

AWS Lambda triggered by S3 to process ‘Word Count’ and notify via SNS

JULEANNY NAVAS
AWS Cloud Computing

Introduction

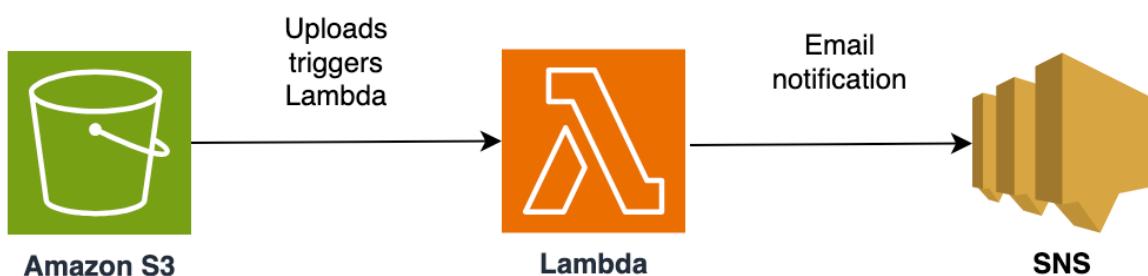
This project showcases the power of AWS **Serverless Computing** by automating word count processing using **AWS Lambda**. Each time a text file is uploaded to an **Amazon S3** bucket, an AWS Lambda function is triggered to count the number of words in the file. The result is then sent via email using **Amazon Simple Notification Service (SNS)**.

This event-driven architecture demonstrates efficient **cloud automation**, eliminating the need for traditional server management while ensuring seamless data processing and notification delivery.

Technologies used

- **AWS Lambda** (Python-based function execution)
- **Amazon S3** (Storage for text files triggering Lambda execution)
- **Amazon SNS** (Notification service for email alerts)
- <https://app.diagrams.net> (Architecture diagram)

Architecture Overview



Step-by-Step Implementation

1. Setting up an S3 bucket:

Amazon Simple Storage Service (**Amazon S3**) is a scalable object storage service that allows users to store and retrieve data with high availability and security. In this project, I used an S3 bucket to store text files, which will trigger the Lambda function upon upload.

Steps:

- Navigate to **S3** in the **AWS Management Console**:
- Click "Create bucket".

The screenshot shows the 'Create a bucket' wizard in the AWS Management Console. At the top, there's a navigation bar with icons for VPC, EC2, S3, Systems Manager, Route 53, and Lambda. Below the navigation bar, on the left, is a sidebar titled 'Amazon S3' with options like 'General purpose buckets', 'Directory buckets', 'Table buckets', 'Access Grants', 'Access Points', 'Object Lambda Access Points', and 'Multi-Region Access Points'. The main content area has a dark background with white text. It features the 'Amazon S3' logo and the tagline 'Store and retrieve any amount of data from anywhere'. Below this, it says 'Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance.' To the right, there's a 'Create a bucket' button, which is highlighted with a yellow border. A text box next to it says: 'Every object in S3 is stored in a bucket. To upload files and folders to S3, you'll need to create a bucket where the objects will be stored.'

- Enter a unique bucket name (**word-count-project-juna**).
- Choose a **region** (ensure all resources are in the same region).
- Keep the **Block Public Access** settings as default (disabled for security).

The screenshot shows the 'Create bucket' configuration wizard in the AWS Management Console. At the top, there's a navigation bar with icons for VPC, EC2, S3, Systems Manager, Route 53, and Lambda. Below the navigation bar, the path is shown as 'Amazon S3 > Buckets > Create bucket'. The main content area is divided into several sections: 'General configuration', 'Object Ownership', and 'Block Public Access settings for this bucket'. In the 'General configuration' section, there are fields for 'Bucket name' (set to 'word-count-project-juna'), 'AWS Region' (set to 'US West (Oregon) us-west-2'), and 'Bucket type' (radio button selected for 'General purpose'). In the 'Object Ownership' section, there are two radio buttons: 'ACLs disabled (recommended)' (selected) and 'ACLs enabled'. In the 'Block Public Access settings for this bucket' section, there is a checked checkbox for 'Block all public access'. Below this, there are four additional checkboxes: 'Block public access to buckets and objects granted through new access control lists (ACLS)', 'Block public access to buckets and objects granted through any access control lists (ACLS)', 'Block public access to buckets and objects granted through new public bucket or access point policies', and 'Block public and cross-account access to buckets and objects through any public bucket or access point policies'. The 'Block all public access' checkbox is highlighted with a red border.

- Click "Create bucket" to finalize.

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

[Cancel](#)

Create bucket

Amazon S3 > Buckets

i [View details](#) X

i Successfully created bucket "word-count-project-juna"

To upload files and folders, or to configure additional bucket settings, choose [View details](#).

► Account snapshot - updated every 24 hours [All AWS Regions](#)

[View Storage Lens dashboard](#)

[General purpose buckets](#) [Directory buckets](#)

General purpose buckets (1) [Info](#) [All AWS Regions](#)

Buckets are containers for data stored in S3.



[Copy ARN](#)

[Empty](#)

[Delete](#)

Create bucket

Find buckets by name

< 1 > ⚙️

▲ Name

▼ AWS Region

▼ IAM Access Analyzer

▼ Creation date

○ [word-count-project-juna](#)

US West (Oregon) us-west-2

[View analyzer for us-west-2](#)

March 19, 2025, 20:03:07 (UTC+01:00)

2. Creating an SNS Topic

Amazon Simple Notification Service (**Amazon SNS**) is a fully managed messaging service that enables event-driven communication between distributed systems, microservices, and users. In this project, SNS is used to send an email notification with the word count result.

Steps:

- Navigate to the **SNS** service in the **AWS Management Console**.
- Click "**Create topic**". Enter **Topic name** (**WordCountNotification**). Click **Next step**.

The screenshot shows the 'Create topic' wizard in the AWS Management Console. On the left, there's a sidebar with 'Amazon SNS' navigation options like Dashboard, Topics, Subscriptions, and Mobile (Push notifications, Text messaging (SMS)). The main area has a title 'Amazon Simple Notification Service' and a subtitle 'Pub/sub messaging for microservices and serverless applications.' A paragraph about Amazon SNS follows. On the right, a large orange-bordered box contains the 'Create topic' form. It has a 'Topic name' field with 'WordCountNotification' and a 'Next step' button below it. Below the form is a link 'Start with an overview'.

- Select "Standard" as the topic type. Click "**Create topic**".

The screenshot shows the 'Create topic' wizard on the 'Details' page. Under 'Type', 'Standard' is selected (radio button highlighted with a blue border). Other options like 'FIFO (first-in, first-out)' are shown with their descriptions. Below this, there's a 'Name' field containing 'WordCountNotification'. The 'Create topic' button is at the bottom right, also highlighted with a blue border.

- Copy the SNS topic **ARN** (for later use).
- Click "**Create subscription**" to add an email recipient:

Amazon SNS > Topics > WordCountNotification

WordCountNotification

Details

Name	WordCountNotification	Display name	-
ARN	arn:aws:sns:us-west-2:719503814044:WordCountNotification	Topic owner	719503814044
Type	Standard		

Subscriptions | Access policy | Data protection policy | Delivery policy (HTTP/S) | Delivery status logging | Encryption | Tags | Integ >

Subscriptions (0)

Edit | Delete | Request confirmation | Confirm subscription | **Create subscription**

- **Protocol:** Email
- **Endpoint:** My email address.
- Click "Create subscription".

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN

arn:aws:sns:us-west-2:719503814044:WordCountNotification

Protocol

The type of endpoint to subscribe

Email

Endpoint

An email address that can receive notifications from Amazon SNS.

@gmail.com

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

Subscription filter policy - optional [Info](#)

This policy filters the messages that a subscriber receives.

Redrive policy (dead-letter queue) - optional [Info](#)

Send undeliverable messages to a dead-letter queue.

Cancel | **Create subscription**

- I confirmed the **subscription email** by clicking the link in the verification email.

AWS Notifications <no-reply@sns.amazonaws.com>

para mí ▾

Traducir al español

You have chosen to subscribe to the topic:
arn:aws:sns:us-west-2:719503814044:WordCountNotification

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
[Confirm subscription](#)

Subscription confirmed!

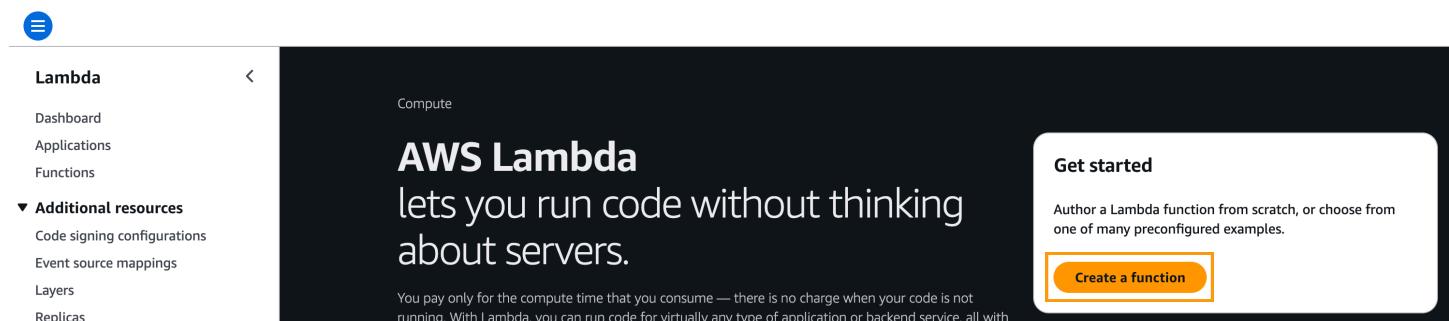
You have successfully subscribed.
Your subscription's id is:
arn:aws:sns:us-west-2:719503814044:WordCountNotification:e754f2de-3635-42b5-9e99-58811239c5a7
If it was not your intention to subscribe, [click here to unsubscribe](#).

3 . Creating the Lambda Function

AWS **Lambda** is a serverless compute service that allows you to run code in response to events without provisioning or managing servers. You can trigger Lambda functions based on events from AWS services like S3, DynamoDB, and SNS, among others. It automatically scales by running code in response to triggers.

Steps:

- Go to **Lambda** in the AWS Management Console. Click **Create function**.



- Select **Author from scratch**.
- Enter the **Function name (word_count_lambda)**.
- Select **Runtime**: Python 3.9 (or the latest available version).
- Under **Change default execution role**, select **Use an existing role**.
- Select the **LambdaAccessRole** (S3 and SNS access).
- Click **Create function**.

This screenshot shows the detailed configuration page for creating a new Lambda function. At the top, it says 'Create function' with a 'Info' link. Below that, there are three options: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. The 'Basic information' section includes fields for 'Function name' (set to 'word_count_lambda') and 'Runtime' (set to 'Python 3.9'). The 'Architecture' section shows 'x86_64' selected. In the 'Permissions' section, under 'Change default execution role', 'Use an existing role' is selected, and 'LambdaAccessRole' is chosen from a dropdown. The 'Existing role' dropdown also lists 'LambdaAccessRole'. At the bottom, there's an 'Additional Configurations' section and a 'Create function' button.

Notes: An AWS Identity and Access Management (IAM) role is required for the Lambda function to access other AWS services. For this project, I chose **LambdaAccessRole**, which provides the following permissions:

- [AWSLambdaBasicExecutionRole](#): This is an AWS managed policy that provides write permissions to Amazon CloudWatch Logs.
- [AmazonSNSFullAccess](#): This is an AWS managed policy that provides full access to Amazon SNS via the AWS Management Console.
- [AmazonS3FullAccess](#): This is an AWS managed policy that provides full access to all buckets via the AWS Management Console.
- [CloudWatchFullAccess](#)

IAM > Roles > LambdaAccessRole

LambdaAccessRole Info

Summary

Creation date
March 19, 2025, 19:45 (UTC+01:00)

Last activity
-

ARN
`arn:aws:iam::719503814044:role/LambdaAccessRole`

Maximum session duration
1 hour

Permissions **Trust relationships** **Tags (1)** **Last Accessed** **Revoke sessions**

Permissions policies (4) [Info](#)

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonS3FullAccess	AWS managed	1
AmazonSNSFullAccess	AWS managed	1
AWSLambdaBasicExecutionRole	AWS managed	1
CloudWatchFullAccess	AWS managed	1

Filter by Type

Search All types

Add permissions

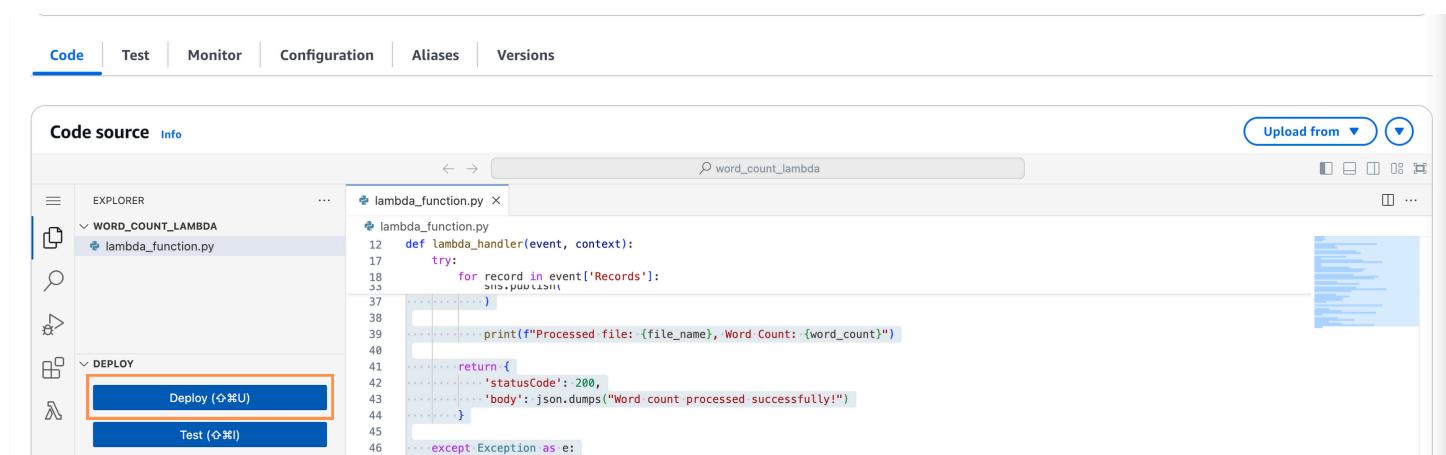
4 . Adding Code to the Lambda Function

Steps:

- Scroll down to the **Code source** section of the Lambda function configuration page.
- Click on the code and replace the default code with the following:

```
1 import json
2 import boto3
3 import os
4
5 # Initialize AWS clients outside the function to optimize performance
6 s3 = boto3.client('s3')
7 sns = boto3.client('sns')
8
9 # Get SNS topic ARN from environment variables
10 SNS_TOPIC_ARN = os.environ.get('SNS_TOPIC_ARN', '')
11
12 def lambda_handler(event, context):
13     """
14     AWS Lambda function triggered by S3 file uploads.
15     Counts the number of words in a text file and sends the result via SNS.
16     """
17     try:
18         for record in event['Records']:
19             bucket_name = record['s3']['bucket']['name']
20             file_name = record['s3']['object']['key']
21
22             # Retrieve file content from S3
23             response = s3.get_object(Bucket=bucket_name, Key=file_name)
24             file_content = response['Body'].read().decode('utf-8')
25
26             # Count words
27             word_count = len(file_content.split())
28
29             # Create and send SNS notification
30             message = f"The word count in the {file_name} file is {word_count}."
31             subject = "Word Count Result"
32
33             sns.publish(
34                 TopicArn=SNS_TOPIC_ARN,
35                 Message=message,
36                 Subject=subject
37             )
38
39             print(f"Processed file: {file_name}, Word Count: {word_count}")
40
41         return {
42             'statusCode': 200,
43             'body': json.dumps("Word count processed successfully!")
44         }
45
46     except Exception as e:
47         print(f"Error processing file: {e}")
48         return {
49             'statusCode': 500,
50             'body': json.dumps(f"Error processing file: {str(e)}")
51         }
```

- Click **Deploy** button.



What does this Python code do?

This **AWS Lambda function** is triggered when a text file is uploaded to an **Amazon S3 bucket**. It **counts the number of words** in the file and sends the result via **Amazon SNS (Simple Notification Service)**.

- **AWS Clients Initialization (Outside the Function for Efficiency):**

- **boto3**: AWS SDK for Python to interact with **S3** and **SNS**.
- Initializing AWS clients **outside the function** reduces latency by reusing connections.

```
5 # Initialize AWS clients outside the function to optimize performance
6 s3 = boto3.client('s3')
7 sns = boto3.client('sns')
```

- **Fetching SNS Topic ARN from Environment Variables:**

- **SNS topic ARN** is stored in an **environment variable**, making the function **more flexible** without hardcoding values.

```
9 # Get SNS topic ARN from environment variables
10 SNS_TOPIC_ARN = os.environ.get('SNS_TOPIC_ARN', '')
11
```

- **Lambda Event Processing & Word Count Execution:**

- Loops through S3 event records to process each uploaded file.
- Extracts bucket name and file name from the event.

```
18     for record in event['Records']:
19         bucket_name = record['s3']['bucket']['name']
20         file_name = record['s3']['object']['key']
21
```

- **Reading File Content from S3:**

- Retrieves the uploaded file from S3 and decodes it as text.

```
22     # Retrieve file content from S3
23     response = s3.get_object(Bucket=bucket_name, Key=file_name)
24     file_content = response['Body'].read().decode('utf-8')
25
```

- **Counting Words in the File:**

- Splits the file content by spaces and counts words.

```
26     # Count words
27     word_count = len(file_content.split())
28
```

- **Sending Notification via SNS:**

- Publishes an **SNS message** containing the **word count result**.

```
29     # Create and send SNS notification
30     message = f"The word count in the {file_name} file is {word_count}."
31     subject = "Word Count Result"
32
33     sns.publish(
34         TopicArn=SNS_TOPIC_ARN,
35         Message=message,
36         Subject=subject
37     )
```

- **Logging & Success Response:**

- Logs the processed file details and returns a 200 OK response.

```
39         print(f"Processed file: {file_name}, Word Count: {word_count}")
40
41     return {
42         'statusCode': 200,
43         'body': json.dumps("Word count processed successfully!")
44     }
```

- **Error Handling:**

- Catches errors and returns a 500 error response if the function fails.

```
46     except Exception as e:
47         print(f"Error processing file: {e}")
48         return {
49             'statusCode': 500,
50             'body': json.dumps(f"Error processing file: {str(e)}")
51         }
```

5. Configuring Environment Variables in AWS Lambda

Configuring the **SNS topic ARN** as an **Environment variable** for the Lambda function:

- Select **Configuration** tab, in the left pane select **Environment variables**. Click the **Edit** button.

The screenshot shows the AWS Lambda console interface. At the top, the path is Lambda > Functions > word_count_lambda. The main area displays the function overview with a diagram showing the function name 'word_count_lambda' and its triggers ('S3'). Below the diagram, there are buttons for '+ Add destination' and '+ Add trigger'. On the right, there are sections for 'Description', 'Last modified' (54 minutes ago), 'Function ARN' (arn:aws:lambda:us-west-2:719503814044:function:word_count_lambda), and 'Function URL' (Info). The bottom navigation bar has tabs for Code, Test, Monitor, Configuration (which is highlighted with a blue border), Aliases, and Versions. On the far left, a sidebar lists General configuration, Triggers, Permissions, Destinations, Function URL, Environment variables (which is highlighted with an orange border), and Tags.

- Add the SNS Topic ARN as an Environment Variable:**
 - Key:** *SNS_TOPIC_ARN*
 - Value:** *arn:aws:sns:us-west-2:719503814044:WordCountNotification* (the SNS topic ARN)
 - Click **Save** button.

The screenshot shows the 'Edit environment variables' dialog for the 'word_count_lambda' function. The title is 'Edit environment variables'. It contains a table with one row, showing a key-value pair. The 'Key' column contains 'SNS_TOPIC_ARN' and the 'Value' column contains 'arn:aws:sns:us-west-2:719503814044:WordCountNotification'. There is a 'Remove' button next to the value. Below the table, there is a link 'Encryption configuration'. In the bottom right corner, there are 'Cancel' and 'Save' buttons, with the 'Save' button highlighted by an orange box.

6 . Configuring the S3 Trigger

A **trigger** is an event that automatically invokes a function. In the case of AWS Lambda, a trigger can be an action performed by an AWS service, such as an S3 upload, a DynamoDB update, or an SNS notification. When the specified event occurs, it automatically executes the Lambda function to process the event.

Configuring the Lambda function to be triggered when a file is uploaded to the S3 bucket:

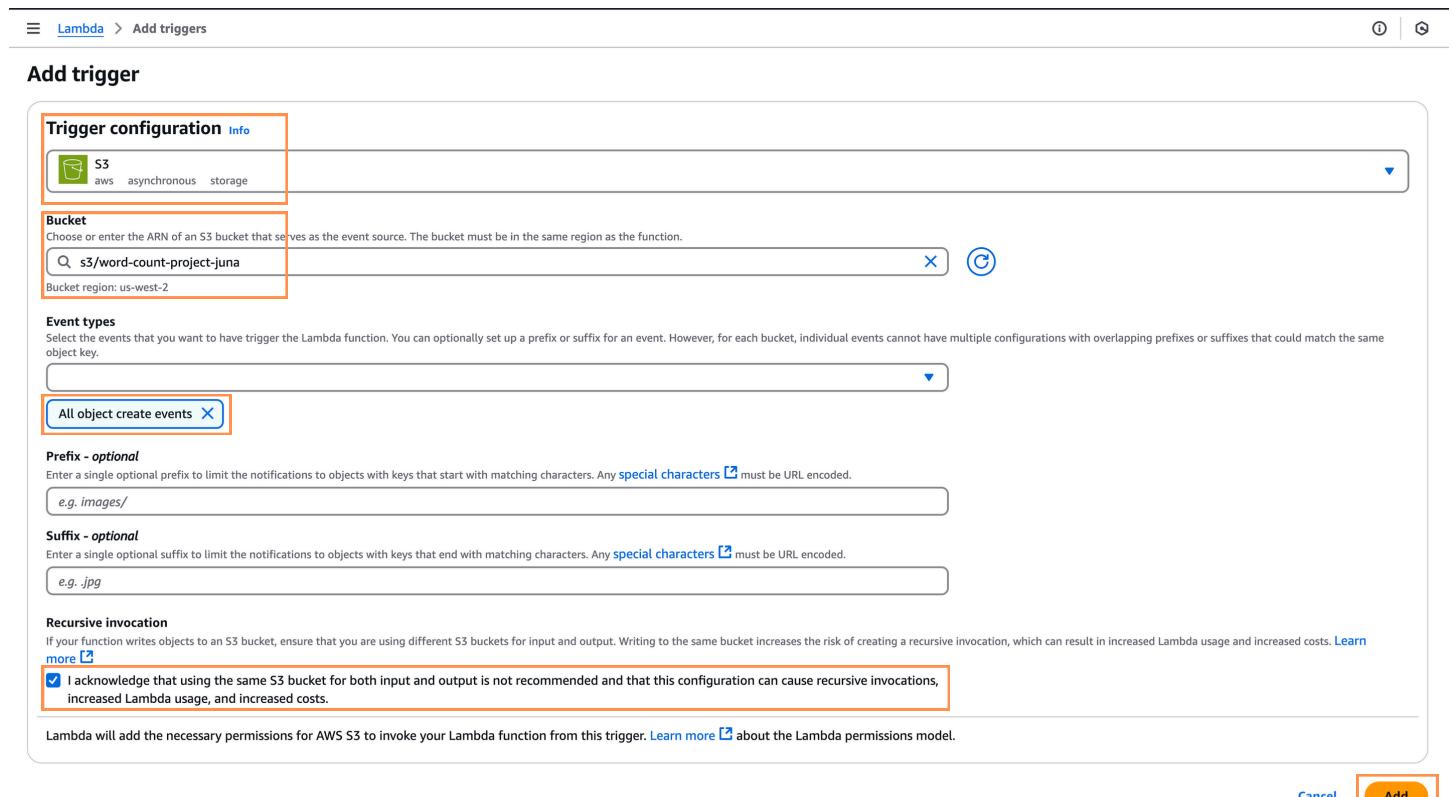
Steps:

- Go to your **Lambda**. Select word_count_lambda. Click **Add trigger**.



The screenshot shows the AWS Lambda Function Overview page for a function named 'word_count_lambda'. The 'Function overview' tab is selected. On the left, there's a 'Diagram' button (which is currently active) and a 'Template' button. Below the diagram is a box containing the function name and a 'Layers' section with '(0)' listed. To the right, there are buttons for 'Throttle', 'Copy ARN', and 'Actions'. Further down, there are 'Export to Infrastructure Composer' and 'Download' buttons. A 'Description' section is present with a link. The 'Last modified' section shows '24 minutes ago'. The 'Function ARN' section displays the ARN: arn:aws:lambda:us-west-2:719503814044:function:word_count_lambda. A 'Layers' section is also visible on the right.

- Choose **S3** as the **source**.
- Bucket: Select the S3 bucket you created before.
- In **Event type**, select **PUT** (for new uploads or '**All object create events**').
- Click **Add**.



The screenshot shows the 'Add trigger' configuration page for the 'word_count_lambda' function. The 'Trigger configuration' section is highlighted with a red box. It shows 'S3' selected as the source. The 'Bucket' section is also highlighted with a red box; it contains a search bar with 's3/word-count-project-juna' and a 'Bucket region: us-west-2' dropdown. The 'Event types' section has a 'All object create events' checkbox checked, which is highlighted with a red box. Below this, there are sections for 'Prefix - optional' and 'Suffix - optional', both with their respective input fields. A note about recursive invocation is present, with a checkbox checked that says '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.' At the bottom, a note states 'Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger.' There are 'Cancel' and 'Add' buttons at the bottom right.

word_count_lambda

[Throttle](#)[Copy ARN](#)[Actions ▾](#)

X

▼ Function overview [Info](#)[Export to Infrastructure Composer](#)[Download ▾](#)[Diagram](#)[Template](#)**Description****Last modified**

31 minutes ago

Function ARN[arn:aws:lambda:us-west-2:719503814044:function:word_count_lambda](#)**Function URL** [Info](#)

-

7 . Testing the Function

Testing ensures that the Lambda function is working as expected and correctly integrated with other AWS services, such as S3 and SNS. It helps identify issues early in the process, allowing you to troubleshoot and fix errors before the system is deployed in a production environment. Additionally, testing verifies that the function processes the input data correctly and triggers the expected notifications.

Steps:

- **Upload a text file to the S3 bucket:**

- Go to **S3** and select your bucket.

The screenshot shows the Amazon S3 console. On the left, there's a sidebar with options like General purpose buckets, Directory buckets, Table buckets, etc. The main area shows an account snapshot and a list of general purpose buckets. One bucket, 'word-count-project-juna', is selected and highlighted with a yellow box. The 'Create bucket' button at the top right is also highlighted.

- Click **Upload**.

The screenshot shows the objects page for the 'word-count-project-juna' bucket. The sidebar on the left is identical to the previous screenshot. The main area shows the bucket name and a table for managing objects. At the top right, there's a prominent 'Upload' button highlighted with a yellow box.

- Click **Add Files**, choose a text file and click **Upload**.

Amazon S3 > Buckets > word-count-project-juna > Upload

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, 208.0 B)
All files and folders in this table will be uploaded.

Name	Folder	Type	Size
24-words-file.txt	-	text/plain	208.0 B

[Remove](#) [Add files](#) [Add folder](#)

Destination Info

Destination
[s3://word-count-project-juna](#)

▶ **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](#)

Upload succeeded
For more information, see the [Files and folders](#) table.

Upload: status

After you navigate away from this page, the following information is no longer available.

Summary

Destination	Succeeded	Failed
s3://word-count-project-juna	1 file, 208.0 B (100.00%)	0 files, 0 B (0%)

[Close](#)

Files and folders (1 total, 208.0 B)

Name	Folder	Type	Size	Status	Error
24-words-file.txt	-	text/plain	208.0 B	Succeeded	-

Check SNS Notifications:

- As expected, I received an email with the word count result.

AWS Notifications <no-reply@sns.amazonaws.com>
para mí ▾

The word count in the 24-words-file.txt file is 24.

Conclusions & Lessons Learned

1. Serverless Computing Simplifies Deployment

No need to manage servers, scaling, or infrastructure.

2. Event-Driven Automation is Efficient

AWS Lambda automatically processes files as soon as they are uploaded.

3. Seamless AWS Service Integration

S3, Lambda, and SNS work together for a fully automated pipeline.

4. Security & Best Practices Matter

IAM roles must be properly configured for Lambda, S3, and SNS.

SNS email subscriptions require user confirmation.

Final Thoughts

This project highlights the **efficiency and scalability** of **AWS Serverless Computing**. By leveraging **Lambda, S3, and SNS**, I successfully **automated word count processing** while eliminating the need for traditional server management.