

- Lambda Python
  - Use the designer
    - Invoke the Lambda function
  - Handler
    - 1) AWS Lambda Function Handler in Python
  - AWS Lambda Function **event** in Python
    - 1.event object with Ad-Hoc Lambda Trigger
    - 2.event object with AWS Service Trigger
  - 1b) AWS Lambda **context** Object in Python
    - Context Properties

## Lambda Python

- Navigate to **AWS Lambda Service** and select **Create Function**
  - For Function name, enter **my-lambda-function**.
  - For Runtime, confirm that **Python 3.9** is selected.
- Create a Lambda function with default function code using AWS Console.
- Lambda creates a Python function and an **execution role** that grants the function permission to upload logs to CloudWatch Log Group.
- Choose **Create function**. AWS will create a default file **lambda\_function.py**

--

- Default Lambda Function Python Code will be as below:

```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

--

- To Check the IAM Role created, navigate to **Configuration > Permissions > IAM Role**.
- The Trust Relationship document for this IAM Role will have below details.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```

        "Service": "lambda.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
}
]
}

```

--

- The IAM Policy attached to this IAM Role is as below:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:CreateLogGroup",
      "Resource": "arn:aws:logs:ap-south-1:ACCOUNT_NUMBER:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:ap-south-1:ACCOUNT_NUMBER:log-
group:/aws/lambda/test-new-lambda-python-function:*"
      ]
    }
  ]
}

```

- When Lambda Function will run, Cloudwatch Log Group Name Format will be :  
***/aws/lambda/<LAMBDA\_FUNCTION\_NAME>***
- The Lambda function assumes the execution role when you invoke your function, and uses the execution role to create credentials for the AWS SDK and to read data from event sources.

--

## Use the designer

- The Designer shows an overview of your function. You can use it to configure triggers, change configurations, add or modify code, create test events etc.
- Modify/add some python commands in the below.

```
import json
```

```
def lambda_handler(event, context):
    # TODO implement
    a=10
    b=20
    c=a+b
    print("Value of c is",c)
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

--

## Invoke the Lambda function

- To test the Lambda function using the console:
  - To invoke a function, choose **Test**, In the **Configure test event** dialog box, choose **Create new test event**. In Event template, Keep the default option. Enter an Event name and keep the format as JSON:

```
{
  "key1": "value1",
  "key2": "value2"
}
```

- The Code that is running, is executed in AWS Data Center in that specific Region.
- The above JSON is the Event passed to the Lambda Function when it is executed as a **Dictionary** Variable

--

The screenshot displays the AWS Lambda console interface. At the top, there's a 'Code source' tab and an 'Upload from' button. Below this is a menu bar with options: File, Edit, Find, View, Go, Tools, and Window. To the right of the menu are 'Test' and 'Deploy' buttons. The main area shows the 'Execution result' for a test event named 'test\_event'. The status is 'Succeeded', with 'Max memory used: 36 MB' and 'Time: 1.66 ms'. The 'Response' is a JSON object: `{ "statusCode": 200, "body": "\"Hello from Lambda!\"" }`. The 'Function Logs' section provides detailed execution information: `START RequestId: 22ed1a70-e8eb-4d72-85af-78667ef81da2 Version: $LATEST`, `Value of c is 30`, `END RequestId: 22ed1a70-e8eb-4d72-85af-78667ef81da2`, and `REPORT RequestId: 22ed1a70-e8eb-4d72-85af-78667ef81da2 Duration: 1.66 ms Billed Duration: 2 ms Memory Size: 256 MB M`. The 'Request ID' is `22ed1a70-e8eb-4d72-85af-78667ef81da2`.

- AWS Lambda Code Execution Logs will be stored in CW Log Group.
  - Navigate to **Monitor** > **View Logs in CloudWatch**

---

## Handler

### 1) AWS Lambda Function Handler in Python

- At the time you create a Lambda function, you specify a handler, which is a function in your code, that AWS Lambda can invoke when the service executes your code.
- The Lambda function handler name specified at the time you create a Lambda function is derived from the following:
  - the name of the Python file in which the Lambda handler function is located
  - the name of the Python handler function

```
lambda_function.lambda_handler
```

```
filename.functionName
```

--

General Syntax :

```
def handler_name(event, context):  
    return some_value
```

- If we modify the function name as **lambda\_handler\_test** and try to run the same.

```
{  
  "errorMessage": "Handler 'lambda_handler' missing on module 'lambda_function'",  
  "errorType": "Runtime.HandlerNotFound",  
  "requestId": "99d3dcf5-d5c9-4551-b95f-85644404d23e",  
  "stackTrace": []  
}
```

- Ensure the Function definition name and lambda handler property function name is same.

Runtime settings <a href="#">Info</a>			<a href="#">Edit</a>
Runtime Python 3.9	Handler <a href="#">Info</a> lambda_function_test.lambda_handler	Architecture <a href="#">Info</a> x86_64	

--

## AWS Lambda Function **event** in Python

- **event** – AWS Lambda uses this parameter to pass in event data to the handler. This parameter is usually of the Python **dict** type.

### 1.event object with Ad-Hoc Lambda Trigger

- When a Lambda Function is invoked manually using AWS Console by using **Test**, a custom Event can be passed similar to Test JSON as below :

```
{
  "key1": "value1",
  "key2": "value2",
  "key3": "value3"
}
```

- The above Event JSON is available as **event** variable inside the **lambda\_handler** function.
- Use **print(event)** inside Python Code to check the value of the **event** variable i.e Dictionary.

--

```
import json

def lambda_handler(event, context):
    print("event object is",event)
    print("event type is",type(event))
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

- Execution is as below:

```
START RequestId: a3e4281a-f1c3-4ff6-a421-32de60cc8c50 Version: $LATEST
event object is {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
event type is <class 'dict'>
END RequestId: a3e4281a-f1c3-4ff6-a421-32de60cc8c50
```

--

## 2.event object with AWS Service Trigger

- Lambda Function supports **Triggers/Invocation** from AWS Services.
- When a Lambda Function is invoked via a AWS Service, the **event** variable contains the content and structure of details of the trigger.
- When an AWS service invokes your function, the event structure varies by service.
- If a Lambda Function is **invoked by S3 Service**, the event data will be as below:

--

```

{
  'Records': [{
    'eventVersion': '2.1',
    'eventSource': 'aws:s3',
    'awsRegion': 'us-east-1',
    'eventTime': '2019-12-14T01:54:29.721Z',
    'eventName': 'ObjectCreated:Put',
    'userIdentity': {
      'principalId': 'A3HLTU27MENXZ1'
    },
    'requestParameters': {
      'sourceIPAddress': '111.225.11.82'
    },
    'responseElements': {
      'x-amz-request-id': 'BB4FFADF2A079713',
      'x-amz-id-2':
'QUopUd3JXVpE6W9UAD6cCa5q19CYbQt3Y7TyuDjhFPPeKGV717h07602Qsq9vxVntsQ9XxMfj0Y='
    },
    's3': {
      's3SchemaVersion': '1.0',
      'configurationId': 'ad5d40d4-8e21-4f13-afcd-4cdc65d1d157',
      'bucket': {
        'name': 's3bkt',
        'ownerIdentity': {
          'principalId': 'A3HLTU27MENXZ1'
        },
        'arn': 'arn:aws:s3:::s3bkt'
      },
      'object': {
        'key': 'AWS_KeyPairs/aws-NV.ppk',
        'size': 1464,
        'eTag': '809d9725c217af3ab4f70f52a0d5e08f',
        'versionId': 'DoSaRg91K2aGVesfp_ttE0QCa8ggnCyZ',
        'sequencer': '005DF440D599CFB897'
      }
    }
  ]
}

```

--

- To view the S3 Event Json as above, follow below steps:
  - Navigate to Amazon S3 bucket using the console, Create OR Use Existing S3 Bucket.
  - Go to **S3 Properties > Event Notifications > Create event notification**
    - Provide details about name, event type.

- Under Destination, select Lambda Function and provide the Lambda Function Name.

### Destination

**i** Before Amazon S3 can publish messages to a destination, you must grant the Amazon S3 principal the necessary permissions to call the relevant API to publish messages to an SNS topic, an SQS queue, or a Lambda function. [Learn more](#)

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

test-new-lambda-python-function ▼

[Cancel](#) [Save changes](#)


- Navigate back to the Lambda Function that was created.


[Lambda](#) > [Functions](#) > test-new-lambda-python-function


## test-new-lambda-python-function

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

▼ **Function overview** [Info](#)

 test-new-lambda-python-function

 Layers (0)


 S3

[+ Add trigger](#)

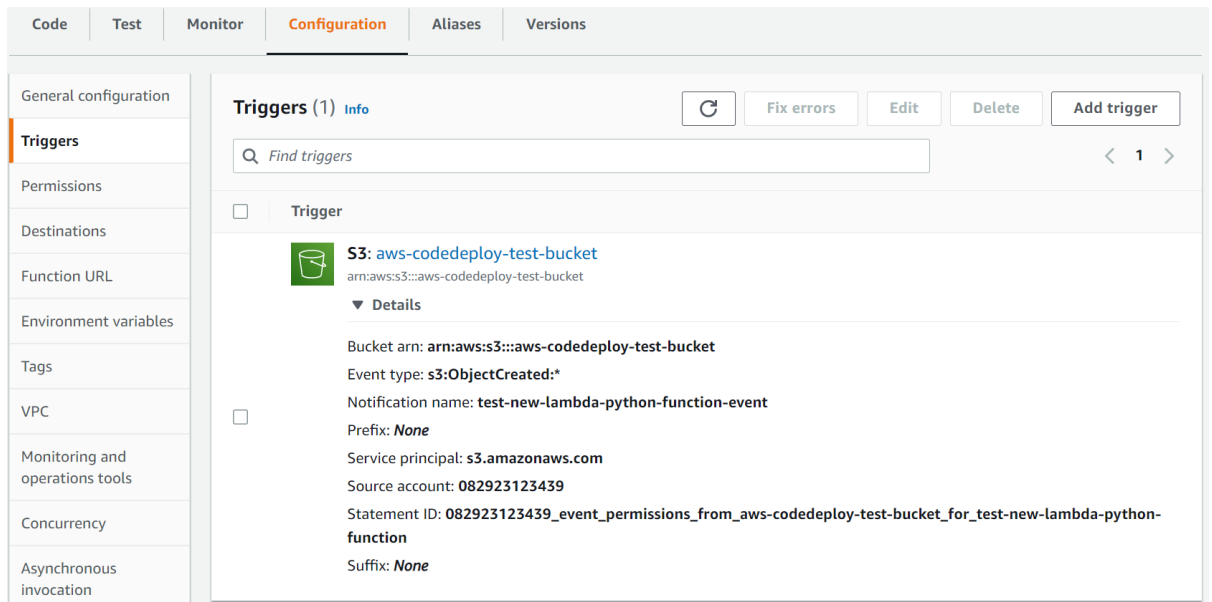
[+ Add destination](#)

Description  
-

Last modified  
9 minutes ago

Function ARN  
 arn:aws:lambda:ap-south-1:082923708139:func  
tion:test-new-lambda-python-function

Function URL [Info](#)  
-



- Lambda Function Trigger for S3 Bucket Event can be configured either from AWS Lambda Function Page or from S3 bucket Properties Page.
- Upload a file on S3 bucket and check for Execution logs for this event s3 trigger will be in cloudwatch log group.

## 1b) AWS Lambda `context` Object in Python

- `context` – AWS Lambda uses this parameter to provide runtime information to your handler.
- When Lambda runs your function, it passes a context object to the handler. This object provides methods and properties that provide information about the invocation, function, and execution environment.

--

### Context Properties

- `function_name` – The name of the Lambda function.
- `function_version` – The version of the function.
- `invoked_function_arn` – The Amazon Resource Name (ARN) used to invoke the function.
- `memory_limit_in_mb` – The amount of memory that's allocated for - the function.
- `aws_request_id` – The identifier of the invocation request.
- `log_group_name` – The log group for the function.
- `log_stream_name` – The log stream for the function instance.

--

```
import json,time

def lambda_handler(event, context):
    print("Lambda Function Version:", context.function_name)
    print("Lambda Function Version:", context.function_version)
    print("Resource ARN that invoked the current Function", context.invoked_function_arn)
```



```
    print("Memory allocated to this Lambda Function is :",  
context.memory_limit_in_mb)  
    print("Invokation request identifier :", context.aws_request_id)  
    print("Log Group Name:", context.log_group_name)  
    print("Log stream name:", context.log_stream_name)
```