



AWS Lambda – Extending Code

1. Now navigate to DynamoDB. There you need to create a table.

DynamoDB > Tables

Tables (0) [Info](#)

Find tables by table name Any tag key Any tag value Actions Delete Create table

Na... ▲ Status Partition key Sort key Indexes Deletion protection Read capacity mode Write capacity mode Total si

You have no tables in this account in this AWS Region.

Create table

2. Name this table. Then give the partition key. After that give the sort key. And then just create your table.

Table details [Info](#)

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.
 Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.)

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.
 String
1 to 255 characters and case sensitive.

Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.
 String
1 to 255 characters and case sensitive.

3. After that come back to AWS Lambda and there create a function or choose any previous function if you have.
4. There you need to change the code to this.

```
import json
import boto3

def lambda_handler(event, context):
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    object_key = event['Records'][0]['s3']['object']['key']
```

```

object_size = event['Records'][0]['s3']['object']['size']

print(bucket_name)
print(object_key)
print(object_size)

dynamodb = boto3.client('dynamodb', region_name='ap-south-1')

params = {
    'Item': {
        'BucketName': {'S': bucket_name},
        'ObjectKey': {'S': object_key},
        'ObjectSize': {'N': str(object_size)}
    },
    'TableName': 'S3BucketData'
}

response = dynamodb.put_item(**params)

return {
    'statusCode': 200,
    'body': json.dumps('Item added to DynamoDB table')
}

```



The screenshot shows the AWS Lambda function editor interface. The top navigation bar includes 'Tools', 'Window', 'Test' (which is selected), and 'Deploy'. Below the bar, there's a tab labeled 'lambda_function' with a dropdown arrow, 'Environment Var' with a plus sign, and a gear icon. The main area contains the Python code for the lambda function, with line numbers from 1 to 30 on the left. The status bar at the bottom right shows '1:12 Python Spaces: 4'.

```

1 import json
2 import boto3
3
4 def lambda_handler(event, context):
5     bucket_name = event['Records'][0]['s3']['bucket']['name']
6     object_key = event['Records'][0]['s3']['object']['key']
7     object_size = event['Records'][0]['s3']['object']['size']
8
9     print(bucket_name)
10    print(object_key)
11    print(object_size)
12
13    dynamodb = boto3.client('dynamodb', region_name='ap-south-1')
14
15    params = {
16        'Item': {
17            'BucketName': {'S': bucket_name},
18            'ObjectKey': {'S': object_key},
19            'ObjectSize': {'N': str(object_size)}
20        },
21        'TableName': 'S3BucketData'
22    }
23
24    response = dynamodb.put_item(**params)
25
26    return {
27        'statusCode': 200,
28        'body': json.dumps('Item added to DynamoDB table')
29    }
30

```

5. Now you need to go to IAM roles and check that you should have these policies attached to your Role.
6. This role should also be attached with your Lambda function.

Permissions policies (3) Info			
You can attach up to 10 managed policies.			
Filter by Type			
<input type="text"/> Search	All types		
<input type="checkbox"/>	Policy name ▾	Type	Attached entities ▾
<input type="checkbox"/>	AmazonDynamoDBFullAccess	AWS managed	3
<input type="checkbox"/>	AmazonS3ReadOnlyAccess	AWS managed	2
<input type="checkbox"/>	AWSLambdaBasicExecutionRole	AWS managed	2

7. After that navigate to S3 and open any bucket of your choice and open its properties then scroll down to Event Notifications.
8. Then in there you need to give it a name.

General configuration

Event name

Event name can contain up to 255 characters.

Prefix - *optional*

Limit the notifications to objects with key starting with specified characters.

Suffix - *optional*

Limit the notifications to objects with key ending with specified characters.

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

Event types

Specify at least one event for which you want to receive notifications. For each group, you can choose an event type for all events, or you can choose one or more individual events.

Object creation

<input type="checkbox"/> All object create events s3:ObjectCreated:*	<input checked="" type="checkbox"/> Put s3:ObjectCreated:Put
	<input checked="" type="checkbox"/> Post s3:ObjectCreated:Post

10. Now choose your destination as Lambda Function. Then you need to specify the Lambda function.
11. 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

12. Then you need to upload something in your S3 bucket.
13. Once you are done then go to DynamoDB open your table and click on Explore table items.
14. After that click on Run.
15. You will see that you have an item that DynamoDB has returned.
16. You have the bucket name, the object key and the object size.

S3BucketData Autopreview View table details

▼ Scan or query items

Scan **Query**

Select a table or index Table - S3BucketData Select attribute projection All attributes

▶ Filters

Run Reset

Completed. Read capacity units consumed: 0.5 X

Items returned (1) C Actions ▾ Create item ◀ 1 ▶ @ X

	BucketName (String)	ObjectKey (String)	ObjectSize	
<input type="checkbox"/>	demouser1222	Code.py	270	...

17. You are now consuming events from S3 via AWS Lambda and then taking the data within that event, those events that are being streamed onto the Lambda function and writing it onto DynamoDB.