

AdvDevOps Case Study 12: Serverless Logging with S3 and Lambda

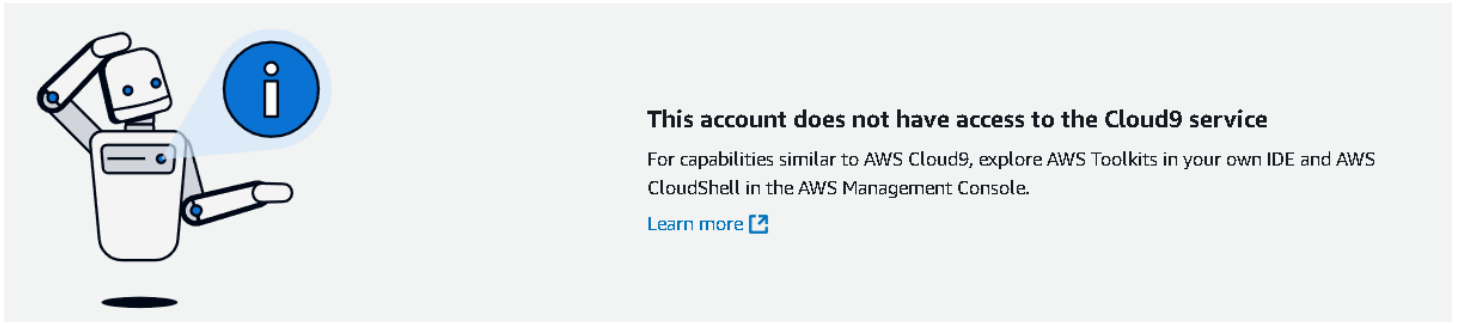
Name: Soham Satpute

Roll No: 52/D15A

- **Concepts Used:** AWS Lambda, S3, and AWS Cloud9.
- **Problem Statement:** "Set up a Lambda function using AWS Cloud9 that triggers when a text file is uploaded to an S3 bucket. The Lambda function should read the file's content and log it."
- **Tasks:**
 - Create a Lambda function in Python using AWS Cloud9.
 - Configure an S3 bucket as the trigger for the Lambda function.
 - Upload a text file to the S3 bucket and verify that the Lambda function logs the content.

Note:

AWS **Cloud9** has been **discontinued**, so we will now use **EC2** for our development environment.



Introduction:

Overview:

This case study focuses on the implementation of a serverless architecture using AWS services, specifically AWS Lambda and Amazon S3, to automate the logging of text file contents uploaded to an S3 bucket. The project capitalizes on the event-driven model of AWS Lambda, which allows functions to be triggered automatically in response to events, such as file uploads. This approach eliminates the need for manual processes, streamlining workflows and reducing the time from data generation to logging.

The chosen architecture supports a variety of applications, from monitoring and auditing file uploads to real-time data analytics. By leveraging the scalability and flexibility of AWS, this implementation can efficiently handle varying loads of file uploads, making it suitable for diverse use cases such as content management systems, log aggregation, and data pipeline automation.

Key Features and Applications:

1. Event-Driven Architecture:

- The solution utilizes an event-driven architecture, where the Lambda function is automatically triggered by specific events in the S3 bucket, specifically `s3:ObjectCreated:*`. This ensures immediate processing of the uploaded files without manual intervention, enhancing responsiveness and efficiency.

2. Real-Time Data Processing:

- By reading and logging the contents of text files in real-time, the system provides immediate feedback to users or systems that rely on this data. This is crucial for applications that require timely information, such as monitoring changes in configurations, updates in content, or logging user activities.

3. Scalability:

- AWS Lambda inherently scales to accommodate varying loads. As file uploads increase, the Lambda function can handle multiple concurrent executions, allowing the system to grow without requiring changes to the underlying infrastructure.

4. Cost Efficiency:

- The serverless nature of this solution means that users only pay for the compute time consumed by the Lambda function when it is invoked, rather than maintaining dedicated servers. This cost model is especially advantageous for sporadic or unpredictable workloads.

5. Seamless Integration with AWS Services:

- The solution integrates smoothly with other AWS services. For instance, Amazon S3 serves as a durable and highly available storage solution, while AWS CloudWatch provides logging and monitoring capabilities, enabling easy tracking of function executions and error handling.

6. Robust Logging and Monitoring:

- Utilizing AWS CloudWatch, the Lambda function logs all relevant activities, including the contents of the uploaded files. This logging capability allows for comprehensive monitoring and auditing, making it easier to track file processing and identify issues quickly.

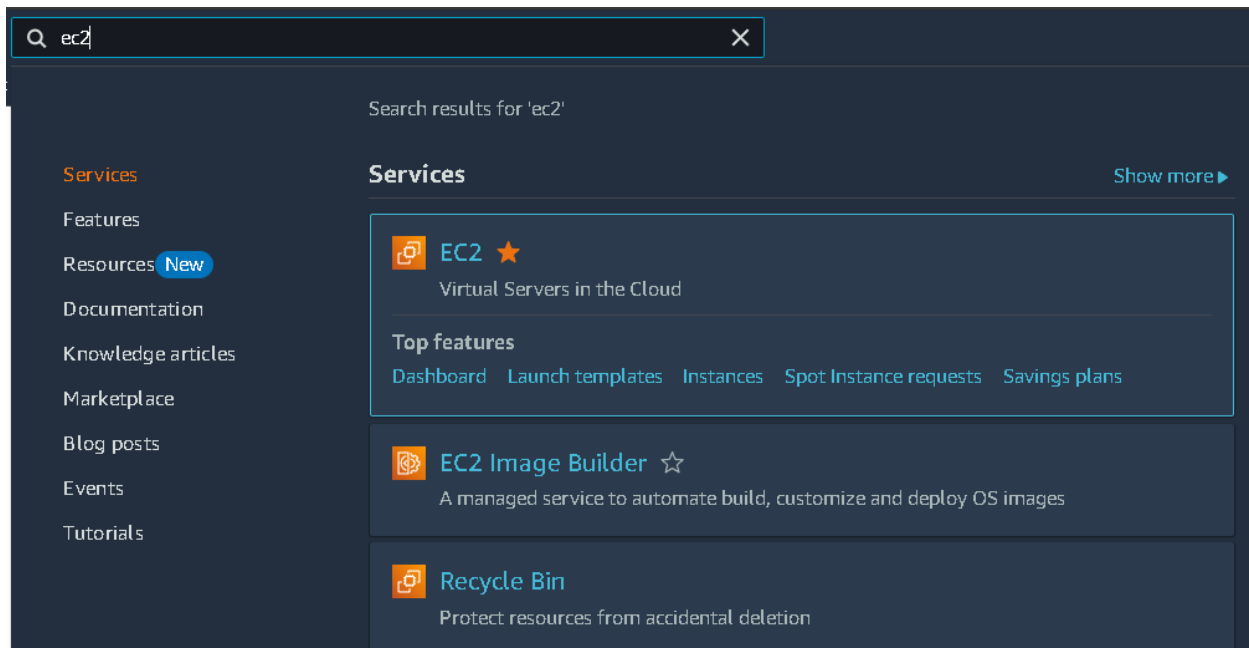
Steps:

1. **Create an S3 Bucket:**
 - Set up an S3 bucket (named soham-lambucket) to store uploaded text files.
2. **Write the Lambda Function:**
 - Created a Python Lambda function to read the content of text files uploaded to the S3 bucket and log that content using CloudWatch.
3. **Set Permissions:**
 - Configured an IAM policy allowing the Lambda function to access the S3 bucket and log content.
4. **Configure S3 Trigger:**
 - Set the S3 bucket to trigger the Lambda function on the s3:ObjectCreated:* event, ensuring the function runs whenever a new file is uploaded.
5. **Upload a Test File:**
 - Uploaded a text file (e.g., test1.txt) to the S3 bucket to test the functionality.
6. **Verify Logs:**
 - Checked CloudWatch logs to confirm that the Lambda function executed successfully and logged the content of the uploaded file.

STEPS:

1. Launch an EC2 Instance

1.1 Login to AWS Console and go to EC2 service.



1.2 Click on "Launch Instance".

- AMI: Choose Amazon Linux 2.
- Instance Type: Select t2.micro (eligible for free tier).
- Key Pair: Create a new key pair (or select an existing one). You'll need this for SSH access.
- Network Settings:
 - Choose default VPC.
 - Security Group: Create a new security group:
 - Inbound Rules:
 - SSH (TCP port 22): Allow from your IP.
 - HTTP (TCP port 80): Optional, allows browser access.
 - HTTPS (TCP port 443): Optional, for secure traffic.
 - Outbound Rules:
 - Allow all outbound traffic (default).

EC2 > ... > Launch an instance

Launch an instance

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Name

SohamLambda

Add additional tags

Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recents

Quick Start

Summary

Number of instances

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.6.2...read more

ami-06b21ccaeff8cd686

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance

Cancel

Launch instance

Preview code

Key pair (login)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

instkey

Create new key pair

Network settings

Edit

Network

vpc-0cff82e96e6a67ffa

Subnet

No preference (Default subnet in any availability zone)

Auto-assign public IP

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups)

Create security group

Select existing security group

Services

Search

[Alt+S]

Network

Info

vpc-03d1ce76af665f00f

Subnet

Info

No preference (Default subnet in any availability zone)

Auto-assign public IP

Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups)

Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Select existing security group

We'll create a new security group called 'launch-wizard-3' with the following rules:

☒ Allow SSH traffic from

Helps you connect to your instance

Anywhere
0.0.0.0/0

☒ Allow HTTPS traffic from the internet

To set up an endpoint, for example when creating a web server

☒ Allow HTTP traffic from the internet

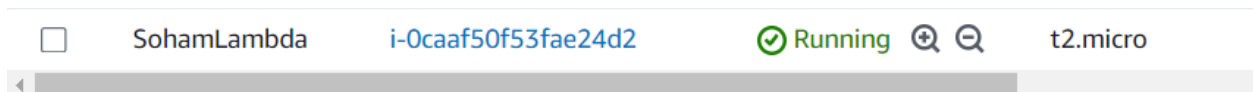
To set up an endpoint, for example when creating a web server

⚠

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

×


1.3 Launch the instance and wait for it to be ready.



1.4 Connect to the EC2 instance via SSH:

```
ssh -i <your-key.pem> ec2-user@<your-ec2-public-dns>
```

```
P:\Users\Soham Satpute\downloads> ssh -i "instkey.pem" ec2-user@ec2-43-205-128-6.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-43-205-128-6.ap-south-1.compute.amazonaws.com (43.205.128.6)' can't be established.
ED25519 key fingerprint is SHA256:/KqjzIkRppQikHBjp+qczfMyt8xOPsgpJ5XH9LQ5z5E.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-43-205-128-6.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
```



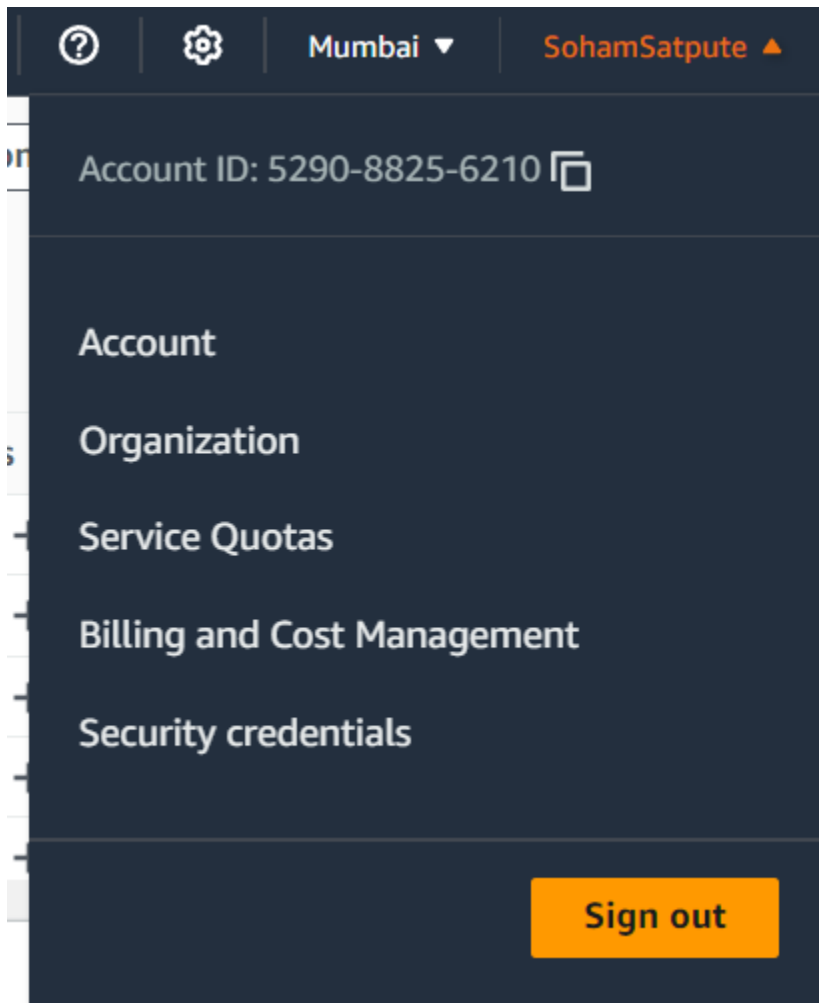
```
#_      #####          Amazon Linux 2023
~ \    /#####|
nnn \   /####|\
nnn  \|  /### |
nnn   \| /#/\__--_\
        V ~ ! t --> https://aws.amazon.com/linux/amazon-linux-2023
       _/_/___/_____/
       _/_/___/_____/
       _/_/___/_____/
       _/_/___/_____/
```

```
[ec2-user@ip-172-31-36-17 ~]$
```

2. Create Access keys for Root user

2.1 Access the Root User Security Credentials:

- In the top-right corner of AWS Management Console, click on your account name or email address, and then click **Security Credentials** from the dropdown menu.



2.2 Manage Root Access Keys:

- Scroll down to the **Access keys for the root account** section.
- If you don't have any existing access keys, click on **Create New Access Key**.
 - This will generate an **Access Key ID** and a **Secret Access Key** for your root user.
- **Download** the keys or **copy** them immediately. You won't be able to see the **Secret Access Key** again after closing this page.

Access keys (0) [Create access key](#)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

Access key ID	Created on	Access key last used	Region last used	Service last used	Status
No access keys					
As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more					
Create access key					

Retrieve access key [Info](#)

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key

Secret access key

AKIAIWMA6DDJJKG4G6FGX

***** [Show](#)

3. Install AWS CLI and Configure EC2

3.1 Update packages and install AWS CLI:

```
sudo yum update -y
sudo yum install aws-cli -y
```

```
[ec2-user@ip-172-31-36-17 ~]$ sudo yum update -y
sudo yum install aws-cli -y
Last metadata expiration check: 0:14:50 ago on Wed Oct 23 03:26:50 2024.
Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 0:14:50 ago on Wed Oct 23 03:26:50 2024.
Package awscli-2-2.15.30-1.amzn2023.0.1.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-36-17 ~]$
```


3.2 Configure AWS CLI:

aws configure

Enter your:

- **AWS Access Key ID**
- **AWS Secret Access Key**
- **Region** (e.g., **us-east-1**)
- **Output format:** **json**

```
[ec2-user@ip-172-31-36-17 ~]$ aws configure
AWS Access Key ID [*****AHKM]:
AWS Secret Access Key [*****6FGX]:
Default region name [us-east-1]: ap-south-1
Default output format [json]:
[ec2-user@ip-172-31-36-17 ~]$
```

3.3 Install Python and pip (since Lambda uses Python):

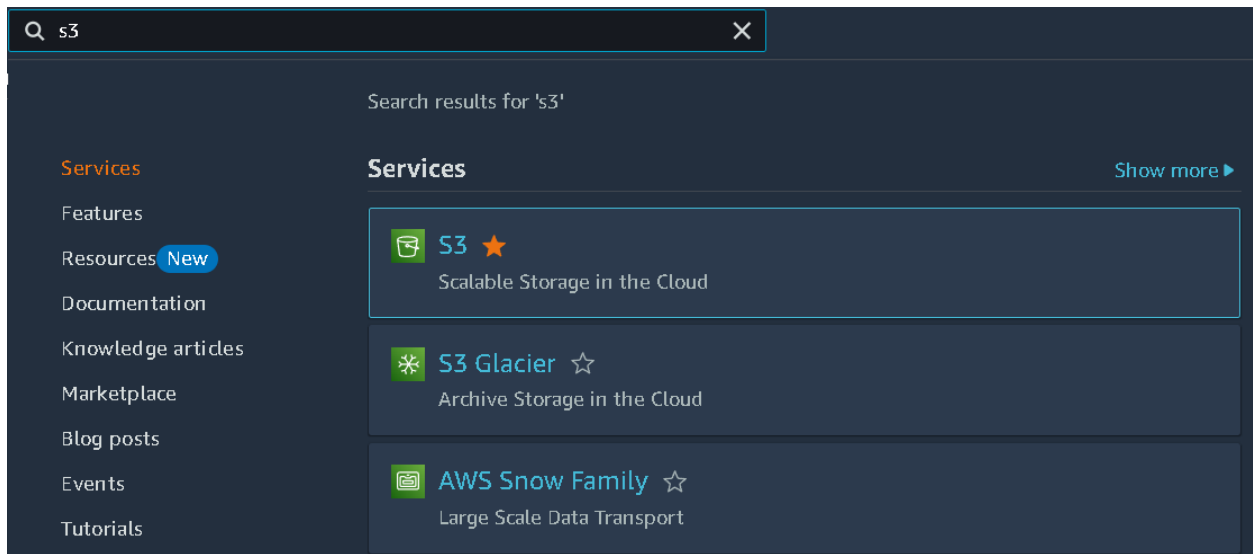
```
sudo yum install python3 -y
```

```
sudo yum install python3-pip -y
```

```
[ec2-user@ip-172-31-36-17 ~]$ sudo yum install python3 -y
sudo yum install python3-pip -y
Last metadata expiration check: 0:28:16 ago on Wed Oct 23 03:26:50 2024.
Package python3-3.9.16-1.amzn2023.0.9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 0:28:16 ago on Wed Oct 23 03:26:50 2024.
Dependencies resolved.
=====
Package                Arch      Version                               Repository      Size
```

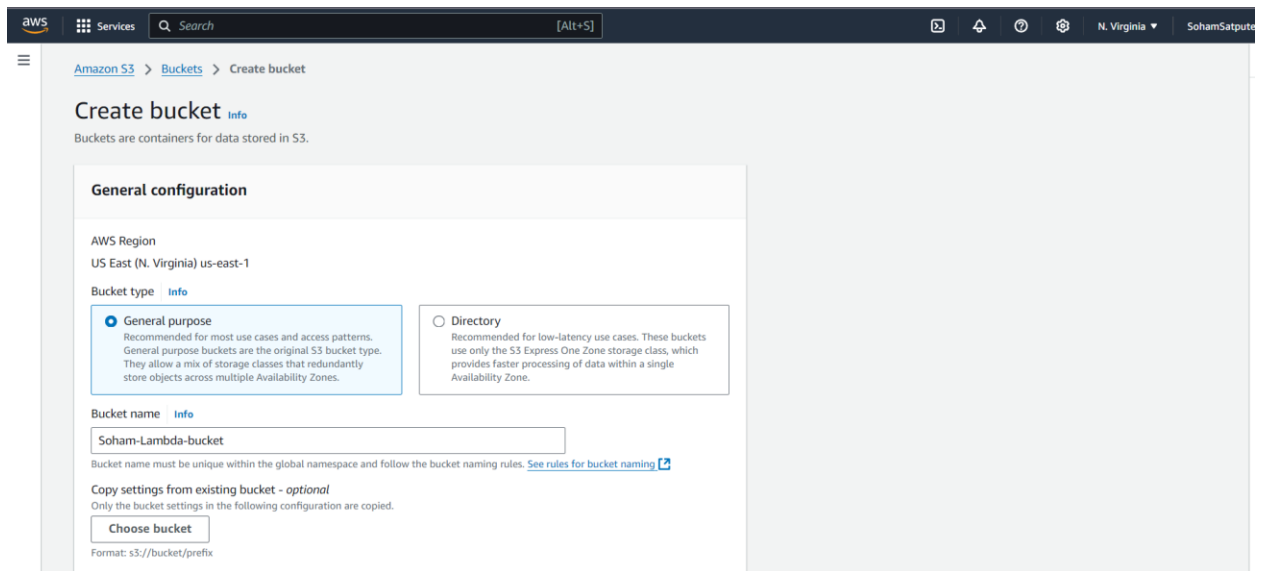
4. Create and S3 Bucket

4.1 In the AWS Management Console, go to **S3**.



4.2 Click **Create bucket**:

- **Bucket Name:** Give a unique name (e.g., `lambda-s3-trigger-bucket`).
- **Region:** Keep the same as your AWS Configuration (e.g., `us-east-1`).
- Keep other settings default.



5. Create the Lambda Function code.

5.1 **On your EC2 instance**, create the Python Lambda function code:

```
nano lambda_function.py
```

```
[ec2-user@ip-172-31-33-47 ~]$ nano lambda_function.py
```

5.2 **Write the following Lambda function** to read the uploaded file from S3:

```
import json
import boto3

s3 = boto3.client('s3')

def lambda_handler(event, context):
    # Get the bucket name and the uploaded file's key
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    file_key = event['Records'][0]['s3']['object']['key']

    # Fetch the file from S3
    file_obj = s3.get_object(Bucket=bucket_name, Key=file_key)
    file_content = file_obj['Body'].read().decode('utf-8')

    # Log the content of the file
    print(f"File Content from {file_key}:")
    print(file_content)

    return {
        'statusCode': 200,
        'body': json.dumps('File processed successfully')
    }
```

5.3 Press **Ctrl+X**, then **Y**, and hit **Enter**.

```
GNU nano 5.8                                lambda_function.py                                Modified
import json
import boto3

s3 = boto3.client('s3')

def lambda_handler(event, context):
    # Get the bucket name and the uploaded file's key
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    file_key = event['Records'][0]['s3']['object']['key']

    # Fetch the file from S3
    file_obj = s3.get_object(Bucket=bucket_name, Key=file_key)
    file_content = file_obj['Body'].read().decode('utf-8')

    # Log the content of the file
    print(f"File Content from {file_key}:")
    print(file_content)

    return {
        'statusCode': 200,
        'body': json.dumps('File processed successfully')
    }
```

File Name to Write: lambda_function.py

^G Help	M-D DOS Format	M-A Append	M-B Backup File
^C Cancel	M-M Mac Format	M-P Prepend	^T Browse

6. Deploy the Lambda function from EC2

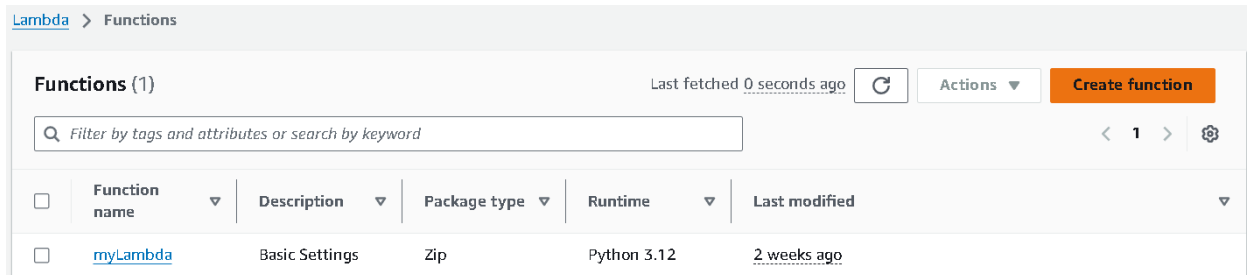
6.1 Package the Lambda function:

zip function.zip lambda_function.py

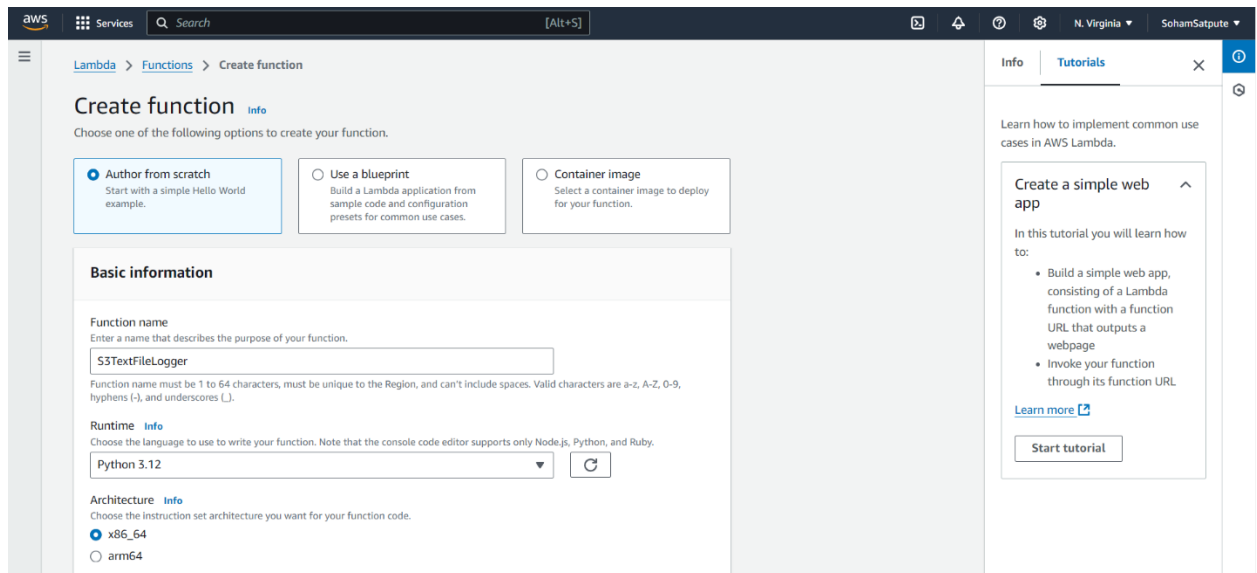
```
[ec2-user@ip-172-31-36-17 ~]$ zip function.zip lambda_function.py
  adding: lambda_function.py (deflated 42%)
[ec2-user@ip-172-31-36-17 ~]$ |
```

6.2 Create a Lambda function in AWS Console:

- Go to **Lambda > Create Function**.



- Choose **Author from Scratch**:
 - **Function Name:** `S3TextFileLogger`
 - **Runtime:** Python 3.12
 - **Execution Role:** Select "Create a new role with basic Lambda permissions."



▼ Change default execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☒ Create a new role with basic Lambda permissions

☐ Use an existing role

☐ Create a new role from AWS policy templates

❗

Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named `S3TextFileLogger-role-6r6slqzr`, with permission to upload logs to Amazon CloudWatch Logs.

► Additional Configurations

Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

Cancel

Create function

- Click **Create Function**.

Lambda > Functions > S3TextFileLogger

S3TextFileLogger

Throttle Copy ARN Actions

▼ Function overview Info

Export to Application Composer Download

Diagram Template

S3TextFileLogger

Layers (0)

+ Add trigger

+ Add destination

Description

-

Last modified

1 second ago

Function ARN

arn:aws:lambda:us-east-1:529088256210:fu
nction:S3TextFileLogger

Function URL

Info

Info Tutorials

Learn how to implement common use cases in AWS Lambda.

Create a simple web app

In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

[Learn more](#)

Start tutorial

●

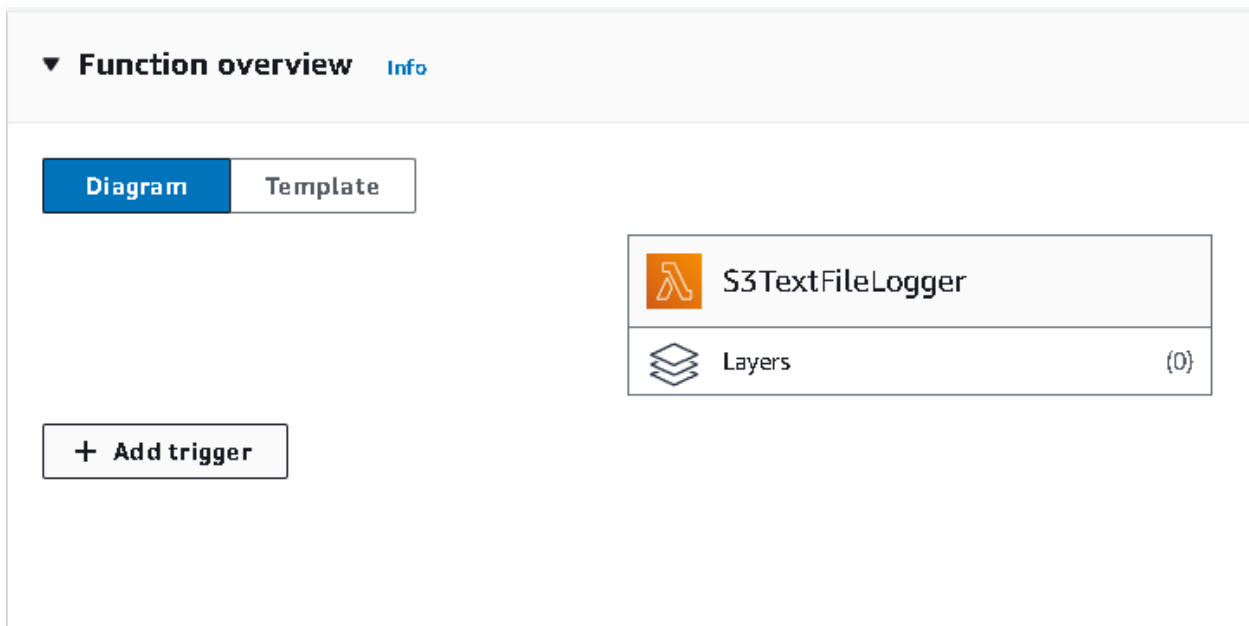
6.3 Upload the function code from EC2 using the AWS CLI:

```
aws lambda update-function-code --function-name S3TextFileLogger --zip-file  
fileb://function.zip
```

```
[ec2-user@ip-172-31-36-17 ~]$ aws lambda update-function-code --function-name S3TextFileLogger --zip-file fileb:///home/  
ec2-user/S3TextFileLogger-df44f7a0-c07a-41df-bbea-f8c16a30b37b.zip --region ap-south-1  
{  
  "FunctionName": "S3TextFileLogger",  
  "FunctionArn": "arn:aws:lambda:ap-south-1:529088256210:function:S3TextFileLogger",  
  "Runtime": "python3.12",  
  "Role": "arn:aws:iam::529088256210:role/service-role/S3TextFileLogger-role-k0hiwlv",  
  "Handler": "lambda_function.lambda_handler",  
  "CodeSize": 299,  
  "Description": "",  
  "Timeout": 3,  
  "MemorySize": 128,  
  "LastModified": "2024-10-23T10:29:31.000+0000",  
  "CodeSha256": "HAPq9EReJVEC5gLavtc/gyd5vZtd9eiUGF932t0jBxY=",  
  "Version": "$LATEST",  
  "TracingConfig": {  
    "Mode": "PassThrough"  
  },  
  "RevisionId": "3b94b729-ec0f-4c10-b609-9c007deceeba",  
  "State": "Active",  
  "LastUpdateStatus": "InProgress",  
  "LastUpdateStatusReason": "The function is being created.",  
  "LastUpdateStatusReasonCode": "Creating",  
  "PackageType": "Zip",  
  "Architectures": [  
    "x86_64"  
  ],  
  "EphemeralStorage": {  
    "Size": 512  
  },  
}
```

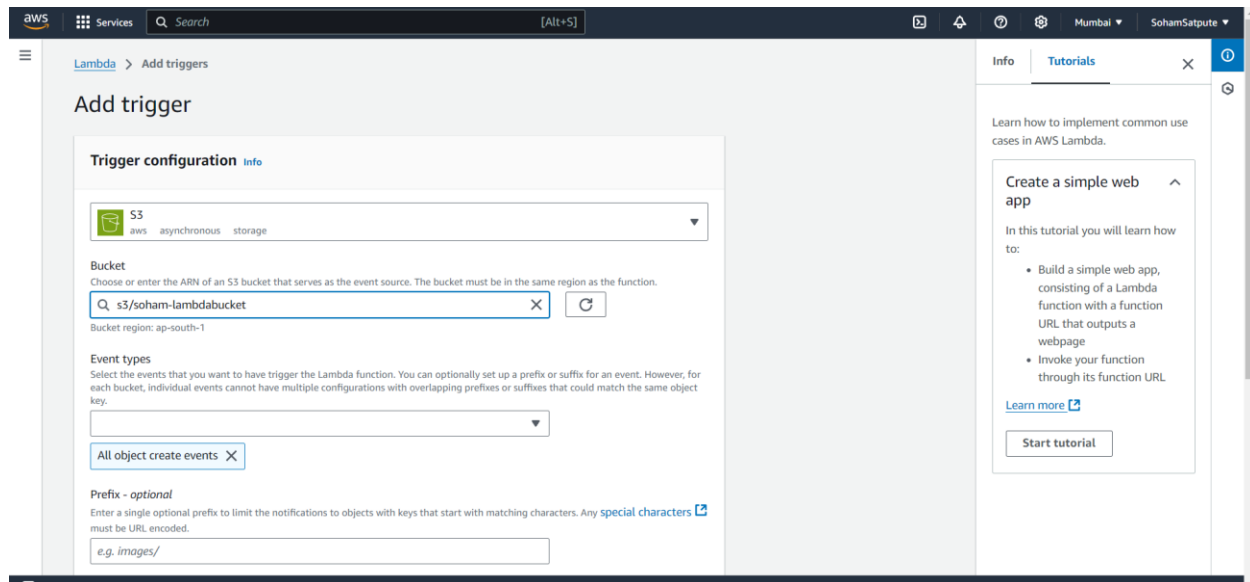
7. Configure S3 as the Trigger

7.1 In **Lambda console**, go to the **Function Overview** section and click **Add Trigger**.

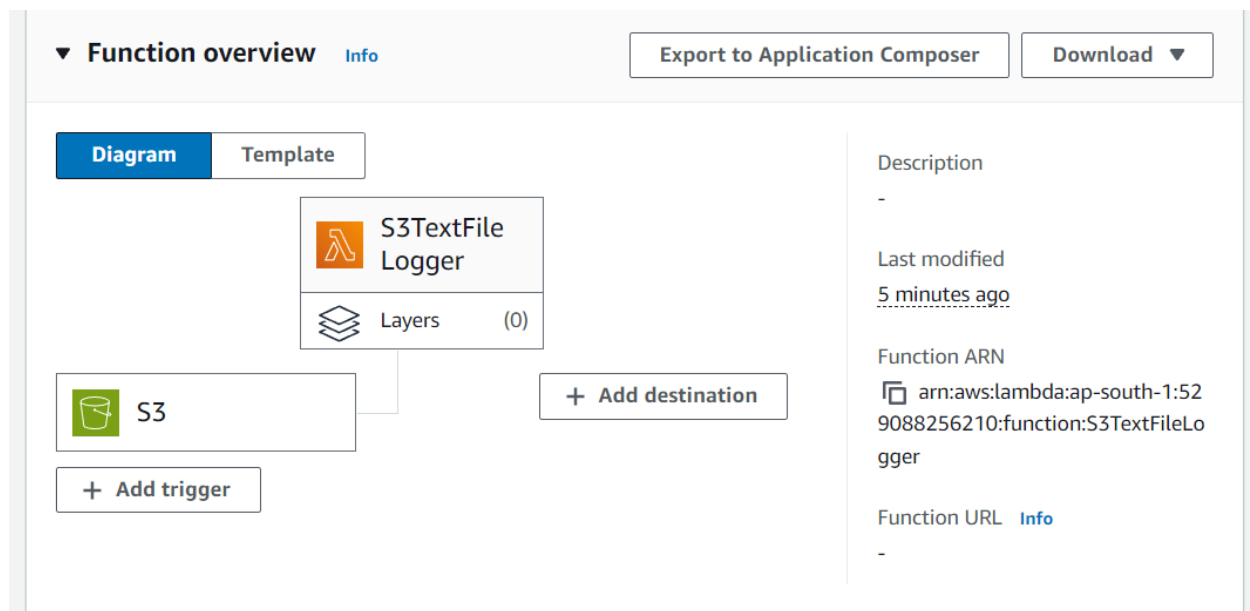


7.2 Choose **S3** as the trigger:

- Select your bucket (**lambda-s3-trigger-bucket**).
- **Event type**: Choose **All object create events**.



7.3 Click **Add** to enable the trigger.



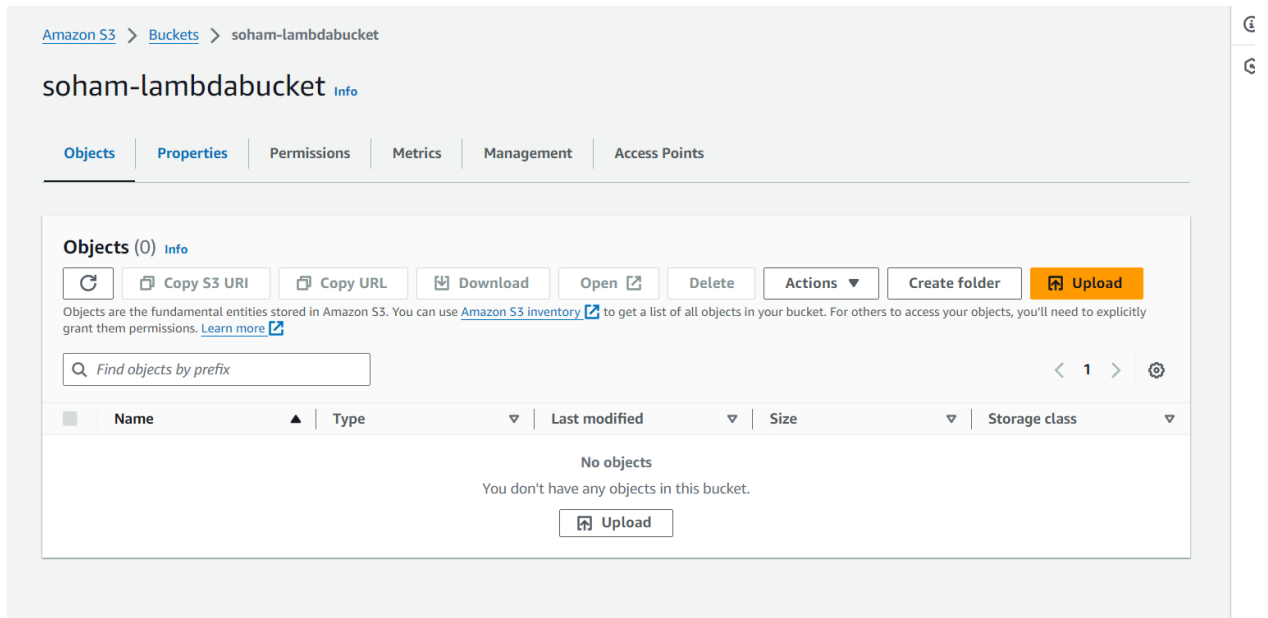
8. Upload a File and Test

8.1 Create a text file in your local host with some content.

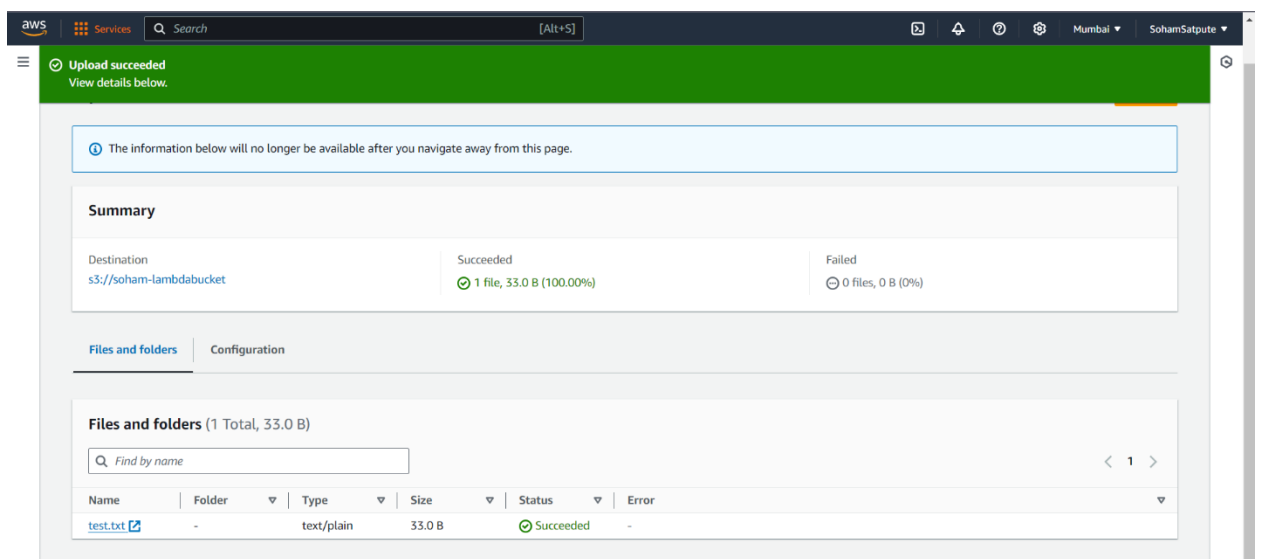
```
C:\Users\Soham Satpute>echo I am Soham Satpute of D15A, 52 > C:\soham22\test.txt
```

8.2 Upload a text file to your S3 bucket:

- Go to **S3** > your bucket > **Upload**.



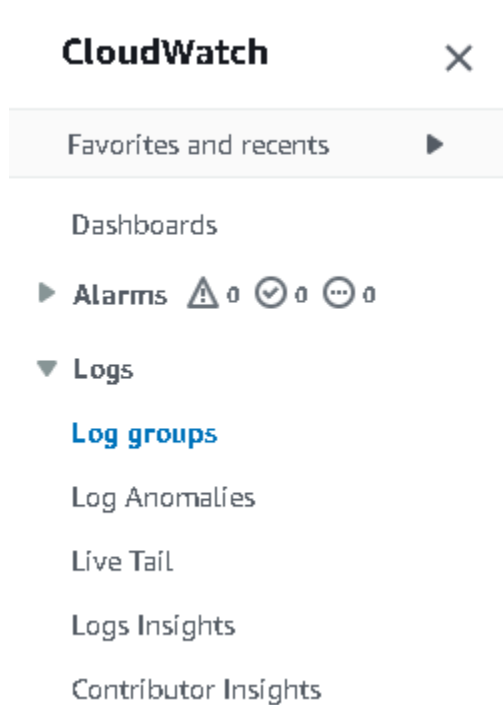
- Upload a **.txt** file with some content (e.g., **hello.txt**)



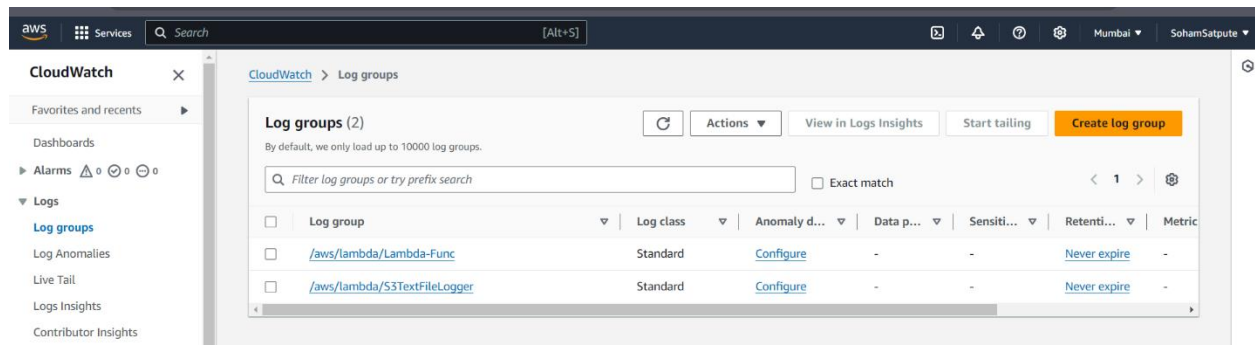
The Lambda function will automatically run when the file is uploaded.

9. Check Logs in CloudWatch

9.1 In the AWS Console, go to **CloudWatch > Logs**.



9.2 Under **Log Groups**, find the log group for your Lambda function (`/aws/lambda/S3TextFileLogger`).



9.3 Open the latest log stream to see the file content logged by the Lambda function.

Log streams (13)

🔄

Delete

Create log stream

Search all log streams

🔍 Filter log streams or try prefix search

☐ Exact match

☐ Show expired

[Info](#)

< 1 > ⚙️

<input type="checkbox"/>	Log stream	▼	Last event time	▼
<input type="checkbox"/>	2024/10/21/[\$LATEST]03e56edd73ec41229af683932332fb0b		2024-10-21 23:53:59 (UTC)	
<input type="checkbox"/>	2024/10/21/[\$LATEST]65c9c7b6bc494833837a129d24e6c85d		2024-10-21 23:41:00 (UTC)	
<input type="checkbox"/>	2024/10/21/[\$LATEST]3c5a58e3a6954fa7b23c4231e3326df2		2024-10-21 23:30:02 (UTC)	
<input type="checkbox"/>	2024/10/21/[\$LATEST]1153b2d0201f4521aea9ea93e27d2f5b		2024-10-21 23:18:02 (UTC)	
<input type="checkbox"/>	2024/10/21/[\$LATEST]f38a7dc710a84cc18f979d56a10eff48		2024-10-21 23:02:03 (UTC)	
<input type="checkbox"/>	2024/10/21/[\$LATEST]56dc2b563a8e439fb4e2b8147de60cfb		2024-10-21 22:46:05 (UTC)	
<input type="checkbox"/>	2024/10/14/[\$LATEST]107c561d41654c50ac89c93d41387448		2024-10-14 23:53:54 (UTC)	

You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

Clear1m30m1h12hCustomUTC timezoneDisplay

▶	Timestamp	Message
		No older events at this moment. Retry
▶	2024-10-23T13:10:37.342Z	INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:ap-south-1::runtime:188d9ca2e2714ff5637bd2bb...
▶	2024-10-23T13:10:37.845Z	START RequestId: 8dae30f7-fd41-4239-bdbf-5d70fbe9c35f Version: \$LATEST
▶	2024-10-23T13:10:37.845Z	[INFO] 2024-10-23T13:10:37.845Z 8dae30f7-fd41-4239-bdbf-5d70fbe9c35f Received event: {"Records": [{"eventVersion": "2.1", "e...
▼	2024-10-23T13:10:37.845Z	[INFO] 2024-10-23T13:10:37.845Z 8dae30f7-fd41-4239-bdbf-5d70fbe9c35f Bucket: soham-lambucket, File Key: test1.txt
	[INFO] 2024-10-23T13:10:37.845Z	8dae30f7-fd41-4239-bdbf-5d70fbe9c35f Bucket: soham-lambucket, File Key: test1.txt
▼	2024-10-23T13:10:38.408Z	[INFO] 2024-10-23T13:10:38.408Z 8dae30f7-fd41-4239-bdbf-5d70fbe9c35f File Content from test1.txt: I am SOham from D15A 52
	[INFO] 2024-10-23T13:10:38.408Z	8dae30f7-fd41-4239-bdbf-5d70fbe9c35f File Content from test1.txt: I am SOham from D15A 52
▶	2024-10-23T13:10:38.429Z	END RequestId: 8dae30f7-fd41-4239-bdbf-5d70fbe9c35f
▶	2024-10-23T13:10:38.429Z	REPORT RequestId: 8dae30f7-fd41-4239-bdbf-5d70fbe9c35f Duration: 583.59 ms Billed Duration: 584 ms Memory Size: 128 MB Max M...
		No newer events at this moment. Auto retry paused. Resume

Back to top