* In **Account-A**:
  + Created an **Amazon S3 bucket** (Bucket-A)
  + Created an **IAM Role** (Role-A)
  + Created an **AWS Lambda function** (Lambda-A) and assigned Role-A to the function
  + Configured an **Amazon S3 Event** on Bucket-A to trigger Lambda-A for "All object create events"
* In **Account-B**:
  + Created an **Amazon S3 bucket** (Bucket-B) with a bucket policy (see below)

**IAM Role – “role-A” {inline policy}**

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": "s3:GetObject",

"Resource": "arn:aws:s3::: SOURCE-BUCKET-NAM/\*"

},

{

"Effect": "Allow",

"Action": [

"s3:PutObject",

"s3:PutObjectAcl"

],

"Resource": "arn:aws:s3::: DESTINATION-BUCKET-NAME/\*"

}

]

}

**Lambda Function**

Lambda-A is triggered when an object is created in Bucket-A, and copies it to Bucket-B:

import boto3

import urllib

TARGET\_BUCKET = DESTINATION-BUCKET-NAME'

def lambda\_handler(event, context):

# Get incoming bucket and key

source\_bucket = event['Records'][0]['s3']['bucket']['name']

source\_key = urllib.parse.unquote\_plus(event['Records'][0]['s3']['object']['key'])

# Copy object to different bucket using multipart copy

s3\_client = boto3.client('s3')

copy\_source = {

'Bucket': source\_bucket,

'Key': source\_key

}

target\_key = source\_key # Change if desired

try:

# Check the size of the source object

source\_object\_info = s3\_client.head\_object(Bucket=source\_bucket, Key=source\_key)

source\_object\_size = source\_object\_info['ContentLength']

if source\_object\_size > 5 \* 1024 \* 1024:

# Create a multipart copy request if the source object is larger than 5MB

multipart\_upload = s3\_client.create\_multipart\_upload(Bucket=TARGET\_BUCKET, Key=target\_key)

upload\_id = multipart\_upload['UploadId']

# Copy the object in parts

part\_number = 1

copy\_part\_responses = []

while True:

copy\_source\_range\_start = (part\_number - 1) \* 5 \* 1024 \* 1024

copy\_source\_range\_end = min(part\_number \* 5 \* 1024 \* 1024 - 1, source\_object\_size - 1)

copy\_source\_range = f"bytes={copy\_source\_range\_start}-{copy\_source\_range\_end}"

# copy\_source\_range = f"bytes={part\_number \* 5 \* 1024 \* 1024}-{(part\_number + 1) \* 5 \* 1024 \* 1024 -1 }"

copy\_part\_response = s3\_client.upload\_part\_copy(

CopySource=copy\_source,

CopySourceRange=copy\_source\_range,

Bucket=TARGET\_BUCKET,

Key=target\_key,

PartNumber=part\_number,

UploadId=upload\_id

)

copy\_part\_responses.append({

'PartNumber': part\_number,

'ETag': copy\_part\_response['CopyPartResult']['ETag']

})

part\_number += 1

# If the last part is smaller than 5MB, exit the loop

if copy\_source\_range\_end == source\_object\_size - 1:

break

# Complete the multipart copy

s3\_client.complete\_multipart\_upload(

Bucket=TARGET\_BUCKET,

Key=target\_key,

UploadId=upload\_id,

MultipartUpload={'Parts': copy\_part\_responses}

)

else:

# If the source object is smaller than 5MB, copy it in a single part

s3\_client.copy\_object(CopySource=copy\_source, Bucket=TARGET\_BUCKET, Key=target\_key, ACL='bucket-owner-full-control')

except Exception as e:

# Handle any exceptions here

print(f"Error: {e}")

return

**Bucket Policy on destination bucket**

The Bucket Policy on Bucket-B permits access from the Role-A IAM Policy:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"AWS": "arn:aws:iam::ACCOUNT-ID role/ROLE-A "

},

"Action": [

"s3:PutObject",

"s3:PutObjectAcl",

"s3:PutBucketTagging",

"s3:PutObjectTagging"

],

"Resource": [

"arn:aws:s3:::DESTINATION-BUCKET-NAME /\*",

"arn:aws:s3::: DESTINATION-BUCKET-NAME "

]

}

]

}