

# Serverless File Processing & Monitoring System using AWS

## Abstract

This project demonstrates a simple yet powerful serverless architecture using Amazon Web Services (AWS). The system automatically detects file uploads to an Amazon S3 bucket, processes events using AWS Lambda, logs details to Amazon CloudWatch, and sends real-time notifications through Amazon SNS. The project showcases essential AWS concepts, including serverless computing, event-driven automation, cloud monitoring, and secure resource management.

## 1. Introduction

AWS provides a scalable and cost-effective cloud platform for deploying applications without managing servers. Serverless architecture allows functions to run only when triggered, reducing operational overhead. In this project, a file monitoring system is built using AWS S3, Lambda, CloudWatch, and SNS. Whenever a user uploads a file, the system automatically triggers a workflow: processing, logging, and alerting.

This project aligns with AWS Cloud Practitioner concepts and demonstrates hands-on cloud implementation using AWS core services.

## 2. Aim

To build a fully serverless file processing and monitoring system using AWS services that automatically responds to S3 file uploads and provides real-time visibility through monitoring and notifications.

## 3. Objectives

1. To implement a serverless event-driven workflow using AWS Lambda.
2. To store and manage files using Amazon S3.
3. To enable real-time logs and monitoring using Amazon CloudWatch.
4. To configure Amazon SNS for alert notifications.
5. To demonstrate basic cloud security using IAM roles and permissions.

#### 4. AWS Services Used

- **Amazon S3** – File storage and event trigger
- **AWS Lambda** – Event-driven compute function
- **Amazon SNS** – Email notifications to the user
- **Amazon CloudWatch** – Logs, monitoring, and dashboards
- **AWS IAM** – Secure access and execution roles

#### 5. System Architecture

##### Workflow:

1. User uploads a file to Amazon S3
2. S3 sends an event to AWS Lambda
3. Lambda processes the event and logs details
4. Logs appear in CloudWatch
5. Lambda sends notification through Amazon SNS
6. User receives an email alert

#### 6. Implementation Steps

##### Step 1 — S3 BUCKET SETUP (Storage Lead)

##### Steps to Perform

##### 1. Create S3 Bucket

1. Go to **AWS Console** → **S3**
2. Click **Create bucket**
3. Enter bucket name → `serverless-file-bucket-yourname`
4. Region → **us-east-1**
5. Keep **Block all public access** = **ON**
6. Click **Create bucket**

##### 2. Create Folder Structure

Inside the bucket:

- Create folder: uploads/
- Create folder: processed/

### 3. Upload Test Files

1. Go to folder uploads/
2. Click **Upload → Add files**
3. Upload test1.txt, sample2.txt

### 4. Verify Bucket

- Check that files show up
- Confirm bucket name for other members

## Step 2 — IAM SETUP (Security Lead)

### Steps to Perform

#### 1. Create IAM Role for Lambda

1. Go to **IAM → Roles → Create role**
2. Select **AWS Service → Lambda**
3. Attach policies:
  - AmazonS3FullAccess
  - CloudWatchLogsFullAccess
  - AmazonSNSFullAccess
4. Name the role:  
**LambdaS3ExecutionRole**

#### 2. Create IAM User (optional for demo)

- Username → project-user
- Permission → S3FullAccess
- Enable **MFA**
- Download credentials

#### 3. Security Verification

- Ensure no public access to bucket

- Ensure Lambda uses the created role

### Step 3 — LAMBDA FUNCTION (Automation Lead)

#### Steps to Perform

##### 1. Create Lambda Function

1. Go to **Lambda**
2. Click **Create function**
3. Choose **Author from scratch**
4. Name → FileEventProcessor
5. Runtime → **Python 3.9**
6. Execution role → **Use existing role → LambdaS3ExecutionRole**

##### 2. Add S3 Trigger

1. Go to Lambda → **Triggers**
2. Add Trigger → **S3**
3. Select bucket: serverless-file-bucket-yourname
4. Event type: **All object create events**
5. Prefix: uploads/

##### 3. Add Code

###### Paste this code:

```
import json
```

```
import boto3
```

```
import urllib.parse
```

```
sns = boto3.client('sns')
```

```
TOPIC_ARN = "YOUR_SNS_TOPIC_ARN"
```

```
def lambda_handler(event, context):
```

```
    for rec in event['Records']:
```

```
        bucket = rec['s3']['bucket']['name']
```

```
        key = urllib.parse.unquote_plus(rec['s3']['object']['key'])
```

```

event_name = rec['eventName']

if event_name.startswith("ObjectCreated"):
    message = f"File uploaded: {key} in {bucket}"
elif event_name.startswith("ObjectRemoved"):
    message = f"File deleted: {key} from {bucket}"
else:
    message = f"Other event: {event_name}"

sns.publish(
    TopicArn=TOPIC_ARN,
    Message=message,
    Subject="S3 Notification"
)

return {"statusCode": 200}

```

Click **Deploy**.

#### 4. Test Lambda

- Upload a new file to S3 → uploads/test3.txt
- Check logs in CloudWatch

### Step 4 — CLOUDWATCH MONITORING (Observability Lead)

#### Steps to Perform

##### 1. View Lambda Logs

1. Go to **CloudWatch** → **Logs**
2. Click log group: /aws/lambda/FileEventProcessor
3. Open latest log stream
4. Verify log:  
**"New file uploaded: uploads/test3.txt"**

##### 2. Create CloudWatch Dashboard

1. Go to **Dashboards** → **Create Dashboard**

2. Name: ServerlessMonitoringDashboard

3. Add widgets:

- Lambda Invocations
- Lambda Errors
- S3 Metrics
- SNS Delivery Status

### 3. Create Alarm (Optional)

1. Go to **Alarms** → **Create alarm**
2. Select Errors metric for Lambda
3. Condition → Errors  $\geq$  1
4. Notification → SNS topic

## Step 5: SNS NOTIFICATIONS (Alerting Lead)

### Steps to Perform

#### 1. Create SNS Topic

1. Go to **SNS** → **Topics** → **Create topic**
2. Type → **Standard**
3. Name → FileUploadAlerts

#### 2. Add Email Subscription

1. SNS → Subscriptions → Create subscription
2. Protocol → Email
3. Enter team email IDs
4. Each member must click **Confirm Subscription** in email

#### 3. Connect SNS to Lambda

Replace in code:

```
TopicArn='YOUR_SNS_TOPIC_ARN',
```

With your actual SNS topic ARN.

#### 4. Test Email Alert

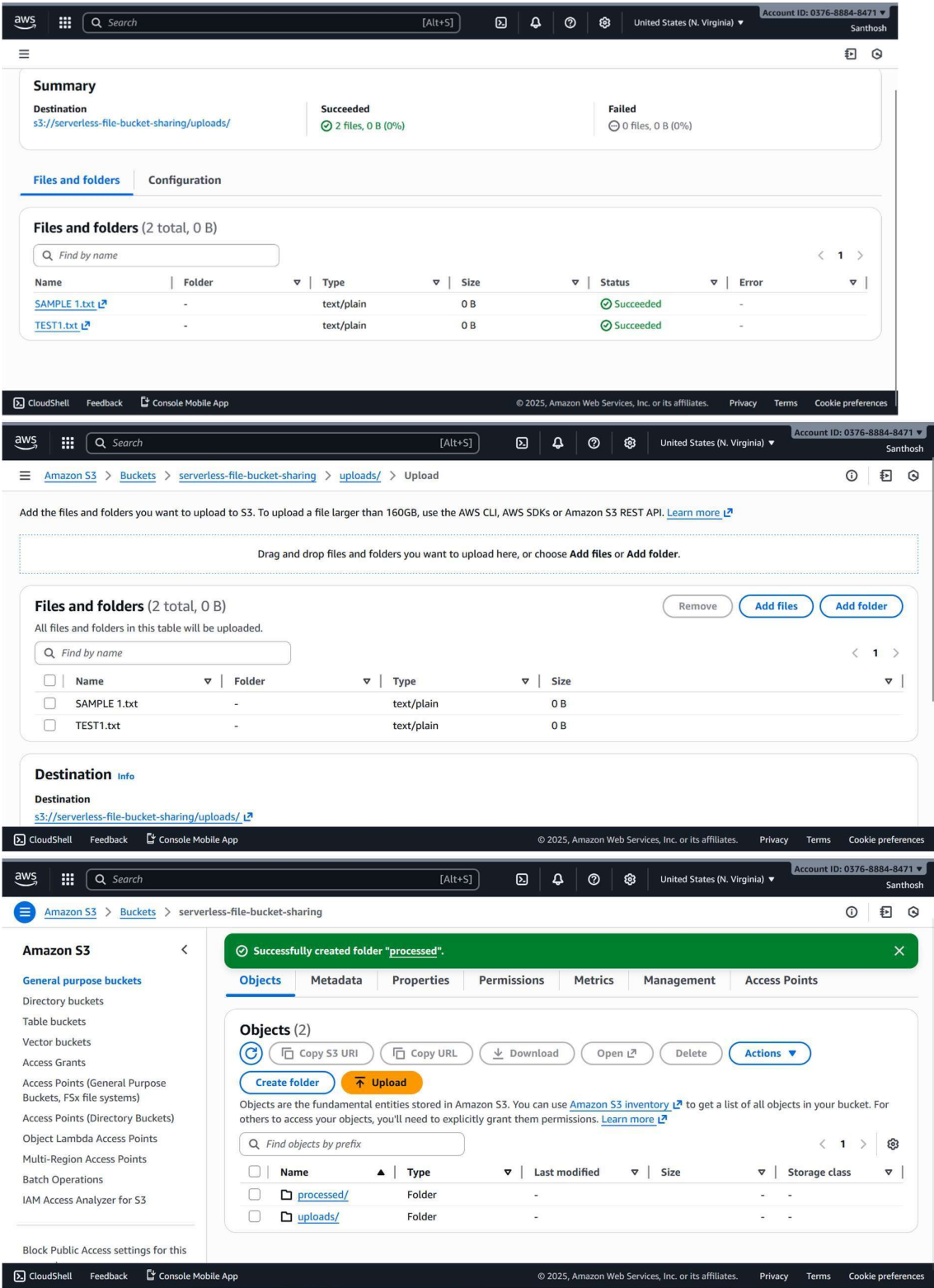
- Upload AlertTest.txt to S3
- All team members should receive email:  
**“New file uploaded: uploads/AlertTest.txt”**

## 7. OUTPUT

- S3 successfully stored uploaded files
- Lambda function triggered automatically
- CloudWatch generated real-time logs
- SNS sent email notifications instantly
- Full serverless workflow executed without manual intervention

# 8. Screenshot:

## Step 1 — S3 BUCKET SETUP (Storage Lead)





Amazon S3

General purpose buckets

Directory buckets

Table buckets

Vector buckets

Access Grants

Access Points (General Purpose Buckets, FSx file systems)

Access Points (Directory Buckets)

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

General purpose buckets

All AWS Regions

Directory buckets

General purpose buckets (1) Info

Buckets are containers for data stored in S3.

Find buckets by name

Name	AWS Region	Creation date
serverless-file-bucket-sharing	US East (N. Virginia) us-east-1	November 24, 2025, 15:07:24 (UTC+05:30)

Account snapshot Info

Updated daily

View dashboard

Storage Lens provides visibility into storage usage and activity trends.

External access summary - new Info

Updated daily

External access findings help you identify bucket permissions that allow public access or access from other AWS accounts.

Block Public Access settings for this

Amazon S3 > Buckets > Create bucket

Secure your objects with two separate layers of encryption. For details on pricing, see [DSSE-KMS pricing on the Storage tab of the Amazon S3 pricing page.](#)

☒ Server-side encryption with Amazon S3 managed keys (SSE-S3)

☐ Server-side encryption with AWS Key Management Service keys (SSE-KMS)

☐ Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)

Bucket Key

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

☐ Disable

☒ Enable

Advanced settings

After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

Cancel

Create bucket

Amazon S3 > Buckets > Create bucket

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region

US East (N. Virginia) us-east-1

Bucket type Info

☒ General purpose

Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

☐ Directory

Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info

serverless-file-bucket-sharing

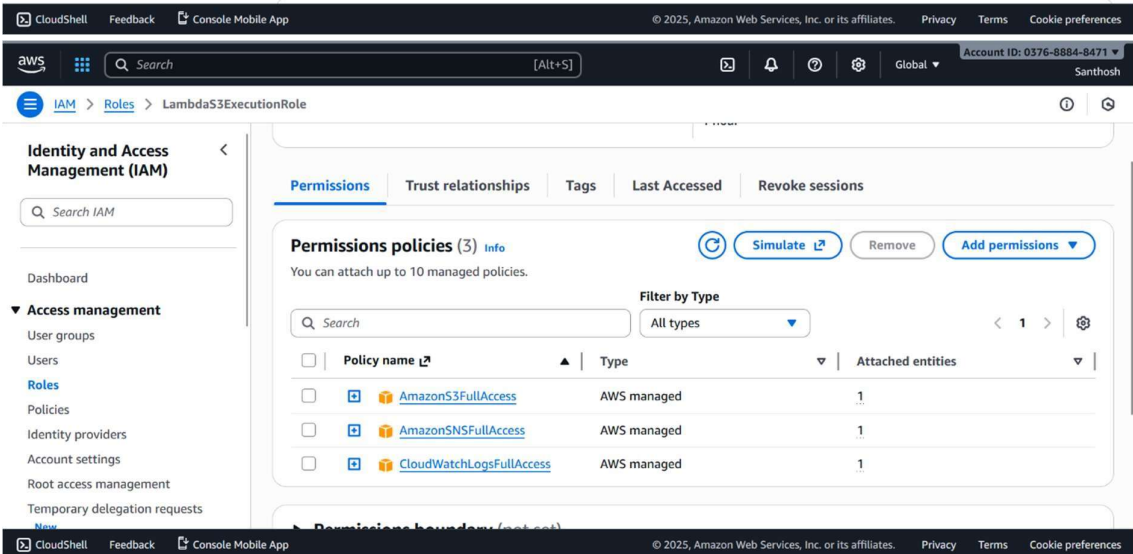
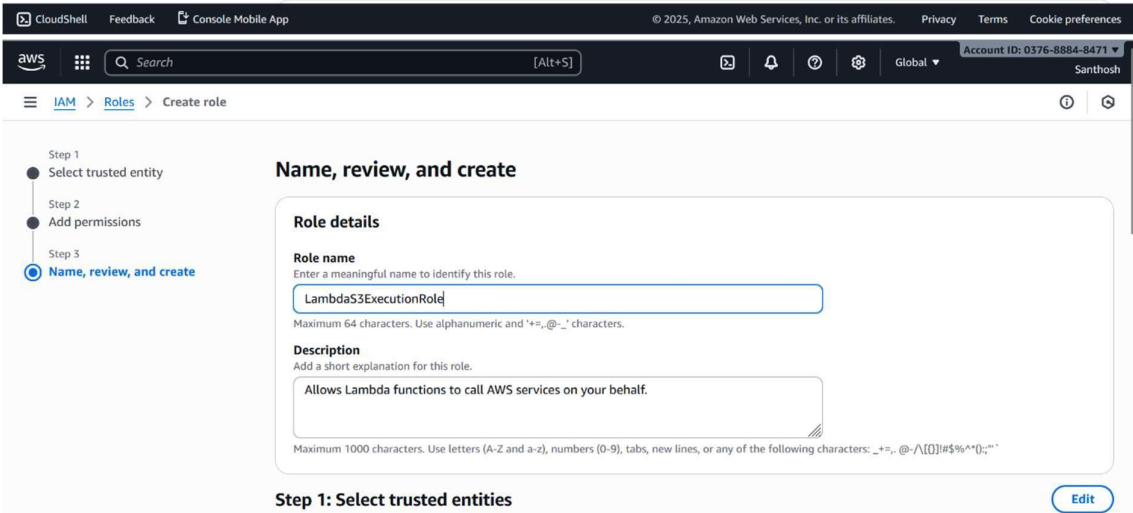
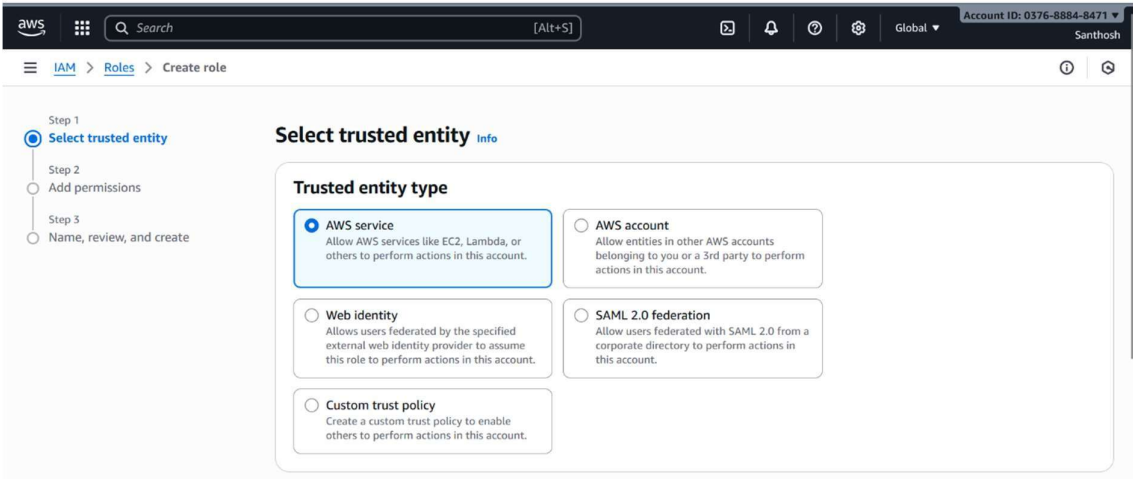
Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn more](#)

Copy settings from existing bucket - optional

Only the bucket settings in the following configuration are copied.

Choose bucket

Step 2 — IAM SETUP (Security Lead)



### Step 3 — LAMBDA FUNCTION (Automation Lead)

aws

Search

[Alt+S]

United States (N. Virginia)

Account ID: 0376-8884-8471

Santhosh

Lambda > Functions > Create function

▼ 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

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

LambdaS3ExecutionRole

[View the LambdaS3ExecutionRole role](#) on the IAM console.

► Additional configurations

Use additional configurations to set up networking, security, and governance for your function. These settings help secure and customize your Lambda function deployment.

Cancel

Create function

CloudShell

Feedback

Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

aws

Search

[Alt+S]

United States (N. Virginia)

Account ID: 0376-8884-8471

Santhosh

Lambda > Add triggers

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.

s3/serverless-file-bucket-sharing

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.

All object create events

Prefix - optional

Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any [special characters](#) must be URL encoded.

uploads/

Suffix - optional

Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any [special characters](#) must be URL encoded.

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](#) about the Lambda permissions model.

Cancel

Add

CloudShell

Feedback

Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

### Add trigger

aws

Search

[Alt+S]

United States (N. Virginia)

Account ID: 0376-8884-8471

Santhosh

Lambda > Add triggers

Prefix - optional

Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any [special characters](#) must be URL encoded.

uploads/

Suffix - optional

Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any [special characters](#) must be URL encoded.

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](#) about the Lambda permissions model.

Cancel

Add

CloudShell

Feedback

Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

aws [Search] [Alt+S] United States (N. Virginia) Account ID: 0376-8884-8471 Santhosh

Lambda > Functions > FileEventProcessor

Successfully updated the function FileEventProcessor.

### FileEventProcessor

Throttle Copy ARN Actions

Function overview Info

Diagram Template

FileEventProcessor

Layers (0)

S3

+ Add trigger

+ Add destination

Export to Infrastructure Composer Download

Description

Last modified 21 seconds ago

Function ARN arn:aws:lambda:us-east-1:037688848471:function:FileEventProcessor

Function URL Info

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws [Search] [Alt+S] United States (N. Virginia) Account ID: 0376-8884-8471 Santhosh

Lambda > Functions > FileEventProcessor

Successfully updated the function FileEventProcessor.

### FileEventProcessor

lambda\_function.py

```
1 import json
2
3 def lambda_handler(event, context):
4     bucket_name = event['Records'][0]['s3']['bucket']['name']
5     file_name = event['Records'][0]['s3']['object']['key']
6
7     print(f"New note uploaded: {file_name} in bucket: {bucket_name}")
8
9
10    return {
11        'statusCode': 200,
12        'body': json.dumps('Note upload logged successfully!')
13    }
```

Deploy (Ctrl+Shift+U) Test (Ctrl+Shift+I)

TEST EVENTS [NONE SELECTED]

us-east-1.console.aws.amazon.com/console/home?region=us-east-1 © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws [Search] [Alt+S] United States (N. Virginia) Account ID: 0376-8884-8471 Santhosh

CloudWatch > Log groups > /aws/lambda/FileEventProcessor > 2025/11/24/[LATEST]5a4369494a2241b9a2f1aa473aafacca

### CloudWatch

Log events

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](#)

Filter events - press enter to search 1m 1h UTC timezone

Display

Timestamp	Message
No older events at this moment. <a href="#">Retry</a>	
2025-11-24T09:55:34.308Z	INIT_START Runtime Version: python:3.9.v125 Runtime Version ARN: arn:aws:lambda:us-east-1:run...
2025-11-24T09:55:34.389Z	START RequestId: 5bbfba02-d75a-49d5-a223-3a043e9ce0ed Version: LATEST
2025-11-24T09:55:34.390Z	✓ New note uploaded: uploads/SAMPLE+2.txt in bucket: serverless-file-bucket-sharing
2025-11-24T09:55:34.391Z	END RequestId: 5bbfba02-d75a-49d5-a223-3a043e9ce0ed
2025-11-24T09:55:34.391Z	REPORT RequestId: 5bbfba02-d75a-49d5-a223-3a043e9ce0ed Duration: 1.34 ms Billed Duration: 79 ms...
No newer events at this moment. <a href="#">Auto retrying...</a> <a href="#">Pause</a>	

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 4 — CLOUDWATCH MONITORING (Observability Lead)

aws

Search

[Alt+S]

United States (N. Virginia)

Account ID: 0376-8884-8471

Santhosh

CloudWatch

Dashboards

Custom Dashboards

Automatic dashboards

Custom Dashboards (0)

Info

Filter dashboards

Name

Create dashboard

Create dashboard

Create new dashboard

Dashboard name

ServerlessMonitoringDashboard

Valid characters in dashboard names include "0-9A-Za-z-.\_".

Cancel

Create dashboard

Add metric graph

Errors

Persist time range

1h

3h

12h

1d

3d

1w

Custom

UTC timezone

Line

Add math

Add query

Browse (8)

Multi source query

Graphed metrics (1)

Options

Source

FunctionName o/o

Metric name

Errors

FileEventProcessor

AsyncEventAge

No alarms

FileEventProcessor

AsyncEventsDropped

No alarms

FileEventProcessor

AsyncEventsReceived

No alarms

FileEventProcessor

Errors

No alarms

FileEventProcessor

AsyncEventsReceived

No alarms

Cancel

Create widget

aws

Search

[Alt+S]

United States (N. Virginia)

Account ID: 0376-8884-8471

Santhosh

CloudWatch

Dashboards

ServerlessMonitoringDashboard

ServerlessMonitoringD...

1h

3h

12h

1d

3d

1w

Custom

UTC timezone

Autosave: Off

Actions

Save

Errors

Count

1

0.5

0

07:05

10:05

Errors

CloudShell

Feedback

Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

## Step 5: SNS NOTIFICATIONS (Alerting Lead)

aws

Search

[Alt+S]

United States (N. Virginia)

Account ID: 0376-8884-8471

Santhosh

Amazon SNS

Topics

Create topic

0

+

↺

### Create topic

Details

Type [Info](#)

Topic type cannot be modified after topic is created

☐ FIFO (first-in, first-out)

- Strictly-preserved message ordering
- Exactly-once message delivery
- Subscription protocols: SQS

☒ Standard

- Best-effort message ordering
- At-least once message delivery
- Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints

Name

FileUploadAlerts

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (\_).

Display name - optional [Info](#)

To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

My Topic

CloudShell

Feedback

Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

aws

Search

[Alt+S]

United States (N. Virginia)

Account ID: 0376-8884-8471

Santhosh

Amazon SNS

Subscriptions

Create subscription

0

+

↺

### Create subscription

Details

Topic ARN

arn:aws:sns:us-east-1:037688848471:FileUploadAlerts

Protocol

The type of endpoint to subscribe

Email

Endpoint

An email address that can receive notifications from Amazon SNS.

jupg24mca23173@jainuniversity.ac.in

After your subscription is created, you must confirm it. [Info](#)

Subscription filter policy - optional [Info](#)

CloudShell

Feedback

Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

aws

Search

[Alt+S]

United States (N. Virginia)

Account ID: 0376-8884-8471

Santhosh

Amazon SNS

Subscriptions

Create subscription

0

+

↺

### Create subscription

Details

Topic ARN

arn:aws:sns:us-east-1:037688848471:FileUploadAlerts

Protocol

The type of endpoint to subscribe

Email

Endpoint

An email address that can receive notifications from Amazon SNS.

santhosh.govindan0932@gmail.com

After your subscription is created, you must confirm it. [Info](#)

us-east-1.console.aws.amazon.com/console/home?region=us-east-1

© 2025, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences



**Amazon SNS**  
Dashboard  
Topics  
Subscriptions  
**Mobile**  
Push notifications  
Text messaging (SMS)

**Details**  

<b>Name</b> FileUploadAlerts	<b>ARN</b> arn:aws:sns:us-east-1:037688848471:FileUploadAlerts	<b>Display name</b> -	<b>Type</b> Standard
<b>Topic owner</b> 037688848471			

**Subscriptions** (2) Edit Delete Request confirmation Confirm subscription Create subscription  
Search  

ID	Endpoint	Status	Protocol
Deleted	santhosh.govindan0932...	Confirmed	EMAIL

Successfully updated the function FileEventProcessor.

EXPLORER  
FILEEVENTPROCESSOR  
lambda\_function.py  
DEPLOY ✓ Current  
Deploy (Ctrl+Shift+U)  
Test (Ctrl+Shift+I)  
TEST EVENTS [NONE SELECTED]  
Create new test event

lambda\_function.py  
6 def lambda\_handler(event, context):  
7 bucket = event['Records'][0]['s3']['bucket']['name']  
8 file\_name = event['Records'][0]['s3']['object']['key']  
9  
10 message = f"New file uploaded: {file\_name} in bucket: {bucket}"  
11 print(message)  
12  
13 sns.publish(  
14 TopicArn='arn:aws:sns:us-east-1:037688848471:FileUploadAlerts',  
15 Message=message,  
16 Subject='New File Upload Alert'  
17 )  
18  
19 return {  
20 'statusCode': 200,  
21 'body': json.dumps('File processed successfully!')  
22 }  
23

Successfully updated the function FileEventProcessor.

## Final verification:

The screenshot displays two web interfaces. The top interface is the AWS Management Console, showing a green notification banner for a successful upload. Below the banner, the 'Files and folders' section shows a table with one file: 'New Text Document.txt', which is 0 B in size and has a status of 'Succeeded'. The bottom interface is a Gmail inbox, showing an email from 'AWS Notifications' with the subject 'New File Upload Alert'. The email body states: 'New file uploaded: uploads/New+Text+Document.txt in bucket: serverless-file-bucket-sharing'. It also includes an unsubscribe link and a support link. On the right side of the Gmail interface, the Gemini chat sidebar is visible, showing options to 'Summarize this email', 'What can Gemini do in Gmail', and 'List action items from this email'.

**AWS Console Upload Succeeded**

Destination: [s3://serverless-file-bucket-sharing/uploads/](#)

Succeeded: 1 file, 0 B (0%)

Failed: 0 files, 0 B (0%)

**Files and folders** (1 total, 0 B)

Name	Folder	Type	Size	Status	Error
<a href="#">New Text Document.txt</a>	-	text/plain	0 B	Succeeded	-

**Gmail: New File Upload Alert**

**AWS Notifications** to me 3:53 PM (0 minutes ago)

New file uploaded: uploads/New+Text+Document.txt in bucket: serverless-file-bucket-sharing

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe: <https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:037688848471:FileUploadAlerts:48fb960b-14d2-4814-be8f-c496fe3e720e&Endpoint=santhosh.govindan0932@gmail.com>

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>

**Gemini Sidebar:**

- Summarize this email in more detail
- What can Gemini do in Gmail
- List action items from this email

Enter a prompt here

Gemini in Workspace can make mistakes, so double-check responses. [Learn more](#)

## 9. Conclusion

This project demonstrates a complete serverless event-driven workflow using AWS cloud services. With S3 for storage, Lambda for processing, CloudWatch for monitoring, and SNS for notifications, the system showcases a practical real-world cloud solution. The implementation proves how AWS serverless computing can simplify development and automate workflows with minimal setup and zero server management.



