# 😊 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.

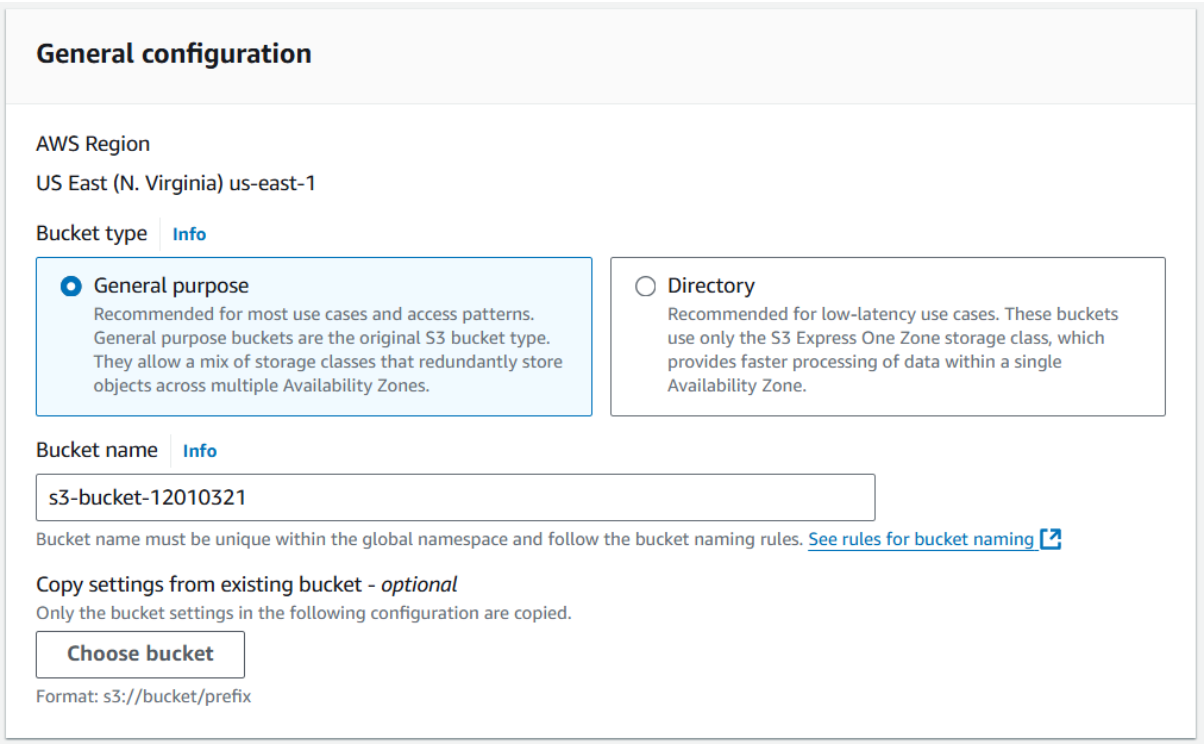
---

**3. Process Flow**

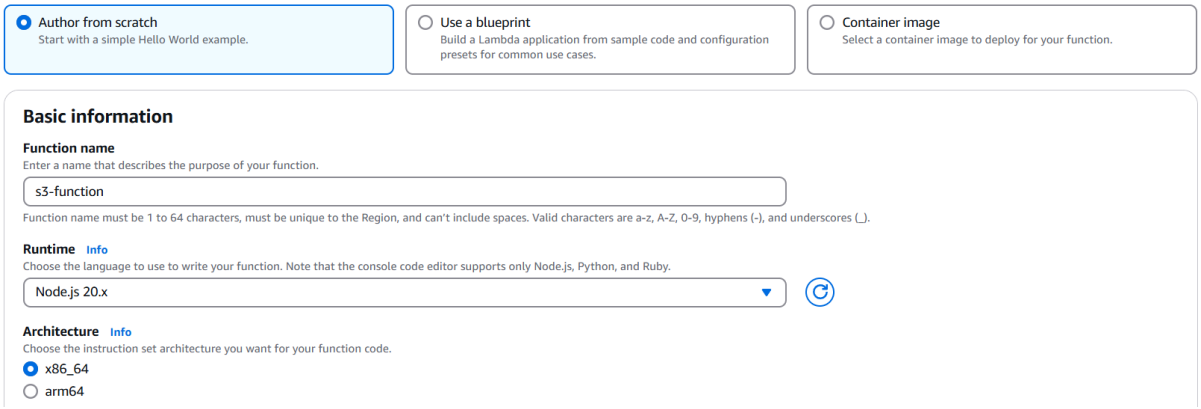
1. **Upload**: Client uploads an object to the S3 bucket.
2. **Trigger**: S3 generates an ObjectUploaded event to invoke the Lambda function.
3. **Lambda Execution**:
    - Receives the event.
    - Executes business logic to extract metadata from the uploaded file.
    - Stores the metadata in DynamoDB.
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:

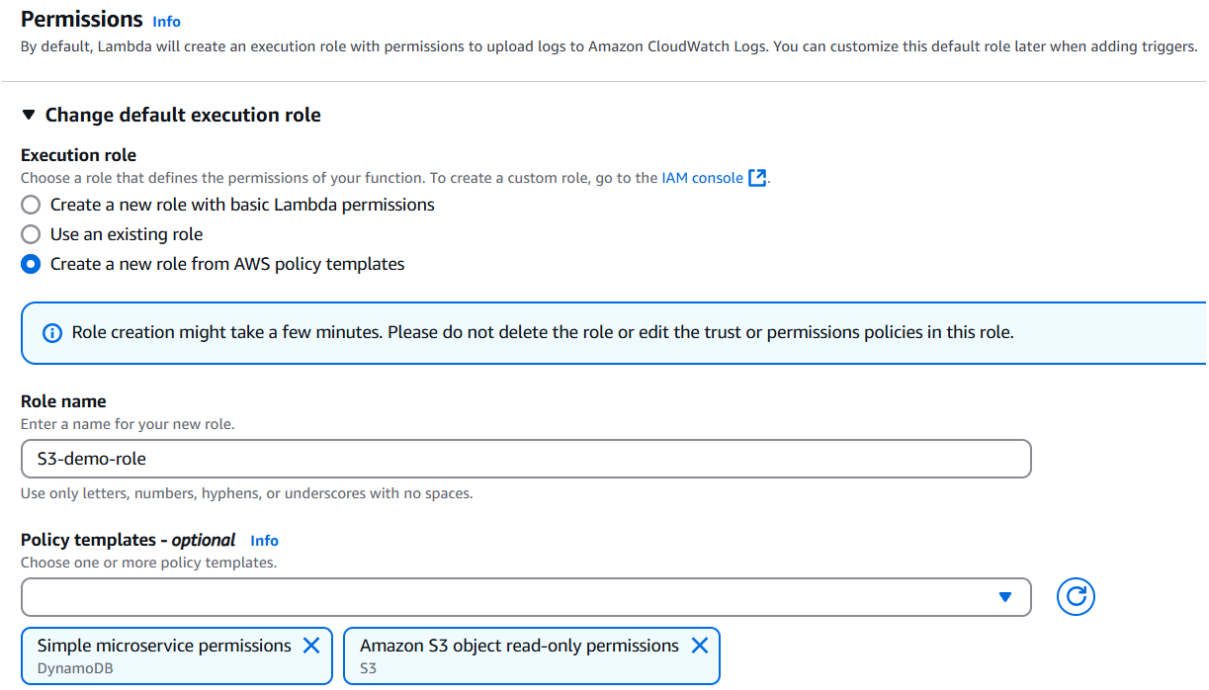
1. Now we will start with creating the architecture for our lab.
2. First, we need to create a bucket. In your console, search for S3 and click on Create Bucket. Give a unique name to your bucket and then create it.




3. 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.

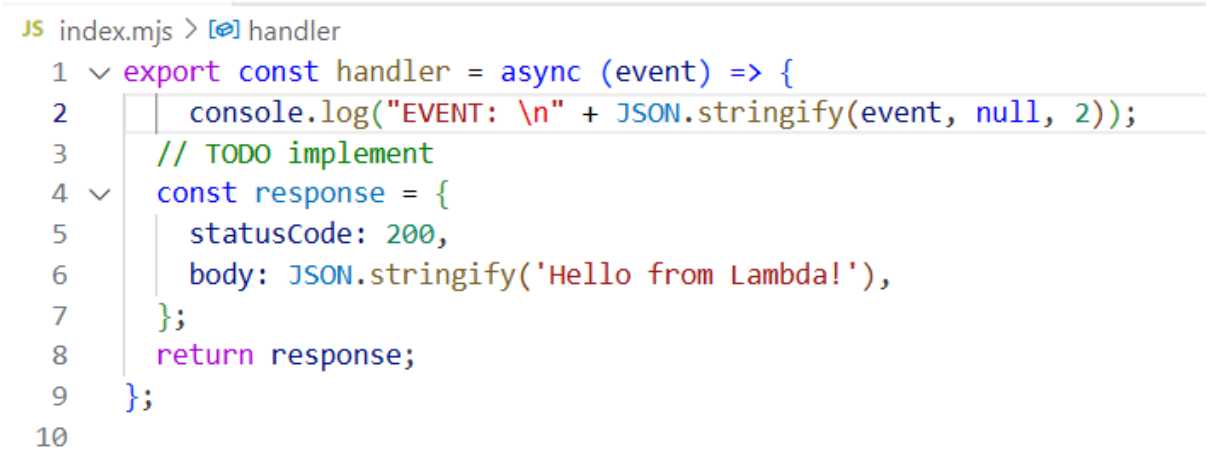



4. In the permission we will create a new role, choose Create a new role from AWS Policy templates give your role a name, and choose the same policy templates you can see below and create your function.

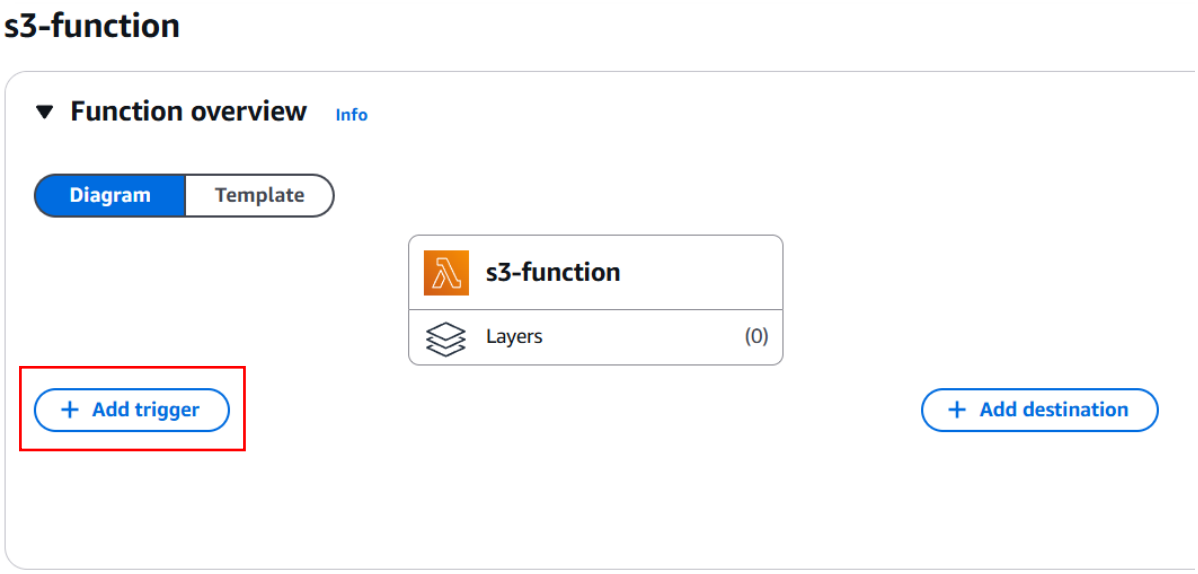



5. Go to the code section in your function and add the 2nd line in your code as you can see below. This will help us to see the logs in CloudWatch.

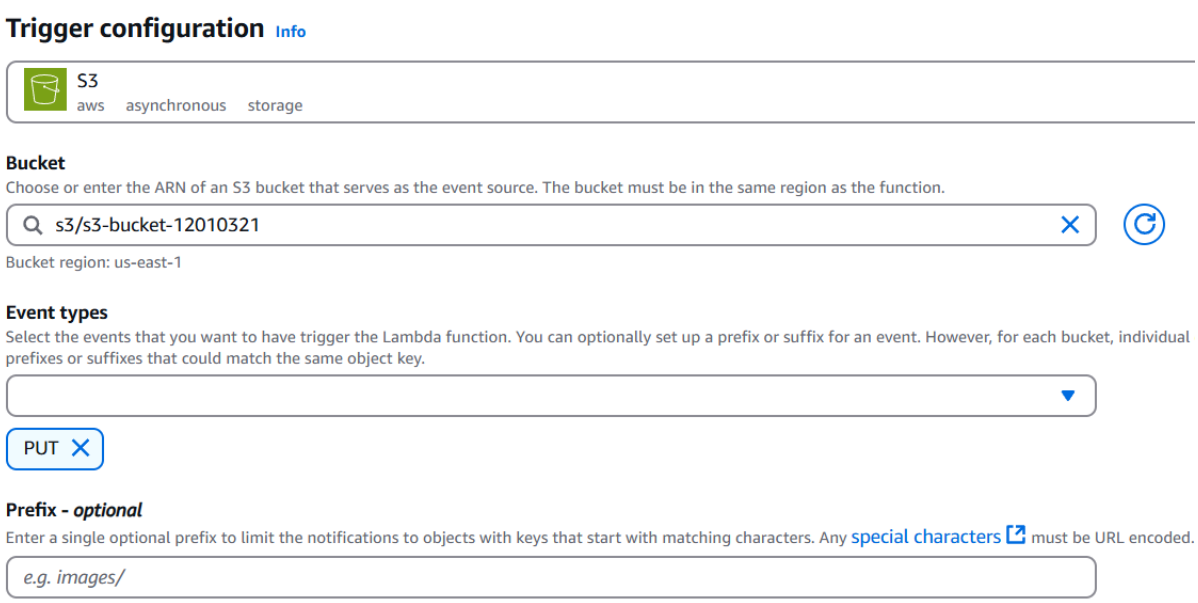
   **console.log("EVENT: \n" + JSON.stringify(event, null, 2));**



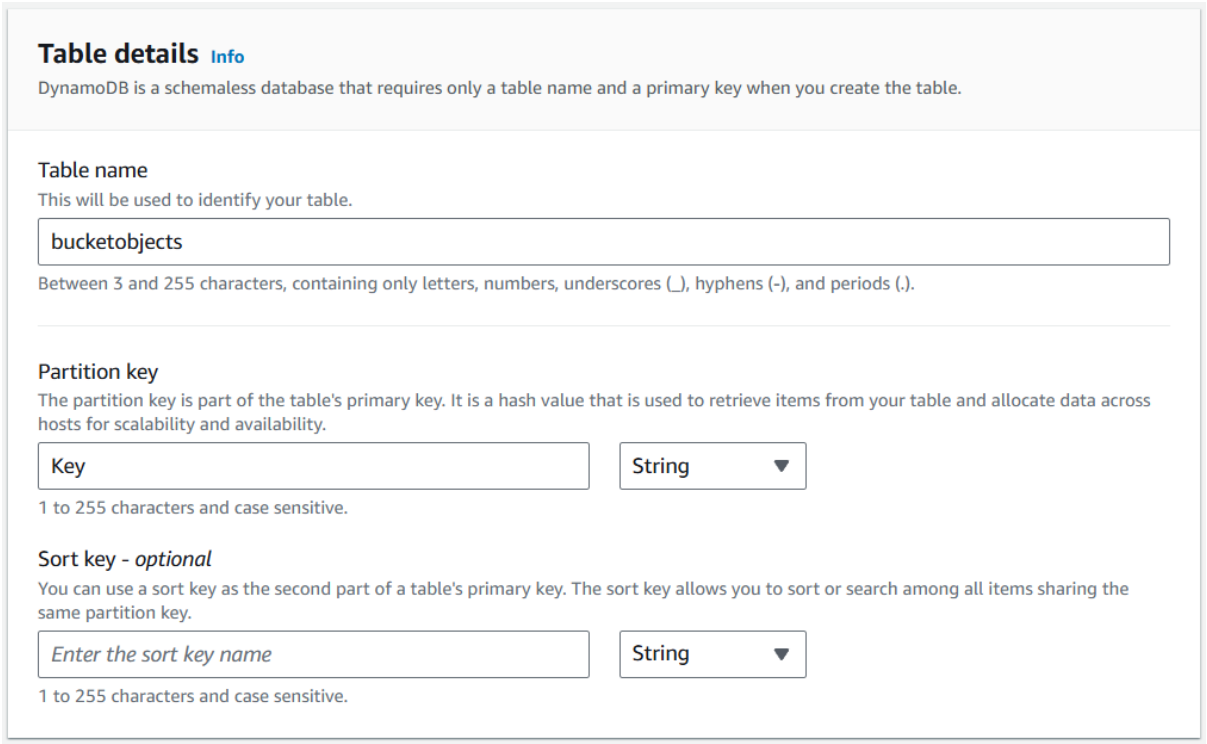

6. Now we need to add a trigger, so click on the Add trigger button.



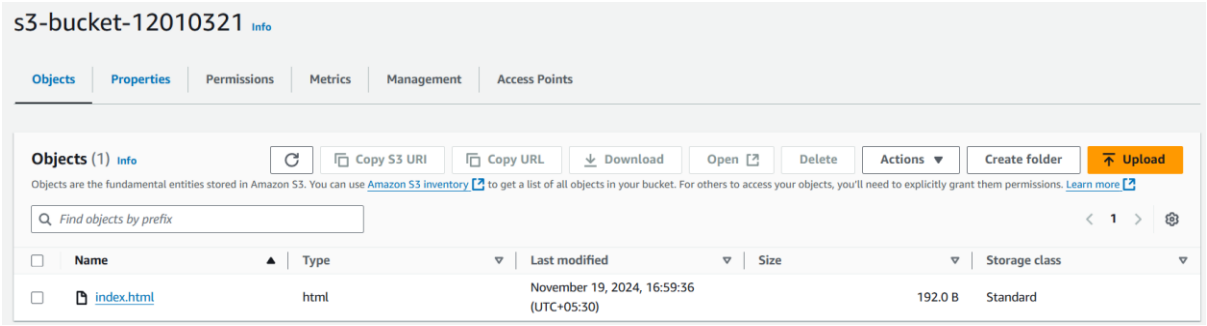

7. You need to add a trigger for S3 and in the bucket choose your newly created bucket and, in the event, types choose PUT. Then check the box for acknowledgment and click on Add.



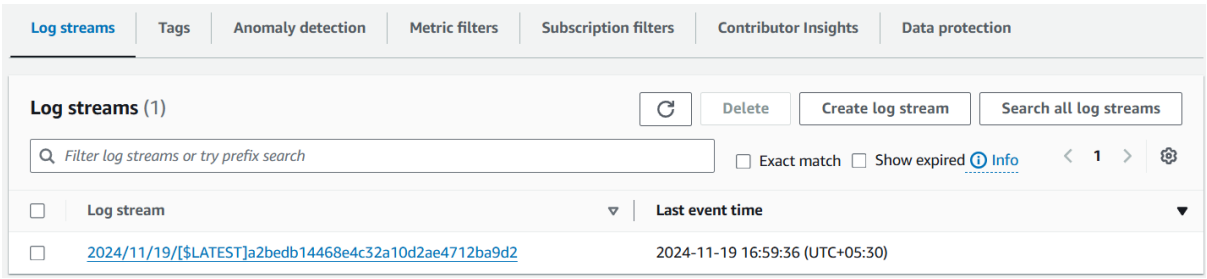

8. So, now we will create our Dynamo DB table. Go to it and click on Create. You need to give a name to your table and give Partition key as “Key”.
9. Then scroll down, in the Table settings choose **Customize settings**, and for the **Read/Write capacity choose On-demand.**
10. Just create your table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




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.



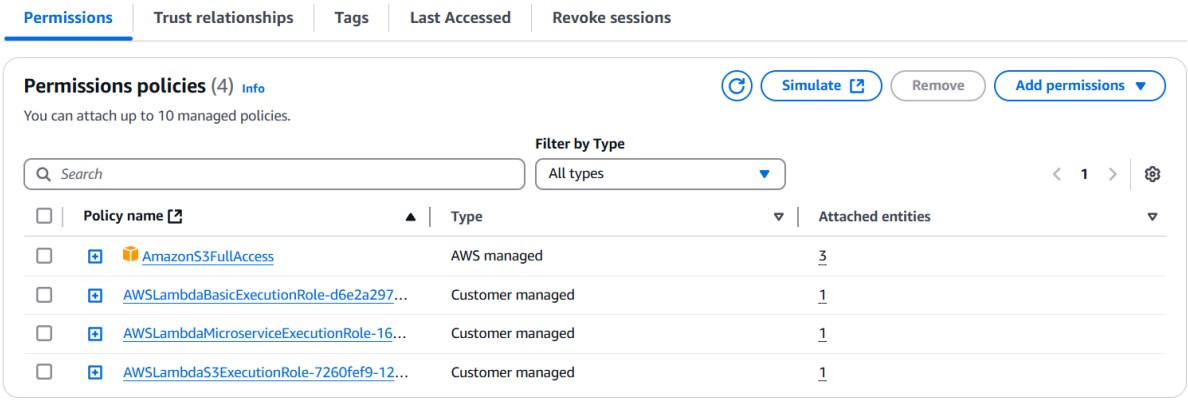

13. Here you can see that a log stream has been generated at the same time when uploaded an object to our S3 bucket.



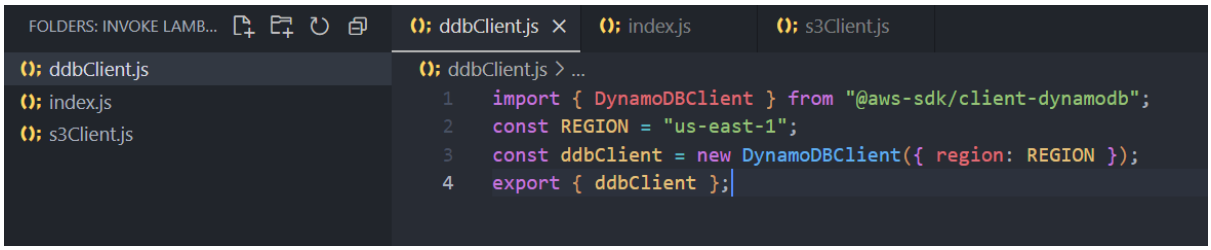

14. If you open the log event you will see that records have been created, it means that our lambda function gets triggered.

```
{
    "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/MN0LCQOIw9VhHw80xTmeTl/DSyTLhn0i/9DQRsQNs0zWQAeHKdQrREz4Z/13uSHqr8bKj1IKu3Ib0NyLY0BWV150NsOWm"
            },
            "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": "4ce90c022f0dcd7d6e3ef4790764c070",
                    "sequencer": "00673C769F920A846E"
                }
            }
        }
    ]
}
```


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




16. Then you need to create a folder on your laptop and open it in VS Code.
17. 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.



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

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**

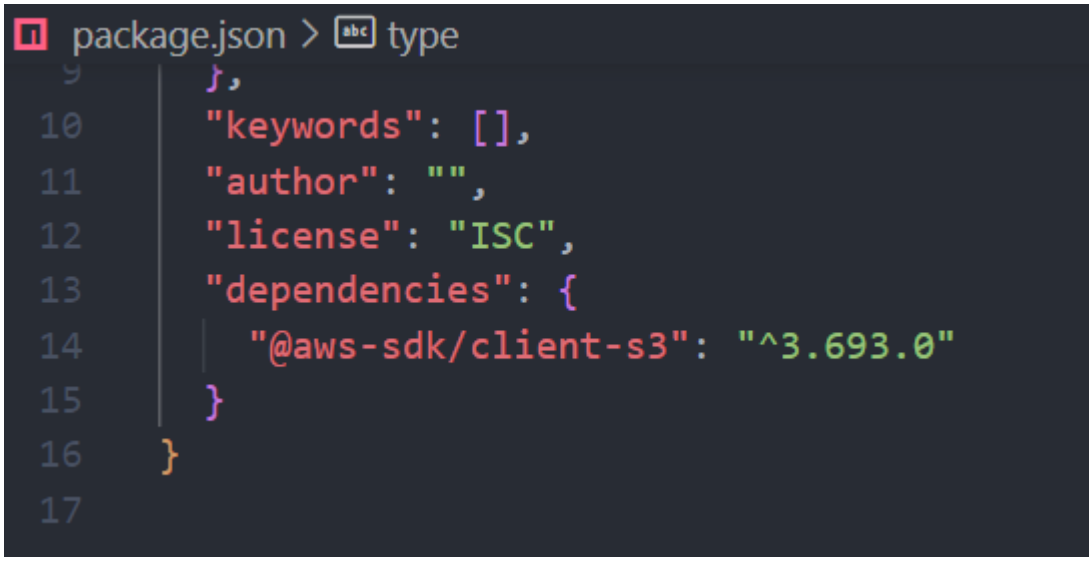
```
PS D:\AWS Serveless\Invoke Lambda Asynchronously Using Amazon S3 to Trigger Lambda> npm install @aws-sdk/client-s3

added 103 packages, and audited 104 packages in 3s

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

found 0 vulnerabilities
```

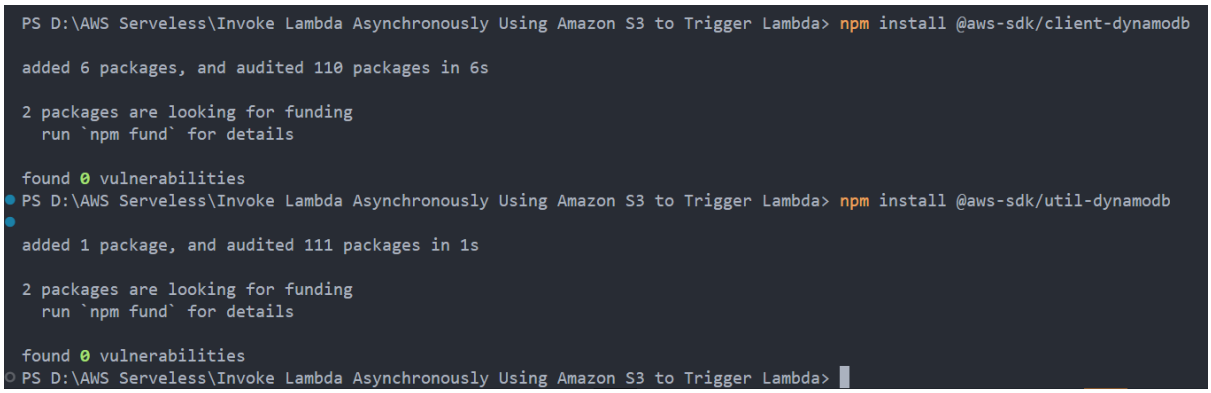
21. Also, inside your package.json file as well you will that the dependencies has been installed.




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

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

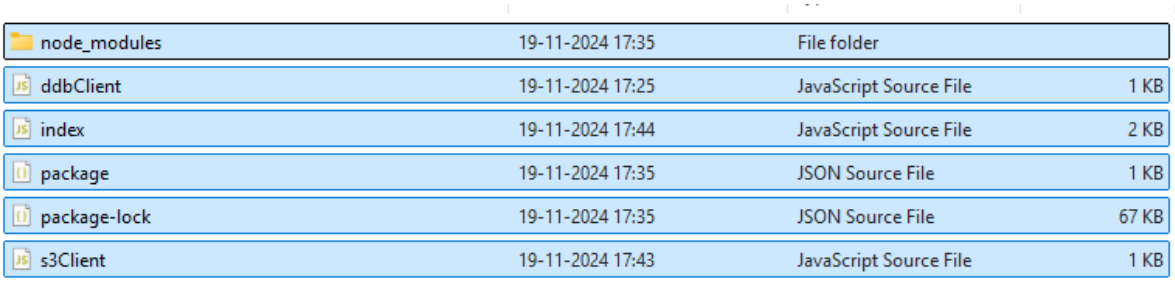
    **npm install @aws-sdk/util-dynamodb**




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

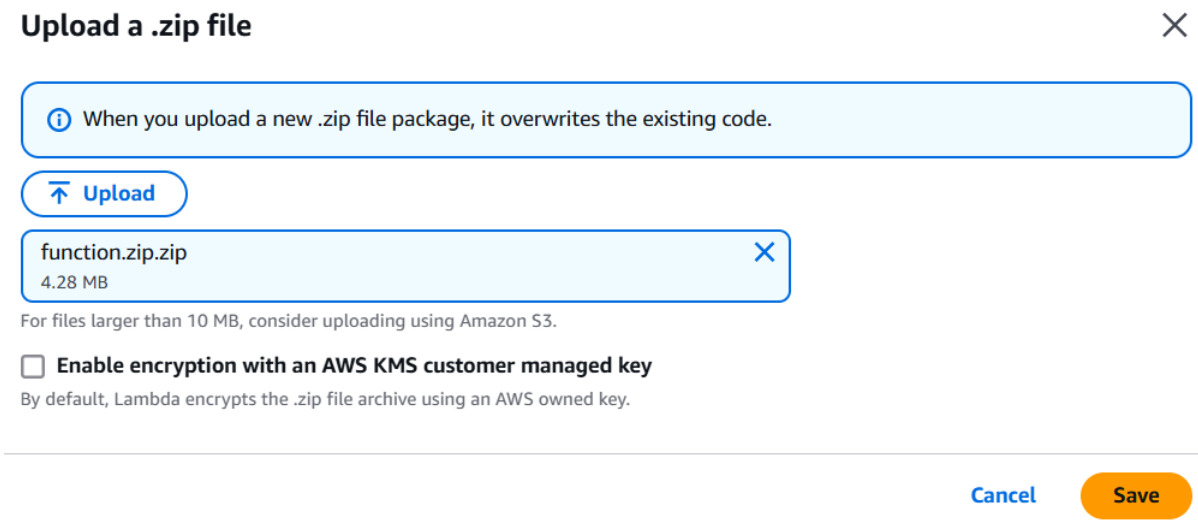
```
package.json > type
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@aws-sdk/client-dynamodb": "^3.693.0",
    "@aws-sdk/client-s3": "^3.693.0",
    "@aws-sdk/util-dynamodb": "^3.693.0"
  }
}
```

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.

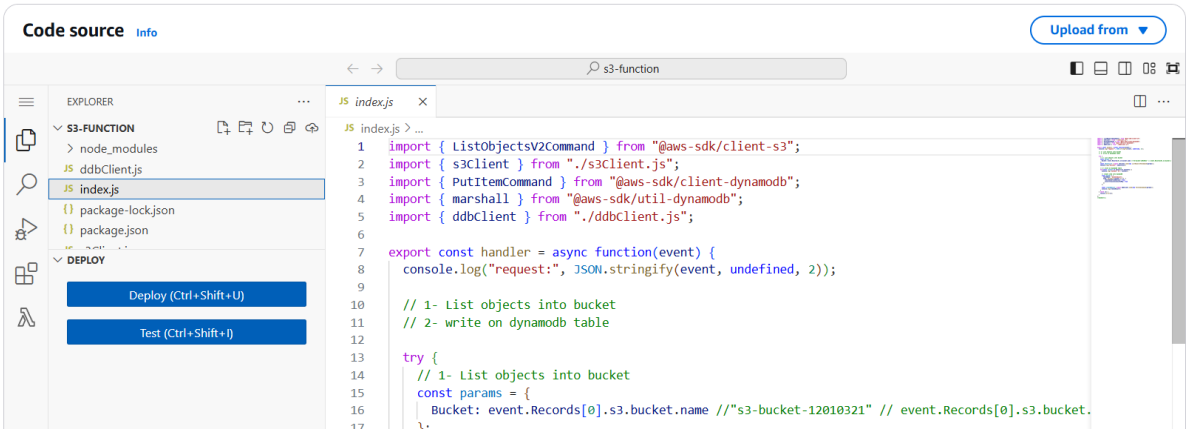


| | | | |
|---|---|---|---|
| node_modules | 19-11-2024 17:35 | File folder | |
| ddbClient | 19-11-2024 17:25 | JavaScript Source File | 1 KB |
| index | 19-11-2024 17:44 | JavaScript Source File | 2 KB |
| package | 19-11-2024 17:35 | JSON Source File | 1 KB |
| package-lock | 19-11-2024 17:35 | JSON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

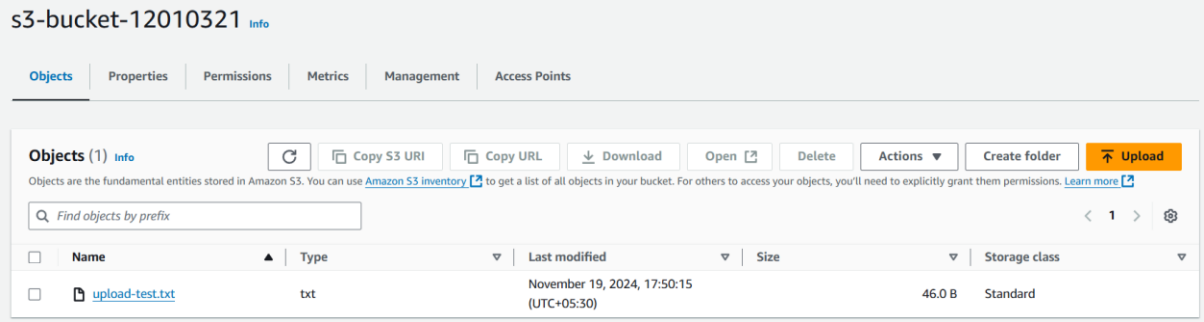
25. Now you need to upload this zip folder to your lambda function. In your lambda function go to the code section and choose upload from Zip file.
26. Choose your file and click on Save.



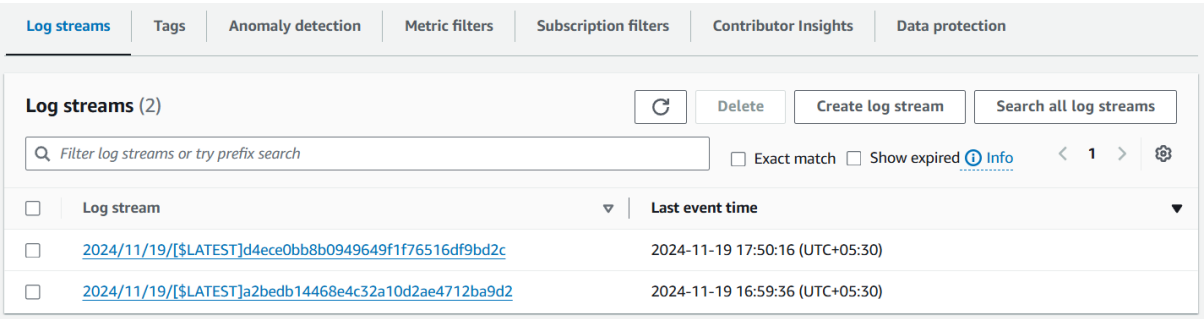
27. Refresh your page and you will be able to see that your zip file has been uploaded successfully.



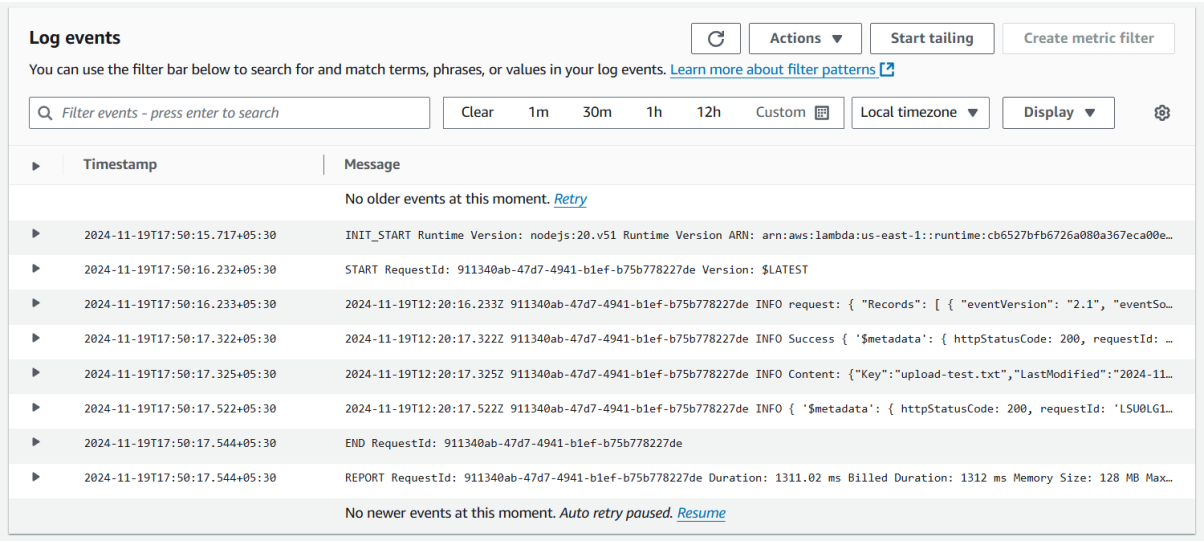

28. Now we will perform end-to-end testing for our environment. For that, we can upload an item to our S3 bucket. Below you can see that we have uploaded a text file.




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.



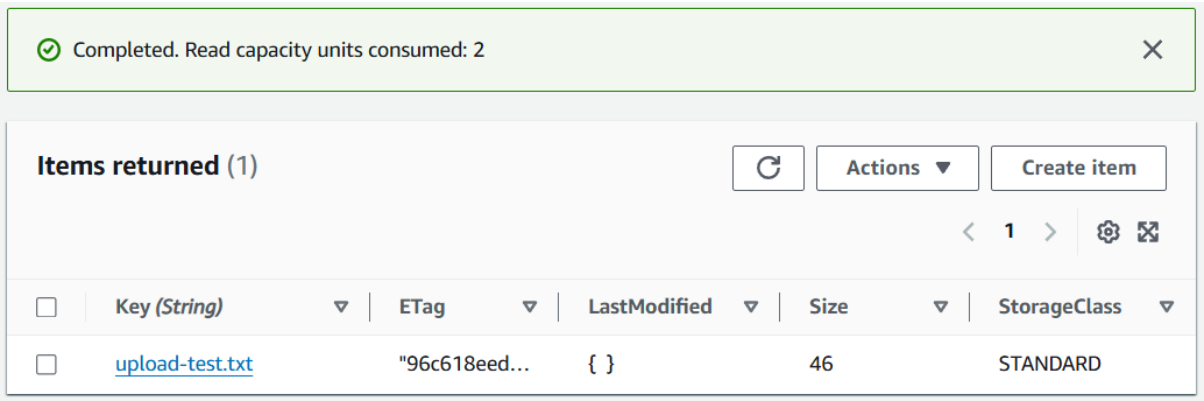

30. Our lambda function was triggered when we uploaded the file and the object we uploaded gets written onto the Dynamo DB table.





```
{
    "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": "BMrJ/4YhWObL56ye1ebXnp50JB9UL7IBfOLjLgtAfNNWRXUo2QU/ioB8mmlOnwsR41wj/7IZxRcj5kpu+FebVBs9NRQOAq2c"
            },
            "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.




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.