

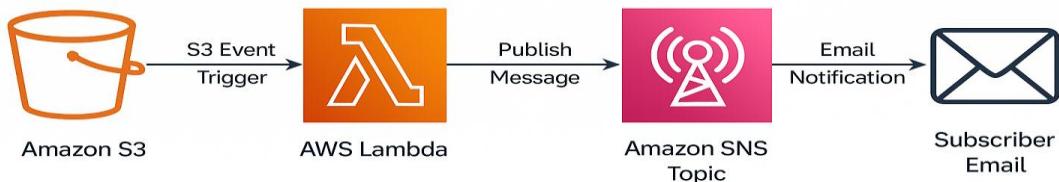
# PROJECT: AWS EVENT-DRIVEN ARCHITECTURE

## *Integration of S3, SNS & Lambda / File Upload Notification System*

By: Kiran Rakh

### OBJECTIVE

- Build a serverless, event-driven architecture using AWS services:
- - Uploading a file to S3 → Triggers a Lambda function → Sends a notification using SNS  
→ Delivers email alert to the subscriber.
- Overall Architecture Diagram:



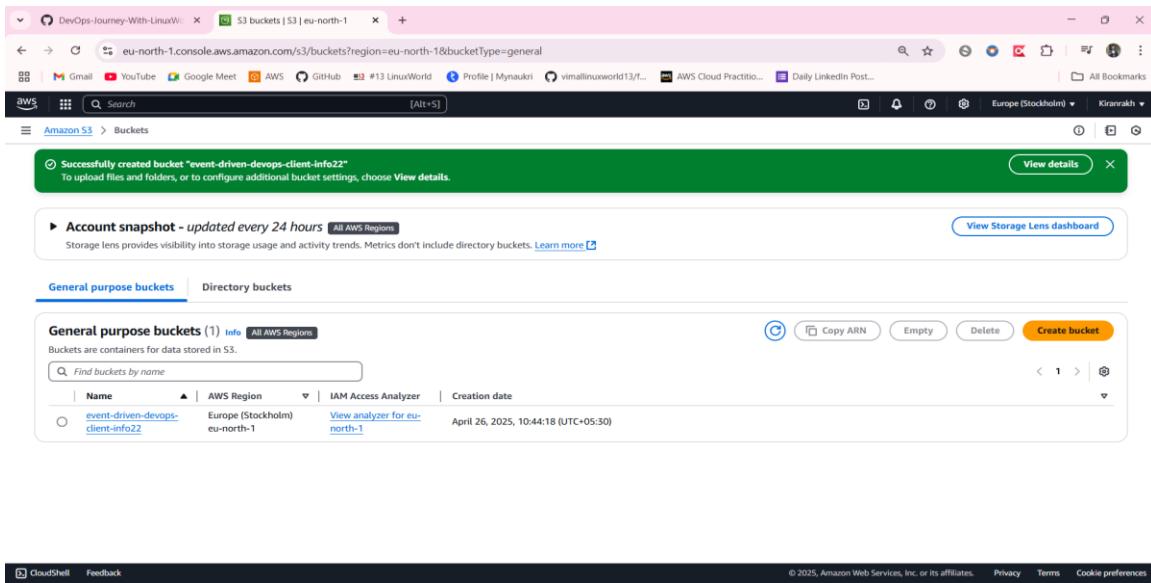
### AWS SERVICES USED

- - Amazon S3 (Simple Storage Service)
- - Amazon SNS (Simple Notification Service)
- - AWS Lambda (Python Runtime)
- - AWS CloudWatch (for logs)
- - IAM (Identity and Access Management)

## STEP-BY-STEP IMPLEMENTATION

### **1** S3: CREATE BUCKET

- 1. Go to AWS Console → Services → S3
- 2. Click “Create bucket”
- 3. Enter a unique name (e.g., event-driven-kiran-bucket)
- 4. Leave other settings as default
- 5. Click “Create bucket”



The screenshot shows the AWS S3 buckets page. A green success message at the top states: "Successfully created bucket 'event-driven-devops-client-info22'. To upload files and folders, or to configure additional bucket settings, choose View details." Below this, an "Account snapshot - updated every 24 hours" section provides storage usage and activity trends. The main table lists "General purpose buckets" with one entry:

| Name                              | AWS Region                       | IAM Access Analyzer                          | Creation date                        |
|-----------------------------------|----------------------------------|--|--------------------------------------|
| event-driven-devops-client-info22 | Europe (Stockholm)<br>eu-north-1 | <a href="#">View analyzer for eu-north-1</a> | April 26, 2025, 10:44:18 (UTC+05:30) |

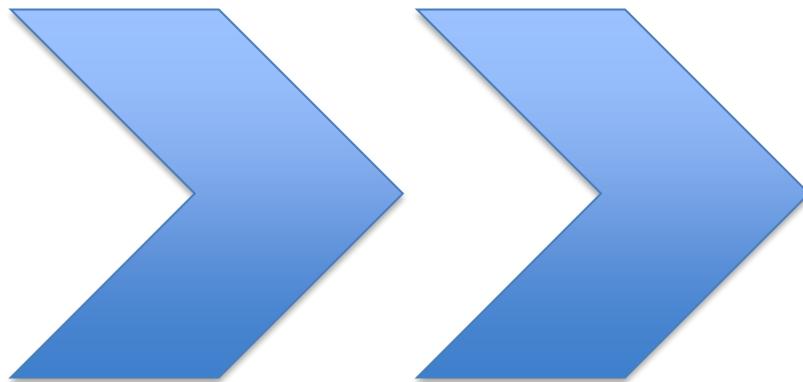
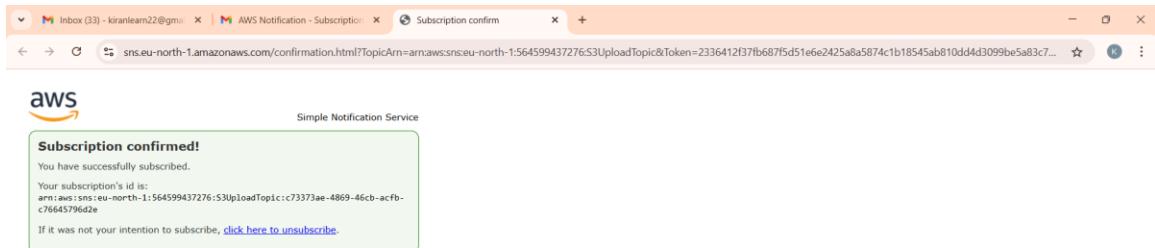
### **2** SNS: CREATE TOPIC & SUBSCRIPTION

- 1. Go to Services → SNS
- 2. Click “Create topic” → Type: Standard, Name: S3UploadTopic
- 3. Click “Create topic”
- 4. Go to your topic → Click “Create subscription”
- 5. Protocol: Email → Endpoint: kiranrakh155@gmail.com → Click “Create subscription”
- 6. Approve the subscription from your email inbox

- SNS topic and email subscription:

The screenshot shows the Amazon SNS Topics page. A green success message at the top states "Topic S3UploadTopic created successfully. You can create subscriptions and send messages to them from this topic." Below this, the "S3UploadTopic" details are shown, including its Name (S3UploadTopic), ARN (arn:aws:sns:eu-north-1:564599437276:S3UploadTopic), and Type (Standard). On the right, there are "Edit", "Delete", and "Publish message" buttons. Below the details, there are tabs for "Subscriptions", "Access policy", "Data protection policy", "Delivery policy (HTTP/S)", "Delivery status logging", "Encryption", "Tags", and "Integrations". The "Subscriptions" tab is selected, showing a table with one row: "No subscriptions found. You don't have any subscriptions to this topic." At the bottom, there are links for "CloudShell", "Feedback", and copyright information: "© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

The screenshot shows the "Create subscription" page for the S3UploadTopic. In the "Details" section, the "Topic ARN" is set to arn:aws:sns:eu-north-1:564599437276:S3UploadTopic, the "Protocol" is set to "Email", and the "Endpoint" is kirantdev22@gmail.com. A note below the endpoint says "After your subscription is created, you must confirm it." In the "Subscription filter policy - optional" section, it says "This policy filters the messages that a subscriber receives." In the "Redrive policy (dead-letter queue) - optional" section, it says "Send undeliverable messages to a dead-letter queue." At the bottom right, there are "Cancel" and "Create subscription" buttons. At the very bottom, there are links for "CloudShell", "Feedback", and copyright information: "© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".



### 3 LAMBDA: CREATE NEW FUNCTION

- 1. Go to Services → Lambda → Create function
- 2. Name: S3ToSNSNotification, Runtime: Python 3.x
- 3. Click “Create function”
- Lambda function code and configuration:

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The 'Basic information' section is visible, where the function name is set to 'S3ToSNSNotification'. Other settings include Python 3.10 as the runtime and x86\_64 as the architecture. A sidebar on the right provides a tutorial on creating a simple web app.

The screenshot shows the 'S3ToSNSNotification' function configuration page. The 'Code' tab is selected, displaying the function's code source. The code is a Python script named 'lambda\_function.py' that uses the Boto3 library to publish messages to an SNS topic when a file is uploaded to an S3 bucket. A success message at the top indicates the function was updated successfully.

```
lambda_function.py
1 import boto3
2
3 sns = boto3.client("sns")
4
5 def lambda_handler(event, context):
6     sns.publish(
7         TopicArn='arn:aws:sns:your-region:<your-account-id>:S3UploadTopic',
8         Message='A new file has been uploaded to the S3 bucket.',
9         Subject='New S3 File Alert'
10     )
11     print("Kiran this side from AWS team ... calling SNS ...!!!!")
```

## 4 IAM: PERMISSION SETUP FOR LAMBDA

- 1. Go to Lambda → Select your function → Configuration → Permissions
- 2. Click the role name → Add permissions → Attach policies
- 3. Search for and attach 'AmazonSNSFullAccess'
- 4. IAM role and policy attachment:

The screenshot shows the AWS IAM 'Add permissions' interface. At the top, there's a breadcrumb navigation: IAM > Roles > S3ToSNSNotification-role-klh6v3mh > Add permissions. The main title is 'Attach policy to S3ToSNSNotification-role-klh6v3mh'. Below it, a section titled 'Current permissions policies (1)' shows one policy attached: 'AmazonSNSFullAccess'. A large section below is titled 'Other permissions policies (1/1049)' and contains a search bar with 'sns' and a filter 'Type: AWS managed'. It lists several policies:

| Policy name  | Type        | Description                                 |
|--|-------------|---|
| AmazonSNSFullAccess                                      | AWS managed | Provides full access to Amazon SNS via...   |
| AmazonSNSSReadonlyAccess                                 | AWS managed | Provides read only access to Amazon S...    |
| AmazonSNSRole  | AWS managed | Default policy for Amazon SNS service...    |
| AWSElasticBeanstalkRoleSNS                               | AWS managed | (Elastic Beanstalk operations role) Allo... |
| AWSToTDeviceDefenderPublishFindingsToSNSMitigationAction | AWS managed | Provides messages publish access to S...    |

At the bottom right are 'Cancel' and 'Add permissions' buttons.

## 5 LINK S3 TO LAMBDA (EVENT NOTIFICATION)

- 1. Go to S3 → Your bucket → Properties → Event notifications
- 2. Create event notification → Name: TriggerLambdaOnUpload
- 3. Event type: PUT (Object Created)
- 4. Destination: Lambda Function → Choose: S3ToSNSNotification

- 5. Save configuration || S3 event notification setup:

**Create event notification** Info

To enable notifications, you must first add a notification configuration that identifies the events you want Amazon S3 to publish and the destinations where you want Amazon S3 to send the notifications.

**General configuration**

**Event name**  
TriggerLambdaOnUpload

**Prefix - optional**  
Limit the notifications to objects with key starting with specified characters.  
image/

**Suffix - optional**  
Limit the notifications to objects with key ending with specified characters.  
.jpg

**Event types**  
Specify at least one event for which you want to receive notifications. For each group, you can choose an event type for all events, or you can choose one or more individual events.

**Object creation**

All object create events  
s3:ObjectCreated\*

Put  
s3:ObjectCreated.Put

Post  
s3:ObjectCreated.Post

Copy  
s3:ObjectCreated.Copy

Move  
s3:ObjectCreated.Move

## 6 TEST IT

- 1. Go to S3 → Your bucket → Upload any file (e.g., test3.txt)
- 2. Lambda should be triggered automatically
- File upload to S3:

**Upload** Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDKs or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose [Add files](#) or [Add folder](#).

**Files and folders (1 total, 60.7 KB)**  
All files and folders in this table will be uploaded.

| Name              | Folder | Type       | Size    |
|-------------------|--------|------------|---------|
| 169644766845.jpeg | -      | image/jpeg | 60.7 KB |

**Destination** Info

**Destination**  
<s3://event-driven-devops-client-info22>

**Destination details**  
Bucket settings that impact new objects stored in the specified destination.

**Permissions**  
Grant public access and access to other AWS accounts.

**Properties**  
Specify storage class, encryption settings, tags, and more.

[Cancel](#) [Upload](#)

## 7 VERIFY IN CLOUDWATCH

- 1. Go to Services → CloudWatch → Log groups
- 2. Find log group for S3ToSNSNotification → Open logs
- 3. Check print statements and SNS execution logs
- CloudWatch logs output:

The screenshot shows the AWS CloudWatch Logs interface. On the left, there's a navigation sidebar with sections like AI Operations, Logs, Metrics, X-Ray traces, Events, Application Signals, Network Monitoring, and Insights. The main area is titled "Log events" and displays a table of log entries. The columns are "Timestamp" and "Message". The first message is a placeholder: "No older events at this moment. Retry". Subsequent messages are timestamped from April 2025 and show various AWS Lambda execution details, including start and end requests, errors (e.g., "LAMBDA\_ERROR" for unhandled exceptions), and report requests. The log entries are color-coded with grey for standard logs and blue for error logs. The interface includes a search bar, filter options, and various buttons for actions like "Actions", "Start tailing", and "Create metric filter".

The screenshot shows a Gmail inbox with one unread email. The subject of the email is "New S3 File Alert". The sender is "AWS Notifications <no-reply@aws.amazon.com> to me". The email body contains the following text:

A new file has been uploaded to the S3 bucket.  
--  
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:  
<https://sns.eu-north-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:eu-north-1:564599437276:S3UploadTopic:c73373ae-4869-46cb-acfb-c76645796d2&Endpoint=kiranrdev22@gmail.com>

At the bottom of the email, there's a note: "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>". The Gmail interface includes a compose button, a search bar, and various navigation and settings icons.

## FINAL USE CASE

- ✓ Automatic email notifications for every file uploaded to S3
- ✓ Use cases: automation pipelines, audit alerts, monitoring
- ✓ Fully managed serverless stack using AWS-native services

## EXTRAS / CUSTOMIZATION TIPS

- - You can rename function, bucket, topic, and handler
- - Edit handler in Lambda if function name is changed
- - Customize message in Lambda code
- - Keep all resources in the same region

## IAM ROLES & KEYS

- - Lambda role must have AmazonSNSFullAccess
- - S3 automatically gets permission to trigger Lambda during setup
- - No access keys required due to IAM roles