

AWS Lambda and Events

1. Now you need to create a function in Lambda with Python as your runtime environment.
2. Now in the permission section, click on use an existing role and choose your role. After that click on create function.

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.

[demo-lambda-role](#) [View the demo-lambda-role role](#) [on the IAM console.](#)

3. Once your function is created then scroll down and write code. Do not deploy the code.

```
import json
import boto3

def lambda_handler(event, context):
    bucketName= event['Records'][0]['s3']['bucket']['name']
    objectKey=event['Records'][0]['s3']['object']['key']
    objectSize=event['Records'][0]['s3']['object']['size']
    print(bucketName)
    print(objectKey)
    print(objectSize)
```

Code source [Info](#)

File Edit Find View Go Tools Window Test Deploy Changes not deployed

Go to Anything (Ctrl-P) Environment

demo-example / [lambda_function.py](#)

```
1 import json
2 import boto3
3
4 def lambda_handler(event, context):
5     bucketName= event['Records'][0]['s3']['bucket']['name']
6     objectKey=event['Records'][0]['s3']['object']['key']
7     objectSize=event['Records'][0]['s3']['object']['size']
8     print(bucketName)
9     print(objectKey)
10    print(objectSize)
```

4. After that you need to navigate to S3. Choose any bucket of your choice and then open its properties. Then scroll to Event notifications.
5. Now click on create event notification.

The screenshot shows the 'Event notifications' section of the AWS S3 console. At the top right are 'Edit', 'Delete', and 'Create event notification' buttons. Below is a table with columns 'Name', 'Event types', 'Filters', 'Destination type', and 'Destination'. A message at the top says 'No event notifications' and 'Choose Create event notification to be notified when a specific event occurs.' A 'Create event notification' button is highlighted with a red box.

6. Here you just need to give the event name.

The screenshot shows the 'General configuration' step. It includes fields for 'Event name' (containing 'demo-s3-event'), 'Prefix - optional' (containing 'images/'), and 'Suffix - optional' (containing '.jpg').

7. Then in event types just select PUT and POST. After that scroll down to the bottom.

The screenshot shows the 'Event types' configuration. It lists 'Object creation' events. Under 'Object creation', 'All object create events' is listed with 's3:ObjectCreated:*'. Next to it are two checked boxes: 'Put' (with sub-type 's3:ObjectCreated:Put') and 'Post' (with sub-type 's3:ObjectCreated:Post').

8. Now choose your destination as Lambda Function. Then you need to specify the Lambda function.
9. For that you just need to choose your Lambda function. Then click on Save changes.

Destination
Choose a destination to publish the event. [Learn more](#)

- Lambda function**
Run a Lambda function script based on S3 events.
- SNS topic**
Fanout messages to systems for parallel processing or directly to people.
- SQS queue**
Send notifications to an SQS queue to be read by a server.

Specify Lambda function

- Choose from your Lambda functions
- Enter Lambda function ARN

Lambda function

demo-example

Cancel Save changes

10. After that you can come back to Lambda function and deploy your code.
11. Once your code is deployed then you need to go back to S3 and quickly upload any file in that.

Amazon S3 > Buckets > demouser1221

demouser1221 [Info](#)

Objects (1) [Info](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

| Name | Type | Last modified | Size | Storage class |
|---------|------|---|---------|---------------|
| Code.py | py | February 11, 2024, 20:06:42 (UTC+05:30) | 270.0 B | Standard |

12. Now to check that your Lambda function worked or not.
13. Go back to lambda and the navigate to Monitor. There you need to click on view CloudWatch logs.

Code | Test | **Monitor** | Configuration | Aliases | Versions

Monitor [Info](#)

[View CloudWatch logs](#) [View X-Ray traces](#) [View Lambda Insights](#)

Filter metrics by [Function](#)

14. In CloudWatch log groups you will see some logs.
15. But you need to open the latest one first.

| Log streams | Tags | Anomaly detection | Metric filters | Subscription filters | Contributor Insights | Data protection |
|---|---------------------------------------|--|---|----------------------|----------------------|-----------------|
| Log streams (2) | | | | | | |
| <input type="button" value="C"/> | <input type="button" value="Delete"/> | <input type="button" value="Create log stream"/> | <input type="button" value="Search all log streams"/> | | | |
| <input type="text"/> Filter log streams or try prefix search | <input type="checkbox"/> Exact match | <input type="checkbox"/> Show expired | Info | < 1 > | | |
| <input type="checkbox"/> Log stream | Last event time | | | | | |
| <input type="checkbox"/> 2024/02/11/[\${LATEST}]d39bb95702f745f7993bf308d60aeb82 | 2024-02-11 20:06:43 (UTC+05:30) | | | | | |
| <input type="checkbox"/> 2024/02/11/[\${LATEST}]5f0c53247b6549fcad0f5c2edff6c0c4d | 2024-02-11 19:19:09 (UTC+05:30) | | | | | |

16. Now based on your code that you deployed on Lambda it will generate the output.
 17. Like here you can see that name of bucket and the name of object in it.

| 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 | |
| <input type="button" value="C"/> | <input type="button" value="Actions"/> |
| <input type="button" value="Start tailing"/> | <input type="button" value="Create metric filter"/> |
| <input type="text"/> Filter events | <input type="button" value="Clear"/> <input type="button" value="1m"/> <input type="button" value="30m"/> <input type="button" value="1h"/> <input type="button" value="12h"/> <input type="button" value="Custom"/> <input type="button" value="Local timezone"/> <input type="button" value="Display"/> <input type="button" value=""/> |
| ▶ | Timestamp |
| | Message |
| | No older events at this moment. Retry |
| ▶ | 2024-02-11T20:06:43.383+05:30 INIT_START Runtime Version: python:3.12.v18 Runtime Version ARN: arn:aws:lambda:ap-south-1::runtime:776a3759221679a634181f858... |
| ▶ | 2024-02-11T20:06:43.665+05:30 START RequestId: 0643ae75-9af1-45ef-8153-b8ba05920f03 Version: \${LATEST} |
| ▶ | 2024-02-11T20:06:43.666+05:30 demouser1221 |
| ▶ | 2024-02-11T20:06:43.666+05:30 Code.py |
| ▶ | 2024-02-11T20:06:43.666+05:30 270 |
| ▶ | 2024-02-11T20:06:43.681+05:30 END RequestId: 0643ae75-9af1-45ef-8153-b8ba05920f03 |
| ▶ | 2024-02-11T20:06:43.681+05:30 REPORT RequestId: 0643ae75-9af1-45ef-8153-b8ba05920f03 Duration: 15.97 ms Billed Duration: 16 ms Memory Size: 128 MB Max Memo... |
| | No newer events at this moment. Auto retry paused . Resume |