

1. Bucket Creation

3 buckets are created for the task. One to store the original file, second to replicate and third to change the case of the text.

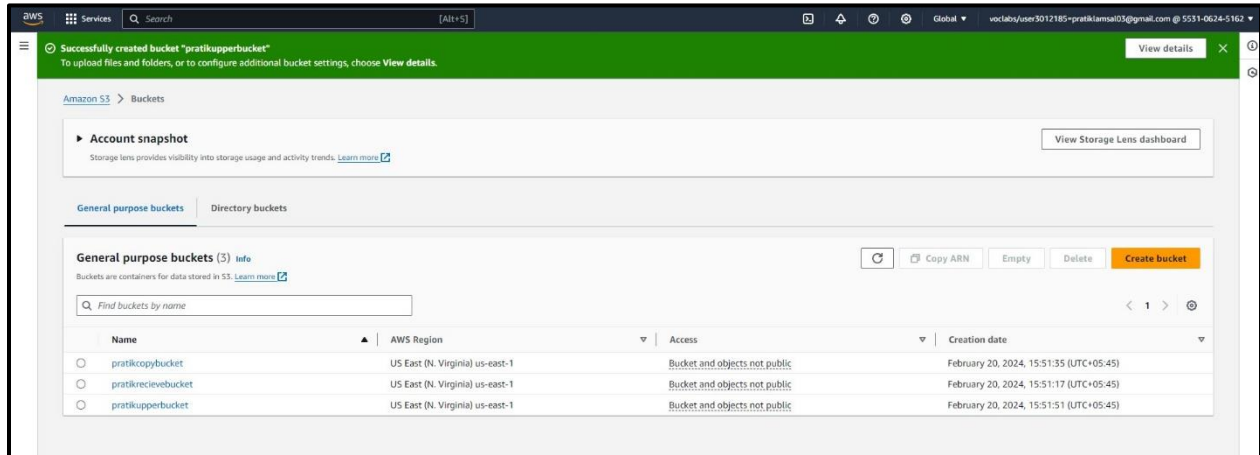


Figure 1 Three Bucket Creation

2. Lambda Function Creation

A lambda function is created to perform the task. Python 3.12 runtime is selected.

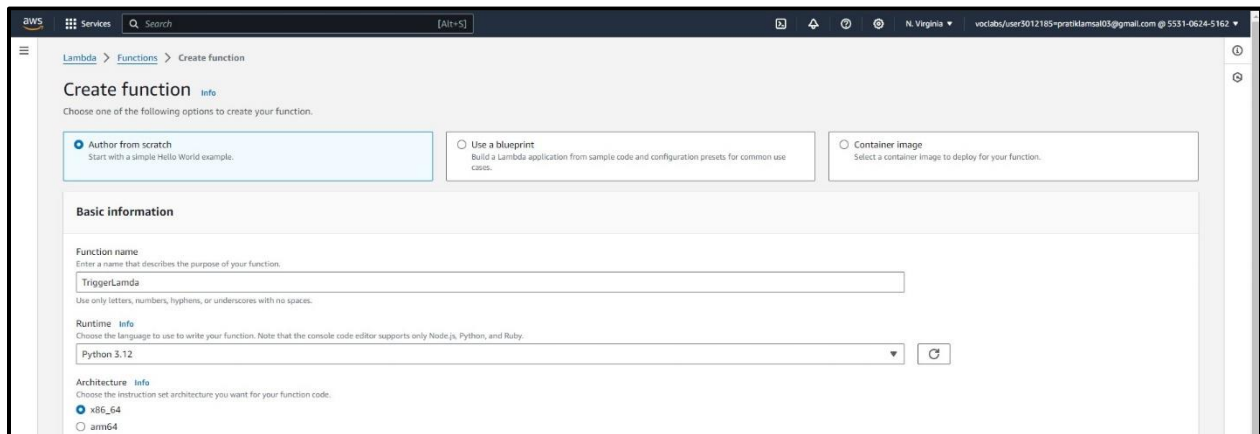


Figure 2 Lambda Function Creation

3. Lambda Function

An existing role (LabRole) is assigned in Permissions and the lambda function is created.

The screenshot shows the 'Permissions' tab in the AWS Lambda console. It includes a section for 'Change default execution role' with options to create a new role or use an existing role. The 'Existing role' dropdown is set to 'LabRole'. Below this is the 'Advanced settings' section with checkboxes for 'Enable Code signing', 'Enable function URL', 'Enable tags', and 'Enable VPC'. At the bottom right are 'Cancel' and 'Create function' buttons.

Figure 3 Lambda Function LabRole

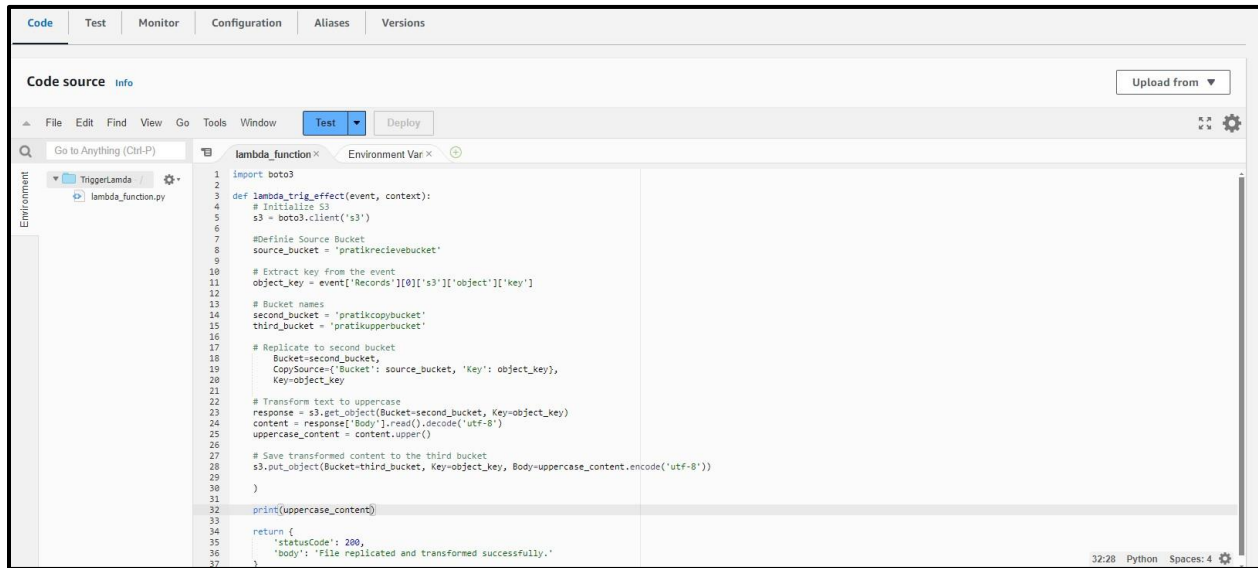
4. Lambda Function Created

The screenshot shows the 'TriggerLamda' function overview in the AWS Lambda console. It displays the function name, a diagram of the function with layers, and a list of destinations. The 'Function overview' tab is active, showing the function's description, last modified time, and function ARN. The 'Code source' tab is also visible at the bottom.

Figure 4 Successful Lambda Function Creation

5. Code Source in Lambda Function

Code added to Lambda Function and deployed.



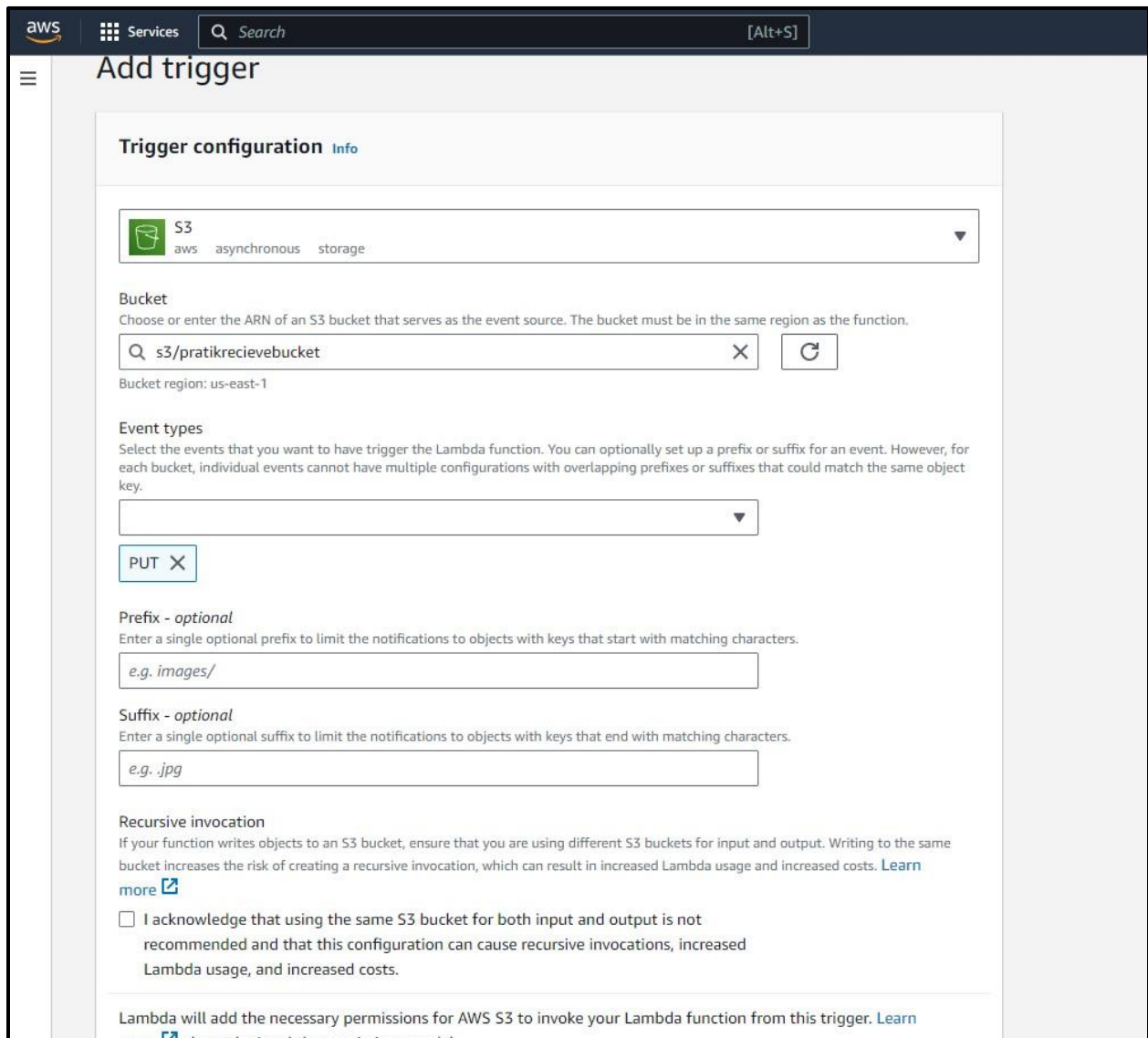
The screenshot displays the AWS Lambda console interface. At the top, there are tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code' tab is selected. Below the tabs, there's a 'Code source' section with an 'Info' link and an 'Upload from' button. The main area shows the code for the 'lambda_function' environment variable. The code is written in Python and uses the boto3 library to interact with Amazon S3. It defines a lambda function 'lambda_trig_effect' that takes an event and context as input. The function initializes an S3 client, defines a source bucket, extracts a key from the event, and then performs a series of operations: it gets an object from a second bucket, transforms the content to uppercase, and saves the transformed content to a third bucket. Finally, it prints the uppercase content and returns a status code of 200 with a success message.

```
1 import boto3
2
3 def lambda_trig_effect(event, context):
4     # Initialize S3
5     s3 = boto3.client('s3')
6
7     # Define Source Bucket
8     source_bucket = 'pratikreclivebucket'
9
10    # Extract key from the event
11    object_key = event['Records'][0]['s3']['object']['key']
12
13    # Bucket names
14    second_bucket = 'pratikcopybucket'
15    third_bucket = 'pratikupperbucket'
16
17    # Replicate to second bucket
18    Bucket=second_bucket,
19    CopySource={'Bucket': source_bucket, 'Key': object_key},
20    Key=object_key
21
22    # Transform text to uppercase
23    response = s3.get_object(Bucket=second_bucket, Key=object_key)
24    content = response['body'].read().decode('utf-8')
25    uppercase_content = content.upper()
26
27    # Save transformed content to the third bucket
28    s3.put_object(Bucket=third_bucket, Key=object_key, Body=uppercase_content.encode('utf-8'))
29
30
31
32    print(uppercase_content)
33
34    return {
35        'statusCode': 200,
36        'body': 'File replicated and transformed successfully.'
37    }
```

Figure 5 Code Source in Lambda Function

6. Adding Trigger to Lambda Function

A trigger is configured for the Lambda Function.



The screenshot shows the 'Add trigger' configuration page in the AWS Lambda console. The page is titled 'Add trigger' and has a 'Trigger configuration' section with an 'Info' link. The trigger is configured for an S3 bucket. The bucket name is 's3/pratikrecievebucket' (note the typo in the image) and the bucket region is 'us-east-1'. The event types are set to 'PUT'. There are optional fields for a prefix ('e.g. images/') and a suffix ('e.g. .jpg'). A 'Recursive invocation' section contains a checkbox for acknowledging that using the same S3 bucket for both input and output is not recommended. At the bottom, there is a note that Lambda will add the necessary permissions for AWS S3 to invoke the Lambda function from this trigger, with a 'Learn more' link.

aws Services Search [Alt+S]

Add trigger

Trigger configuration Info

S3
aws asynchronous storage

Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.

Q s3/pratikrecievebucket X ↻

Bucket region: us-east-1

Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

▼

PUT X

Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

e.g. images/

Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

e.g. .jpg

Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)

☐ I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. [Learn more](#)

Figure 6 Trigger Configuration

7. File Uploaded to First Bucket

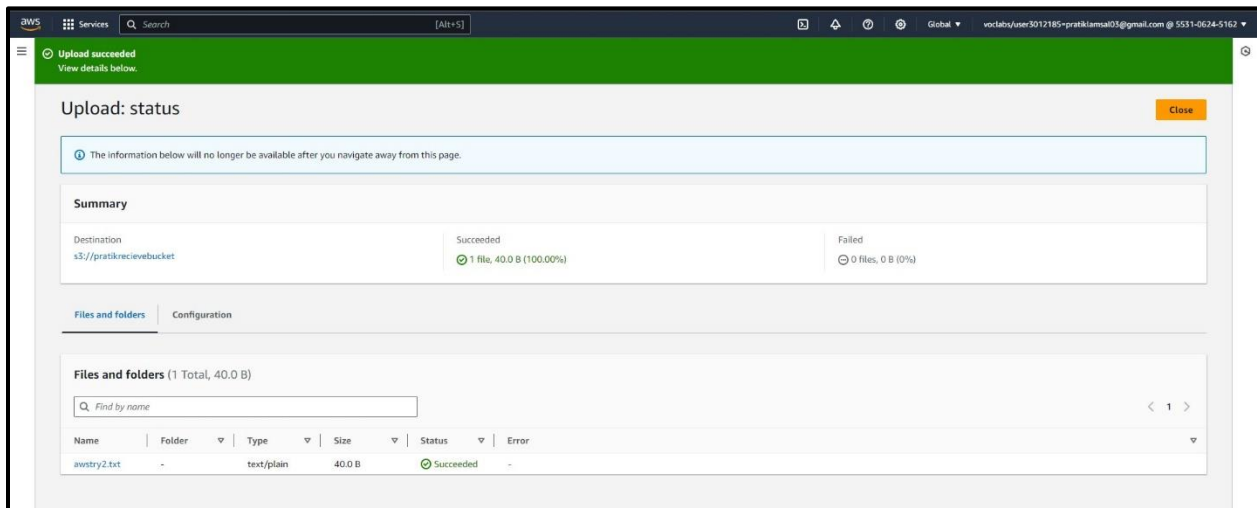


Figure 7 Successful File Upload in First Bucket

8. Checking Logs in CloudWatch

Logs is now checked to see if it is successfully executed.

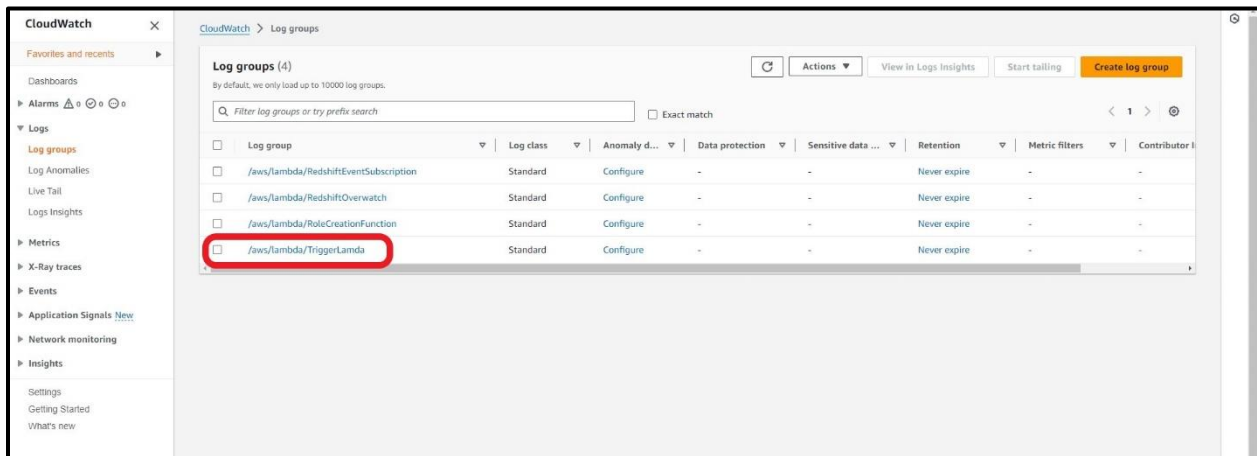


Figure 8 CloudWatch Lambda Log

9. Errors in Log Events

Multiple errors found in logs.

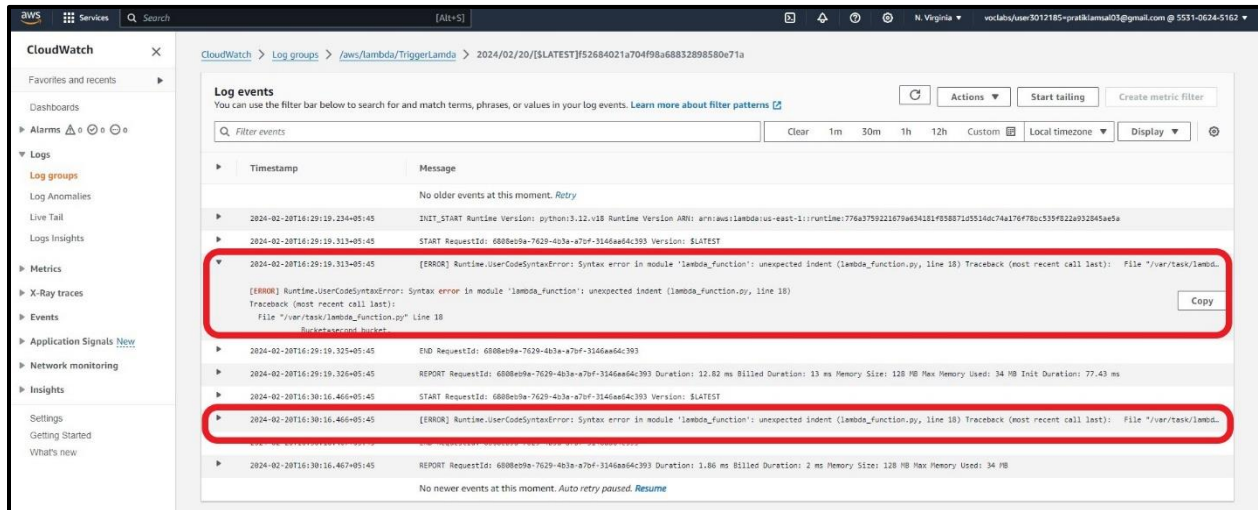


Figure 9 Errors in Log Events

10. Code Fixed and Handler Changed

The error in the code is fixed and the Handler is changed according to the code deployed.

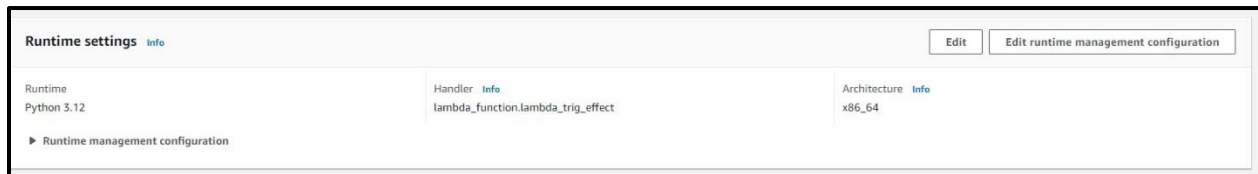


Figure 10 Handler Change

11. Successful Execution

On checking the Log Events once again, we can see lack of errors and printed, changed text file content.

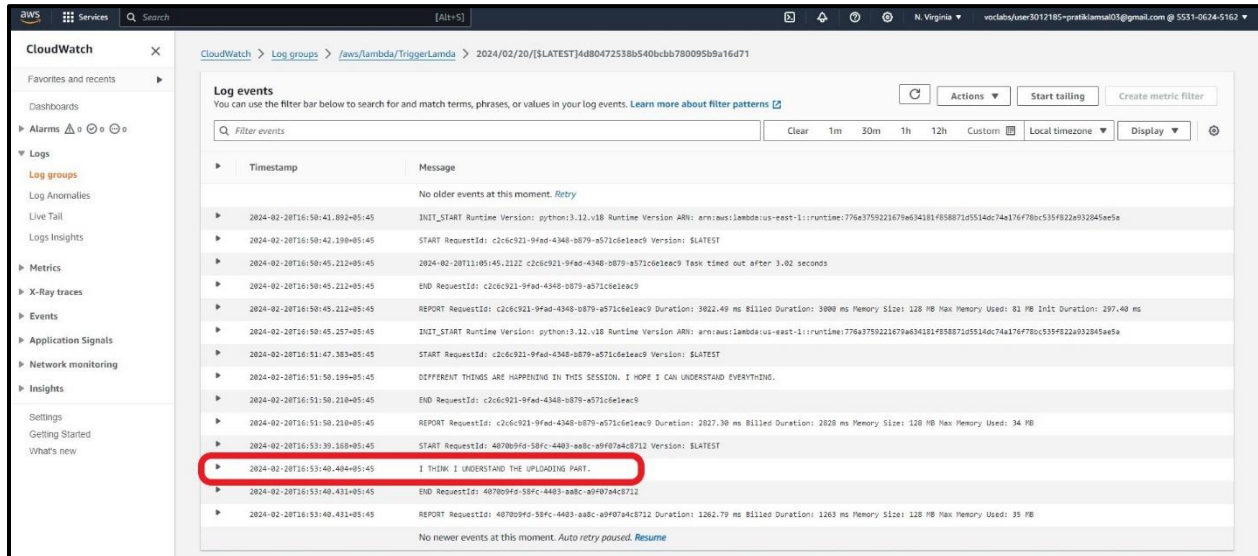


Figure 11 Successful Execution

12. Attempting to Download File

The changed file into uppercase from the third bucket is attempted to be downloaded into the local machine.

```
#Downloading to Local Machine
local_file_path = 'C:\Users\craze\OneDrive\Desktop\aws_class'

# Download file from S3 bucket to local machine
download_file_from_s3(third_bucket, object_key, local_file_path)

return {
    'statusCode': 200,
    'body': 'File replicated and transformed successfully.'
}
```

Figure 12 Code to Download into Local Machine

13. Until now, couldn't do it.