**EX NO 6 AUTOMATION AND OPTIMIZATION WITH S3**

**Aim**:

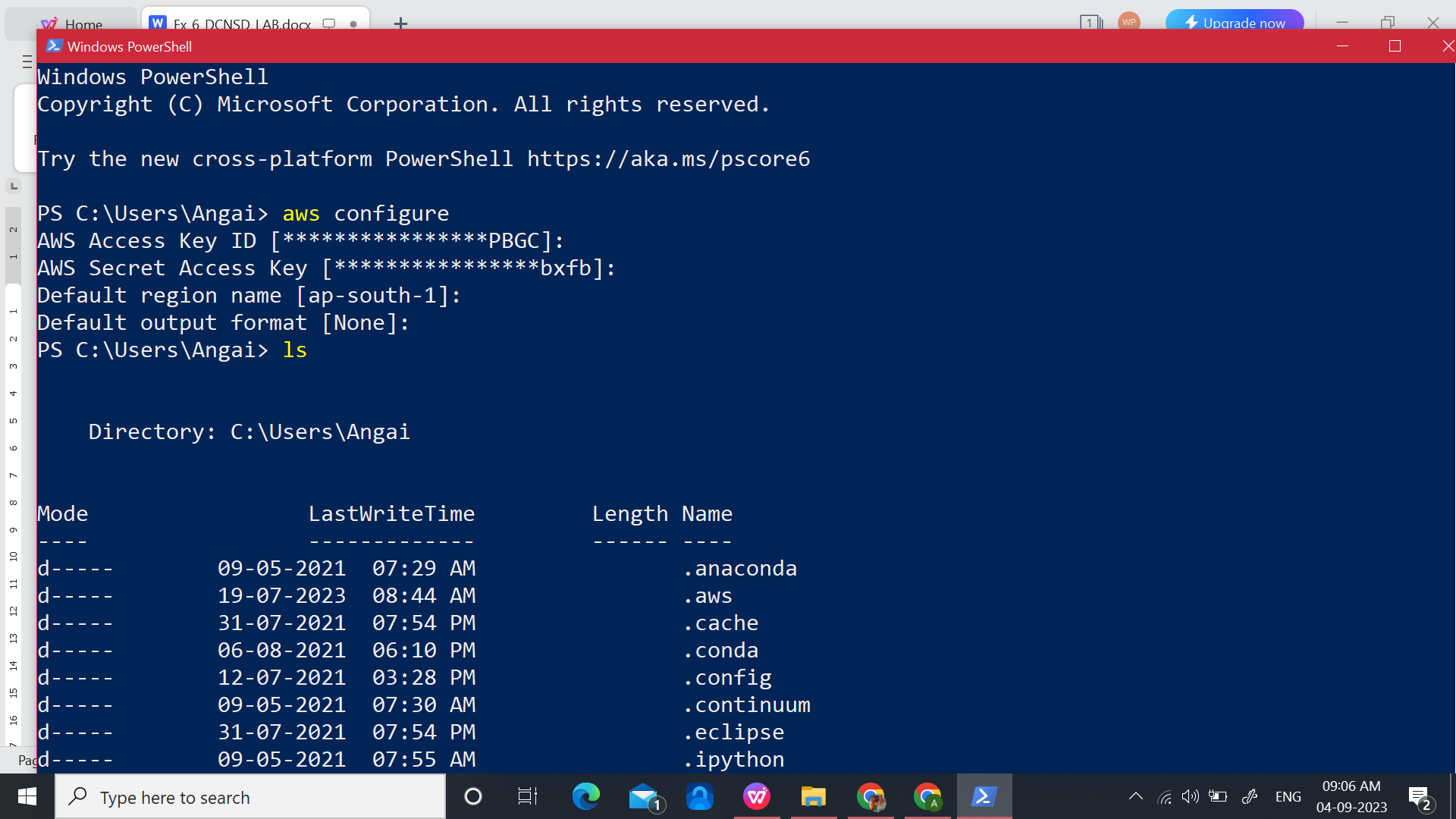
# To automate the movement of Objects Between S3 Buckets using sync command and Lambda function.

**YOU CAN USE AWS LEARNERS LAB**

**AUTOMATION USING SYNC COMMAND**:

There may be instances where we need to move data from one [S3 bucket](https://saturncloud.io/glossary/s3-bucket" \t "https://saturncloud.io/blog/automating-the-movement-of-objects-between-s3-buckets-a-comprehensive-guide/_blank) to another.

1. Open the [Amazon S3 console](https://s3.console.aws.amazon.com/s3" \t "https://repost.aws/knowledge-center/_blank).
2. Choose Create Bucket.
3. Choose a [DNS-compliant name](https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucketnamingrules.html" \t "https://repost.aws/knowledge-center/_blank) for your new bucket (bucket1).
4. Click the created bucket and Upload a new file (welcome.txt)
5. Choose Create Bucket.
6. Choose a [DNS-compliant name](https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucketnamingrules.html" \t "https://repost.aws/knowledge-center/_blank) for your new bucket (bucket2).
7. Click and check that the bucket is empty (no objects are there).
8. Open the command prompt and configure aws



1. Create a directory (folder) named ‘sample1’ and create any file inside the directory.

### **AWS sync command**

### The AWS sync command may be a command utilized in the AWS S3 storage. This Command is employed to sync directories to S3 buckets, prefixes, and the other way around . AWS sync command recursively copies new and updated files from the source ( Directory or Bucket/Prefix ) to the destination ( Directory or Bucket/Prefix ).

### **AWS Sync command usage:**

1. **syncs the bucket to the local current directory**

aws s3 sync . s3://bucket1

Output:

upload: test.txt to s3://bucket1/test.txt

The above AWS sync command syncs objects of the bucket to files in a local directory by **uploading** the local files to s3.

1. **syncs bucket1 to bucket2.**

aws s3 sync s3://mybucket s3://mybucket2

Output:

copy: s3://bucket1/test.txt to s3://bucket2/test.txt

copy: s3://bucket1/welcome.txt to s3://bucket2/welcome.txt

The above AWS sync command syncs objects from one bucket to another bucket by **copying** s3 objects.

1. **syncs the current local directory to the bucket.**

aws s3 sync s3://mybucket .

Output:

download: s3://bucket1/test.txt to test.txt

download: s3://bucket1/welcome.txt to welcome.txt

**AUTOMATION USING LAMBDA FUNCTION**:

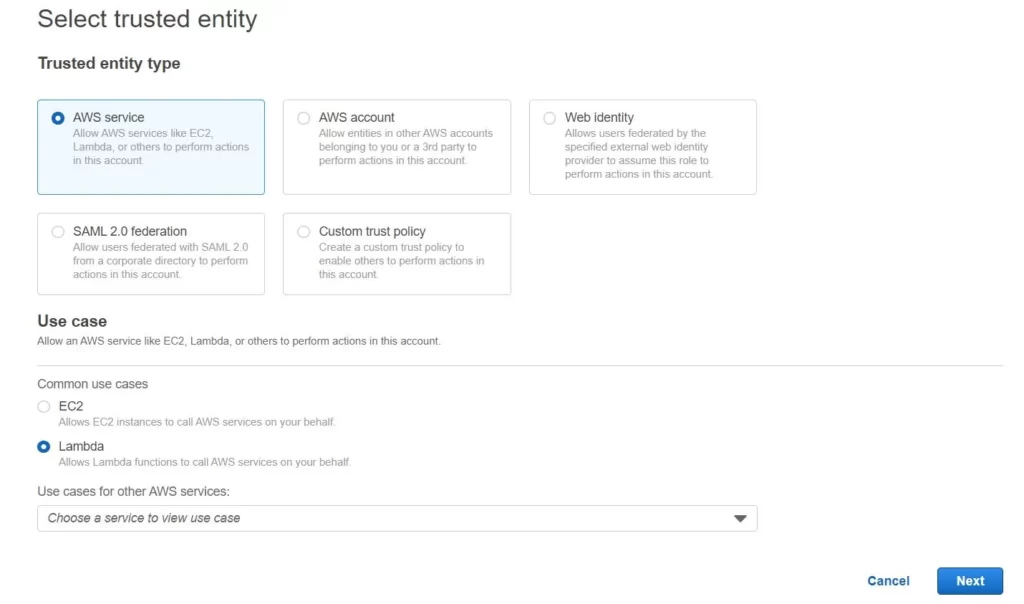
There may be instances where we need to move data from one [S3 bucket](https://saturncloud.io/glossary/s3-bucket" \t "https://saturncloud.io/blog/automating-the-movement-of-objects-between-s3-buckets-a-comprehensive-guide/_blank) to another.

**Bucket Creation:**

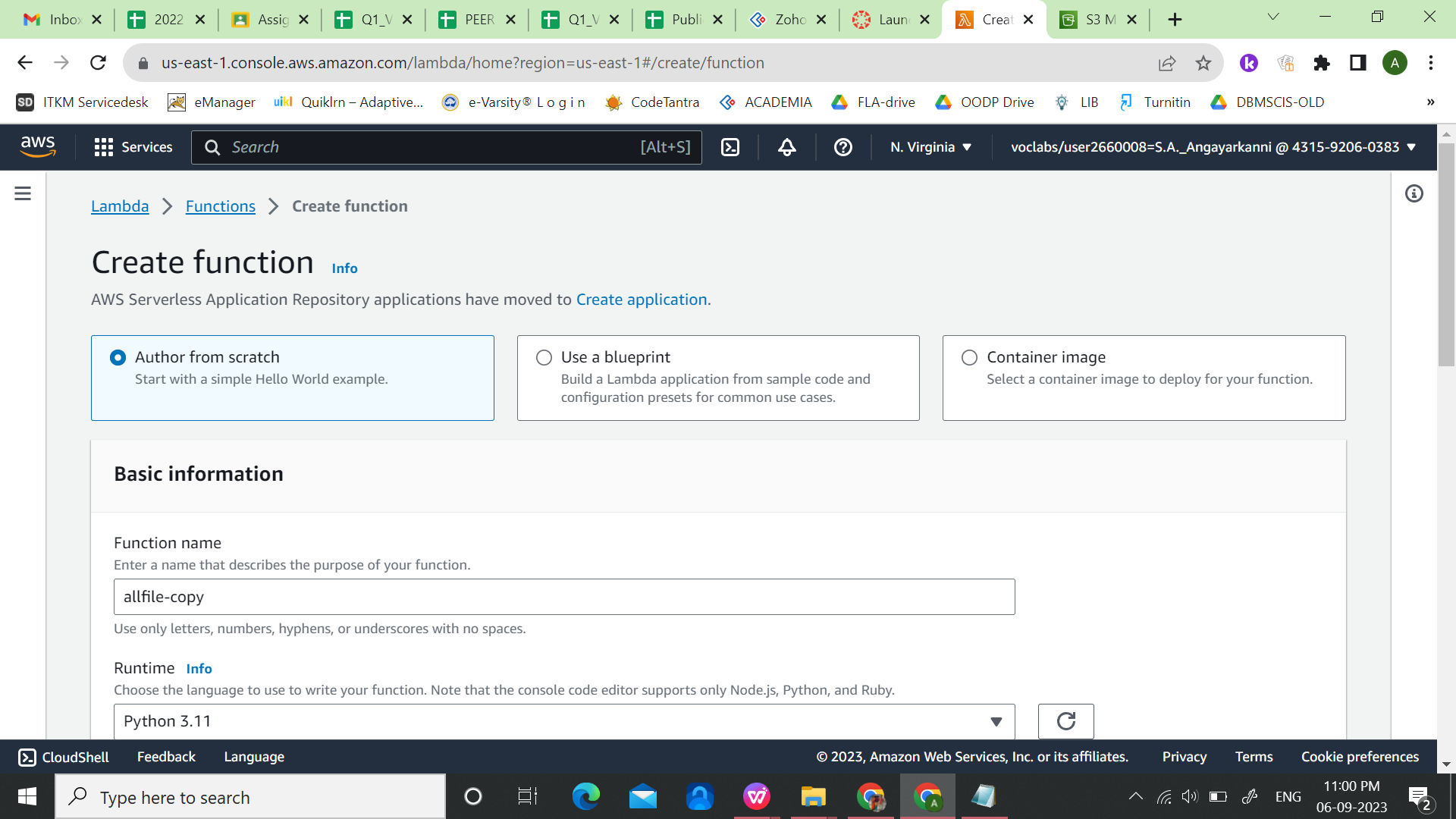
1. Open the [Amazon S3 console](https://s3.console.aws.amazon.com/s3" \t "https://repost.aws/knowledge-center/_blank).
2. Choose Create two Bucket.
3. Choose a [DNS-compliant name](https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucketnamingrules.html" \t "https://repost.aws/knowledge-center/_blank) for your new bucket (bucket-src).
4. Click the created bucket and Upload a new file (welcome.txt)
5. Choose Create Bucket.
6. Choose a [DNS-compliant name](https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucketnamingrules.html" \t "https://repost.aws/knowledge-center/_blank) for your new bucket (bucket-dest). Click and check that the bucket is empty (no objects are there).
7. Make sure you select a correct region. It is best not to enable public access without the need to ‘Block all public access’.

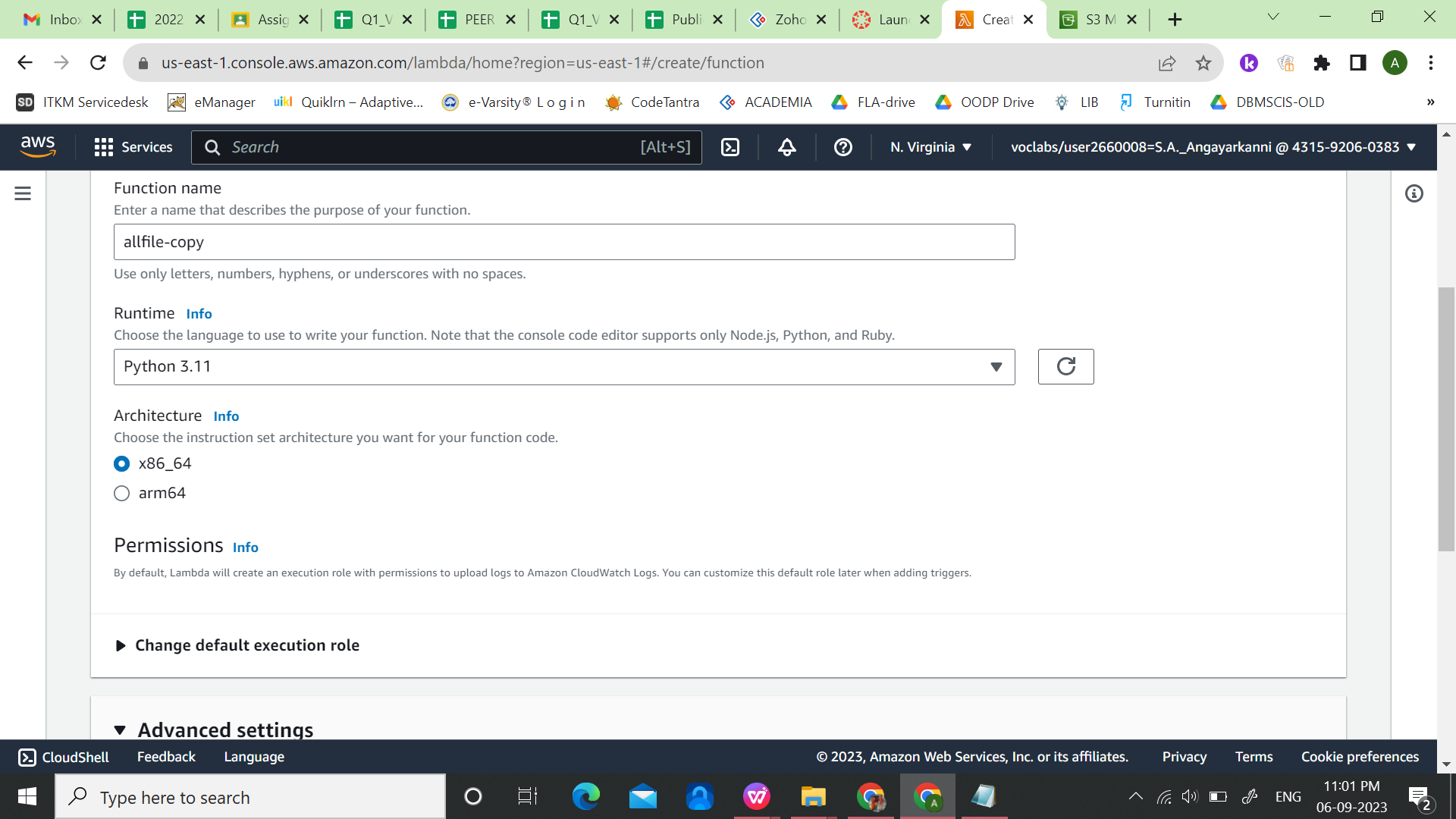
**Lambda Function Creation:**

1. Open the IAM console.

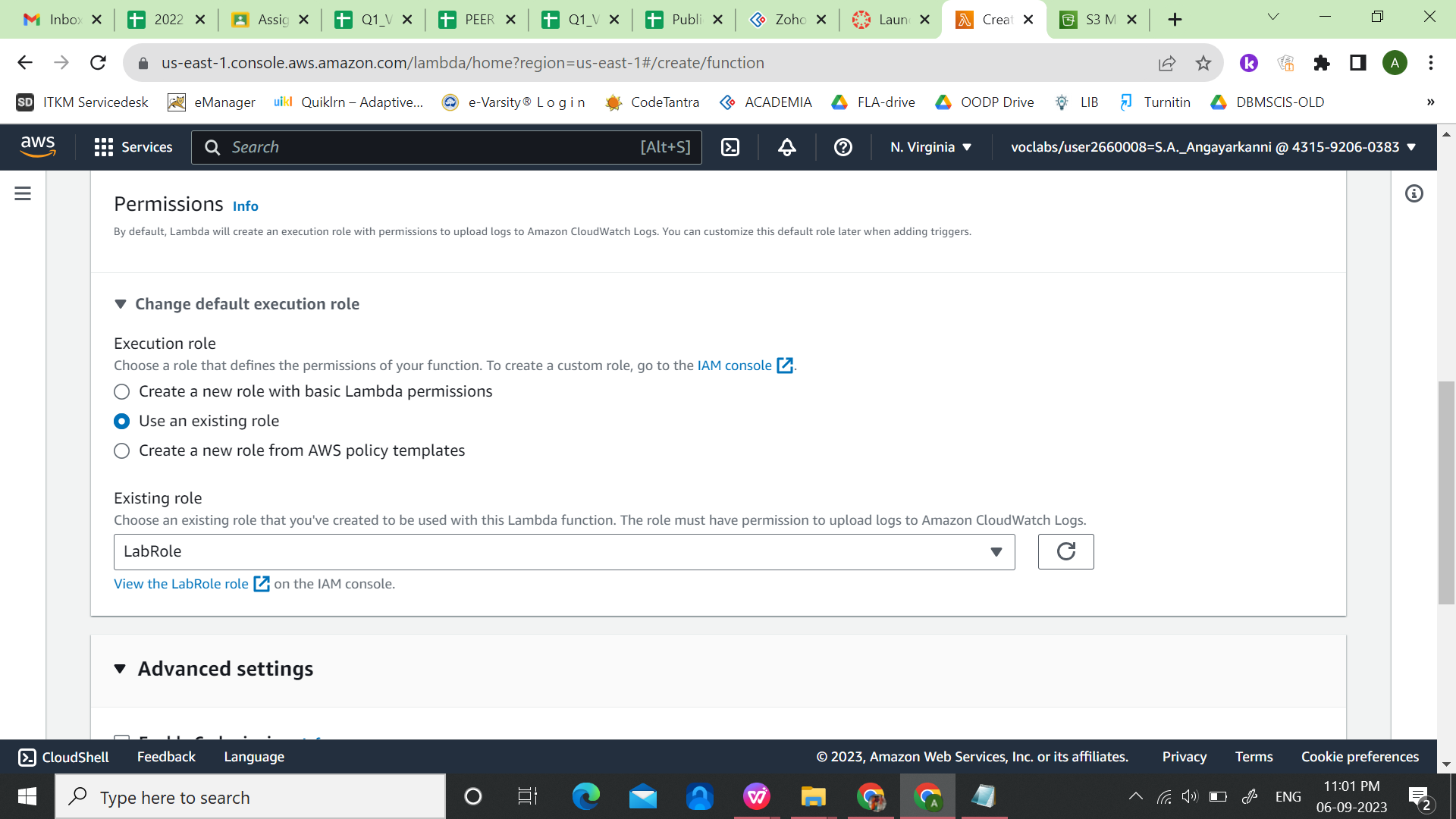


1. Create a new IAM role for the lambda function. Select ‘Author from Scratch’. Function name is given as ‘allfile-copy’. Select ‘Python’.

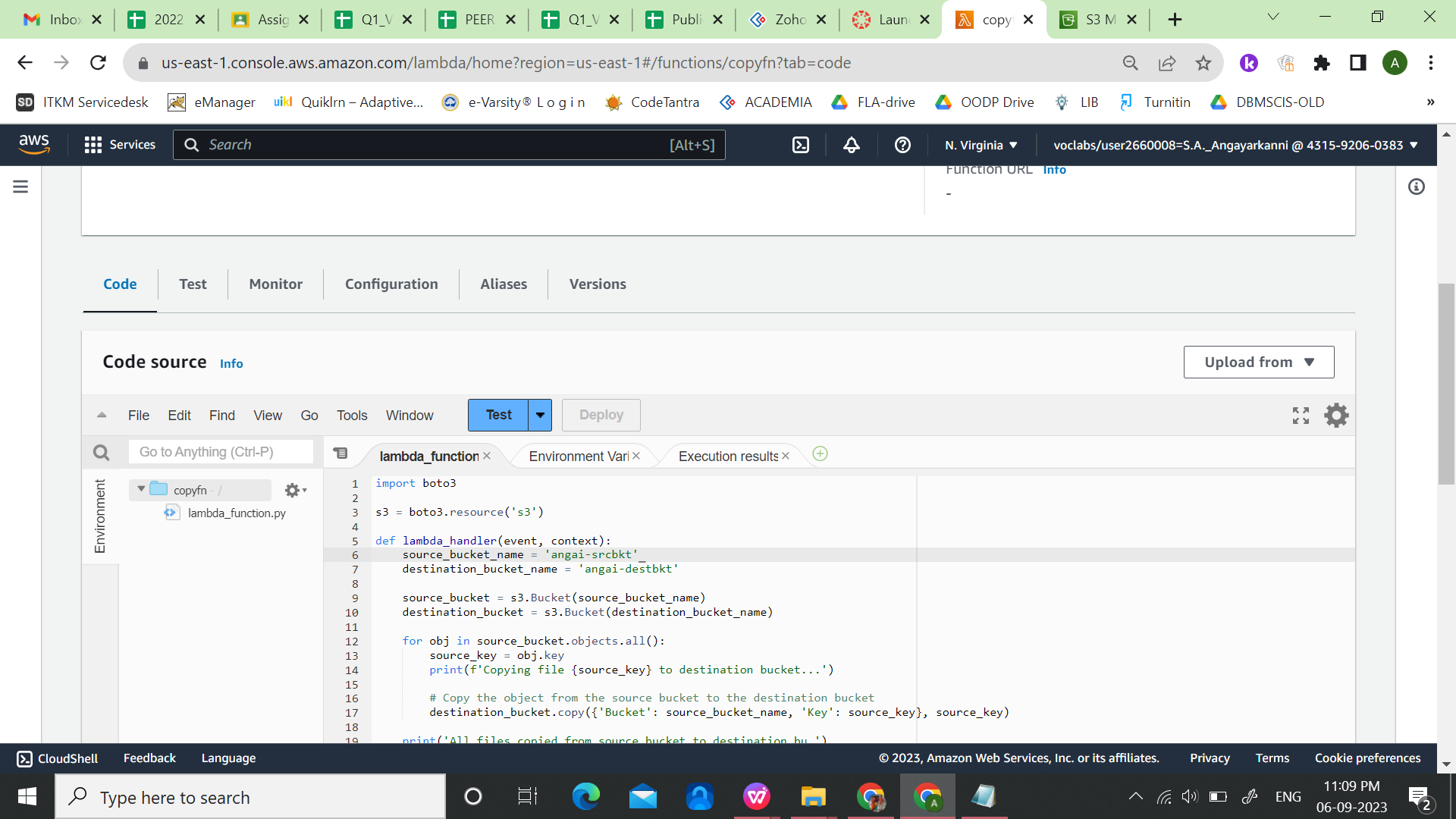




1. Expand ‘Change default execution role’. Select ‘Use an existing role’. Select ‘LabRole’ from the list of roles.

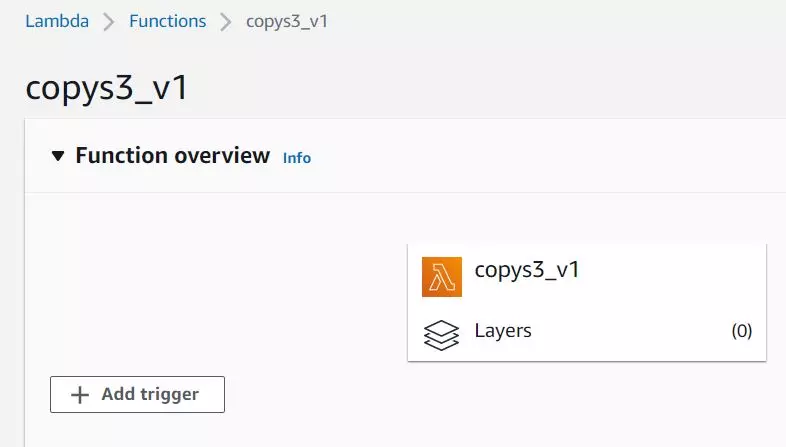


1. Click ‘Create Function’.
2. Under the Code. Type the given python code.



|  |
| --- |
| **Python code to automatically copy all objects from source bucket to destination bucket** |
| import boto3  s3 = boto3.resource('s3')  def lambda\_handler(event, context):  source\_bucket\_name = 'angai-srcbkt'  destination\_bucket\_name = 'angai-destbkt'  source\_bucket = s3.Bucket(source\_bucket\_name)  destination\_bucket = s3.Bucket(destination\_bucket\_name)  for obj in source\_bucket.objects.all():  source\_key = obj.key  print(f'Copying file {source\_key} to destination bucket...')    # Copy the object from the source bucket to the destination bucket  destination\_bucket.copy({'Bucket': source\_bucket\_name, 'Key': source\_key}, source\_key)  print('All files copied from source bucket to destination bucket.') |

1. Then ‘Deploy’ the code.
2. Once you have made sure that everything is working correctly, then you can add the automation. In ‘Function overview‘, you click Add trigger:



1. Select S3 from the list and the name of your source bucket, that is where you will be adding files.
2. You set the event type to PUT because you only want the function to be triggered when objects are added.
3. Click Add
4. You can test the function by adding files to S3 via the UI, or using CLI. Whenever an object is created in source bucket, it get automatically copied to destination bucket.

|  |
| --- |
| **Python code to automatically copy objects from source bucket to destination bucket, whenever an object is deleted from source bucket** |
| import boto3  import os  def lambda\_handler (event, context):  s3 = boto3.resource('s3')  source\_bucket = 'angai-srcbkt'  source = s3.Bucket('angai-srcbkt')  destination = s3.Bucket('angai-destbkt')    for obj in source.objects.all():  copy\_source = {  'Bucket': 'angai-srcbkt',  'Key': obj.key  }  destination.copy(copy\_source, obj.key) |