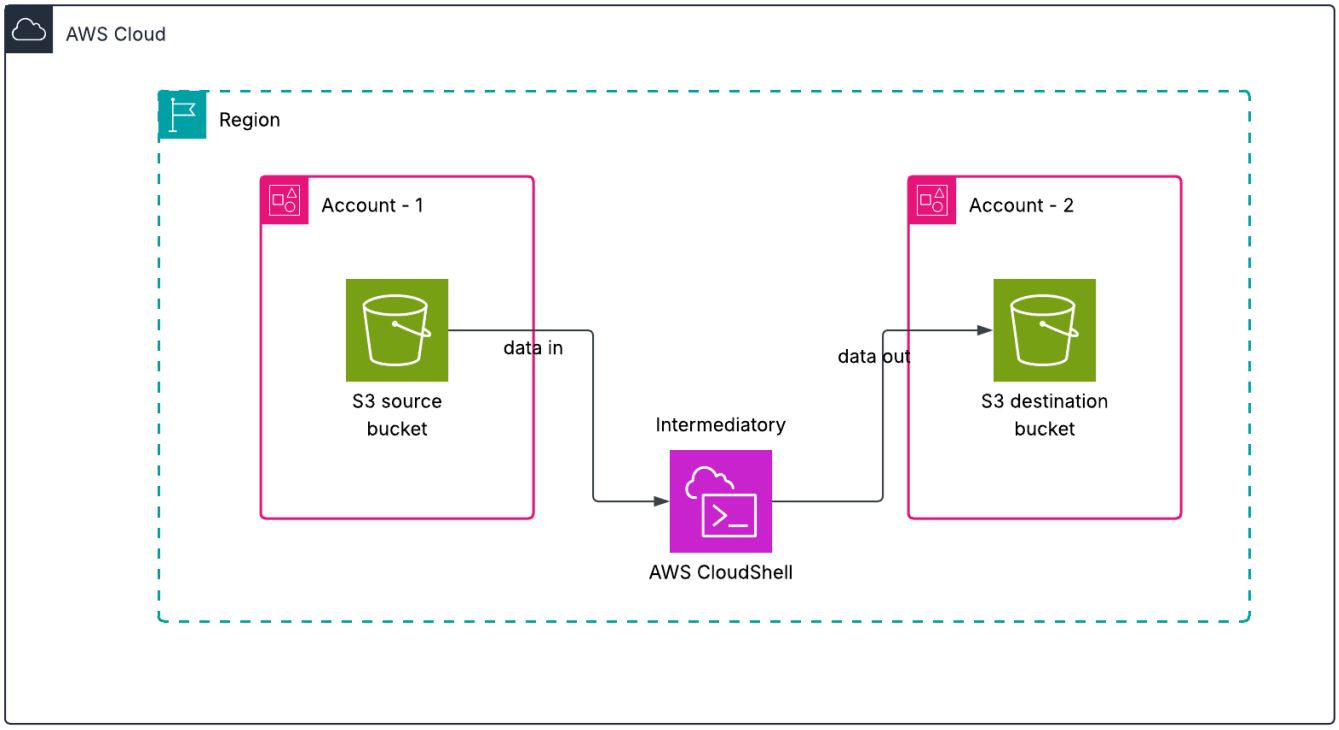


Fig: Flow of data through AWS CloudShell with s3 sync command.

Pros of using the Cloudshell method:

- Since the data travels within the AWS network, it would incur very no data transfer charge.
- The data transfer rate is also very high, around 8 GB of data in less than 30 seconds.

Bash Script:

```
aws s3 sync s3://sourcebucket/ s3://destinationbucket/
OR
aws s3 cp s3://sourcebucket/ s3://destinationbucket/
```

Note: When we do `aws s3 cp` OR `aws s3 sync`
These commands automatically use **multipart upload** for files over a certain size (**default 8 MB parts**). So we don't need to manage it, it happens behind the scenes.

Context: The source bucket is in Account 1 and the destination bucket is in Account 2. Block all Public Access is kept default (ON) as well as other bucket configurations.

For this **AWS CloudShell method** to execute successfully we need to do the following steps:

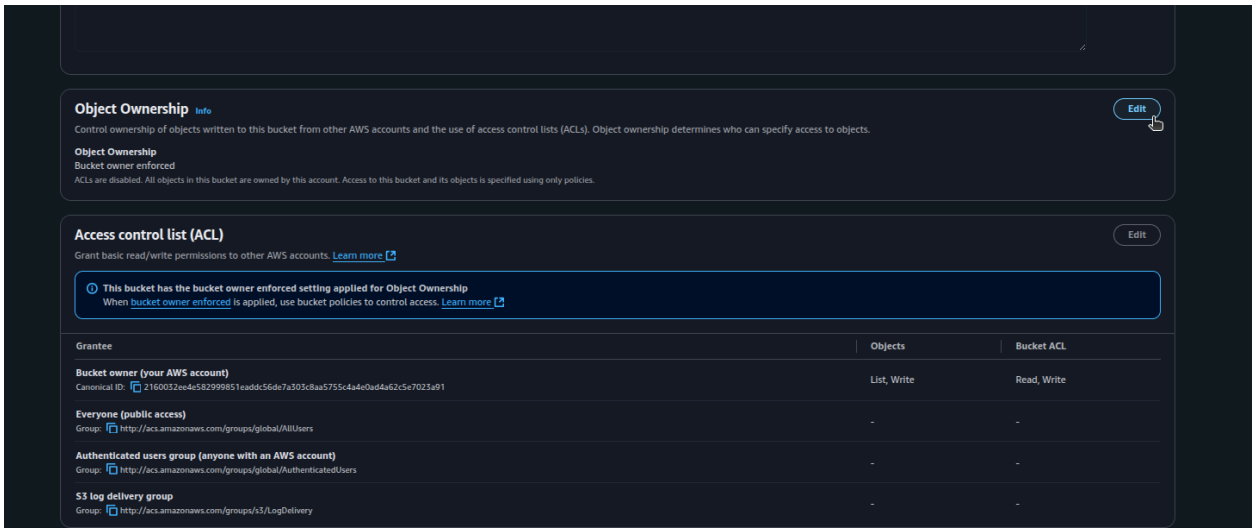
1. **Insert Bucket policy:**
   If you are uploading from account 1's bucket to account 2's bucket then you must insert the resource based policy (S3 bucket policy) **in the destination bucket** that allows for at least 3 actions, namely: s3:PutObjectAcl, s3:PutObject and s3:ListBucket for **Account 1.** Here's the S3 bucket policy example for **destination bucket**:

```
{
	"Version": "2012-10-17",
	"Statement": [
		{
			"Sid": "AllowAccountAToWrite",
			"Effect": "Allow",
			"Principal": {
				"AWS": "arn:aws:sts::<Account ID>:<Account name>”
			},
			"Action": [
				"s3:PutObject",
				"s3:PutObjectAcl"
			],
			"Resource": "arn:aws:s3:::destinationbucket/*"
		},
		{
			"Sid": "AllowAccountAToList",
			"Effect": "Allow",
			"Principal": {
				"AWS": "arn:aws:sts::<Account ID>:<Account name>"
			},
			"Action": "s3:ListBucket",
			"Resource": "arn:aws:s3:::destinationbucket"
		}
	]
}
```

2. **Enforce Object Ownership (Dealing with object ownerships while doing cross account S3 object migration):**
   Go to S3 console → Destination Bucket → Permissions → Object Ownership → set Object Ownership to **Bucket owner enforced.**

   This allows for us to **Ignore the uploader's identity** and always make the **bucket owner** the owner of the uploaded object.

Otherwise if this is not enabled, although the objects would have been copied to the destination bucket but the object ownership would still be Account 1.



3. **Run the following command:**

Run this command in **CloudShell in Account 1** as we have set up the bucket policy in Account 2. Make sure Account 1 and 2 both have appropriate access for their own buckets.

```
aws s3 sync s3://sourcebucket/ s3://destinationbucket/
```

Optional parameters for this command includes

```
aws s3 cp s3://sourcebucket/ s3://destinationbucket/ --acl bucket-owner-full-control
```

This additional parameter gives the bucket owner full control of the uploaded object, while the upload process is being done.

This concludes the migration process.

Overall, This is a very cost effective and secure alternative for small data transfers between S3 buckets in a single region.