

The Task of automatic Birthday wishes on website mailing system includes automatically uploading files from one S3 bucket to another at a fixed scheduled time with the help of Amazon EventBridge and lambda. This would in turn trigger the Amazon SNS to send a mail to the specified email id containing the URL of the website containing the Birthday wishes hosted on S3.

Team No: 04

Ananya Tiwari

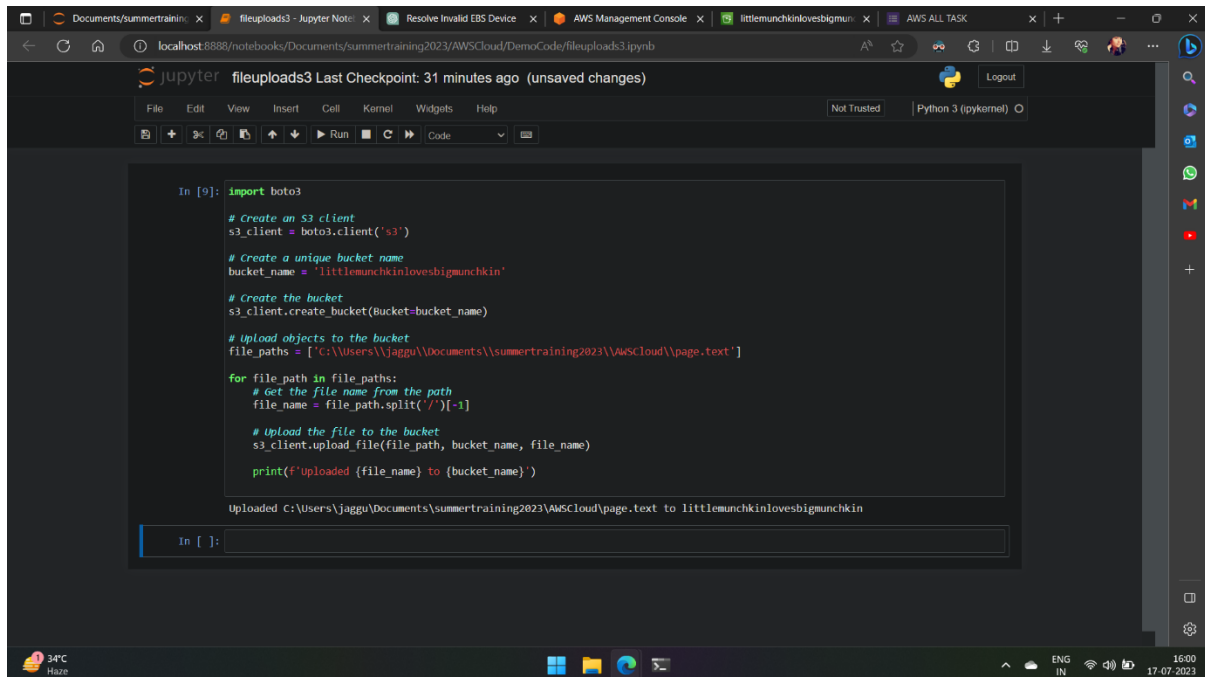
Rangari Bhokre Deepika

Malay Thakur

Manish Didi

To achieve this, we will use AWS S3, AWS Lambda, AWS SNS, AWS EventBridge Scheduler and AWS IAM. Here are the steps to set it up:

Step 1: Create two S3 buckets and upload objects into one using the Boto3 library.



The screenshot shows a Jupyter Notebook interface with a Python script that uses the Boto3 library to create an S3 bucket and upload files. The code is as follows:

```
In [9]: import boto3

# Create an S3 client
s3_client = boto3.client('s3')

# Create a unique bucket name
bucket_name = 'littlemunchkinlovesbigmunchkin'

# Create the bucket
s3_client.create_bucket(Bucket=bucket_name)

# Upload objects to the bucket
file_paths = ['C:\\Users\\jaggu\\Documents\\summertraining2023\\AWS\\Cloud\\page.text']

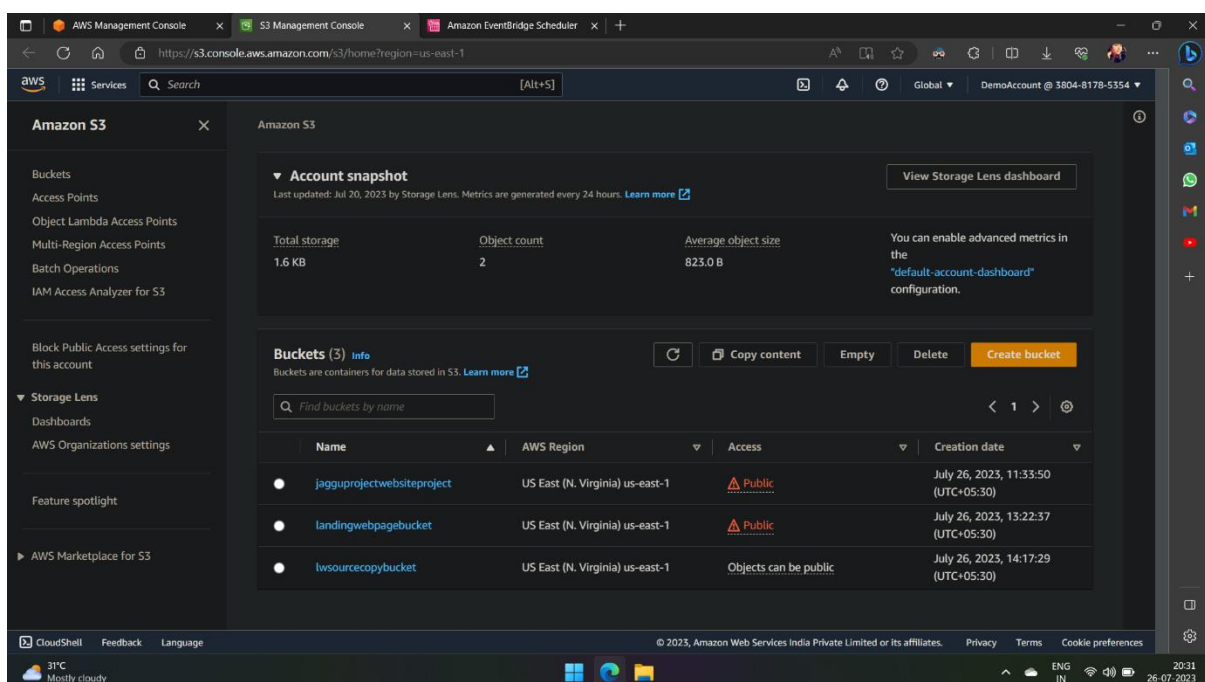
for file_path in file_paths:
    # Get the file name from the path
    file_name = file_path.split('/')[-1]

    # Upload the file to the bucket
    s3_client.upload_file(file_path, bucket_name, file_name)

    print(f'Uploaded {file_name} to {bucket_name}')
```

The output of the script shows that the file 'page.text' has been successfully uploaded to the bucket 'littlemunchkinlovesbigmunchkin'.

-Make sure to replace 'your-unique-bucket-name' with the desired name for your S3 bucket. Also, provide the correct file paths for the objects you want to upload in the file_paths list.

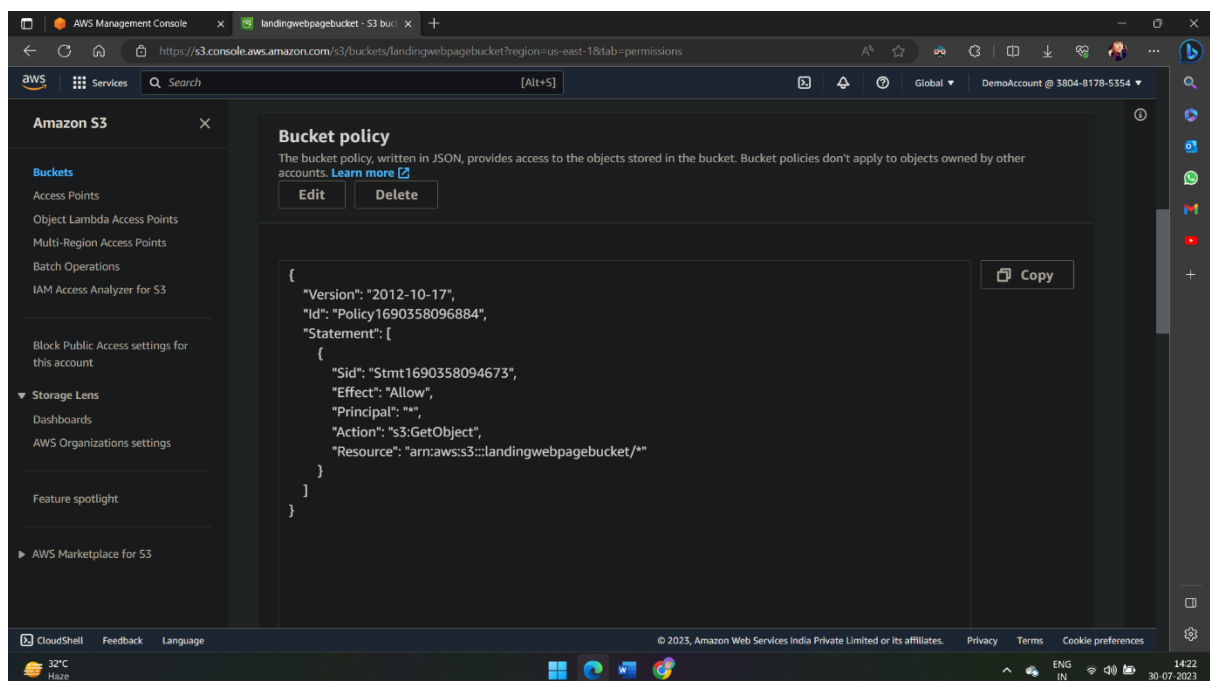


The screenshot shows the AWS Management Console for the S3 service. The left sidebar contains navigation links for Buckets, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, Block Public Access settings for this account, Storage Lens, Dashboards, AWS Organizations settings, Feature spotlight, and AWS Marketplace for S3. The main content area displays the 'Account snapshot' and a list of buckets.

Name	AWS Region	Access	Creation date
jagguprojectwebsiteproject	US East (N. Virginia) us-east-1	Public	July 26, 2023, 11:33:50 (UTC+05:30)
landingwebpagebucket	US East (N. Virginia) us-east-1	Public	July 26, 2023, 13:22:37 (UTC+05:30)
twsourcecopybucket	US East (N. Virginia) us-east-1	Objects can be public	July 26, 2023, 14:17:29 (UTC+05:30)

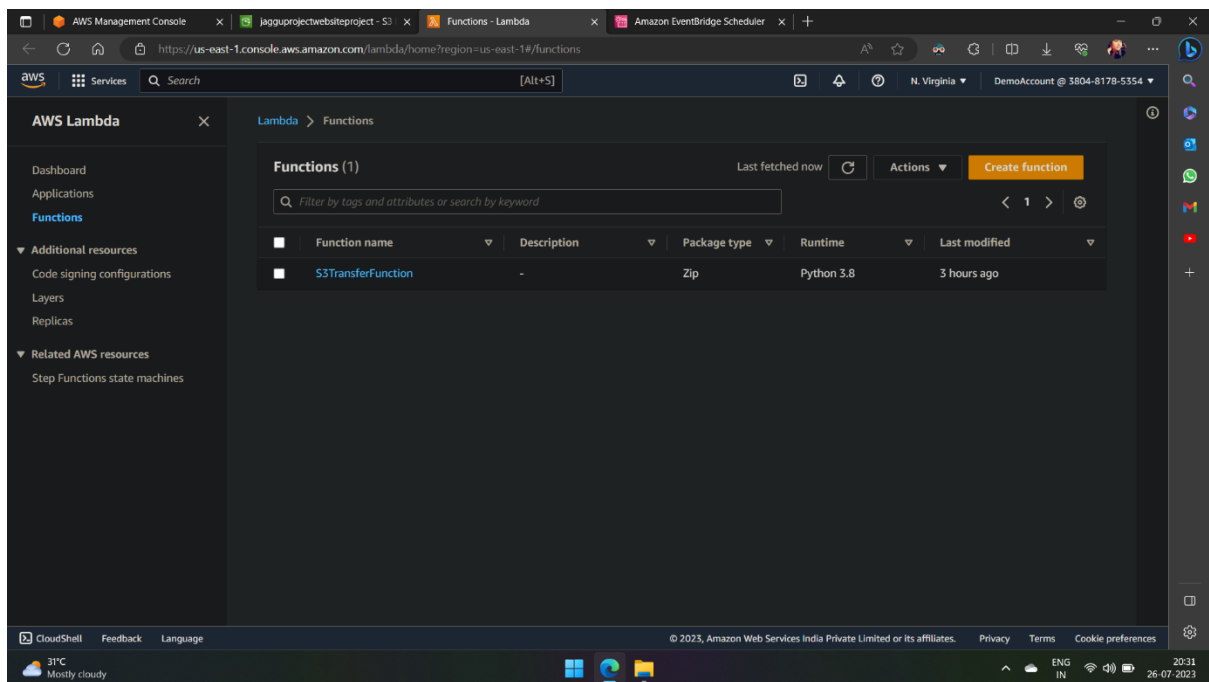
Step 2: Enable Public Access and Static Website Hosting in destination S3 Bucket.

- Go to the AWS Management Console and navigate to the S3 service and select Buckets.
- Select both the S3 buckets one by one and under the “Permission” section select edit Block public access (bucket settings).
- Uncheck all the checkboxes in the same section to enable public access.
- But for full public access, click on edit under “Bucket Policy” and design a policy for each S3 Bucket to enable complete public access.
- Under the “Properties” section enable “Static Website Hosting.”



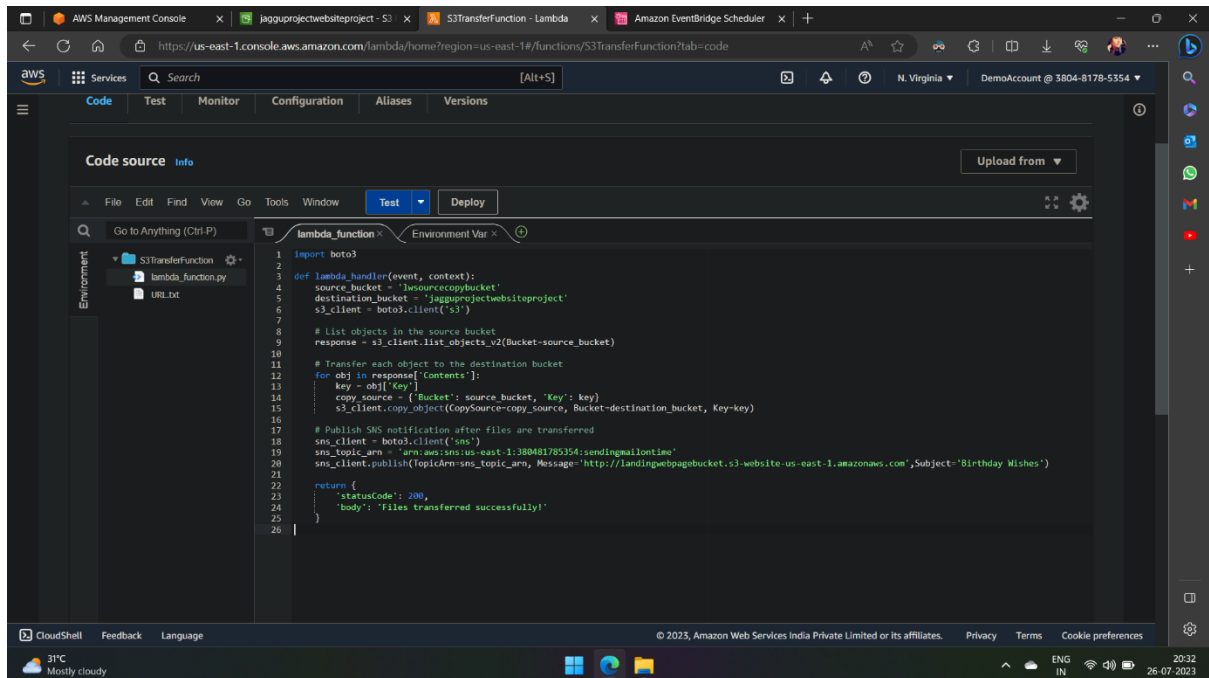
Step 3: Create the Lambda Function.

- Go to the AWS Management Console and navigate to the Lambda service.
- Click on "Create function."
- Choose "Author from scratch."
- Provide a name for the function (e.g., "S3TransferFunction").
- For the runtime, select "Python 3.8" (or any other supported runtime you prefer).
- Click "Create function."



Step 3: Configure the Lambda Function.

- In the Lambda function editor, scroll down to the "Function code" section.
- Replace the existing code with the Python code below:

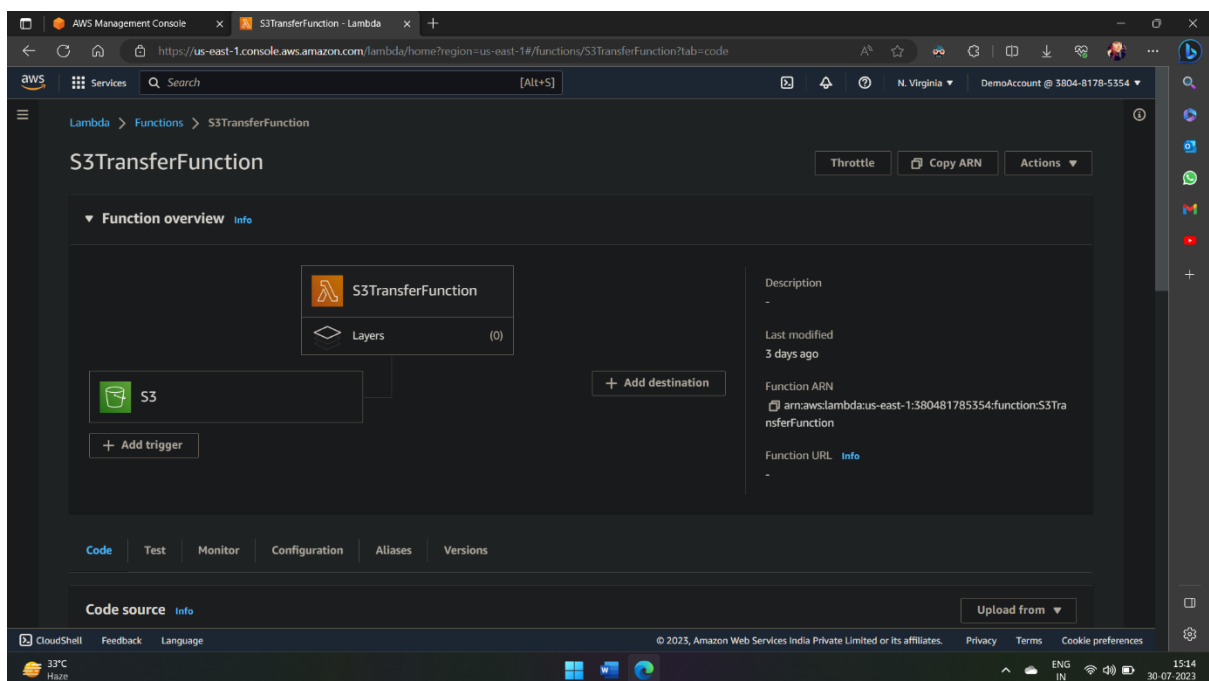


```
1 import boto3
2
3 def lambda_handler(event, context):
4     source_bucket = 'your-source-bucket-name'
5     destination_bucket = 'your-destination-bucket-name'
6     s3_client = boto3.client('s3')
7
8     # List objects in the source bucket
9     response = s3_client.list_objects_v2(Bucket=source_bucket)
10
11     # Transfer each object to the destination bucket
12     for obj in response['Contents']:
13         key = obj['Key']
14         copy_source = {'Bucket': source_bucket, 'Key': key}
15         s3_client.copy_object(CopySource=copy_source, Bucket=destination_bucket, Key=key)
16
17     # Publish SNS notification after files are transferred
18     sns_client = boto3.client('sns')
19     sns_topic_arn = 'arn:aws:sns:us-east-1:380481785354:sendingmailontime'
20     sns_client.publish(TopicArn=sns_topic_arn, Message='http://landingwebpagebucket.s3-website-us-east-1.amazonaws.com', Subject='Birthday Wishes')
21
22     return {
23         'statusCode': 200,
24         'body': 'Files transferred successfully!'
25     }
26
```

- Replace 'your-source-bucket-name' with the name of your source S3 bucket.
- Replace 'your-destination-bucket-name' with the name of your destination S3 bucket.
- Scroll down to the "Basic settings" section and set the "Timeout" to an appropriate value (e.g., 1 minute).

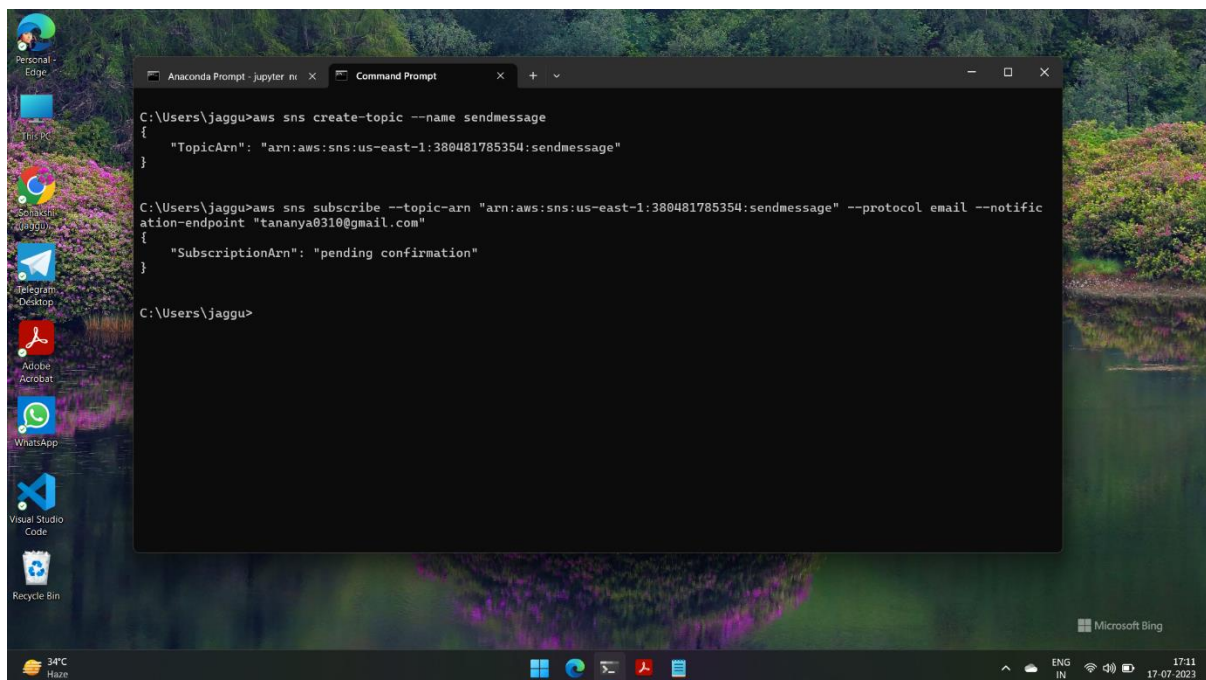
Step 3: Add trigger to the lambda function.

- Go to the AWS Management Console and navigate to the Lambda service.
- Click on "Functions" in the left sidebar and then select your desired lambda function.
- Click on "Add Trigger."
- Choose a source (in this case select "Amazon S3").
- Under "Event Type" select PUT.
- Tick the checkbox for acknowledgement.
- And then click on "Add."



Step 4: Create a SNS topic using CLI.

-Create an SNS topic.



```
C:\Users\jaggu>aws sns create-topic --name sendmessage
{
  "TopicArn": "arn:aws:sns:us-east-1:380481785354:sendmessage"
}

C:\Users\jaggu>aws sns subscribe --topic-arn "arn:aws:sns:us-east-1:380481785354:sendmessage" --protocol email --notification-endpoint "tananya0310@gmail.com"
{
  "SubscriptionArn": "pending confirmation"
}

C:\Users\jaggu>
```

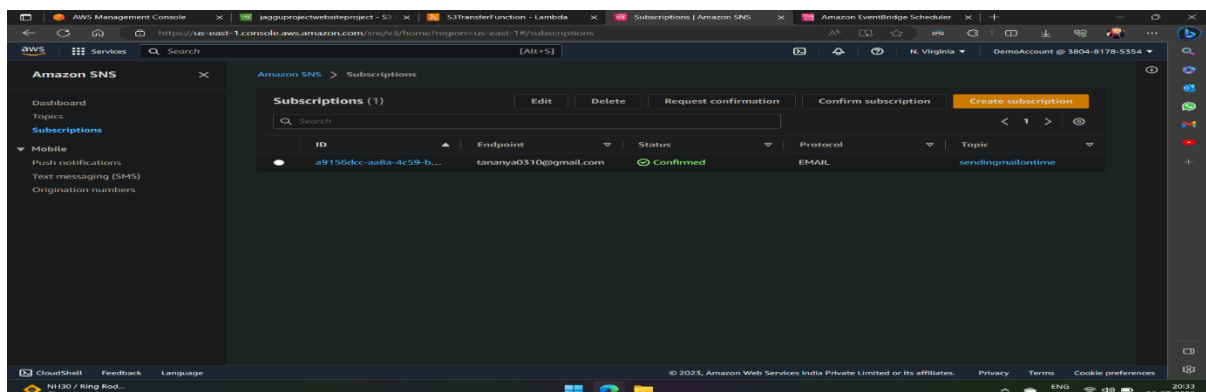
-This command will create an SNS topic and return the topic ARN.

-Replace "ARN_OF_YOUR_TOPIC" with the actual ARN of the SNS topic that was created in step 1.

-Subscribe to the topic using email protocol

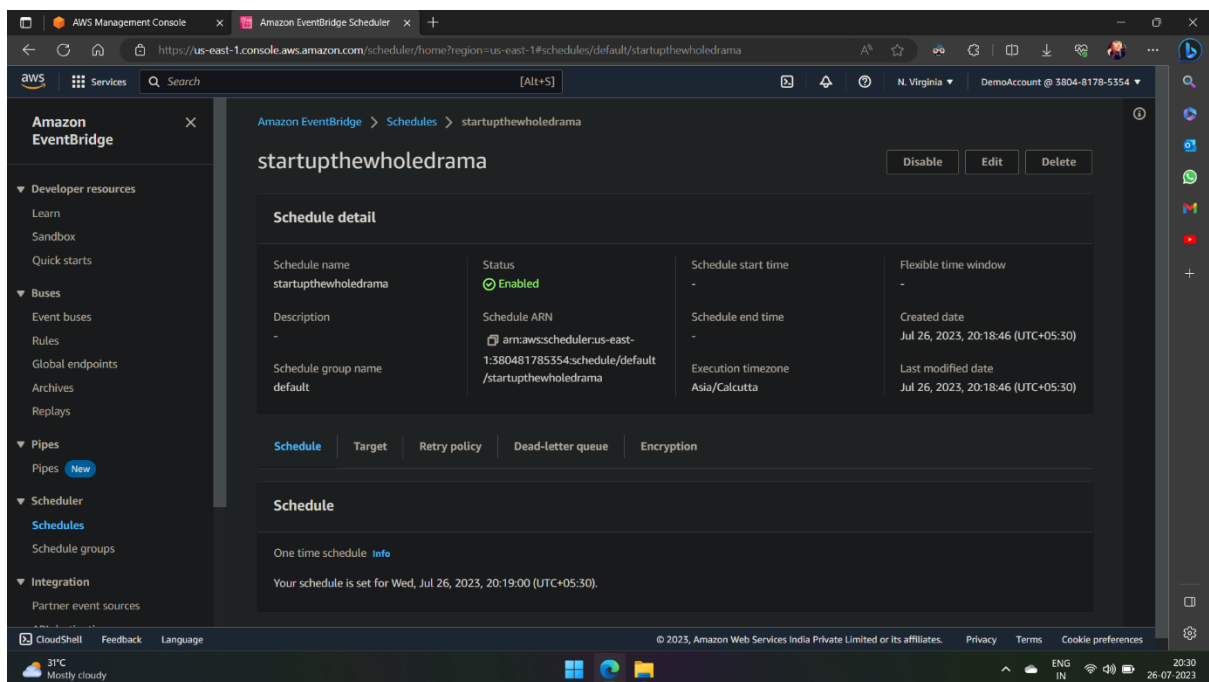
-Make sure you have the AWS CLI installed and properly configured on your Windows Command Prompt with your AWS account credentials before running these commands.

-Note: If you haven't configured the AWS CLI yet, you can run `aws configure` command to set up your access key, secret access key, default region, and output format.



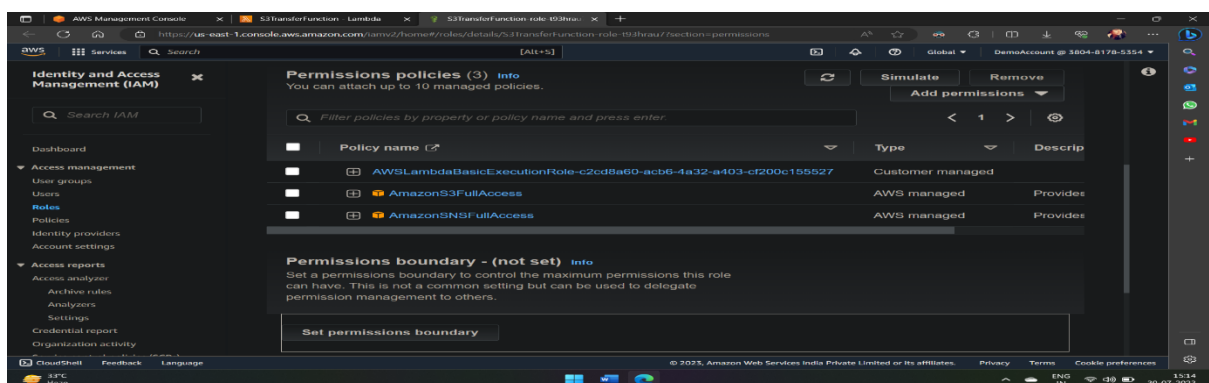
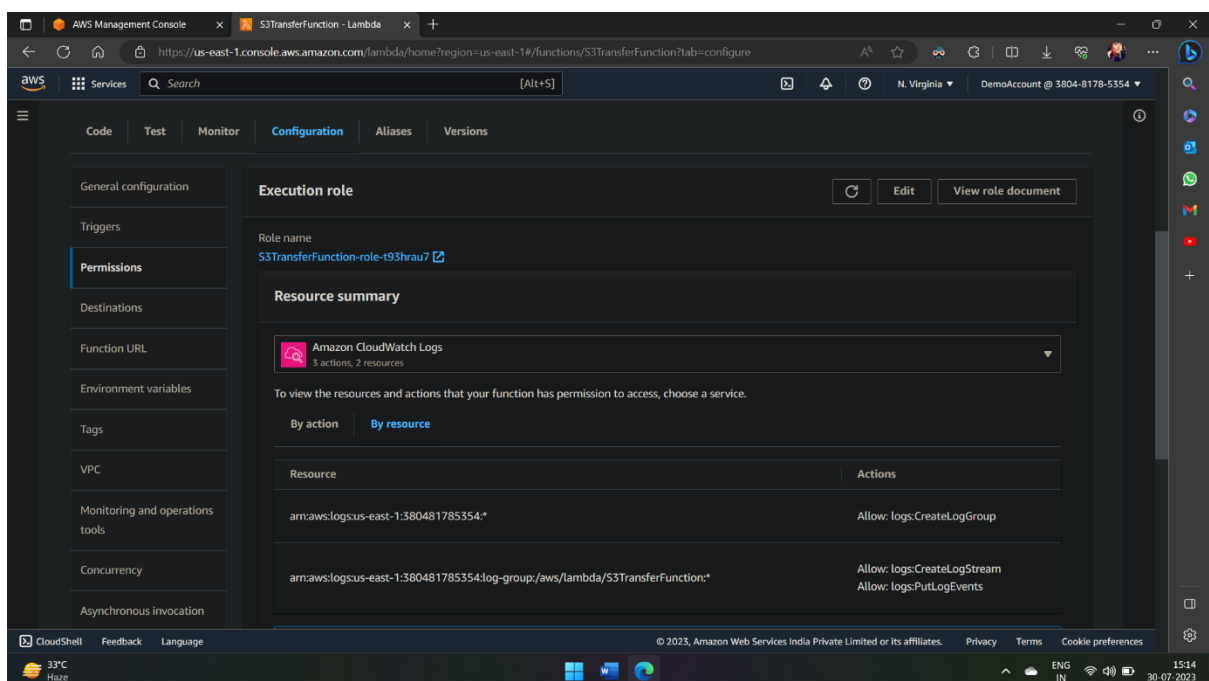
Step 5: Set up the EventBridge Schedule.

- Go to the AWS Management Console and navigate to the EventBridge service.
- Click on "Scheduler" in the left sidebar and then click on "Create schedule."
- Provide a name and description for the schedule.
- Under "Occurrence" select "One Time Schedule."
- Select the desired Date & Time and the suitable Timezone.
- Under "Targets", choose "AWS Lambda Invoke."
- Select the Lambda function you created in Step 2 from the dropdown.
- For now keep the default settings for Retry policy and dead-letter queue (DLQ)
- Under "Permissions" click on "Create new role for this schedule."
- Click on "Create Schedule."



Step 6: Providing the appropriate Permissions to Lambda.

- Go to the AWS Management Console and navigate to the Lambda service.
- Select the created Lambda functions."
- Under the "Configurations" section, select the option "Triggers."
- Now click on the execution role created, it will automatically redirect you to the IAM console.
- Click on "Add Permissions" and the "Attach Policy."
- Select AmazonSNSFullAccess & AmazonS3FullAccess and attach both the policies to the role.



That's it! Now, your Lambda function will be triggered by the EventBridge scheduler at the scheduled time, and it will copy all the files from the source S3 bucket to the destination S3 bucket.

Additionally, ensure that the Lambda function's timeout is sufficient to handle the number of files you expect to transfer within the scheduled timeframe. Adjust the timeout if necessary. Always double-check your configurations and test the setup with caution to avoid any unintended consequences.

