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

sns = boto3.client('sns')

def lambda_handler(event, context):
    bucket = event['Records'][0]['s3']['bucket']['name']
    file_name = event['Records'][0]['s3']['object']['key']
```

```
    message = f"New file uploaded: {file_name} in bucket: {bucket}"
    print(message)

    sns.publish(
        TopicArn='YOUR_SNS_TOPIC_ARN',
        Message=message,
        Subject='New File Upload Alert'
    )

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

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 ≥ 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

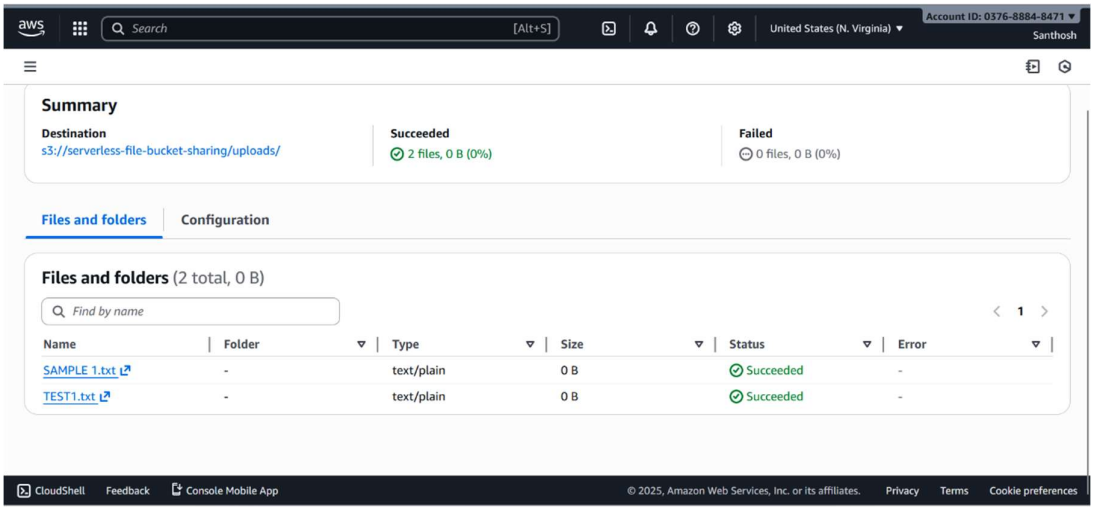
- Upload AlertTest.txt to S3
- All team members should receive email:
  “New file uploaded: uploads/AlertTest.txt”

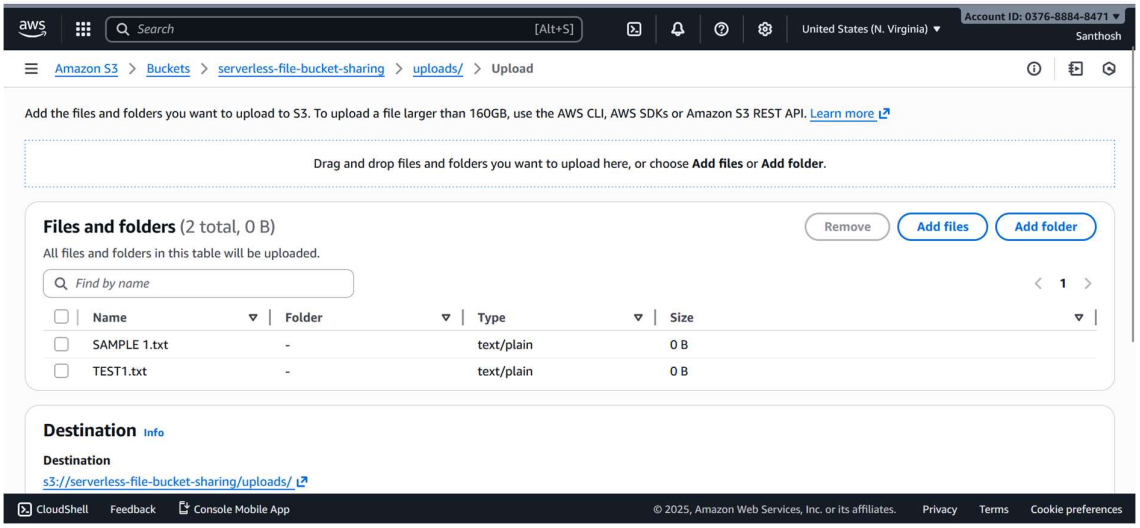
7. OUTPUT

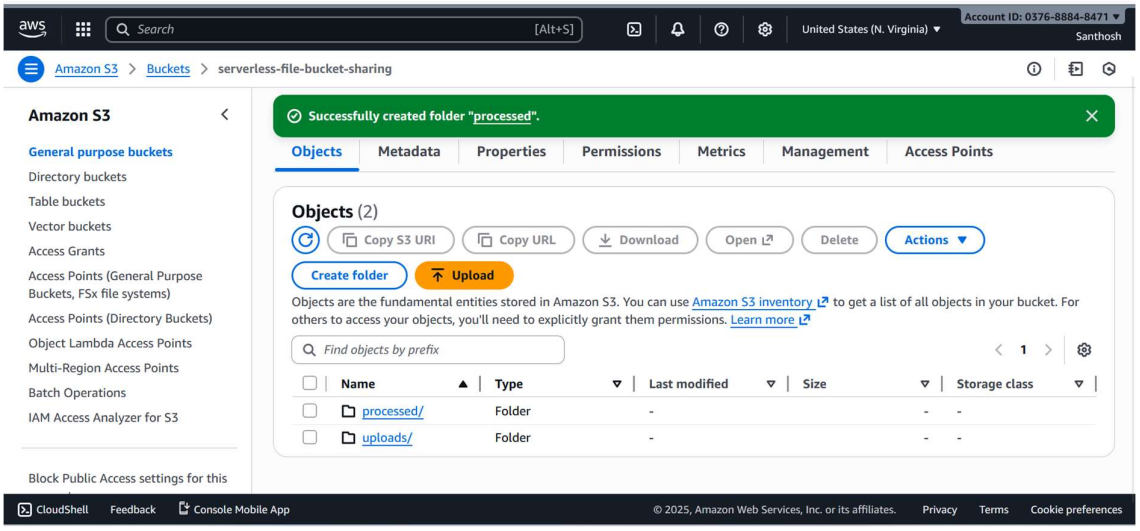
- S3 successfully stored uploaded files
- Lambda function triggered automatically
- CloudWatch generated real-time logs
- SNS sent email notifications instantly
- Full serverless workflow executed without manual intervention

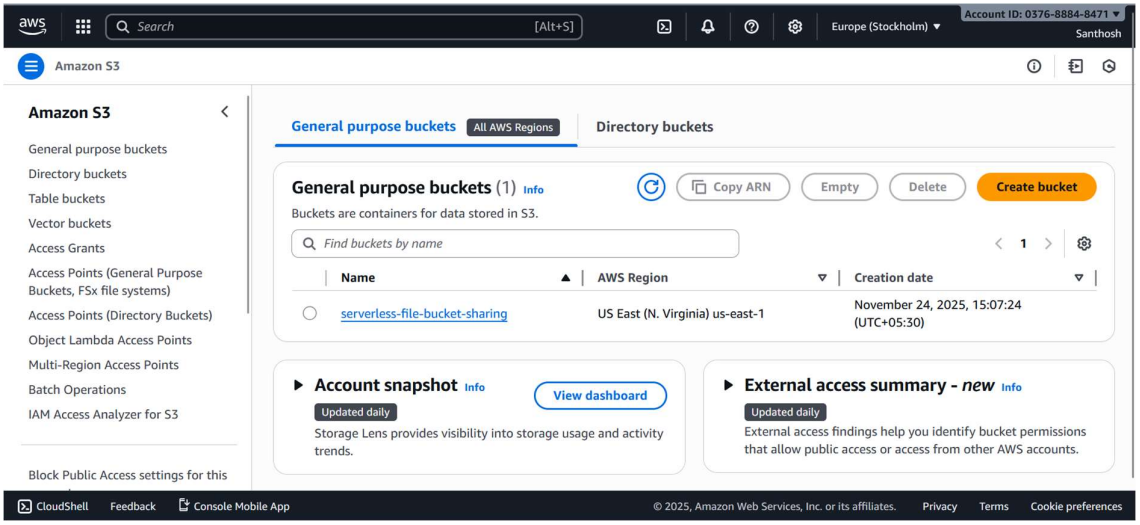
8. Screenshot:

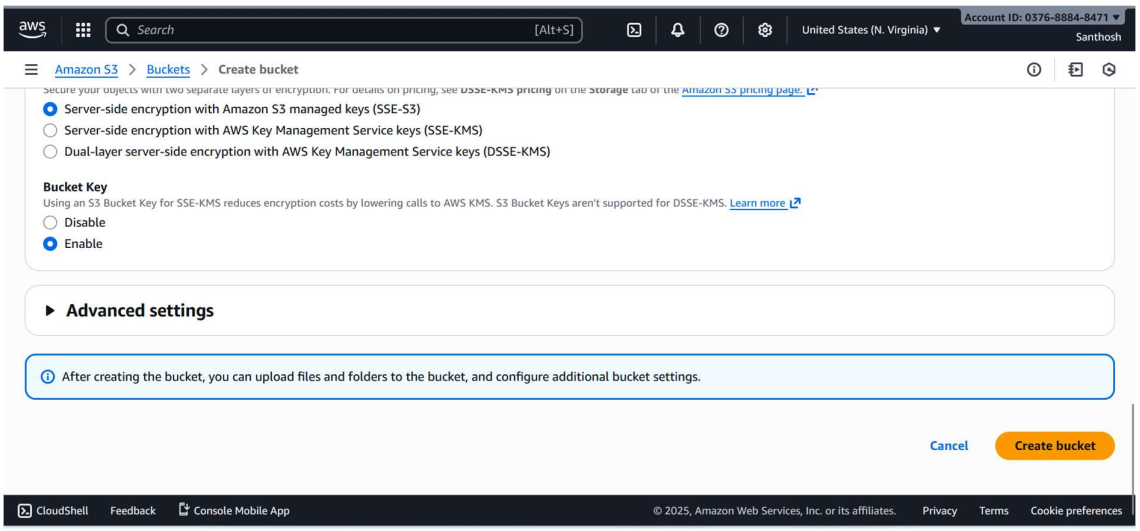
Step 1 — S3 BUCKET SETUP (Storage Lead)

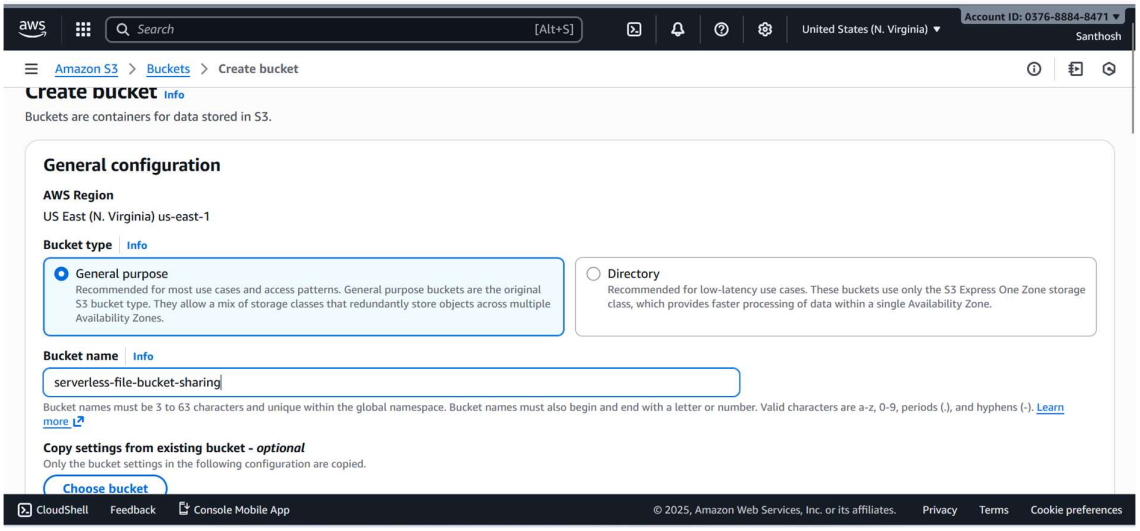




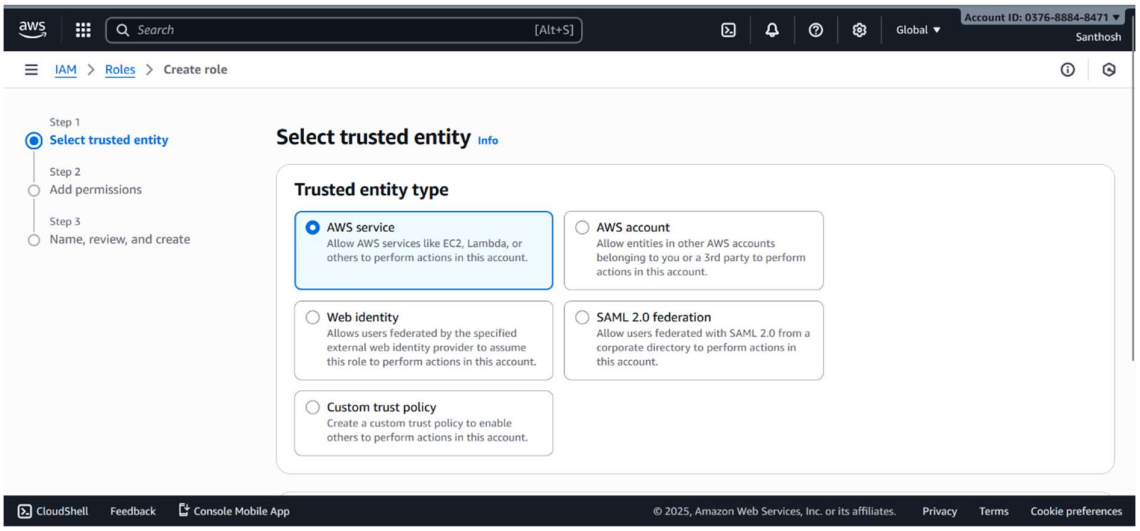


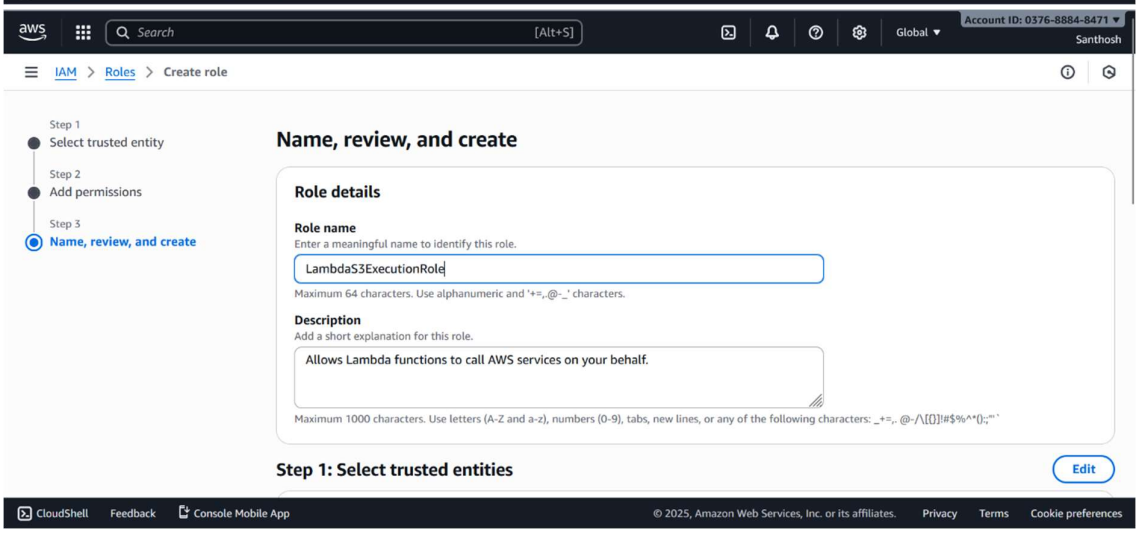

Search [Alt+S]
Europe (Stockholm)
Account ID: 0376-8884-8471
Santhosh
Amazon S3
Amazon S3
General purpose buckets
Directory buckets
Table buckets
Vector buckets
Access Grants
Access Points (General Purpose Buckets, FSx file systems)
Access Points (Directory Buckets)
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this
General purpose buckets
All AWS Regions
Directory buckets
General purpose buckets (1) Info
Buckets are containers for data stored in S3.
Copy ARN
Empty
Delete
Create bucket
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
View dashboard
Updated daily
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.
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

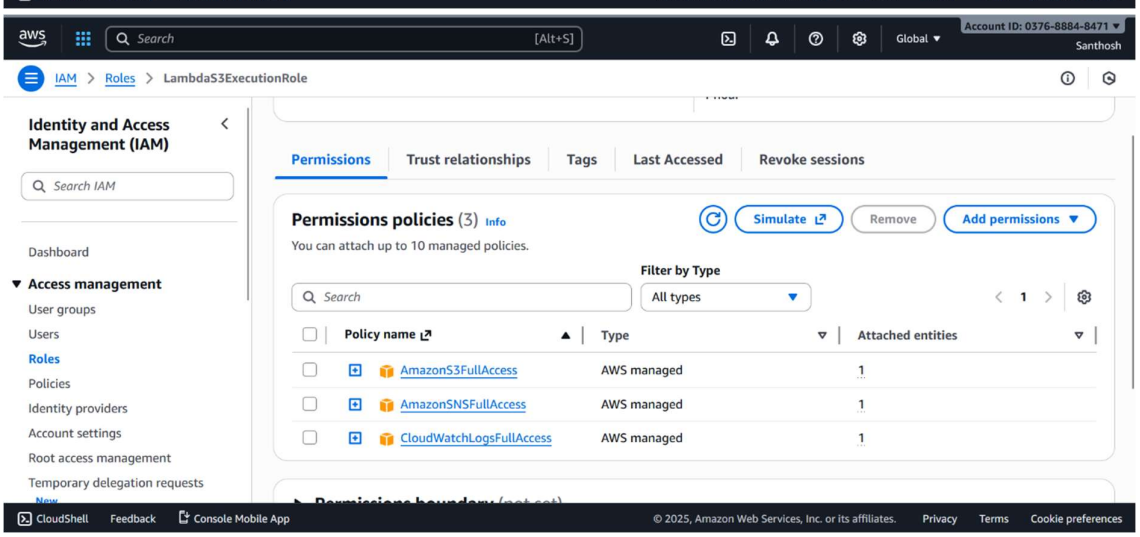

Search [Alt+S]
United States (N. Virginia)
Account ID: 0376-8884-8471
Santhosh
Amazon S3 > Buckets > Create bucket
Server-side encryption with Amazon S3 managed keys (SSE-S3)
Server-side encryption with AWS Key Management Service keys (SSE-KMS)
Dual-layer server-side encryption with AWS Key Management 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
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Search [Alt+S]
United States (N. Virginia)
Account ID: 0376-8884-8471
Santhosh
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
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

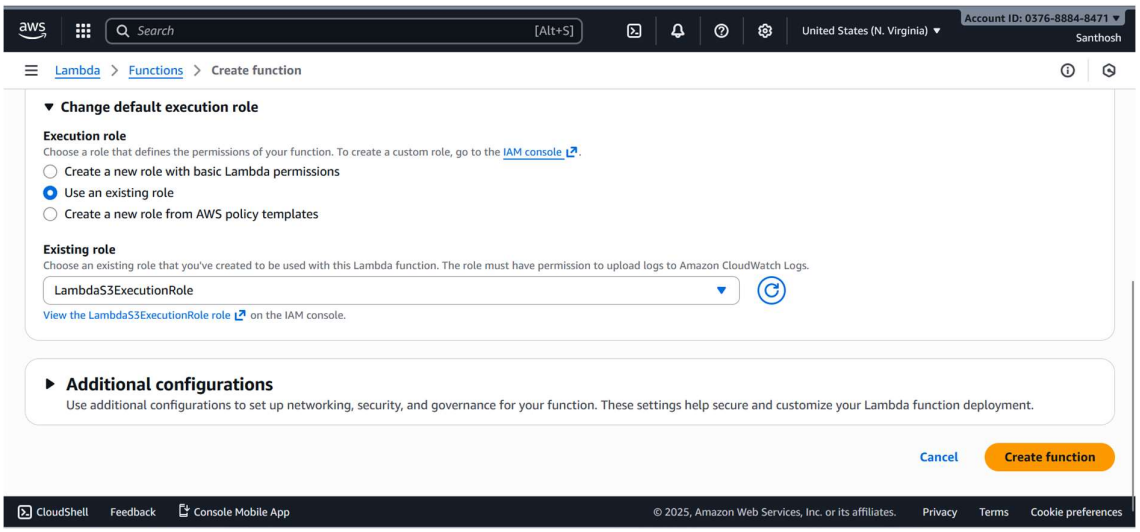
Step 2 — IAM SETUP (Security Lead)

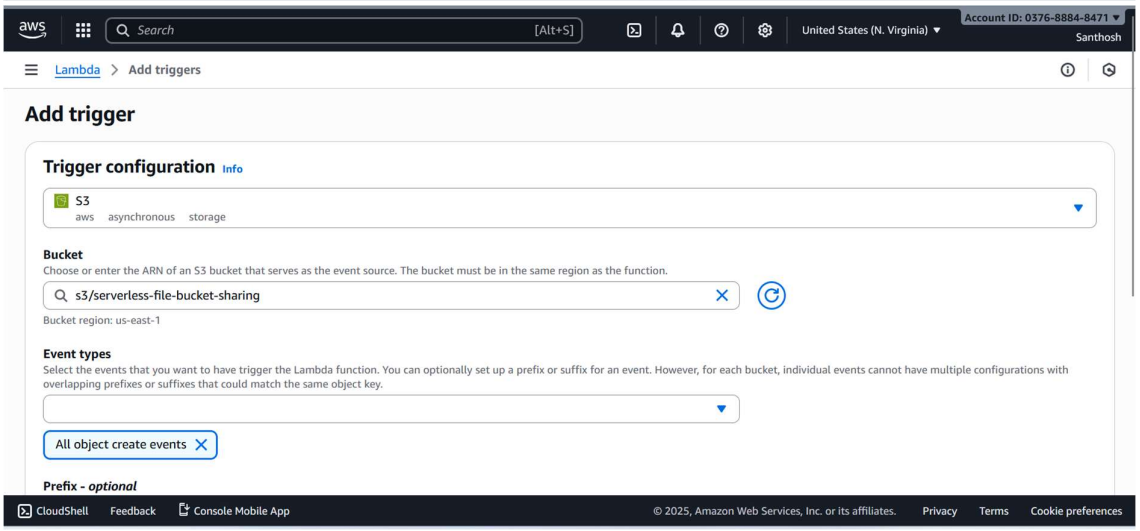


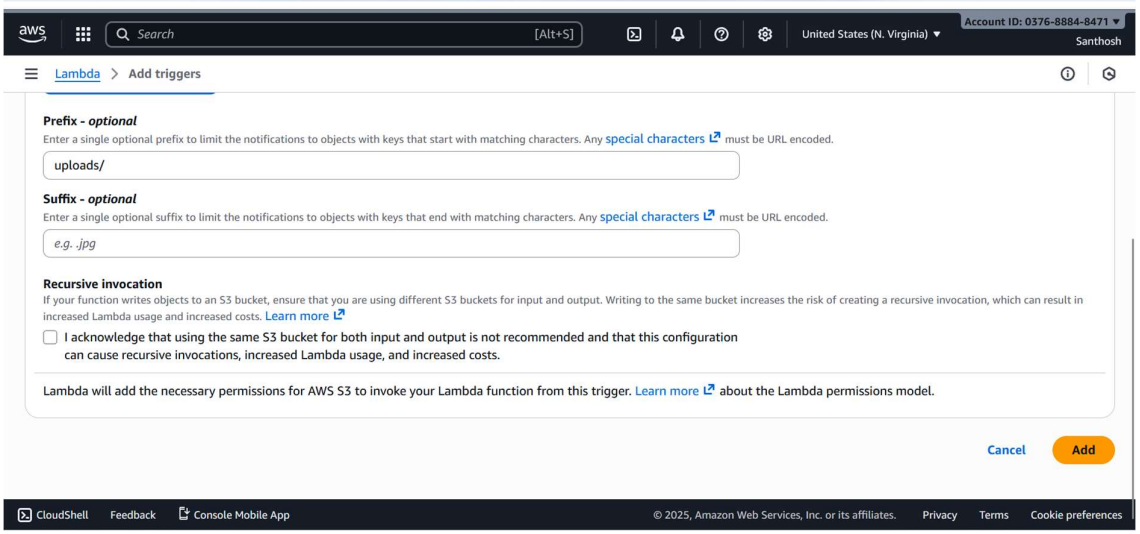


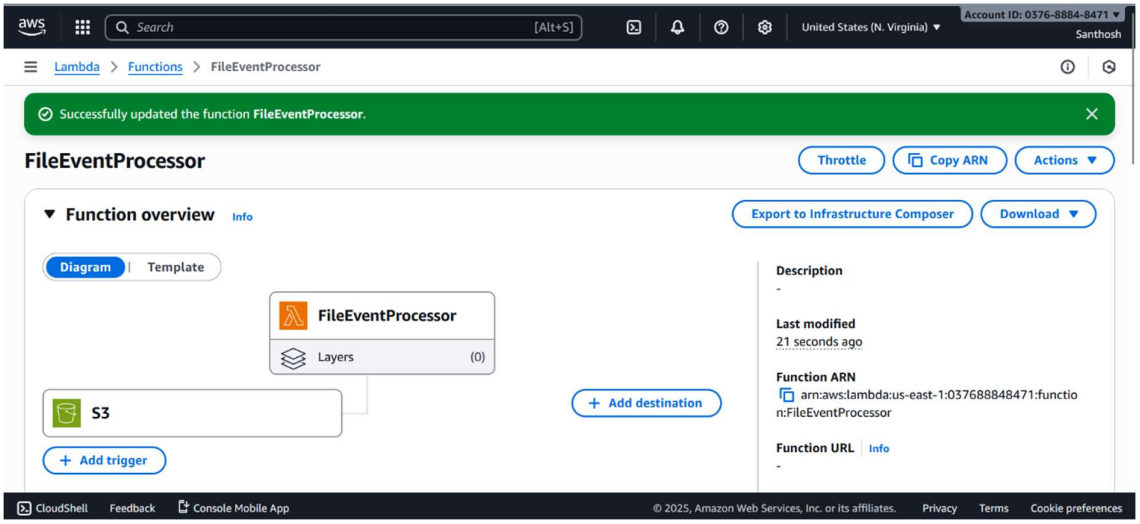


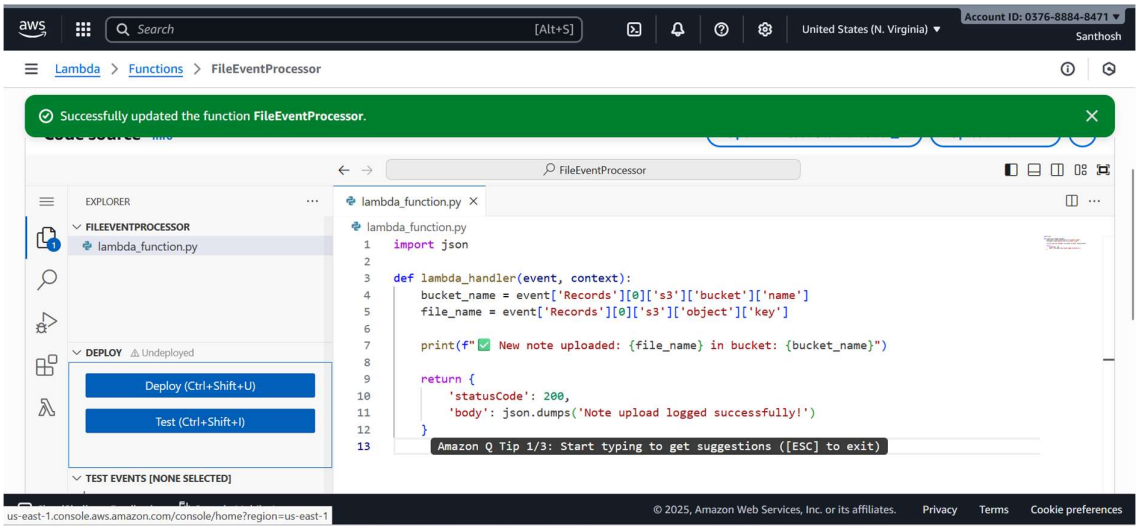
Step 3 — LAMBDA FUNCTION (Automation Lead)

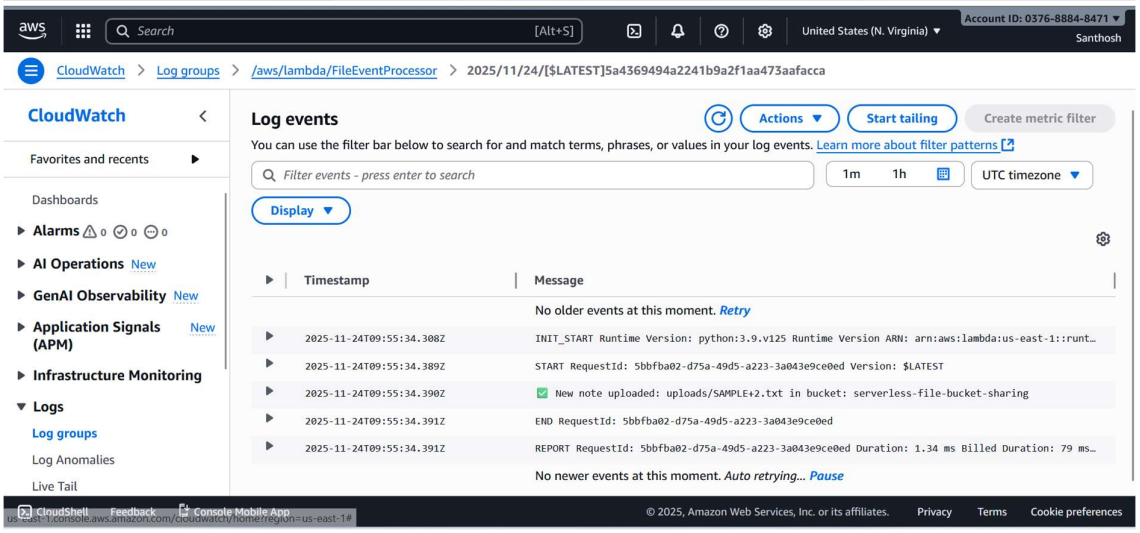




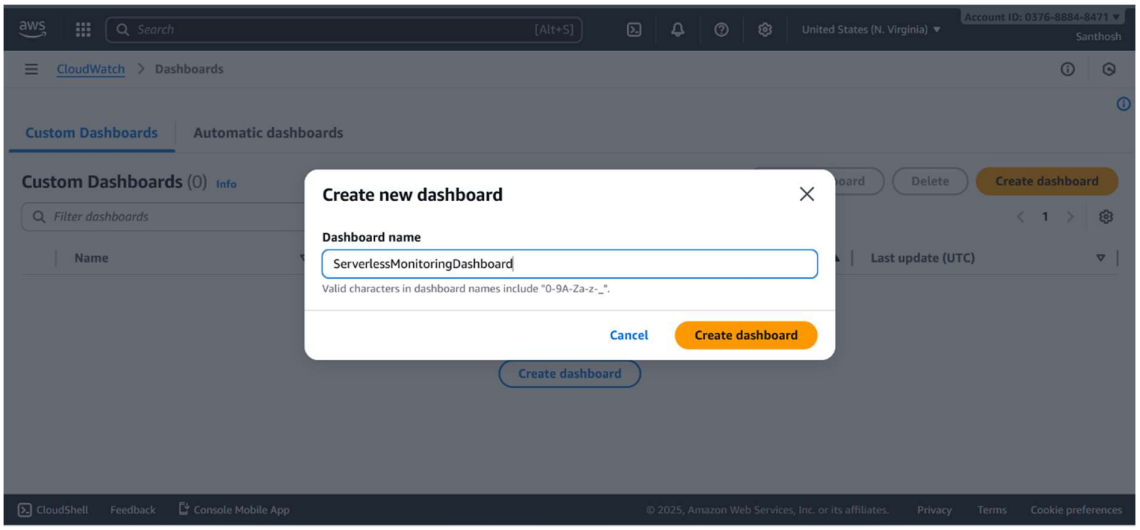


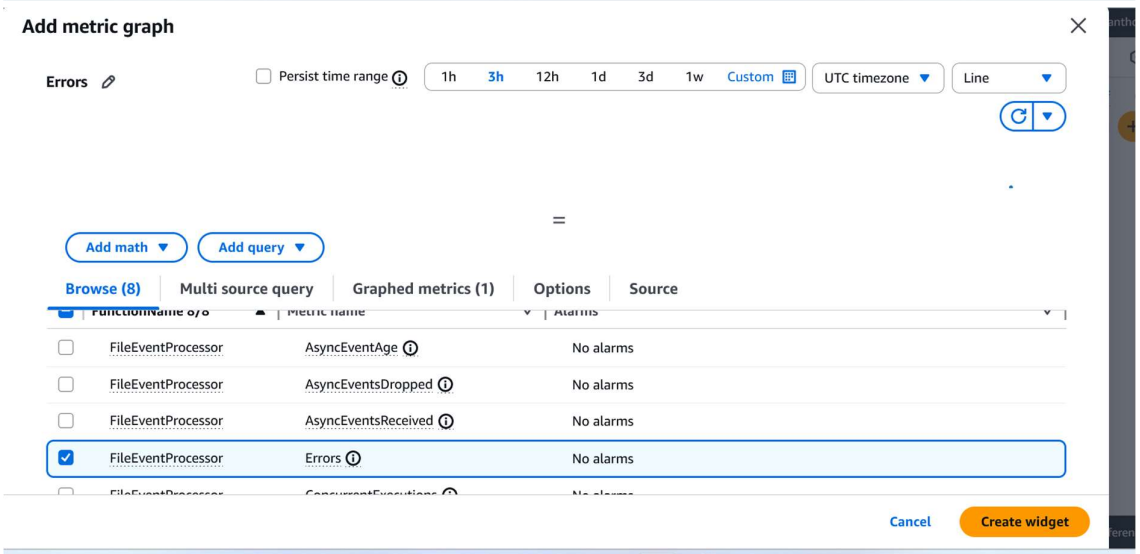

Account ID: 0376-8884-8471
Santhosh
United States (N. Virginia)
Lambda > Functions > FileEventProcessor
Successfully updated the function FileEventProcessor.
FileEventProcessor
Throttle
Copy ARN
Actions
Function overview
Info
Export to Infrastructure Composer
Download
Diagram
Template
FileEventProcessor
Layers
(0)
S3
Add trigger
Add destination
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

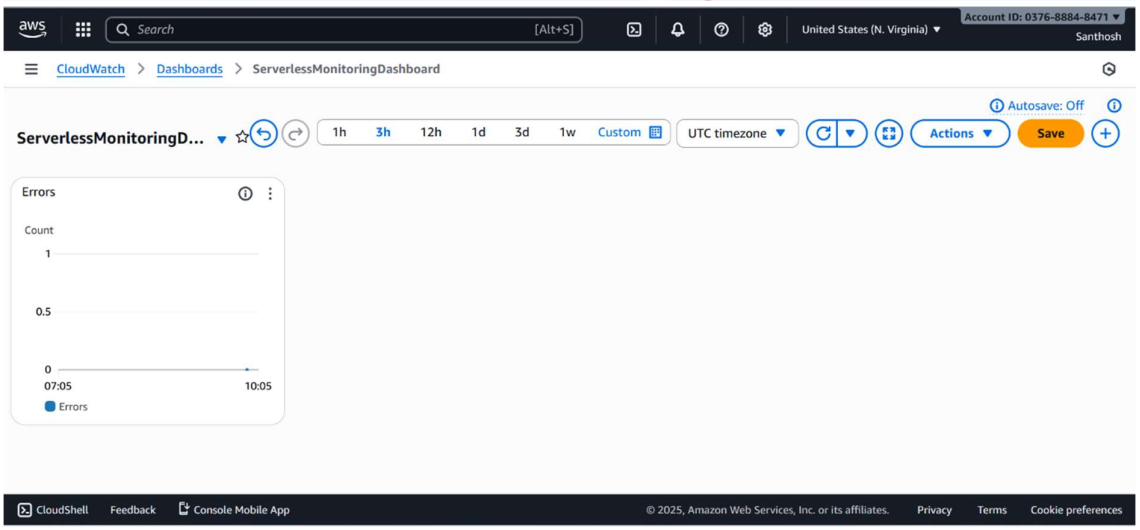

Account ID: 0376-8884-8471
Santhosh
United States (N. Virginia)
Lambda > Functions > FileEventProcessor
Successfully updated the function FileEventProcessor.
FileEventProcessor
EXPLORER
FILEEVENTPROCESSOR
lambda_function.py
DEPLOY
Undeployed
Deploy (Ctrl+Shift+U)
Test (Ctrl+Shift+I)
TEST EVENTS [NONE SELECTED]
lambda_function.py
import json
def lambda_handler(event, context):
bucket_name = event['Records'][0]['s3']['bucket']['name']
file_name = event['Records'][0]['s3']['object']['key']
print(f"✅ New note uploaded: {file_name} in bucket: {bucket_name}")
return {
'statusCode': 200,
'body': json.dumps('Note upload logged successfully!')
}
Amazon Q Tip 1/3: Start typing to get suggestions ([ESC] to exit)
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


Account ID: 0376-8884-8471
Santhosh
United States (N. Virginia)
CloudWatch > Log groups > /aws/lambda/FileEventProcessor > 2025/11/24/[$LATEST]5a4369494a2241b9a2f1aa473aafacca
CloudWatch
Favorites and recents
Dashboards
Alarms
AI Operations
New
GenAI Observability
New
Application Signals (APM)
New
Infrastructure Monitoring
Logs
Log groups
Log Anomalies
Live Tail
Log events
Actions
Start tailing
Create metric filter
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. Retry
2025-11-24T09:55:34.308Z
INIT_START Runtime Version: python:3.9.v125 Runtime Version ARN: arn:aws:lambda:us-east-1::runt…
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. Auto retrying... Pause
CloudShell
Feedback
Console Mobile App
us-east-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1#
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

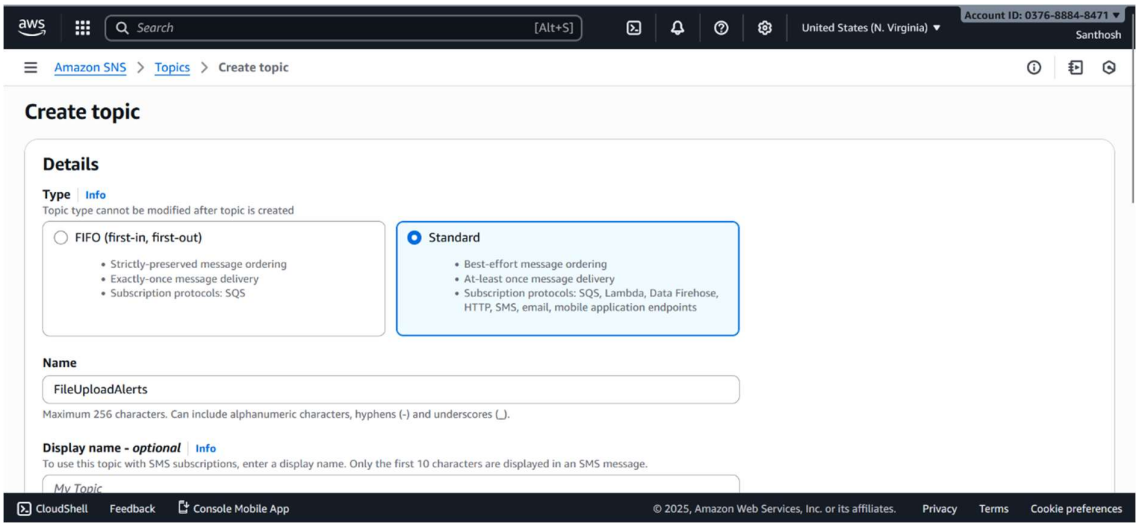
Step 4 — CLOUDWATCH MONITORING (Observability Lead)

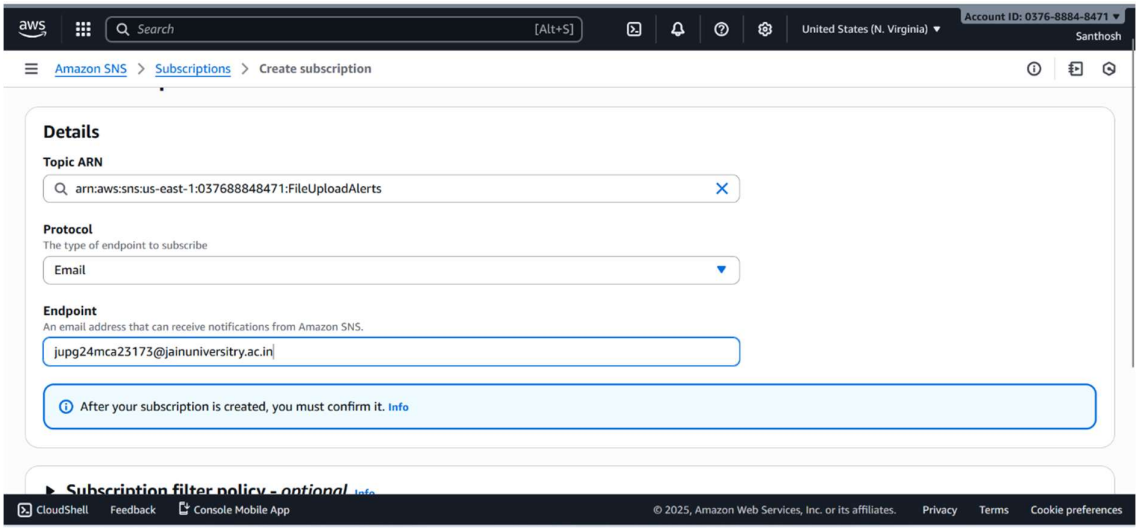


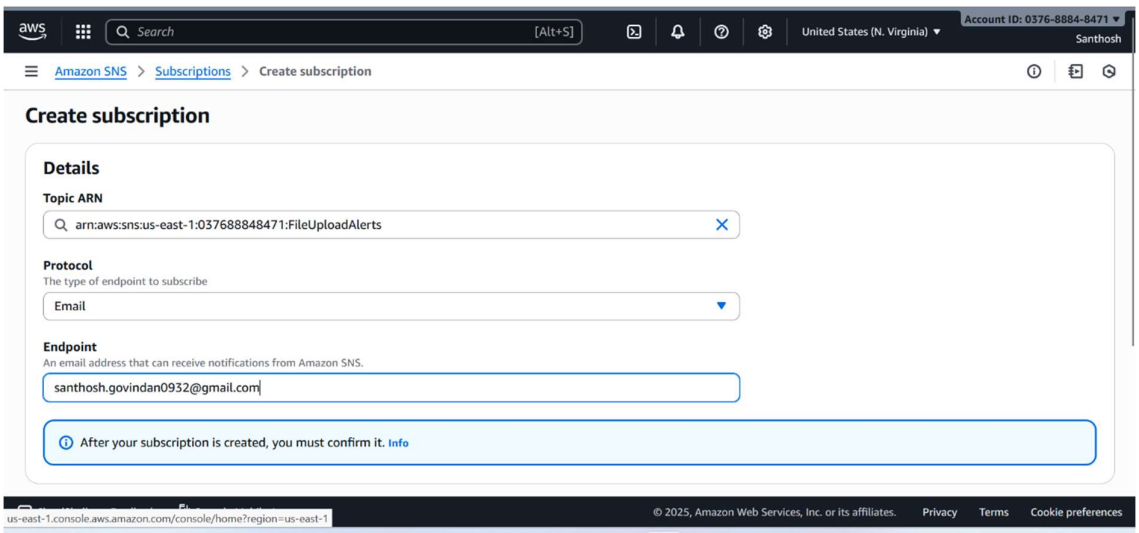


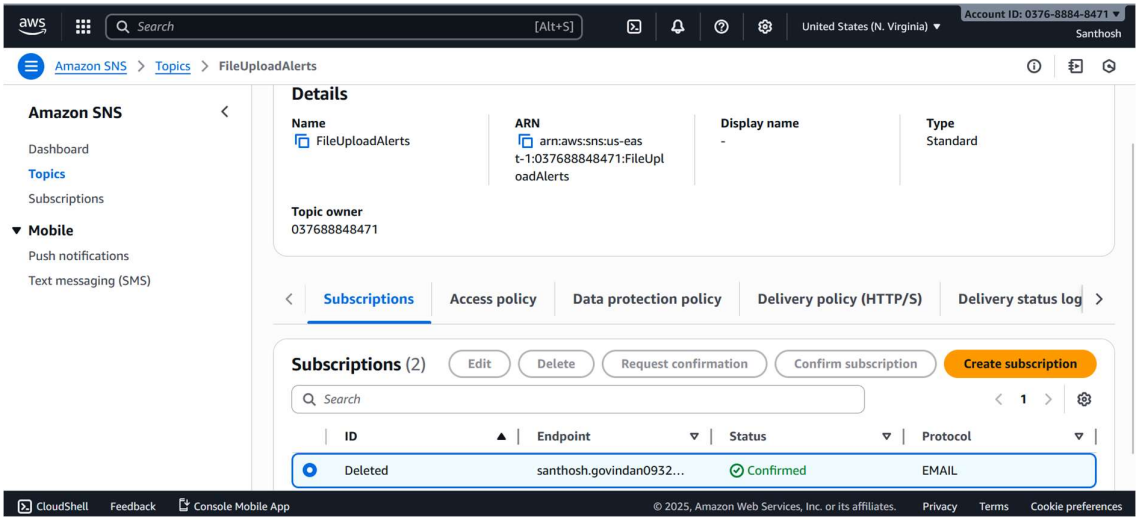


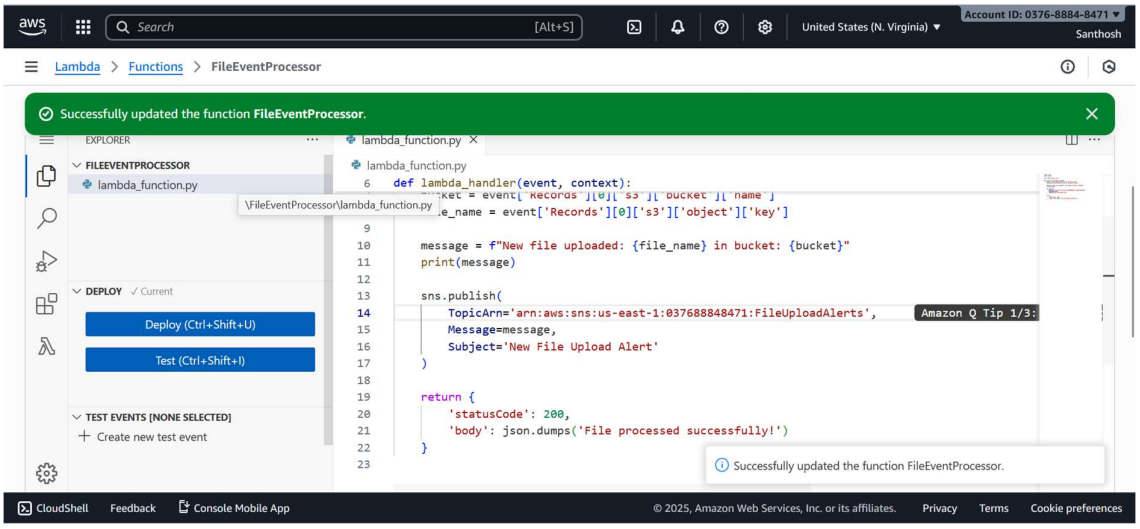
Step 5: SNS NOTIFICATIONS (Alerting Lead)



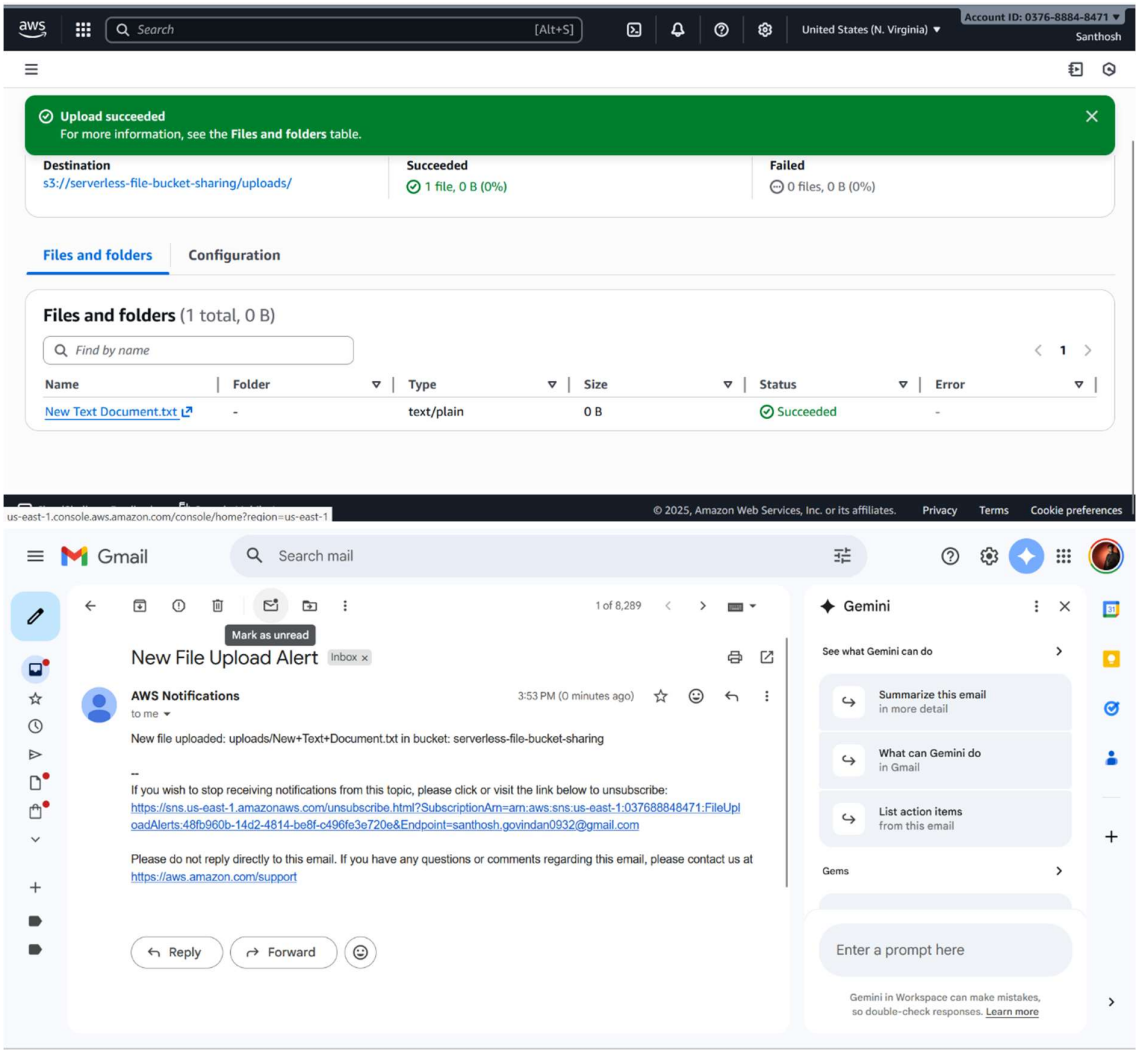





Search [Alt+S]
United States (N. Virginia)
Account ID: 0376-8884-8471
Santhosh
Amazon SNS > Topics > FileUploadAlerts
Amazon SNS
Dashboard
Topics
Subscriptions
Mobile
Push notifications
Text messaging (SMS)
Details
Name
FileUploadAlerts
ARN
arn:aws:sns:us-east-1:037688848471:FileUploadAlerts
Display name
-
Type
Standard
Topic owner
037688848471
Subscriptions
Access policy
Data protection policy
Delivery policy (HTTP/S)
Delivery status log
Subscriptions (2)
Edit
Delete
Request confirmation
Confirm subscription
Create subscription
Search
ID
Endpoint
Status
Protocol
Deleted
santhosh.govindan0932...
Confirmed
EMAIL
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Search [Alt+S]
United States (N. Virginia)
Account ID: 0376-8884-8471
Santhosh
Lambda > Functions > FileEventProcessor
Successfully updated the function FileEventProcessor.
EXPLORER
FILEEVENTPROCESSOR
lambda_function.py
\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
def lambda_handler(event, context):
bucket = event['Records'][0]['s3']['bucket']['name']
file_name = event['Records'][0]['s3']['object']['key']
message = f"New file uploaded: {file_name} in bucket: {bucket}"
print(message)
sns.publish(
TopicArn='arn:aws:sns:us-east-1:037688848471:FileUploadAlerts',
Amazon Q Tip 1/3:
Message=message,
Subject='New File Upload Alert'
)
return {
'statusCode': 200,
'body': json.dumps('File processed successfully!')
}
Successfully updated the function FileEventProcessor.
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Final verification:



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.

⭐ **If you want, I can now generate:**

📌 **A downloadable Word (.docx) file of this report**
📌 **A matching PPT**
📌 **A GitHub README file**

Just tell me: **“Generate Word file now”** and I will create it.