

AdvDevOps Case Study 12: Serverless Logging with S3 and Lambda

- **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:

- In today's cloud-centric world, organizations are increasingly leveraging serverless architectures to streamline their operations and enhance efficiency. This case study explores the implementation of a serverless logging system using AWS Lambda and Amazon S3. The objective is to automate the logging process by creating a Lambda function that is triggered whenever a text file is uploaded to an S3 bucket.
- By utilizing AWS Lambda, a fully managed serverless computing service, developers can run code in response to events without the need for provisioning or managing servers. This architecture allows for seamless scaling and reduced operational costs, making it an ideal solution for handling sporadic workloads. The integration with Amazon S3 enables users to store and retrieve data effortlessly, while AWS Cloud9 provides a powerful environment for developing and deploying Lambda functions.
- The core problem addressed in this study is to set up a Lambda function that automatically logs the contents of uploaded text files, thus enhancing data handling capabilities. The tasks involved include creating a Lambda function using Python, configuring the S3 bucket as the event trigger, and verifying the logging functionality by uploading a sample text file. This implementation not only improves efficiency but

also demonstrates the practical benefits of serverless computing in modern application development.

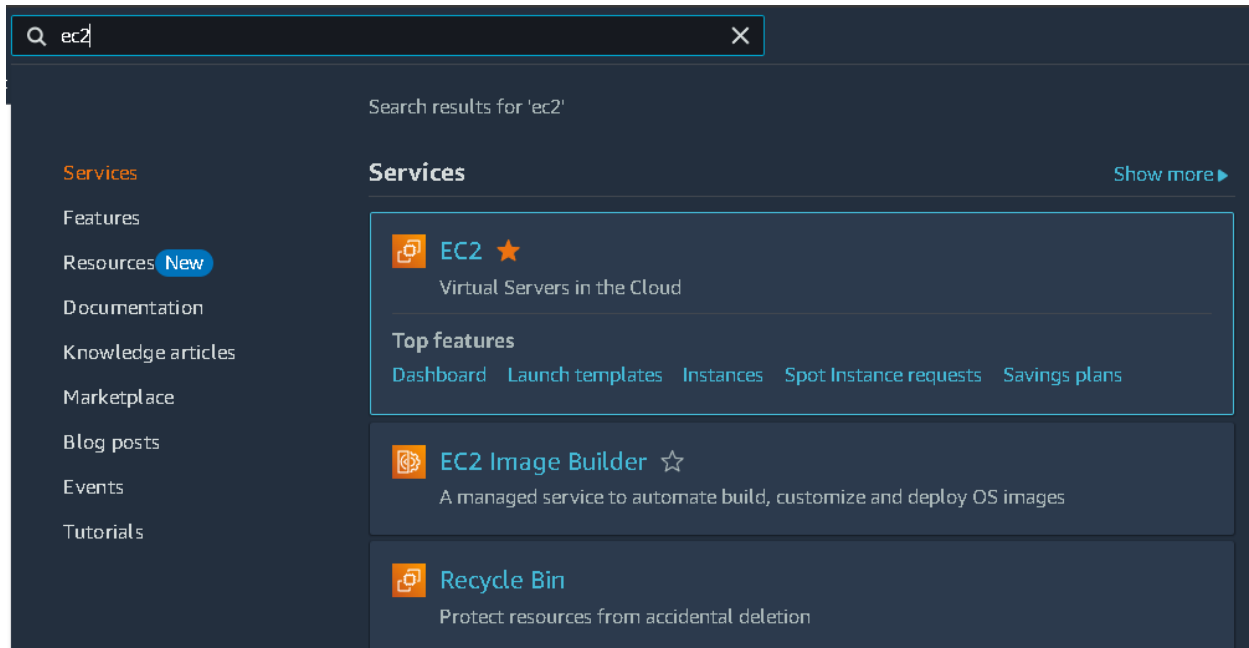
Key Features

- **Event-Driven Architecture:** The system leverages an event-driven approach where the AWS Lambda function is automatically triggered by events, such as the uploading of a text file to an S3 bucket. This eliminates the need for manual intervention, ensuring a seamless workflow.
- **Serverless Computing:** By using AWS Lambda, the solution operates without the need to provision or manage server infrastructure. This allows developers to focus on writing code rather than worrying about server management, leading to increased productivity and reduced operational costs.
- **Scalability:** The serverless architecture automatically scales based on the number of file uploads, handling variable workloads efficiently. This means that the system can accommodate bursts of activity without requiring pre-planned scaling strategies.
- **Real-Time Logging:** As soon as a file is uploaded to the S3 bucket, the Lambda function reads its contents and logs the information in real-time. This immediate logging capability enhances the responsiveness of applications that depend on timely data processing.
- **Integration with AWS Services:** The solution can be easily integrated with other AWS services, such as Amazon CloudWatch for monitoring and alerting, Amazon SNS for notifications, and Amazon DynamoDB for data storage. This allows for the creation of a robust data processing pipeline.
- **Cost Efficiency:** The pay-as-you-go pricing model of AWS Lambda ensures that users only pay for the compute time used during the execution of the logging function. This model is particularly advantageous for applications with sporadic workloads, as it minimizes idle resource costs.
- **Enhanced Monitoring and Troubleshooting:** With integration into Amazon CloudWatch, users can monitor the performance of the Lambda function and track logs generated during execution. This makes it easier to troubleshoot issues and maintain operational visibility.

STEPS:

1. Launch an EC2 Instance

1.1 Login to AWS Console and go to EC2 service.



1.2 Click on "Launch Instance".

- AML: 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).

Aryan Dangat D15A

Services

Search

[Alt+S]

N. Virginia

AryanH20Dangat

EC2 > ... > Launch an instance

Launch an instance

Info

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

Info

Name

aryanlambda

Add additional tags

Application and OS Images (Amazon Machine Image)

Info

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

Quick Start

Amazon Linux

macOS

Ubuntu

Windows

Red Hat

SUSE Linux

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Summary

Number of instances

Info

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.6.2...read more

ami-0eb21ccaff8cd886

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 usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 50 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GiB of bandwidth to the internet.

Cancel

Launch instance

Services

Search

[Alt+S]

N. Virginia

AryanH20Dangat

Launch an instance

Info

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

ary

Create new key pair

Network settings

Info

Network

Info

vpc-019ba79f9984faa38

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-1' 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

Summary

Number of instances

Info

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.6.2...read more

ami-0eb21ccaff8cd886

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

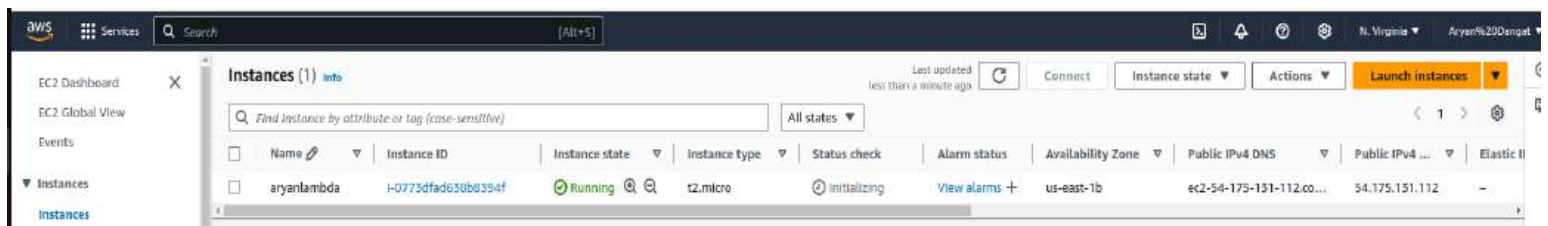
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 50 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GiB of bandwidth to the internet.

Cancel

Launch instance

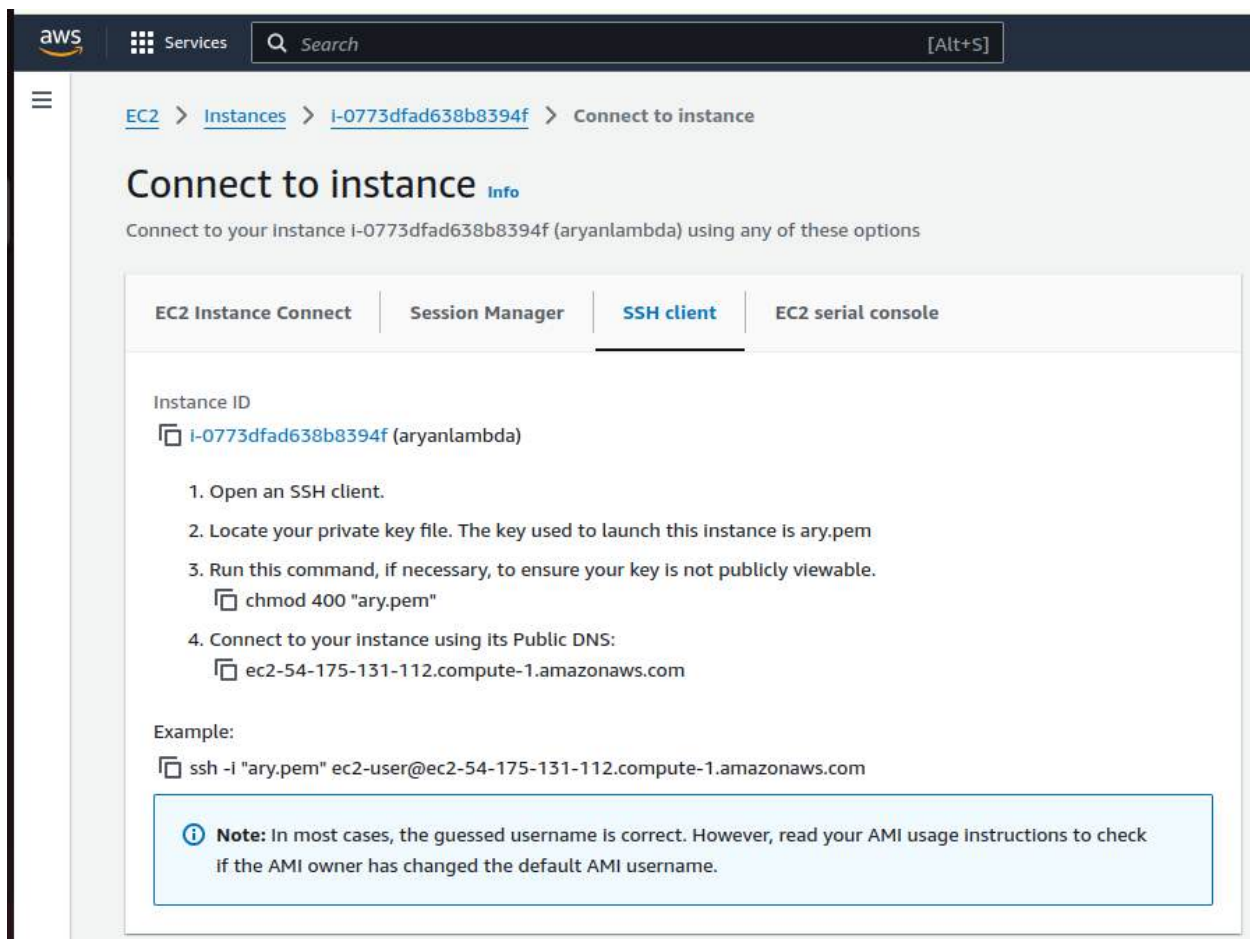
Aryan Dangat D15A

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>
```



- 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
 AKIA4QPHHSOK5AD6PEO4	 ***** Show

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

Download .csv file

Done

3. Install AWS CLI and Configure EC2

3.1 Update packages and install AWS CLI:

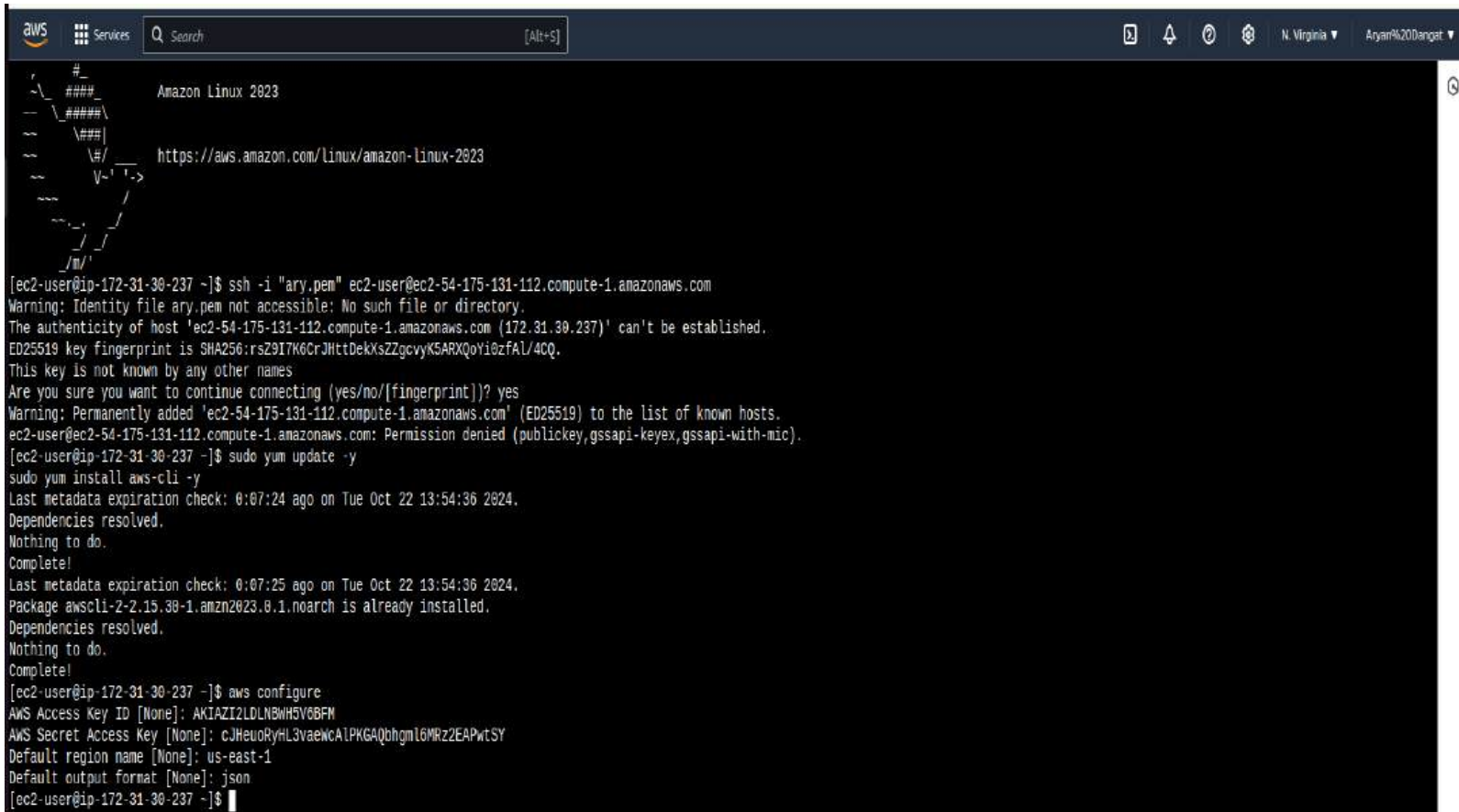
```
sudo yum update -y
sudo yum install aws-cli -y
```

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**



```
#_
_###_ Amazon Linux 2023
_###_
_###_ https://aws.amazon.com/linux/amazon-linux-2023
_###_

[ec2-user@ip-172-31-30-237 ~]$ ssh -i "ary.pem" ec2-user@ec2-54-175-131-112.compute-1.amazonaws.com
Warning: Identity file ary.pem not accessible: No such file or directory.
The authenticity of host 'ec2-54-175-131-112.compute-1.amazonaws.com (172.31.30.237)' can't be established.
ED25519 key fingerprint is SHA256:rs29I7K6CrJHttDekXsZzgcvyKSARXQoYi0zfAL/4CQ.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-175-131-112.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
ec2-user@ec2-54-175-131-112.compute-1.amazonaws.com: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-172-31-30-237 ~]$ sudo yum update -y
sudo yum install aws-cli -y
Last metadata expiration check: 0:07:24 ago on Tue Oct 22 13:54:36 2024.
Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 0:07:25 ago on Tue Oct 22 13:54:36 2024.
Package awscli-2.2.15.30-1.amzn2023.0.1.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-30-237 ~]$ aws configure
AWS Access Key ID [None]: AKIAZI2LDLNBWHSV6BFM
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
[ec2-user@ip-172-31-30-237 ~]$
```

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

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



```
[ec2-user@ip-172-31-33-47 ~]$ sudo yum install python3 -y
sudo yum install python3-pip -y
Last metadata expiration check: 0:26:12 ago on Sun Oct 20 11:13:19 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: 0:26:12 ago on Sun Oct 20 11:13:19 2024.
Dependencies resolved.
=====
Package                Architecture      Version           Repository        Size
=====
Installing:
python3-pip             noarch            21.3.1-2.amzn2023.0.8  amazonlinux      1.8 M
Installing weak dependencies:
libxcrypt-compat        x86_64            4.4.33-7.amzn2023    amazonlinux        92 k
=====
Transaction Summary
=====
Install 2 Packages

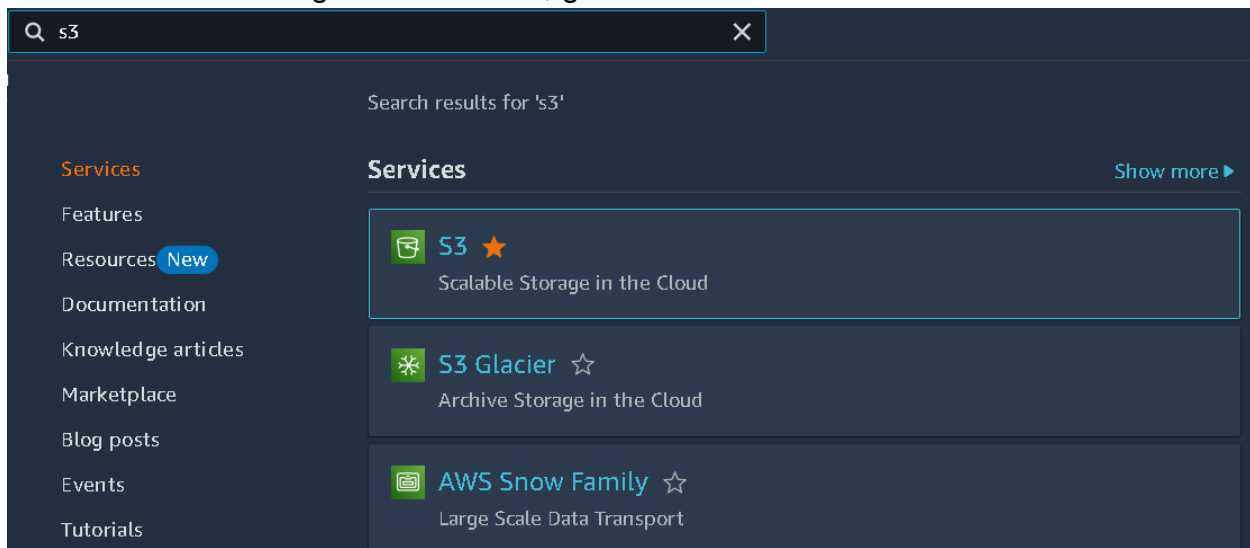
Total download size: 1.9 M
Installed size: 11 M
Downloading Packages:
(1/2): libxcrypt-compat-4.4.33-7.amzn2023.x86_64.rpm 1.5 MB/s | 92 kB  00:00
(2/2): python3-pip-21.3.1-2.amzn2023.0.8.noarch.rpm 15 MB/s | 1.8 MB  00:00
-----
Total                                           11 MB/s | 1.9 MB  00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :                                     1/1
  Installing     : libxcrypt-compat-4.4.33-7.amzn2023.x86_64 1/2
  Installing     : python3-pip-21.3.1-2.amzn2023.0.8.noarch 2/2
  Running scriptlet: python3-pip-21.3.1-2.amzn2023.0.8.noarch 2/2
  Verifying      : libxcrypt-compat-4.4.33-7.amzn2023.x86_64 1/2
  Verifying      : python3-pip-21.3.1-2.amzn2023.0.8.noarch 2/2

Installed:
  libxcrypt-compat-4.4.33-7.amzn2023.x86_64                python3-pip-21.3.1-2.amzn2023.0.8.noarch

Complete!
```

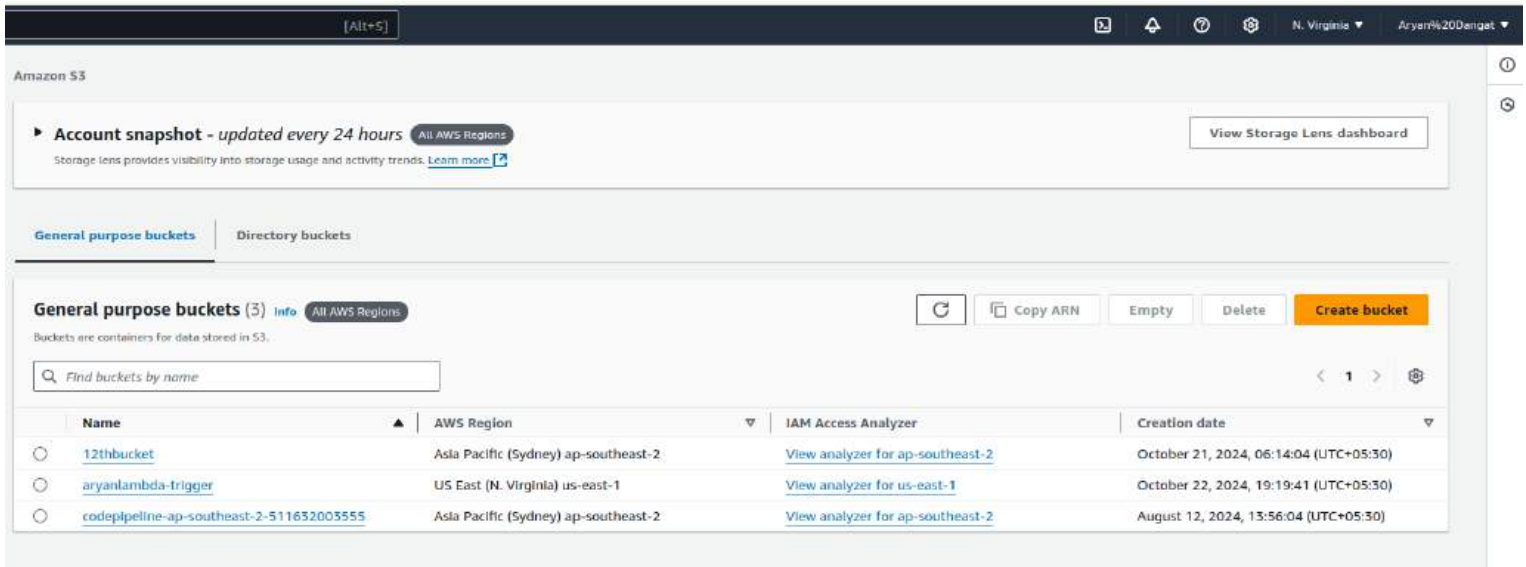
4. Create and S3 Bucket

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.



4.3 Create the bucket.

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
```

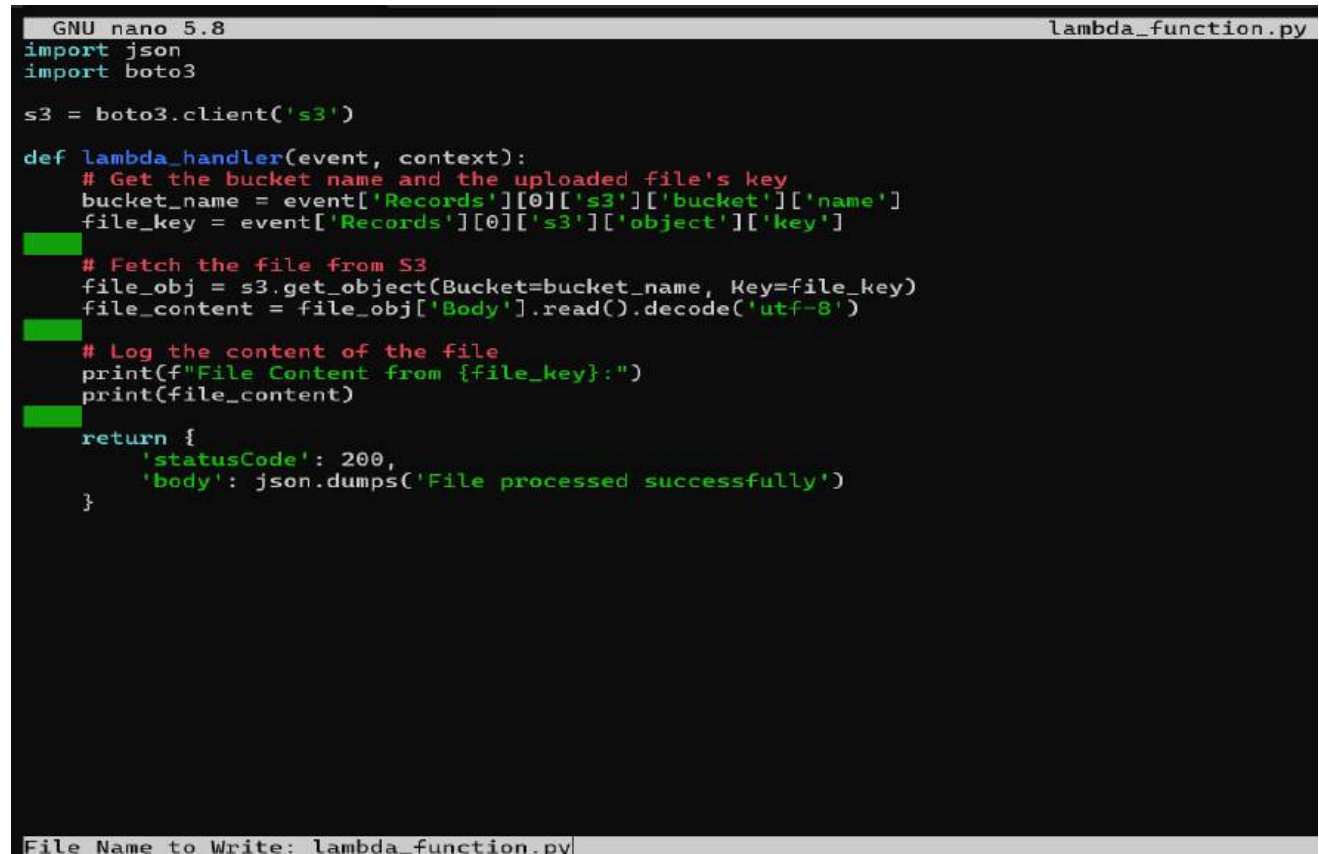
```
bucket_name = event['Records'][0]['s3']['bucket']['name']
file_key = event['Records'][0]['s3']['object']['key']

# 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
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

6. Deploy the Lambda function from EC2

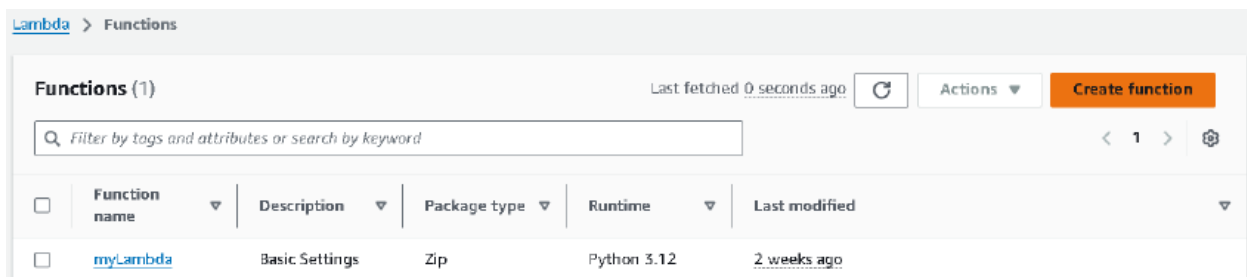
6.1 Package the Lambda function:

zip function.zip lambda_function.py

```
[ec2-user@ip-172-31-33-47 ~]$ zip function.zip lambda_function.py
adding: lambda_function.py (deflated 47%)
[ec2-user@ip-172-31-33-47 ~]$
```

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."

Create function [Info](#)

Choose one of the following options to create your function.

- ☒ **Author from scratch**
Start with a simple Hello World example.
- ☐ **Use a blueprint**
Build a Lambda application from sample code and configuration presets for common use cases.
- ☐ **Container image**
Select a container image to deploy for your function.

Basic information

Function name [Info](#)
Enter a name that describes the purpose of your function.

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
 [Refresh](#)

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.
☒ **x86_64**
☐ arm64

Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ **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

Permissions [Info](#)

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role

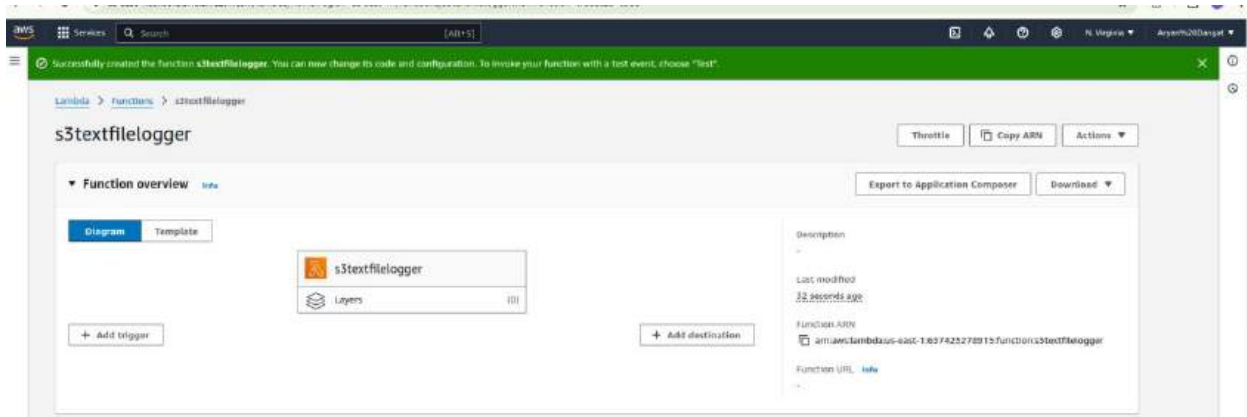
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

- ☒ **Create a new role with basic Lambda permissions**
- ☐ Use an existing role
- ☐ Create a new role from AWS policy templates

i 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-l6qnx3qp`, with permission to upload logs to Amazon CloudWatch Logs.

- Click **Create Function**.



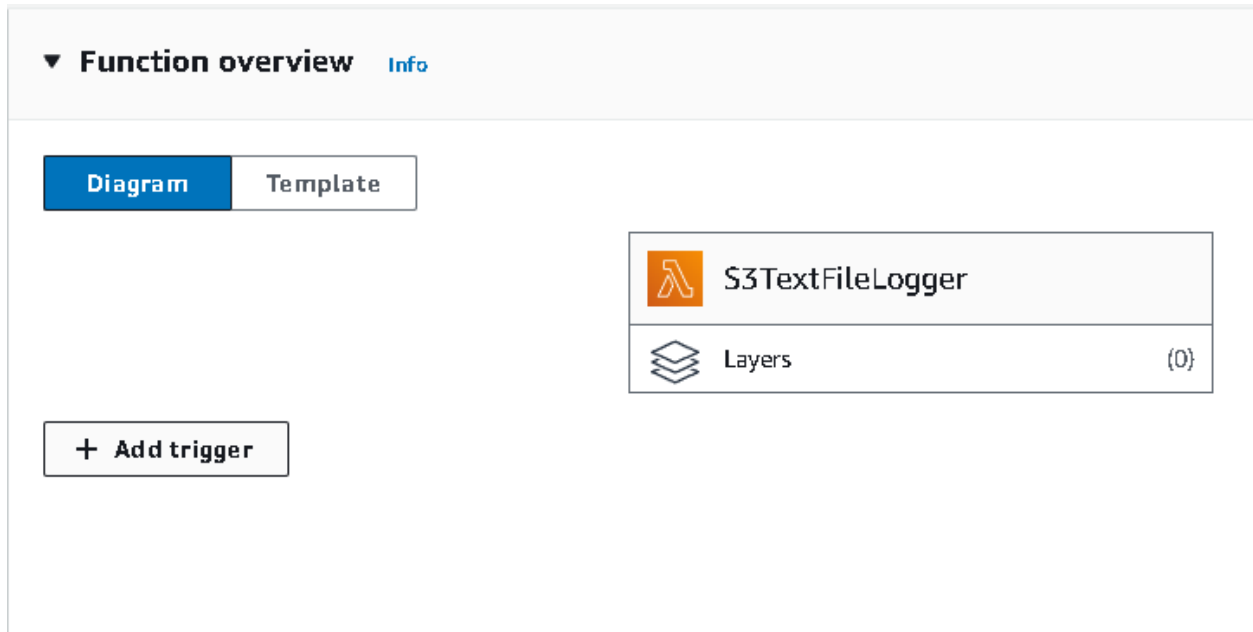
6.3 Upload the function code from EC2 using the AWS CLI:

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

```
[ec2-user@ip-172-31-33-47 ~]$ aws lambda update-function-code --function-name S3TextFileLogger --zip-file fileb://function.zip
{
  "FunctionName": "S3TextFileLogger",
  "FunctionArn": "arn:aws:lambda:eu-north-1:860015268757:function:S3TextFileLogger",
  "Runtime": "python3.12",
  "Role": "arn:aws:iam::860015268757:role/service-role/S3TextFileLogger-role-l6qnx3qp",
  "Handler": "lambda_function.lambda_handler",
  "CodeSize": 524,
  "Description": "",
  "Timeout": 3,
  "MemorySize": 128,
  "LastModified": "2024-10-20T12:13:22.000+0000",
  "CodeSha256": "r3bhnxP0TGjIFMX9rKsiULVb+470gIq4Fn8ufxx4Cuc=",
  "Version": "$LATEST",
  "TracingConfig": {
    "Mode": "PassThrough"
  },
  "RevisionId": "b5c0eee4-ec69-482f-9836-35d63c7752e8",
  "State": "Active",
  "LastUpdateStatus": "InProgress",
  "LastUpdateStatusReason": "The function is being created.",
  "LastUpdateStatusReasonCode": "Creating",
  "PackageType": "Zip",
  "Architectures": [
    "x86_64"
  ],
  "EphemeralStorage": {
    "Size": 512
  },
  "SnapStart": {
    "ApplyOn": "None",
    "OptimizationStatus": "Off"
  },
  "RuntimeVersionConfig": {
    "RuntimeVersionArn": "arn:aws:lambda:eu-north-1:runtime:188d9ca2e2714ff5637bd2bbe06ceb81ec3bc408a0f277dab104c14cd814b081"
  },
  "LoggingConfig": {
    "LogFormat": "Text",
    "LogGroup": "/aws/lambda/S3TextFileLogger"
  }
}
```

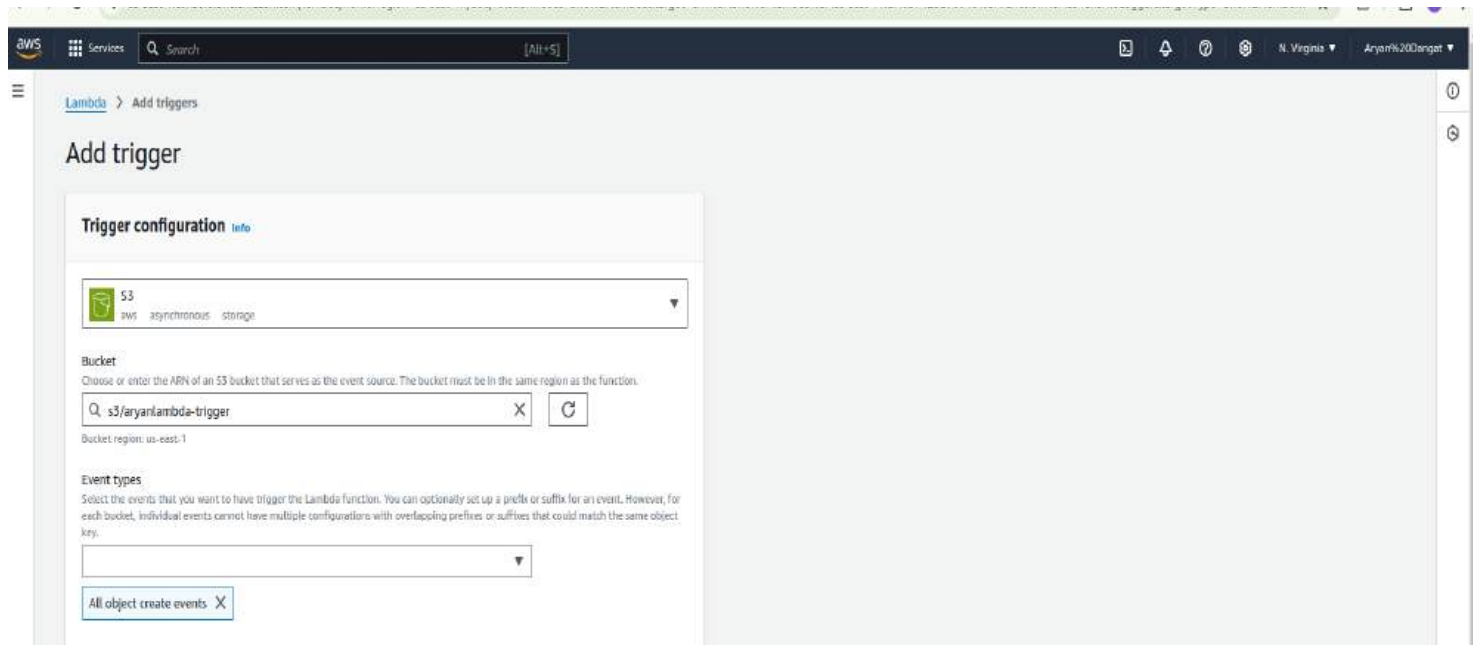
7. Configure S3 as the Trigger

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

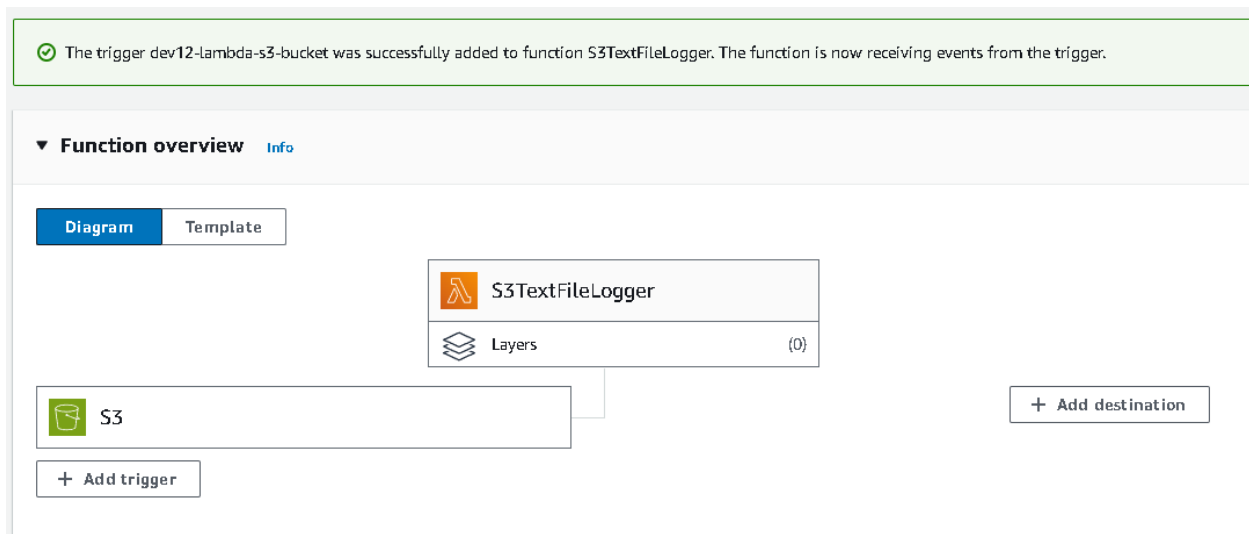


7.2 Choose **S3** as the trigger:

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



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



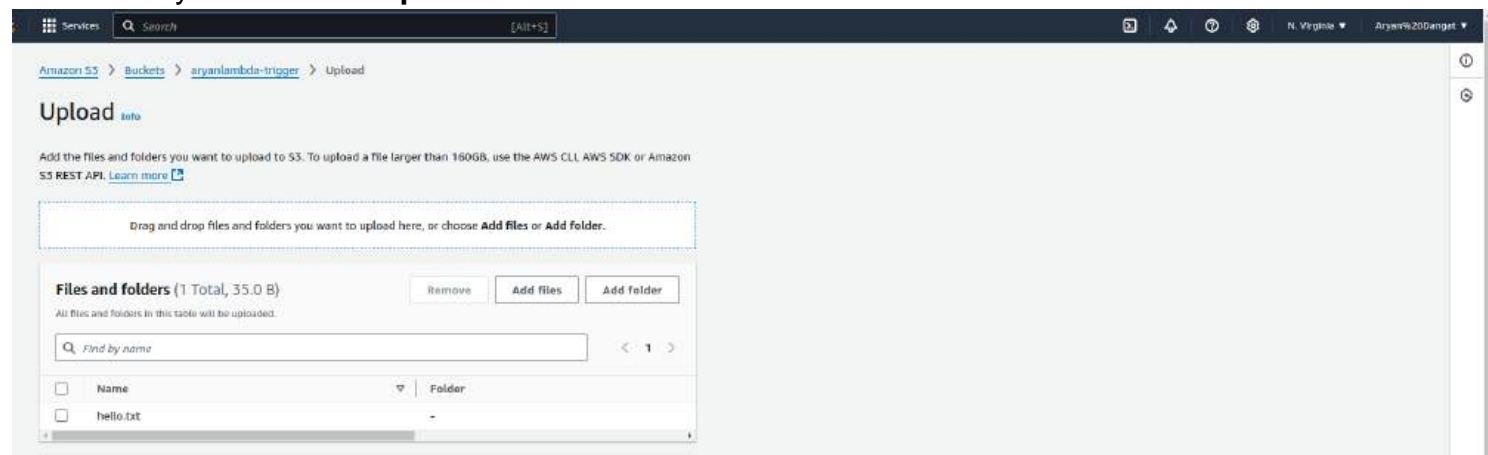
8. Upload a File and Test

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

```
aryan@aryan-Inspiron-3576:~/Downloads$ cat hello.txt
Hello hi there, I am Aryan Dangat.
aryan@aryan-Inspiron-3576:~/Downloads$
```

8.2 Upload a text file to your S3 bucket:

Go to **S3** > your bucket > **Up**



oad.

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

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 SDK 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, 41.0 B) Remove Add files Add folder

All files and folders in this table will be uploaded.

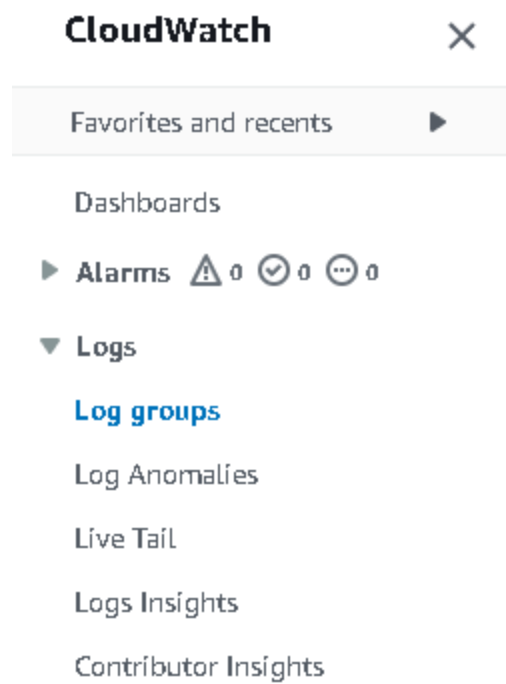
< 1 >

<input type="checkbox"/>	Name	Folder
<input type="checkbox"/>	hello.txt	-

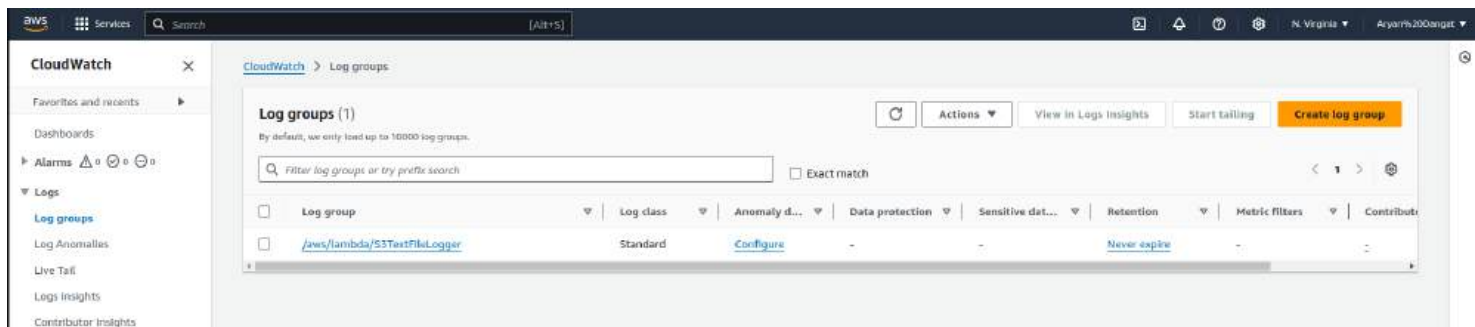
The Lambda function will automatically run when the file is uploaded.

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.

Aryan Dangat D15A

[Alt+S]

CloudWatch > Log groups > /aws/lambda/53TextFileLogger > All events

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

Clear 1m 30m 1h 12h Custom UTC timezone Display

Timestamp	Message	Log stream name
2024-10-22T14:25:53.677Z	INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lam...	2024/10/22/[SLATEST]a535494ede7c4e5b90d7330f661afed7
2024-10-22T14:25:54.216Z	START RequestId: 5388e5b5-924d-4bcf-98d1-334274ce4d35 Version: \$LATEST	2024/10/22/[SLATEST]a535494ede7c4e5b90d7330f661afed7
2024-10-22T14:25:54.858Z	LAMBDA_WARNING: Unhandled exception. The most likely cause is an issue in the...	2024/10/22/[SLATEST]a535494ede7c4e5b90d7330f661afed7
2024-10-22T14:25:54.858Z	[ERROR] ClientError: An error occurred (AccessDenied) when calling the GetObj...	2024/10/22/[SLATEST]a535494ede7c4e5b90d7330f661afed7
2024-10-22T14:25:54.878Z	END RequestId: 5388e5b5-924d-4bcf-98d1-334274ce4d35	2024/10/22/[SLATEST]a535494ede7c4e5b90d7330f661afed7
2024-10-22T14:25:54.878Z	REPORT RequestId: 5388e5b5-924d-4bcf-98d1-334274ce4d35 Duration: 645.56 ms B1...	2024/10/22/[SLATEST]a535494ede7c4e5b90d7330f661afed7
2024-10-22T14:26:59.034Z	START RequestId: 5388e5b5-924d-4bcf-98d1-334274ce4d35 Version: \$LATEST	2024/10/22/[SLATEST]a535494ede7c4e5b90d7330f661afed7
2024-10-22T14:26:59.539Z	LAMBDA_WARNING: Unhandled exception. The most likely cause is an issue in the...	2024/10/22/[SLATEST]a535494ede7c4e5b90d7330f661afed7
2024-10-22T14:26:59.539Z	[ERROR] ClientError: An error occurred (AccessDenied) when calling the GetObj...	2024/10/22/[SLATEST]a535494ede7c4e5b90d7330f661afed7
2024-10-22T14:26:59.559Z	END RequestId: 5388e5b5-924d-4bcf-98d1-334274ce4d35	2024/10/22/[SLATEST]a535494ede7c4e5b90d7330f661afed7
2024-10-22T14:26:59.559Z	REPORT RequestId: 5388e5b5-924d-4bcf-98d1-334274ce4d35 Duration: 525.48 ms B1...	2024/10/22/[SLATEST]a535494ede7c4e5b90d7330f661afed7

Edit the permissions of the s3 bucket to rectify the access denied problem.

[Alt+S]

IAM > Roles

Roles (10)

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Search

Role name	Trusted entities	Last activity
aryan	AWS Service: ec2	71 days ago
AWSCodePipelineServiceRole-ap-southeast-2-AD-CICD	AWS Service: codepipeline	71 days ago
AWSServiceRoleForAmazonSSM	AWS Service: ssm (Service-Linked Ro	-
AWSServiceRoleForSupport	AWS Service: support (Service-Linker	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service	-
CodeDeploy	AWS Service: codedeploy	71 days ago
pract1-role-tz6ojvs3	AWS Service: lambda	4 days ago
Pract11-role-pv1k413d	AWS Service: lambda	Yesterday
s3textfilelogger-role-9rgxejp0	AWS Service: lambda	-
S3TextFileLogger-role-hjfpvvp	AWS Service: lambda	44 minutes ago

Aryan Dangat D15A

[Alt+S]

Global ▾Aryan Dangat

Creation date
October 22, 2024, 19:47 (UTC+05:30)

ARN
arn:aws:iam:637423278915:role/service-role/S3TextFileLogger-role-hjfqpvvp

Last activity
44 minutes ago

Maximum session duration
1 hour

Permissions

Trust relationships

Tags

Last Accessed

Revoke sessions

Permissions policies (2) Info

⌂ Simulate ⌂ Remove Add permissions ▾

You can attach up to 10 managed policies.

Q Search

Filter by Type
All types ▾

< 1 > ⚙

<input type="checkbox"/>	Policy name <a>🔗	Type	Attached entities
<input type="checkbox"/>	AmazonS3FullAccess	AWS managed	1
<input type="checkbox"/>	AWSLambdaBasicExecutionRole-6a02ca...	Customer managed	1

[Alt+S]

N. Virginia ▾Aryan Dangat ▾

Q Filter events - press enter to search

Clear 1m 30m 1h 12h Custom 📅

Local timezone ▾

Display ▾ ⚙

▶ Timestamp | Message

No older events at this moment. [Retry](#)

▼

2024-10-22T20:37:55.799+05:30

INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:188d9ca2e2714ff5637bd2b...

INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:188d9ca2e2714ff5637bd2b...

▼

2024-10-22T20:37:56.310+05:30

START RequestId: 0d6ba82b-76a2-43a6-b330-373cf8d818da Version: \$LATEST

START RequestId: 0d6ba82b-76a2-43a6-b330-373cf8d818da Version: \$LATEST

▼

2024-10-22T20:37:56.859+05:30

File Content from hello.txt:

File Content from hello.txt:

▼

2024-10-22T20:37:56.859+05:30

Hey Hi there, I am Aryan Dangat.

Hey Hi there, I am Aryan Dangat.

▼

2024-10-22T20:37:56.880+05:30

END RequestId: 0d6ba82b-76a2-43a6-b330-373cf8d818da

END RequestId: 0d6ba82b-76a2-43a6-b330-373cf8d818da

▶

2024-10-22T20:37:56.880+05:30

REPORT RequestId: 0d6ba82b-76a2-43a6-b330-373cf8d818da Duration: 569.54 ms Billed Duration: 570 ms Memory Size: 128 MB Max...

No newer events at this moment. Auto retry paused. [Resume](#)

Back to top ^

[Alt+S]

N. Virginia

Aryan Dangat

CloudWatch

>

Log groups

>

/aws/lambda/S3TextFileLogger

>

2024/10/22/[\$LATEST]c404e25e179c432186a6d7809255d424

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

Clear1m30m1h12hCustomLocal timezoneDisplay

Timestamp	Message
No older events at this moment. Retry	
2024-10-22T20:37:55.799+05:30	INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:108d9ca2e2714ff5637bd2bbe06c..
2024-10-22T20:37:56.310+05:30	START RequestId: 0d6ba82b-76a2-43a6-b330-373cf8d818da Version: \$LATEST
2024-10-22T20:37:56.859+05:30	File Content from hello.txt:
2024-10-22T20:37:56.859+05:30	Hey Hi there, I am Aryan Dangat.
2024-10-22T20:37:56.880+05:30	END RequestId: 0d6ba82b-76a2-43a6-b330-373cf8d818da
2024-10-22T20:37:56.880+05:30	REPORT RequestId: 0d6ba82b-76a2-43a6-b330-373cf8d818da Duration: 569.54 ms Billed Duration: 570 ms Memory Size: 128 MB Max Memo..
No newer events at this moment. Auto retry paused. Resume	