



# Invoke Lambda Asynchronously Using Amazon S3 to Trigger Lambda

## 1. Lab Objective

- Implement a serverless architecture using Amazon S3, Lambda, and DynamoDB.
  - Trigger a Lambda function with an S3 event and persist file metadata into DynamoDB.
- 

## 2. Architecture Overview

- **Client Application:** Uploads objects into an Amazon S3 bucket.
  - **S3 Bucket:** Emits an event when an object is uploaded, triggering a Lambda function.
  - **Lambda Function:**
    - Processes the event asynchronously.
    - Saves the uploaded file's metadata into DynamoDB.
  - **DynamoDB:** Stores metadata of uploaded files.
- 

## 3. Process Flow

1. **Upload:** Client uploads an object to the S3 bucket.
2. **Trigger:** S3 generates an ObjectUploaded event to invoke the Lambda function.
3. **Lambda Execution:**
  - Receives the event.
  - Executes business logic to extract metadata from the uploaded file.
  - Stores the metadata in DynamoDB.
4. **Validation:** Check DynamoDB via the AWS Management Console to ensure metadata is saved.
5. **Asynchronous Nature:**
  - No direct response is sent back to the client application.
  - Data persistence is validated through the management console.



## To begin with the Lab:

1. Now we will start with creating the architecture for our lab.
2. First, we need to create a bucket. In your console, search for S3 and click on Create Bucket. Give a unique name to your bucket and then create it.

## General configuration

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 redundantly store  
objects across multiple Availability Zones.

Directory

Recommended for low-latency 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](#)

s3-bucket-12010321

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#) ↗

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: s3://bucket/prefix

- Now we need to create a lambda function. So, open lambda and click on create. Give a name to your function and choose runtime as Node.js.

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.

s3-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.

Node.js 20.x



**Architecture** [Info](#)

Choose the instruction set architecture you want for your function code.

x86\_64

arm64

- In the permission we will create a new role, choose Create a new role from AWS Policy templates give your role a name, and choose the same policy templates you can see below and create your 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

ⓘ Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

#### Role name

Enter a name for your new role.

Use only letters, numbers, hyphens, or underscores with no spaces.

#### Policy templates - optional Info

Choose one or more policy templates.

 Simple microservice permissions XDynamoDB Amazon S3 object read-only permissions XS3

5. Go to the code section in your function and add the 2<sup>nd</sup> line in your code as you can see below. This will help us to see the logs in CloudWatch.

```
console.log("EVENT: \n" + JSON.stringify(event, null, 2));
```

js index.mjs > [🔗] handler

```
1  ↴ export const handler = async (event) => {
2    ↳   console.log("EVENT: \n" + JSON.stringify(event, null, 2));
3    ↳   // TODO implement
4    ↴   const response = {
5      ↳     statusCode: 200,
6      ↳     body: JSON.stringify('Hello from Lambda!'),
7      ↳   };
8      ↳   return response;
9    ↳ };
10
```

6. Now we need to add a trigger, so click on the Add trigger button.

## s3-function

▼ Function overview [Info](#)

[Diagram](#) [Template](#)

 s3-function

 Layers (0)

[+ Add trigger](#)

[+ Add destination](#)

7. You need to add a trigger for S3 and in the bucket choose your newly created bucket and, in the event, types choose PUT. Then check the box for acknowledgment and click on Add.

### Trigger configuration [Info](#)

 S3  
aws asynchronous storage

**Bucket**  
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.

[X](#) [C](#)

Bucket region: us-east-1

**Event types**  
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual prefixes or suffixes that could match the same object key.

[▼](#)

[PUT](#) [X](#)

**Prefix - optional**  
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any [special characters](#) must be URL encoded.

8. So, now we will create our Dynamo DB table. Go to it and click on Create. You need to give a name to your table and give Partition key as “Key”.
9. Then scroll down, in the Table settings choose **Customize settings**, and for the **Read/Write capacity choose On-demand**.
10. Just create your table after choosing the above configurations.

## 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.

bucketobjects

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.

Key

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.

Enter the sort key name

String



1 to 255 characters and case sensitive.

11. Now we need to upload an object to our S3 bucket and check the CloudWatch logs for our lambda function to check whether it gets triggered or not.

12. As you can see below, I just uploaded an index.html file to our S3 bucket. Now we need to open the logs for our function.

s3-bucket-12010321 [Info](#)

Objects Properties Permissions Metrics Management Access Points

Objects (1) [Info](#) [C](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) Actions Create folder [Upload](#)

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](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	<a href="#">index.html</a>	html	November 19, 2024, 16:59:36 (UTC+05:30)	192.0 B	Standard

13. Here you can see that a log stream has been generated at the same time when uploaded an object to our S3 bucket.

Log streams Tags Anomaly detection Metric filters Subscription filters Contributor Insights Data protection

Log streams (1) [C](#) Delete Create log stream Search all log streams

Filter log streams or try prefix search  Exact match  Show expired [Info](#)

<input type="checkbox"/>	Log stream	Last event time
<input type="checkbox"/>	<a href="#">2024/11/19/[LATEST]a2bedb14468e4c32a10d2ae4712ba9d2</a>	2024-11-19 16:59:36 (UTC+05:30)

14. If you open the log event you will see that records have been created, it means that our lambda function gets triggered.

```

{
  "Records": [
    {
      "eventVersion": "2.1",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "2024-11-19T11:29:35.600Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "A13DFHTRP6I83Q"
      },
      "requestParameters": {
        "sourceIPAddress": "103.226.202.202"
      },
      "responseElements": {
        "x-amz-request-id": "G62X3BTVN5WFMCJH",
        "x-amz-id-2": "vC/MN0LCQ0Iw9VhHw80xTmeT1/DSyTLhn0i/9DQRsQNs0zwQaEHKdQrREz4Z/13uSHqr8bKj1IKu3Ib0NyLY0BwV150Ns0Wm"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "8d09a56d-8f56-46f8-b597-c05561c6610c",
        "bucket": {
          "name": "s3-bucket-12010321",
          "ownerIdentity": {
            "principalId": "A13DFHTRP6I83Q"
          },
          "arn": "arn:aws:s3:::s3-bucket-12010321"
        },
        "object": {
          "key": "index.html",
          "size": 192,
          "eTag": "4ce90c022f0dc7d6e3ef4790764c070",
          "sequencer": "00673C769F920A846E"
        }
      }
    }
  ]
}

```

- Now you need to open the Role attached to your Lambda function and add the S3 full access permission to it.

[Permissions](#) | [Trust relationships](#) | [Tags](#) | [Last Accessed](#) | [Revoke sessions](#)

**Permissions policies (4) [Info](#)**

You can attach up to 10 managed policies.

Filter by Type			
<input type="text" value="Search"/>	All types	< 1 >	
<input type="checkbox"/> Policy name <a href="#">AmazonS3FullAccess</a>	AWS managed	3	
<input type="checkbox"/> <a href="#">AWSLambdaBasicExecutionRole-d6e2a297...</a>	Customer managed	1	
<input type="checkbox"/> <a href="#">AWSLambdaMicroserviceExecutionRole-16...</a>	Customer managed	1	
<input type="checkbox"/> <a href="#">AWSLambdaS3ExecutionRole-7260fef9-12...</a>	Customer managed	1	

- Then you need to create a folder on your laptop and open it in VS Code.
- Here you need to create three files as you can see below and for the code, you can copy that from the code files folder you get with this document. Make sure to change the region and the name of your bucket and dynamo DB table.

FOLDERS: INVOKE LAMBDA

- ddbClient.js
- index.js
- s3Client.js

```

dbbClient.js X index.js s3Client.js
0: ddbClient.js > ...
1 import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
2 const REGION = "us-east-1";
3 const ddbClient = new DynamoDBClient({ region: REGION });
4 export { ddbClient };

```

18. Now we need to install the node modules for that we can open the terminal and run this command. After running this command you will see that a package.json file has been created.

```
npm init -y
```

```
PS D:\AWS Serveless\Invoke Lambda Asynchronously Using Amazon S3 to Trigger Lambda> npm init -y
Wrote to D:\AWS Serveless\Invoke Lambda Asynchronously Using Amazon S3 to Trigger Lambda\package.json:

{
  "name": "invoke-lambda-asynchronously-using-amazon-s3-to-trigger-lambda",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\"Error: no test specified\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

19. Open the package.json file and on the line, you need to add a type module to your file and save it.

```
package.json > abc type
1  [
2    "name": "invoke-lambda-asynchronously-using-amazon-s3-to-trigger-lambda",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "type": "module",|_
7      | Debug
8      "scripts": {
9        | "test": "echo \\"Error: no test specified\\" && exit 1"
10       },
11      "keywords": [],
12      "author": "",
13      "license": "ISC"
14]
```

20. Then you need to run the npm install command to install the AWS SDK to your folder. You will see that node modules have been installed inside your folder.

```
npm install @aws-sdk/client-s3
```

```
PS D:\AWS Serveless\Invoke Lambda Asynchronously Using Amazon S3 to Trigger Lambda> npm install @aws-sdk/client-s3
added 103 packages, and audited 104 packages in 3s
2 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
```

21. Also, inside your package.json file as well you will that the dependencies has been installed.

```
1 package.json > type
2   },
3   "keywords": [],
4   "author": "",
5   "license": "ISC",
6   "dependencies": {
7     "@aws-sdk/client-s3": "^3.693.0"
8   }
9 }
10 }
11 }
```

22. After we ran two more commands to install the dependencies for Dynamo DB.

```
npm install @aws-sdk/client-dynamodb
```

```
npm install @aws-sdk/util-dynamodb
```

```
PS D:\AWS Serveless\Invoke Lambda Asynchronously Using Amazon S3 to Trigger Lambda> npm install @aws-sdk/client-dynamodb
added 6 packages, and audited 110 packages in 6s

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS D:\AWS Serveless\Invoke Lambda Asynchronously Using Amazon S3 to Trigger Lambda> npm install @aws-sdk/util-dynamodb
added 1 package, and audited 111 packages in 1s

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS D:\AWS Serveless\Invoke Lambda Asynchronously Using Amazon S3 to Trigger Lambda>
```

23. In your Package.json file you will see the dependencies.

```
1 package.json > type
2
3     "dependencies": {
4         "@aws-sdk/client-dynamodb": "^3.693.0",
5         "@aws-sdk/client-s3": "^3.693.0",
6         "@aws-sdk/util-dynamodb": "^3.693.0"
7     }
8
9 }
```

24. Once you have created everything then you need to zip your files. For that go to the folder location open it and then select all the files and choose to zip them.

node_modules	19-11-2024 17:35	File folder
ddbClient	19-11-2024 17:25	JavaScript Source File 1 KB
index	19-11-2024 17:44	JavaScript Source File 2 KB
package	19-11-2024 17:35	JSON Source File 1 KB
package-lock	19-11-2024 17:35	JSON Source File 67 KB
s3Client	19-11-2024 17:43	JavaScript Source File 1 KB

25. Now you need to upload this zip folder to your lambda function. In your lambda function go to the code section and choose upload from Zip file.  
26. Choose your file and click on Save.

**Upload a .zip file** X

ⓘ When you upload a new .zip file package, it overwrites the existing code.

⤓ Upload

function.zip.zip X  
4.28 MB

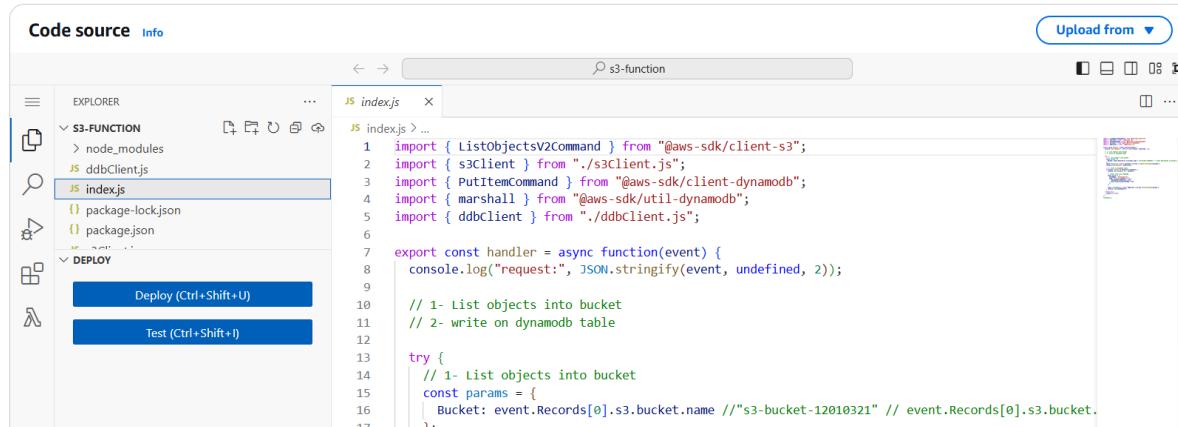
For files larger than 10 MB, consider uploading using Amazon S3.

**Enable encryption with an AWS KMS customer managed key**  
By default, Lambda encrypts the .zip file archive using an AWS owned key.

---

Cancel Save

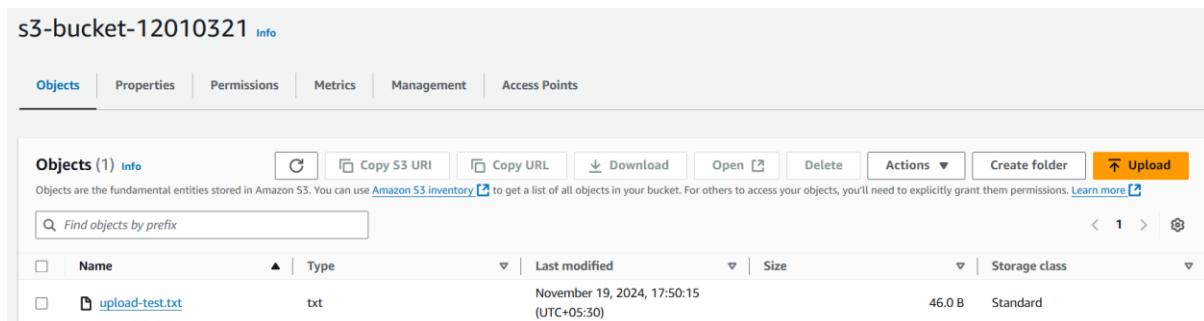
27. Refresh your page and you will be able to see that your zip file has been uploaded successfully.



The screenshot shows the AWS Lambda code editor interface. On the left, the 'EXPLORER' sidebar lists files: node\_modules, ddbClient.js, and index.js (which is selected). Below it, the 'DEPLOY' section has two buttons: 'Deploy (Ctrl+Shift+U)' and 'Test (Ctrl+Shift+I)'. The main area displays the contents of index.js:

```
JS index.js > ...
1 import { ListObjectsV2Command } from "@aws-sdk/client-s3";
2 import { s3Client } from "./s3client.js";
3 import { PutItemCommand } from "@aws-sdk/client-dynamodb";
4 import { marshall } from "@aws-sdk/util-dynamodb";
5 import { ddbClient } from "./ddbClient.js";
6
7 export const handler = async function(event) {
8     console.log("request:", JSON.stringify(event, undefined, 2));
9
10    // 1- List objects into bucket
11    // 2- write on dynamodb table
12
13    try {
14        // 1- List objects into bucket
15        const params = {
16            Bucket: event.Records[0].s3.bucket.name // "s3-bucket-12010321" // event.Records[0].s3.bucket.
17        }
18    
```

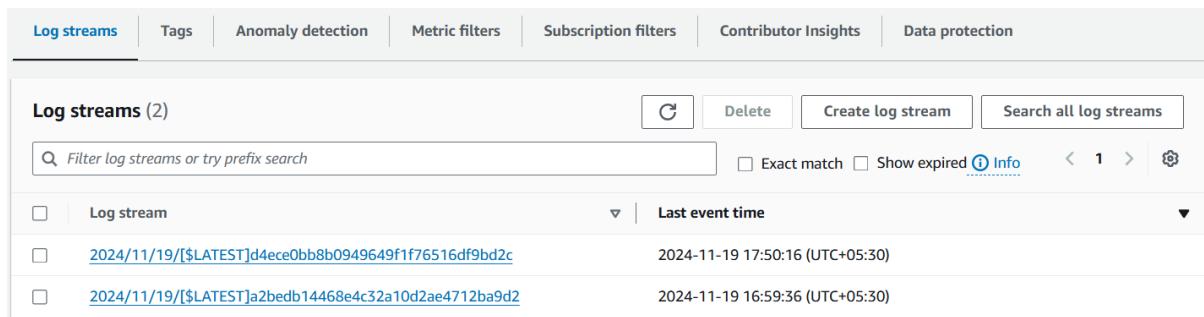
28. Now we will perform end-to-end testing for our environment. For that, we can upload an item to our S3 bucket. Below you can see that we have uploaded a text file.



The screenshot shows the AWS S3 console for the bucket 's3-bucket-12010321'. The 'Objects' tab is selected. The object list shows one item: 'upload-test.txt'. The details for this file are:

Name	Type	Last modified	Size	Storage class
upload-test.txt	txt	November 19, 2024, 17:50:15 (UTC+05:30)	46.0 B	Standard

29. We will move to CloudWatch logs and see the logs for our lambda function. You will see that you have a new log stream open it.



The screenshot shows the AWS CloudWatch Logs 'Log streams' page. There are two log streams listed:

Log stream	Last event time
2024/11/19/[...LATEST]d4ece0bb8b0949649f1f76516df9bd2c	2024-11-19 17:50:16 (UTC+05:30)
2024/11/19/[...LATEST]a2bedb14468e4c32a10d2ae4712ba9d2	2024-11-19 16:59:36 (UTC+05:30)

30. Our lambda function was triggered when we uploaded the file and the object we uploaded gets written onto the Dynamo DB table.

**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](#)

Timestamp	Message
No older events at this moment. <a href="#">Retry</a>	
2024-11-19T17:50:15.717+05:30	INIT_START Runtime Version: nodejs:20.v51 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:cb6527bfb6726a080a367eca00e...
2024-11-19T17:50:16.232+05:30	START RequestId: 911340ab-47d7-4941-b1ef-b75b778227de Version: \$LATEST
2024-11-19T17:50:16.233+05:30	2024-11-19T12:20:16.233Z 911340ab-47d7-4941-b1ef-b75b778227de INFO request: { "Records": [ { "eventVersion": "2.1", "eventSo...
2024-11-19T17:50:17.322+05:30	2024-11-19T12:20:17.322Z 911340ab-47d7-4941-b1ef-b75b778227de INFO Success { '\$metadata': { httpStatusCode: 200, requestId: ...
2024-11-19T17:50:17.325+05:30	2024-11-19T12:20:17.325Z 911340ab-47d7-4941-b1ef-b75b778227de INFO Content: {"Key": "upload-test.txt", "LastModified": "2024-11...
2024-11-19T17:50:17.522+05:30	2024-11-19T12:20:17.522Z 911340ab-47d7-4941-b1ef-b75b778227de INFO { '\$metadata': { httpStatusCode: 200, requestId: 'LSU0LG1...
2024-11-19T17:50:17.544+05:30	END RequestId: 911340ab-47d7-4941-b1ef-b75b778227de Duration: 1311.02 ms Billed Duration: 1312 ms Memory Size: 128 MB Max...
2024-11-19T17:50:17.544+05:30	REPORT RequestId: 911340ab-47d7-4941-b1ef-b75b778227de Duration: 1311.02 ms Billed Duration: 1312 ms Memory Size: 128 MB Max...
No newer events at this moment. <a href="#">Auto retry paused.</a> <a href="#">Resume</a>	

```
{
  "Records": [
    {
      "eventVersion": "2.1",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "2024-11-19T12:20:14.449Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "A13DFHTRP6I83Q"
      },
      "requestParameters": {
        "sourceIPAddress": "103.226.202.202"
      },
      "responseElements": {
        "x-amz-request-id": "6V4M0VW29Q9JBVDK",
        "x-amz-id-2": "BMrJ4YhwObl56ye1ebXnp50JB9UL7IBFOLjLgtAfNNwRXUo2QU/iaB8mm10nwsR41wj/7IZxRcj5kpu+FebVBs9NRQ0Aq2c"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "8d09a56d-8f56-46f8-b597-c05561c6610c",
        "bucket": {
          "name": "s3-bucket-12010321",
          "ownerIdentity": {
            "principalId": "A13DFHTRP6I83Q"
          },
          "arn": "arn:aws:s3:::s3-bucket-12010321"
        },
        "object": {
          "key": "upload-test.txt",
          "size": 46,
          "eTag": "96c618eed8d238ddf0873b73ee21092b",
          "sequencer": "00673C827E6A16B1D6"
        }
      }
    }
  ]
}
```

31. Open the dynamo DB and you will see that an object has been added to the table.

Completed. Read capacity units consumed: 2		X
<b>Items returned (1)</b>		
<input type="checkbox"/>	Key (String)	Actions ▾
<input type="checkbox"/>	<a href="#">upload-test.txt</a>	"96c618eed... { } 46 STANDARD

32. After completing this lab delete all the resources start with deleting your function and dynamo db table then move to S3 and delete your bucket.