

Problem Statement: Create an AWS Lambda Function that takes a message from an event and stores it in the S3 Bucket as a text file

Project Overview:

1. A message is triggered – manually for now
2. AWS lambda function will process the message
3. The python function will store the message as a text file in the S3 Bucket

Pre-requisites:

1. Amazon Free Tier Account.
2. Knowledge on how to create IAM users and add permissions.

Step-By-Step Instructions:

Step 1:

1. Login into AWS console
2. Navigate to S3 bucket
3. Click on create bucket



4. Select General Purpose
5. Provide a unique bucket name
6. Leave the rest as default settings and click on create

Create bucket [info](#)

Buckets are containers for data stored in S3.

General configuration [info](#)

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 independently store objects across multiple Availability Zones.

☐ **Directory**
Recommended for time-series 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](#)

mybucket15032020

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](#) [info](#)

Copy settings from existing bucket [optional](#)
Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: <ID>/bucketname

Object Ownership [info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

☒ **ACLs disabled (recommended)**
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

☐ **ACLs enabled**
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership
Bucket owner enforced

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or its objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#) [info](#)

☒ **Block all public access**

When you turn on all four settings below, each of the following settings are independent of one another:

- ☐ Block public access to buckets and objects granted through new access control lists (ACLs)
- ☐ Block public access to buckets and objects granted through new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using 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 access policies
- ☐ Block public access to buckets and objects granted through any public bucket or access point policies
- ☐ Block public and cross-account access to buckets and objects through any public bucket or access point policies

Block all public access to buckets and objects granted through any public bucket or access point policies

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#) [info](#)

Bucket Versioning

☒ **Enable**

☐ **Disable**

Tags [optional](#) [info](#)

You can use bucket tags to track storage costs and organize buckets. [Learn more](#) [info](#)

No tags associated with this bucket.

[Add tag](#)

Default encryption [info](#)

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type [info](#)

☒ **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)**

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

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](#) [info](#)

☐ **Disable**

☒ **Enable**

Advanced settings

[info](#) After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

[Cancel](#) [Create bucket](#)

7. The bucket is created successfully

Step 2:

8. Navigate to IAM and click on create a role

Select trusted entity [info](#)

Trusted entity type

☒ **AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

☐ **AWS account**
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

☐ **Web identity**
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

☐ **SAML 2.0 federation**
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

☐ **Custom trust policy**
Create a custom trust policy to enable others to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

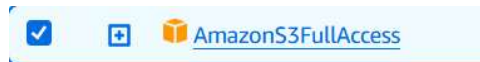
Lambda

Choose a use case for the specified service.

Use case

☒ **Lambda**
Allows Lambda functions to call AWS services on your behalf.

9. Locate the permission **AmazonS3FullAccess**



10. Click on Next and provide a role name
11. Create a role

Step 3:

12. Now we will create a lambda function using python
13. Click on create function
14. Select **Author from scratch**
15. Provide a name to the function
16. Select runtime as python from the dropdown
17. Now attach the IAM role created in previous step
18. Expand change default execution role
19. Select use existing role
20. From the drop down select the IAM role
21. Click on create function

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

Enter a name that describes the purpose of your function.

storemessagesetos3

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.

Python 3.11

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

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.

View the LambdaS3WriteRole role on the IAM console.

▼ Additional Configurations

Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

☐ Enable Code signing

info

Use code signing configurations to ensure that the code has been signed by an approved source and has not been altered since signing.

☐ Enable encryption with an AWS KMS customer managed key

info

By default, Lambda encrypts the .zip file archive using an AWS owned key.

☐ Enable function URL

info

Use function URLs to assign HTTP(S) endpoints to your Lambda functions.

☐ Enable tags

info

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources, track your AWS costs, and enforce attribute-based access control.

☐ Enable VPC

info

Connect your function to a VPC to access private resources during invocation.

Cancel

Create function

Step 4:

22. In the function code editor past the code.

s3upload.py

23. The code will be available in the GitHub for download.

```

1  import boto3 # AWS SDK for Python (used to interact with AWS services)
2  import datetime # Module to handle timestamps for unique file names
3
4  # Initialize the S3 client
5  s3 = boto3.client('s3')
6
7  # Define the name of the S3 bucket where messages will be stored
8  BUCKET_NAME = "your-bucket-name" # Replace with your actual bucket name
9
10 def lambda_handler(event, context):
11     """
12     AWS Lambda function to store a message in an S3 bucket as a text file.
13     This function gets triggered manually or via an event (future use case).
14
15     Parameters:
16     event (dict): Event data passed to the Lambda function (not used in this basic version).
17     context (object): Metadata about the function execution (not used here).
18
19     Returns:
20     dict: Response containing HTTP status code and confirmation message.
21
22     """
23     # Message to be stored in S3 (Static for now; can be made dynamic later)
24     message = "Hello, this is a test message!"
25
26     # Generate a unique timestamp for the file name to avoid overwriting existing files
27     timestamp = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
28
29     # Define the file name where the message will be saved inside the "messages/" folder
30     file_name = f"messages/message_{timestamp}.txt"
31
32     # Upload the message to the specified S3 bucket
33     s3.put_object(
34         Bucket=BUCKET_NAME, # The target S3 bucket
35         Key=file_name, # The path (folder + file name)
36         Body=message # The actual message content to be stored in the file
37     )
38
39     # Return a success response
40     return {
41         "statusCode": 200,
42         "body": f"Message stored in {file_name}" # Confirmation message
43     }

```

Step 5:

24. Within the function, test the code and check if the messages are stored in the S3 bucket

25. Click on deploy

26. Click on Test and configure test event



27. Click on Create New Event

28. Provide an Event Name

29. In the template select Hello World

30. In the Event JSON, replace the code

31. Click on Invoke

32. In the execution results the response will be 200 Status Code

Configure test event

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

☒ Create new event ☐ Edit saved event

Event name

TestingUploadtoS3

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private ☐ Shareable

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

hello-world

Event JSON

```
1 {
2   "message": "Hello, this is a test message!"
3 }
4
```

[Format JSON](#)

4:1 JSON Spaces: 2

[Cancel](#) [Invoke](#) [Save](#)

Test Event Name

Testing

Response

```
{
  "statusCode": 200,
  "body": "Message stored in messages/message_2025-03-12_07-46-48.txt"
}
```

Step 6:

33. Navigate to S3 Bucket

34. You will find a object called messages



35. Click on messages and text file is uploaded



36. Download the message and the content is available as mentioned above. **This is a manual process**, and it can be **done dynamic using the event.get method**.

Step 7:

37. Replace the single line code in the function as shown

```
# Message to be stored in S3 (Static for now; can be made dynamic later)
#message = "Hello, this is a test message!"
message = event.get("message", "Default Message")
```

38. Deploy the code again

39. Create a new test event and in the event json give any message and click on invoke. It will be stored as text files in S3 as shown

```
Hello, this is a test message123456789!
```