**Semester: V**
**Academic Year: 2022-23**
**Class / Branch: TE IT**
**Subject: Advanced Devops Lab (ADL)**
**Name of Instructor:Prof. Manjusha K.**

**Name of Student: Soham Dalvi**
**Student ID: 21104010**

---

**EXPERIMENT NO. 12**

**Aim: To create a Lambda function which will log "An Image has been added" once you add an object to a specific bucket in S3**

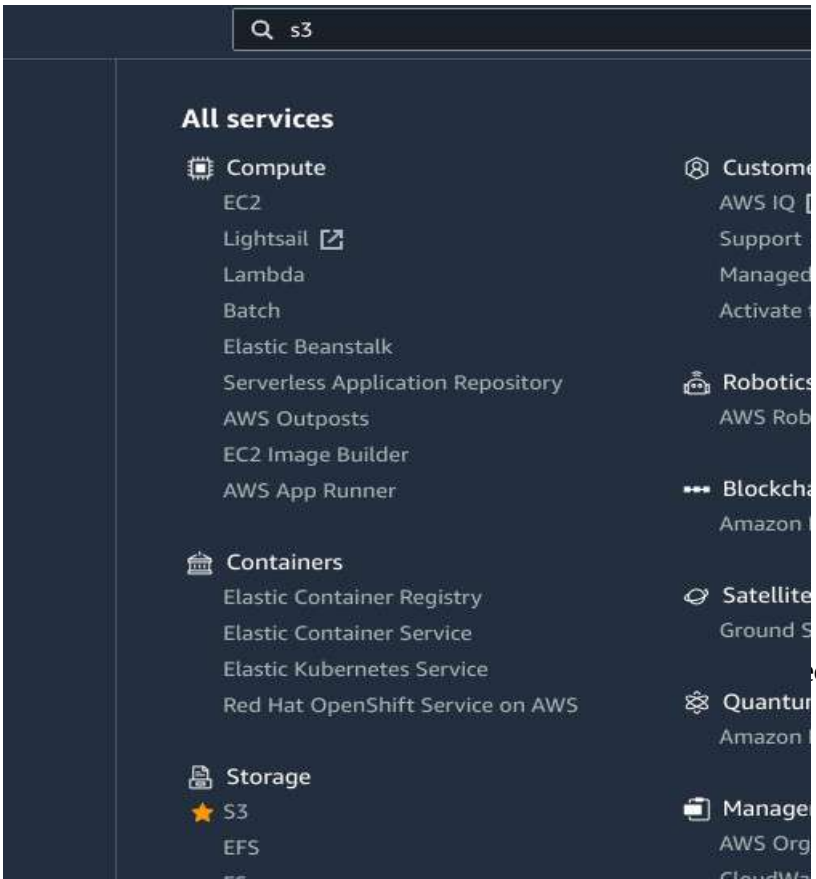
## Theory:

### Creating S3 Bucket

Let us start first by creating a s3 bucket in AWS console using the steps given below –

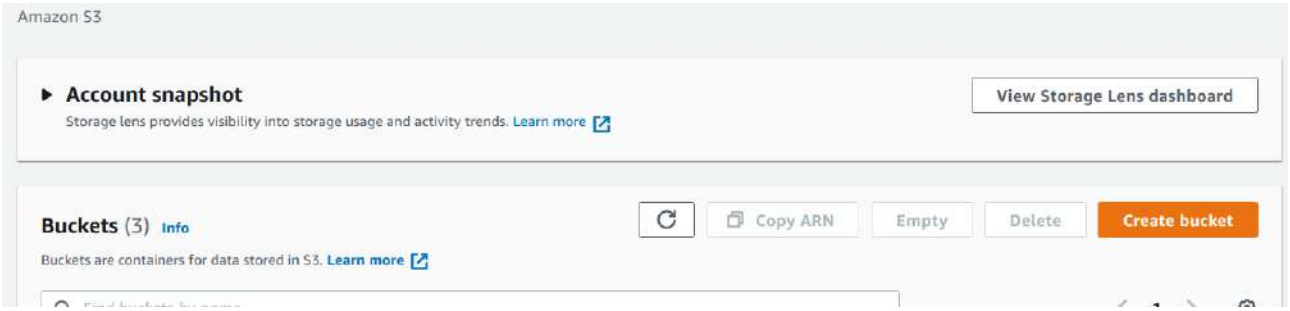
**Step 1**

Go to Amazon services and click **S3** in storage section as highlighted in the image given below –



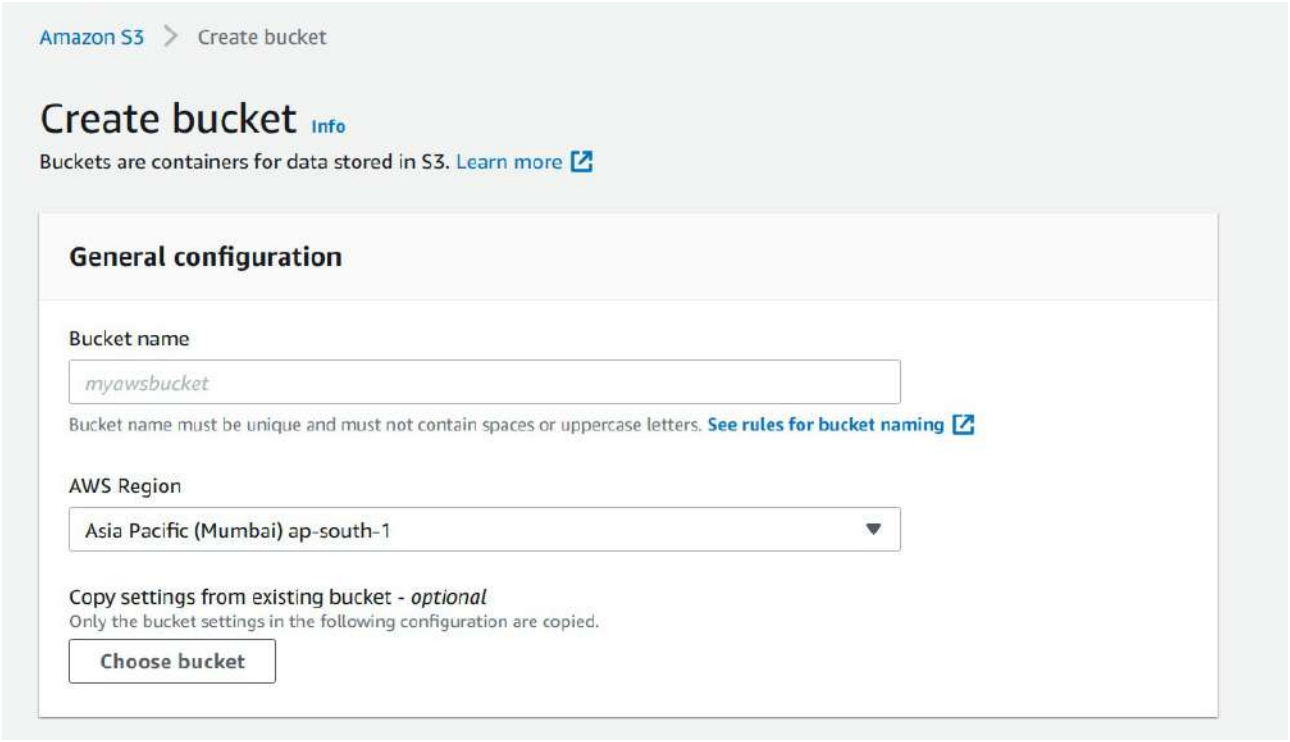
**Step 2**

Click **S3** storage and **Create bucket** which will store the files uploaded.




**Step 3**

Once you click Create bucket button, you can see a screen as follows –

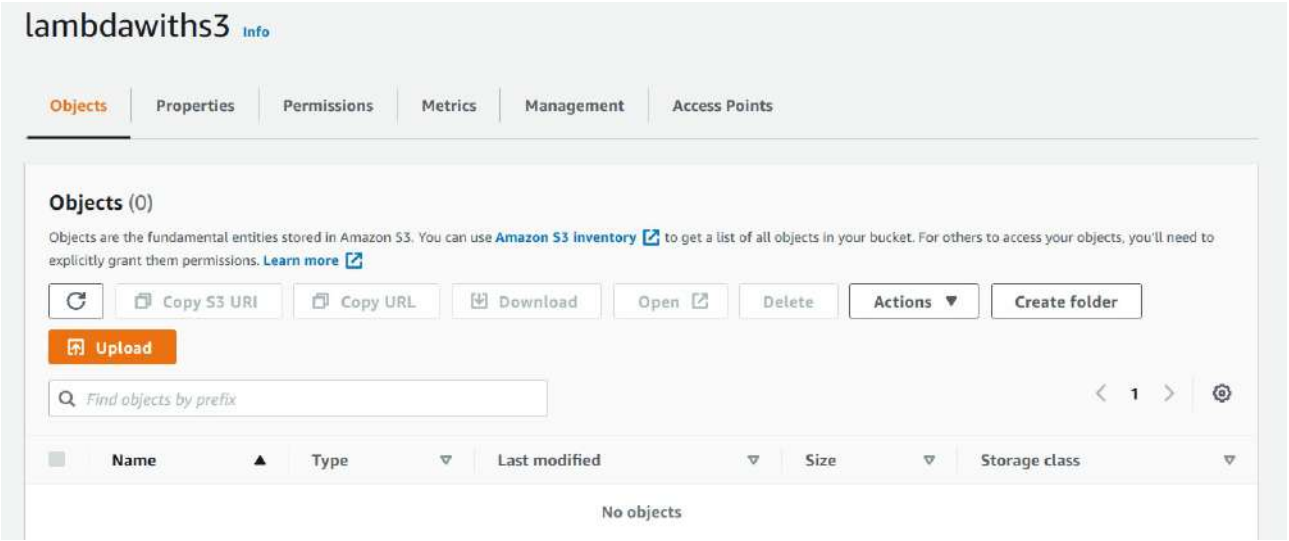


**Step 4**

Enter the details Bucket name, Select the Region and click Create button at the bottom left side. Thus, we have created bucket with name :


**Step 5**

Now, click the bucket name and it will ask you to upload files as shown below –



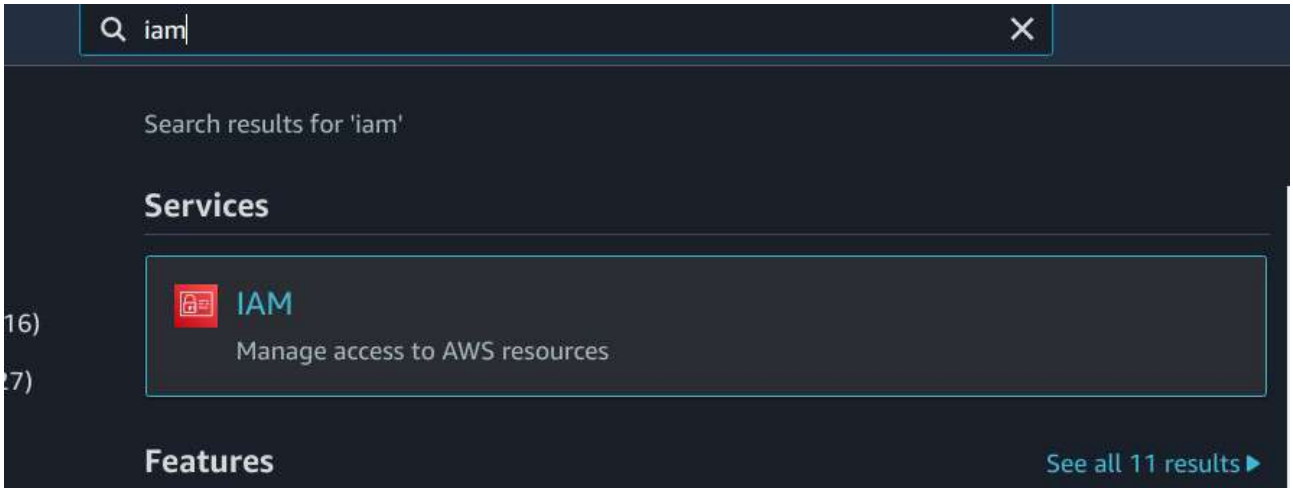

Thus, we are done with bucket creation in S3.

## Create Role that Works with S3 and Lambda

To create role that works with S3 and Lambda, please follow the Steps given below –

**Step 1**

Go to AWS services and select IAM as shown below –

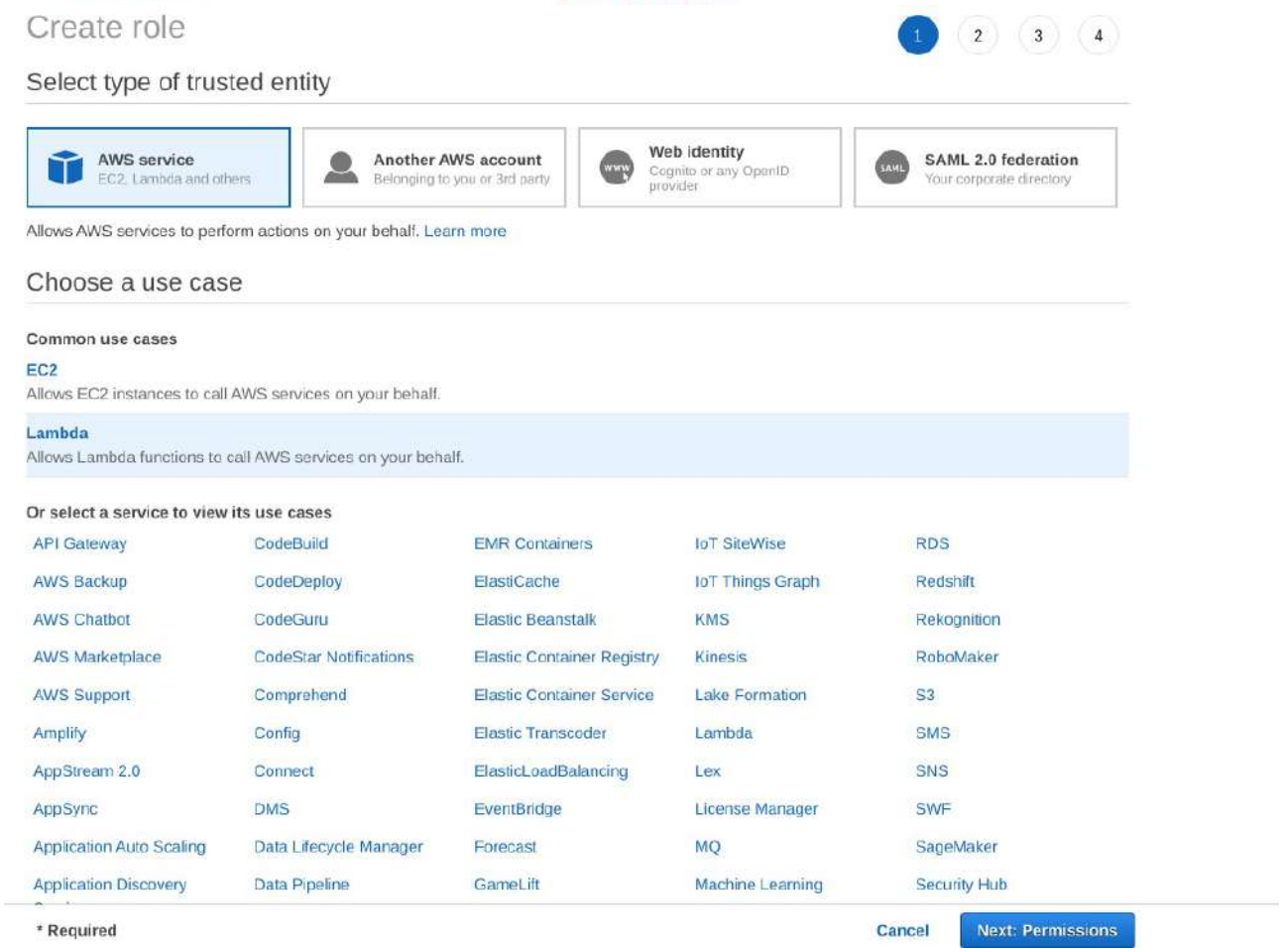



**Step 2**

Now, click **IAM -> Roles** as shown below −


**Step 3**

Now, click **Create role** and choose the services that will use this role. Select Lambda and click **Permission** button.



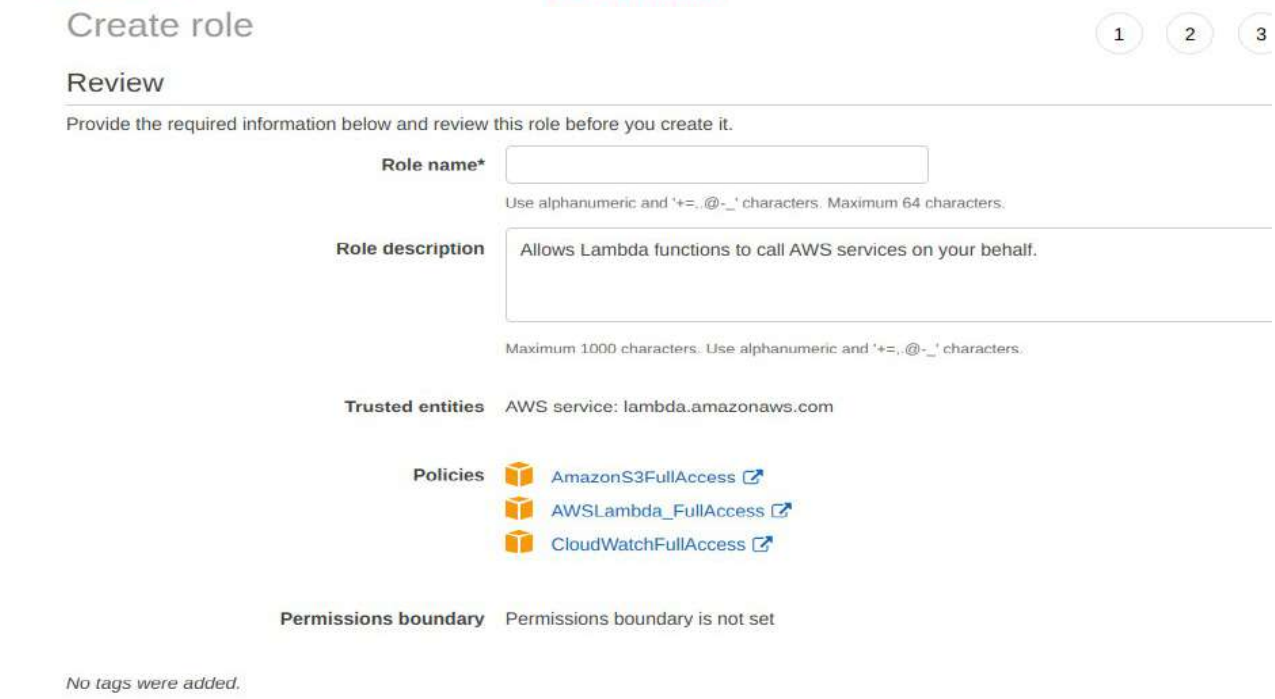

**Step 4**

Add the permission from below and click Review.

**AmazonS3FullAccess, AWSLambdaFullAccess and CloudWatchFullAccess.**

**Step 5**

Observe that we have chosen the following permissions –




Observe that the Policies that we have selected are **AmazonS3FullAccess, AWSLambdaFullAccess and CloudWatchFullAccess.**

**Step 6**

Now, enter the Role name, Role description and click Create Role button at the bottom.

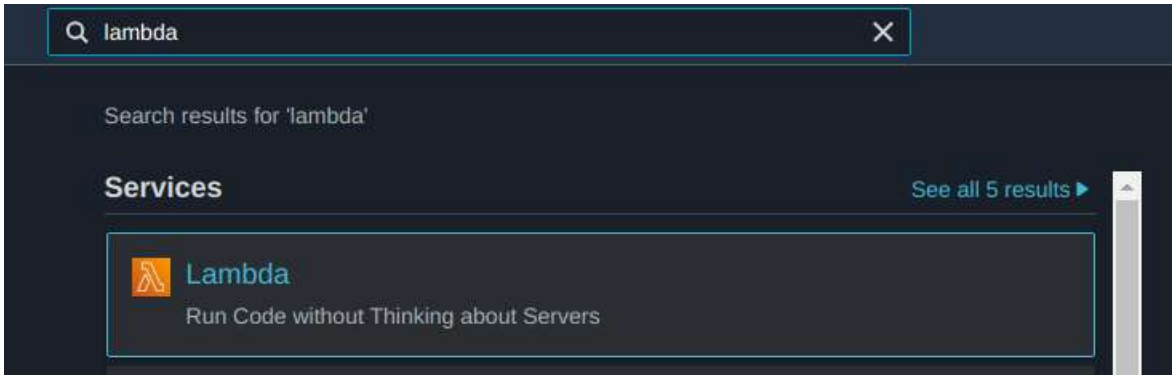

Thus, our role named lambdawiths3service is created.

## Create Lambda function and Add S3 Trigger

In this section, let us see how to create a Lambda function and add a S3 trigger to it. For this purpose, you will have to follow th Steps given below –

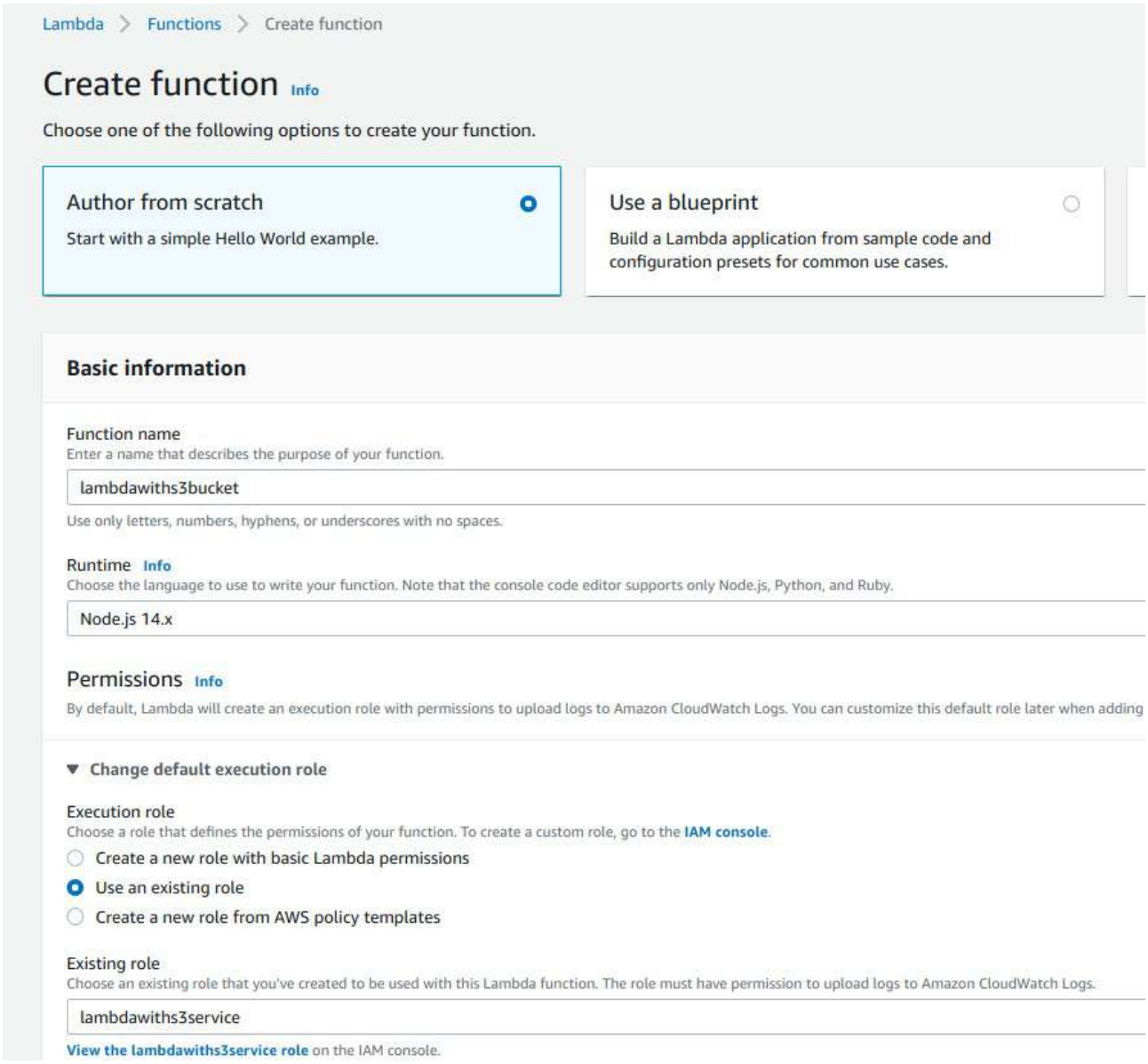
**Step 1**

Go to AWS Services and select Lambda as shown below –



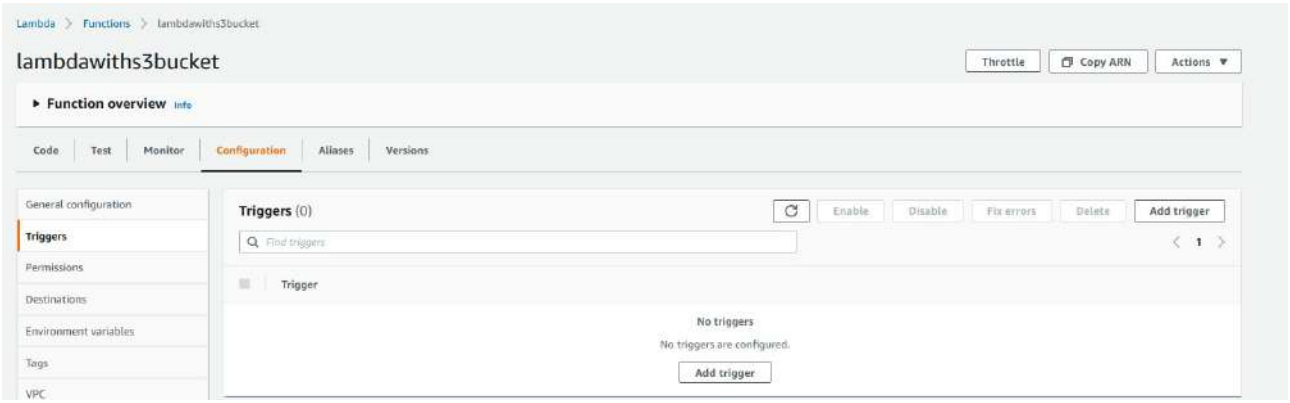

**Step 2**

Click **Lambda** and follow the process for adding **Name**. Choose the **Runtime, Role** etc. and create the function. The Lambda function that we have created is shown in the screenshot below –



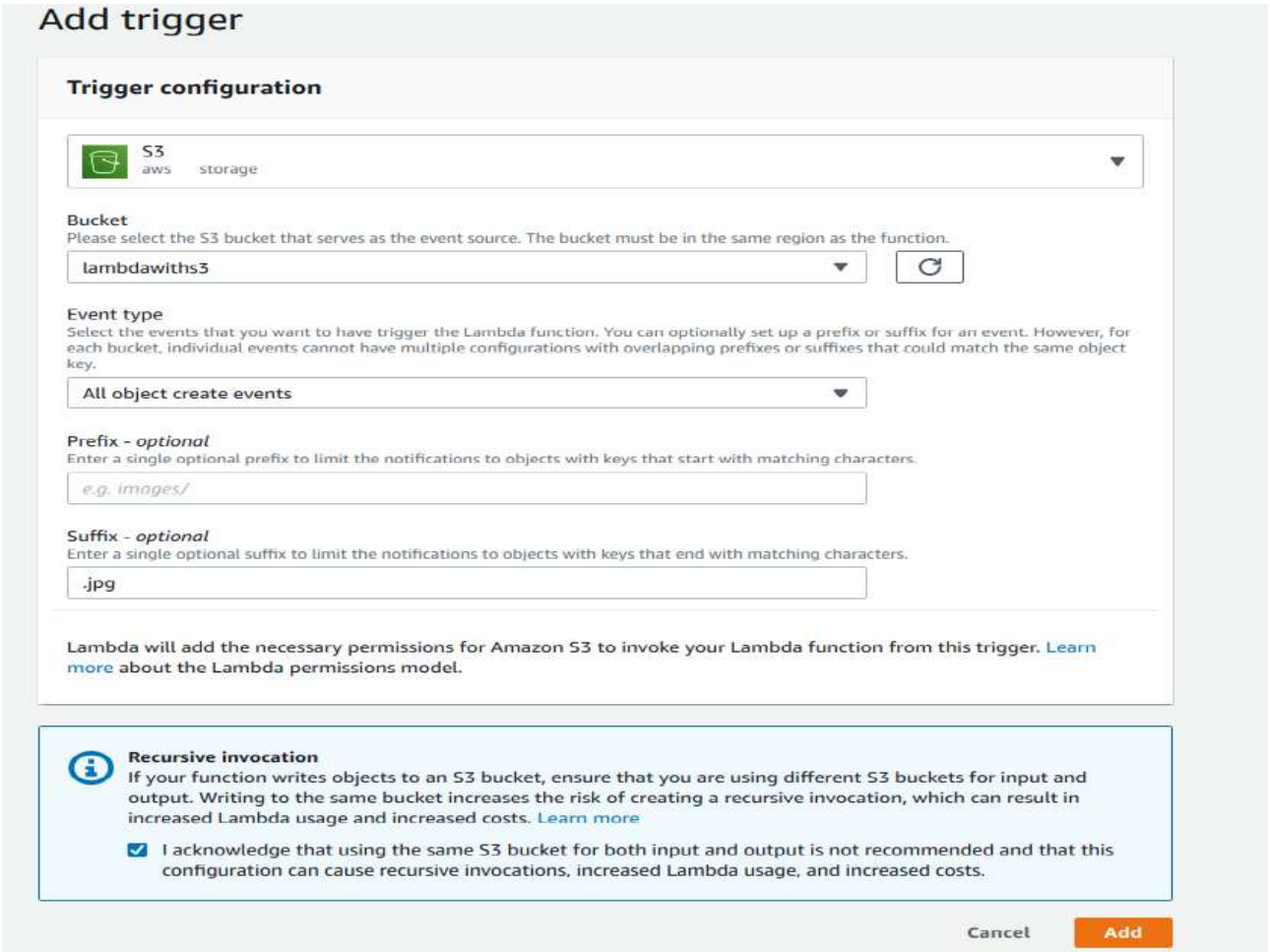

**Step 3**

Now let us add the S3 trigger.




**Step 4**

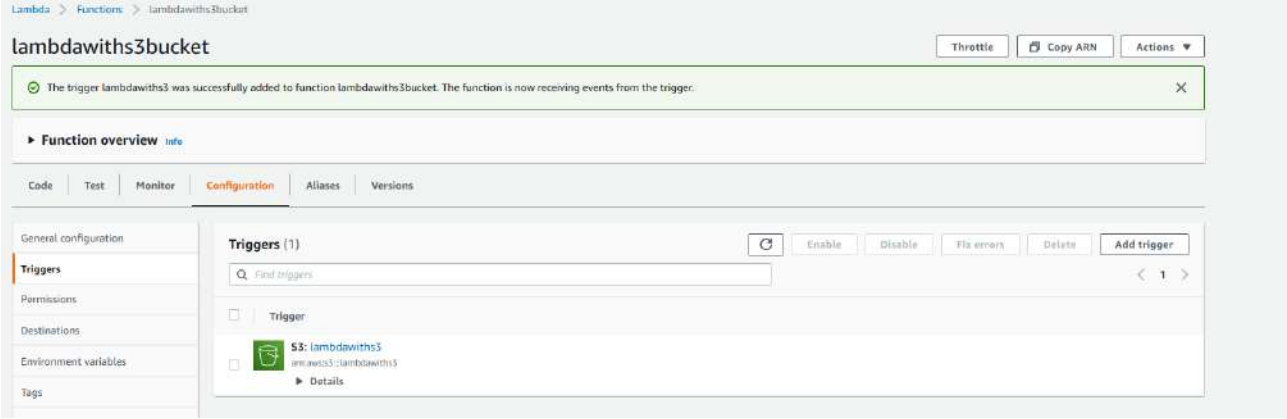
Choose the trigger from above and add the details as shown below –



You can add Prefix and File pattern which are used to filter the files added. For Example, to trigger lambda only for .jpg images. as we need to trigger Lambda for all jpg image files uploaded. Click Add button to add the trigger.

**Step 5**

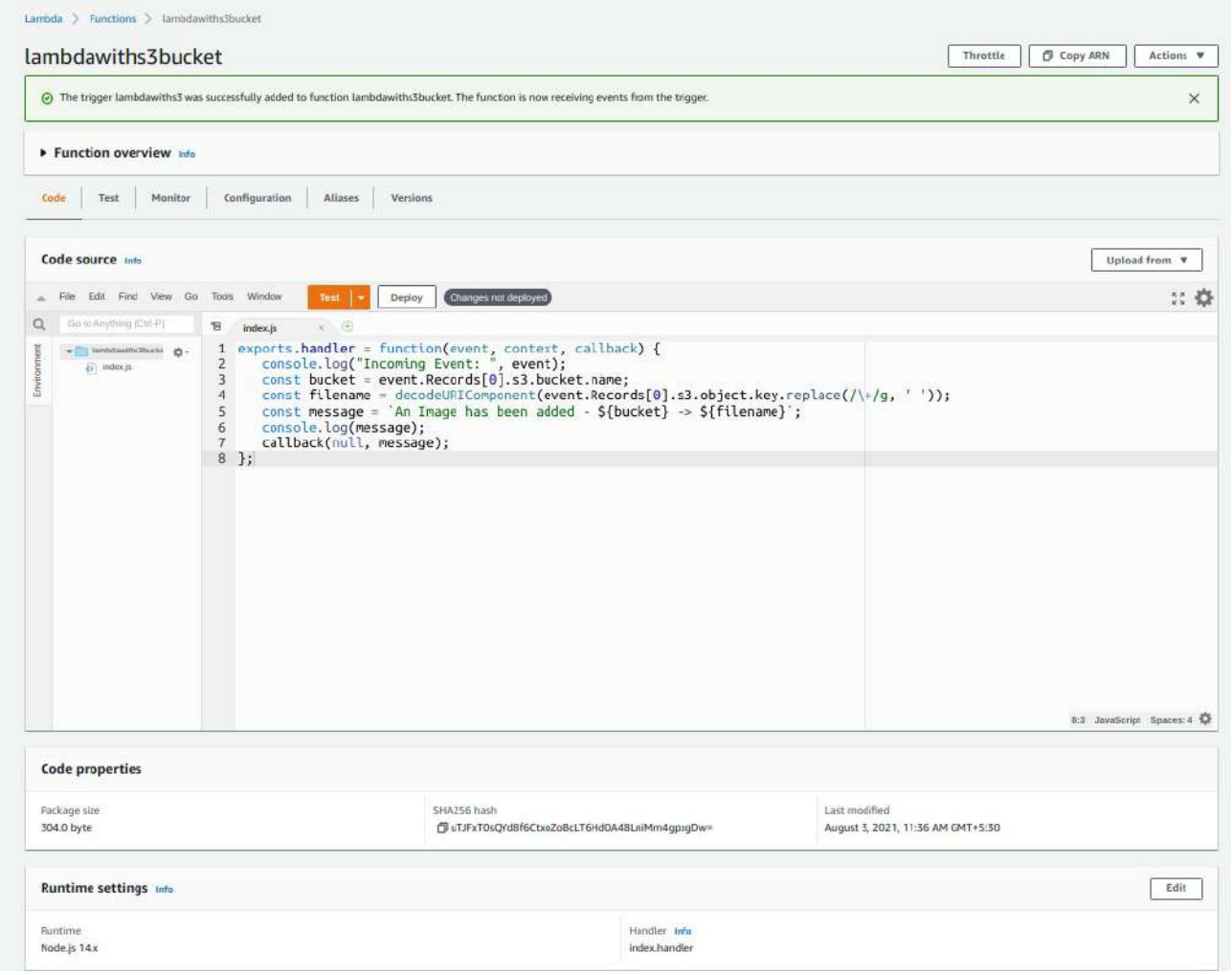
You can find the the trigger display for the Lambda function as shown below –




**Step 6**

Let's add the details for the aws lambda function. Here, we will use the online editor to add our code and use nodejs as the runtime environment.

To trigger S3 with AWS Lambda, we will have to use S3 event in the code as shown below –




**Step 7:**

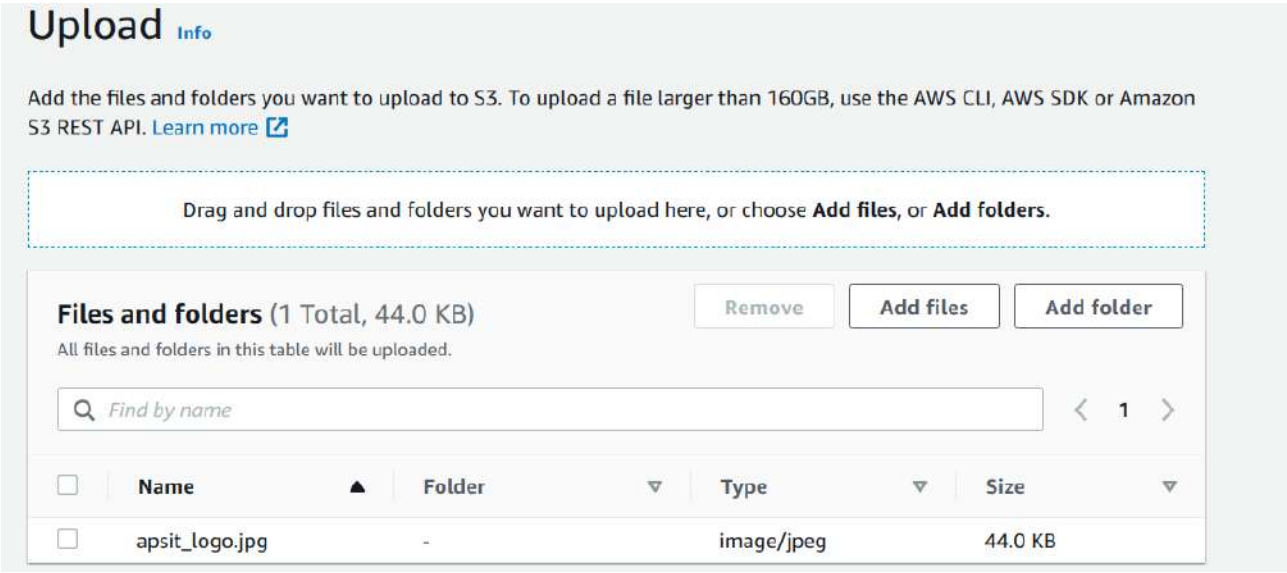
let us save the changes and test the lambda function with S3upload.

**Step 8:**

Now, save the Lambda function. Open S3 from Amazon services and open the bucket we created earlier namely lambdawiths3.

Upload the image in it as shown below −

Click **Add files** to add files. You can also drag and drop the files. Now, click **Upload** button.

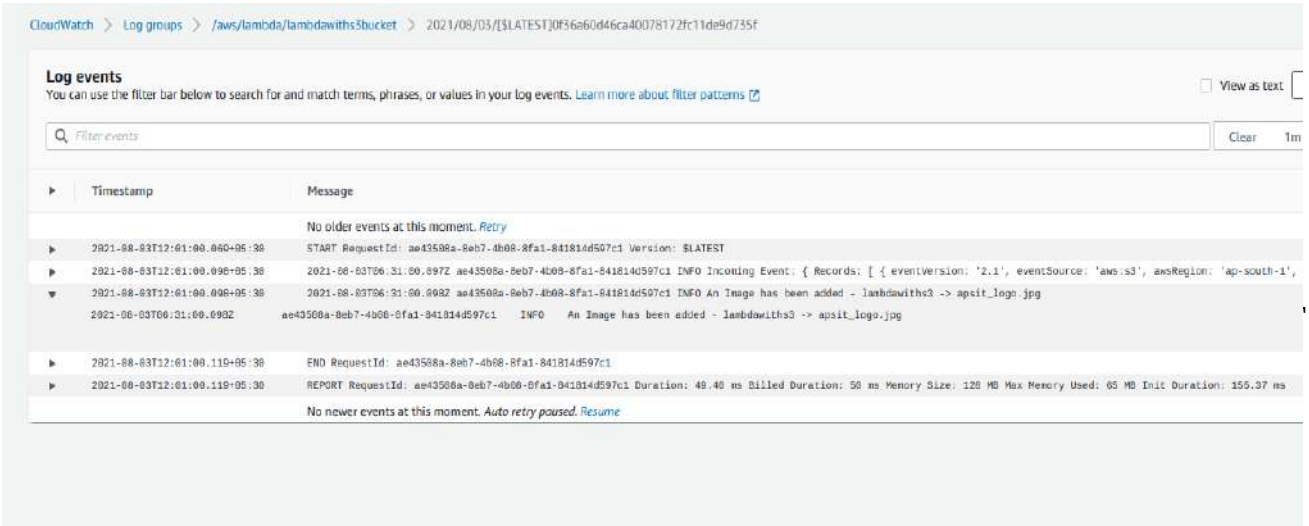



Thus, we have uploaded one image in our S3 bucket.

**Step 9**

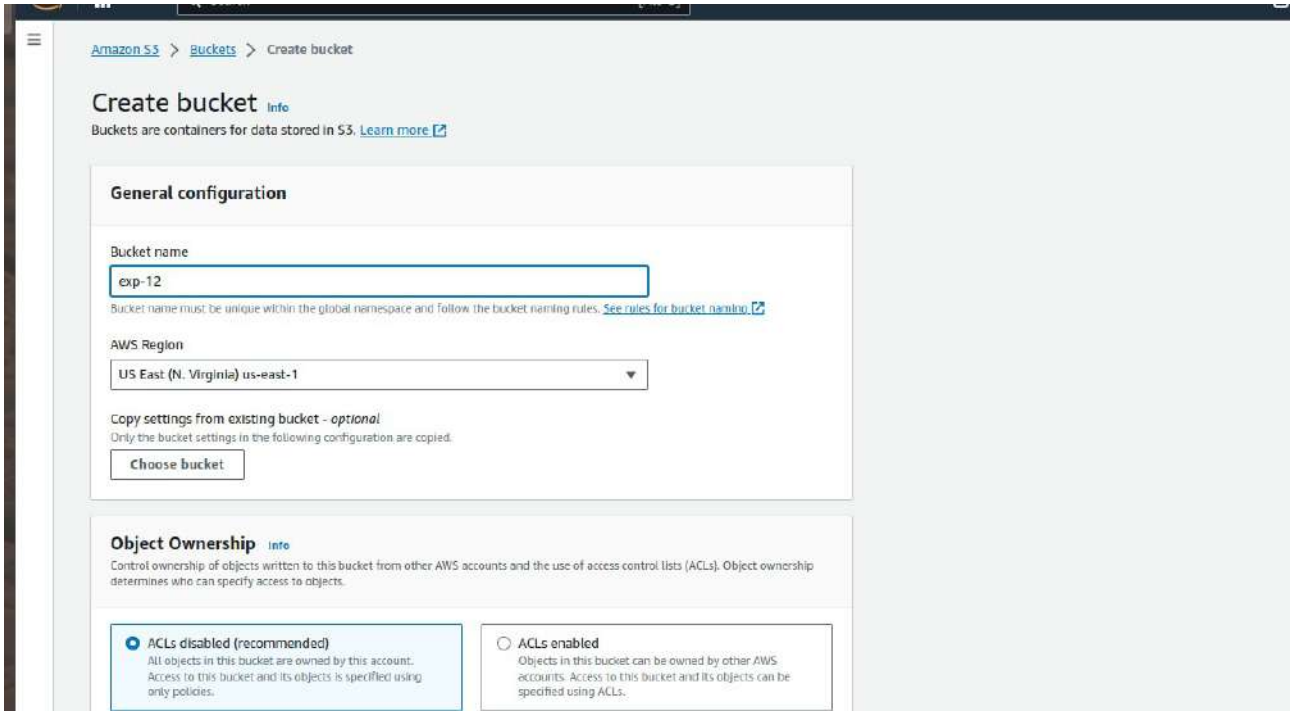
To see the trigger details, go to AWS service and select CloudWatch. Open the logs for the Lambda

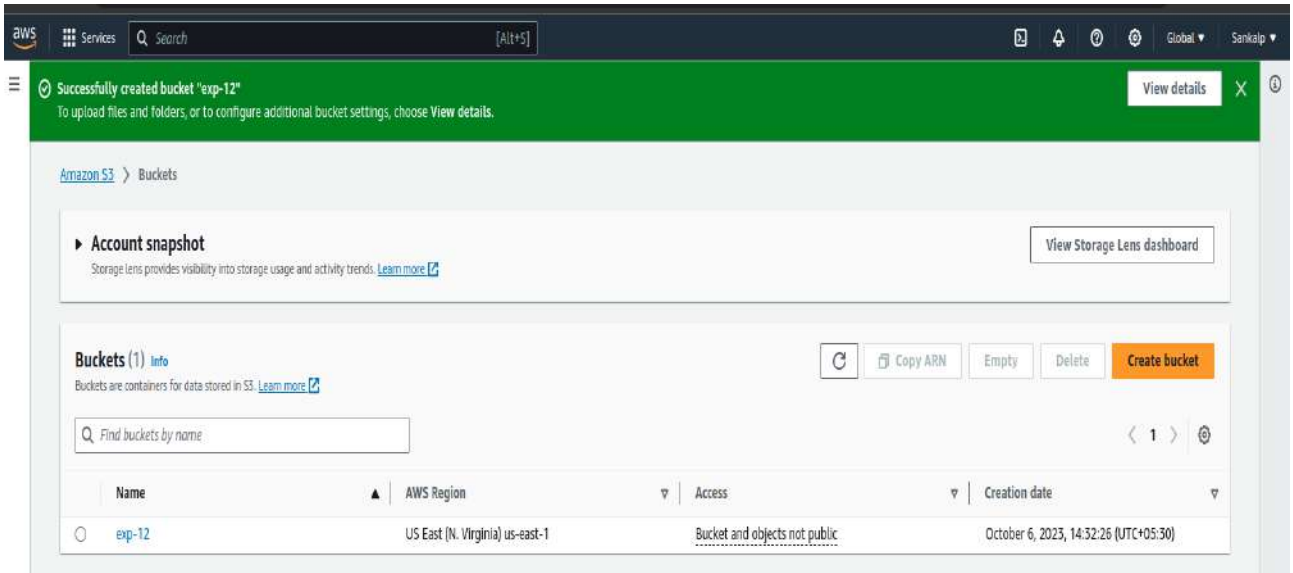
AWS Lambda function gets triggered when file is uploaded in S3 bucket and the details are logged in Cloudwatch as shown below −

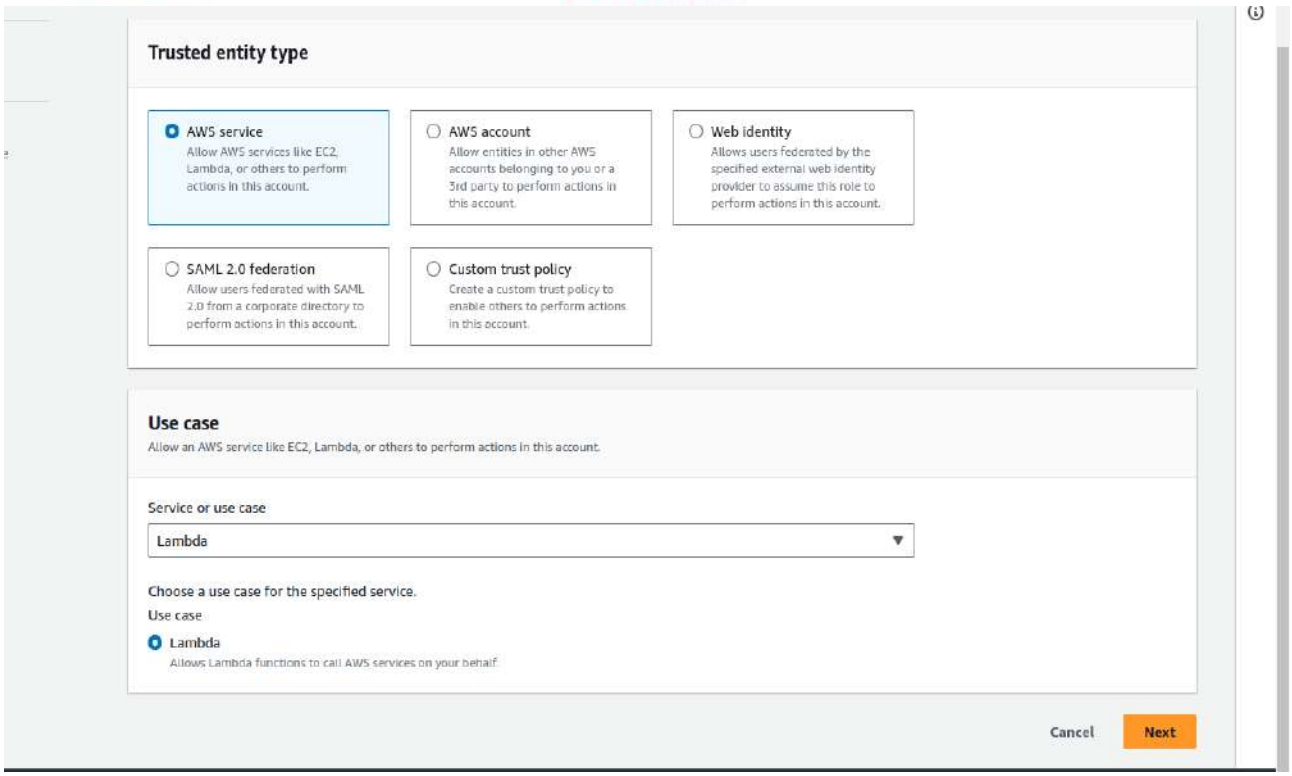


**An image has been Added -> apsit_logo.jpg** you can see in cloudwatch logs.

**OUTPUT:**




Trusted entity type
AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.
Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.
Service or use case
Lambda
Choose a use case for the specified service.
Use case
Lambda
Allows Lambda functions to call AWS services on your behalf.
Cancel
Next

```
15    ]
16 }
```

**Step 2: Add permissions** Edit

Permissions policy summary

| Policy name | Type | Attached as |
|---|---|---|
| AmazonS3FullAccess | AWS managed | Permissions policy |
| AWSLambda_FullAccess | AWS managed | Permissions policy |
| CloudWatchFullAccess | AWS managed | Permissions policy |

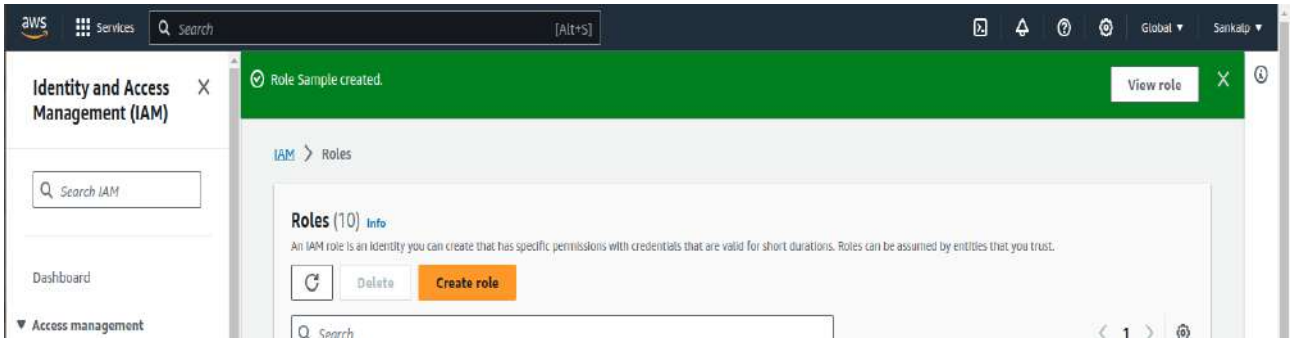
**Step 3: Add tags**

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.
Sample
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

Description
Add a short explanation for this role.
Allows Lambda functions to call AWS services on your behalf.
Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.



Function name
Enter a name that describes the purpose of your function.

exp12

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

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 18.x

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

- x86_64
- arm64

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

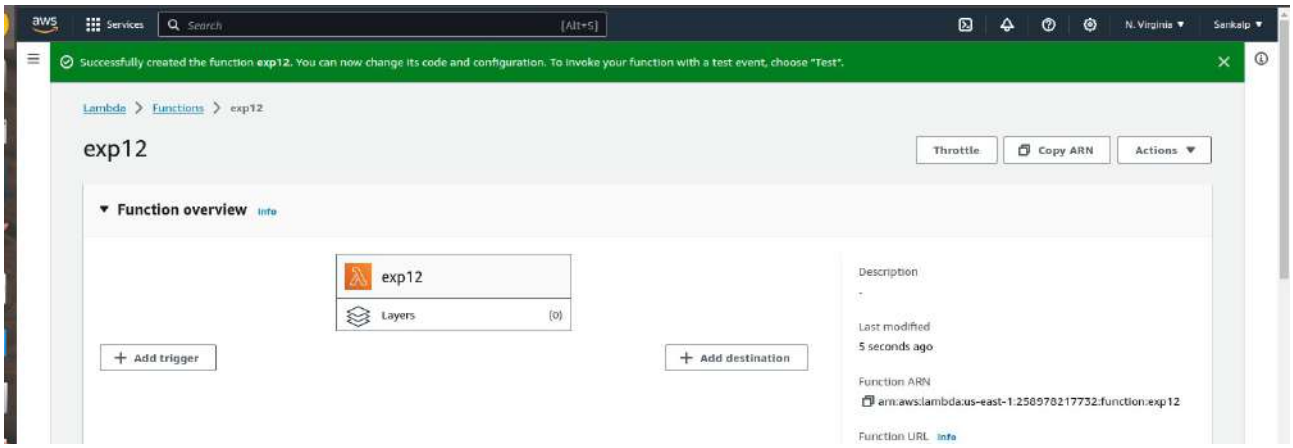
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

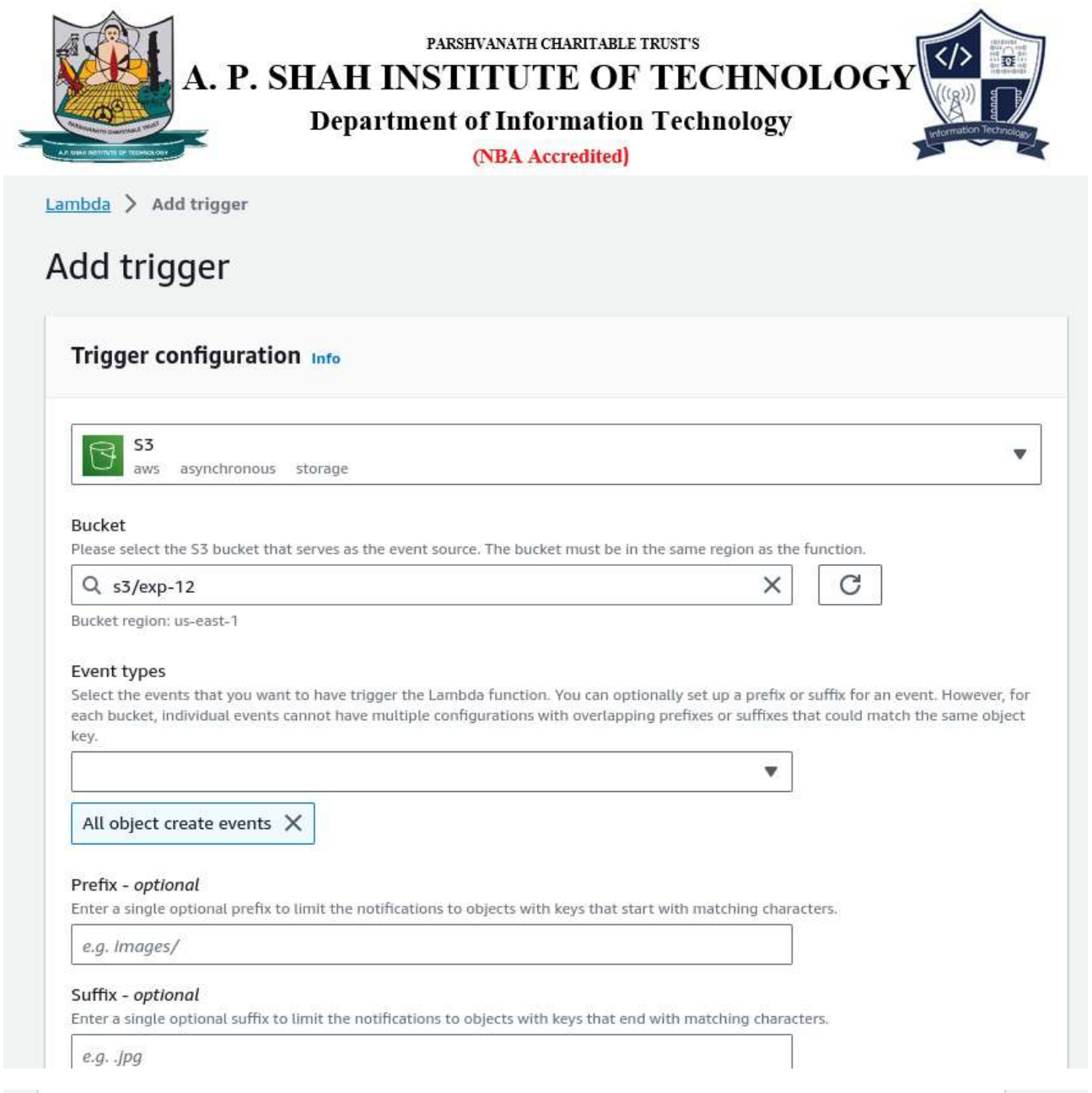
- Create a new role with basic Lambda permissions
- Use an existing role
- Create a new role from AWS policy templates

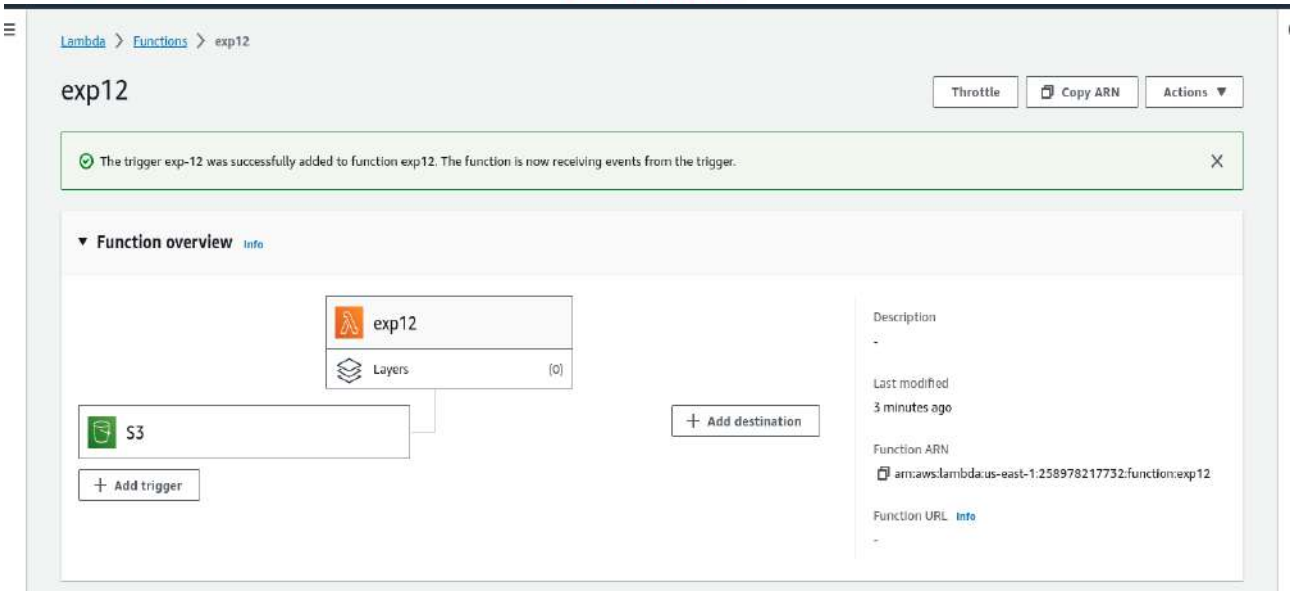
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

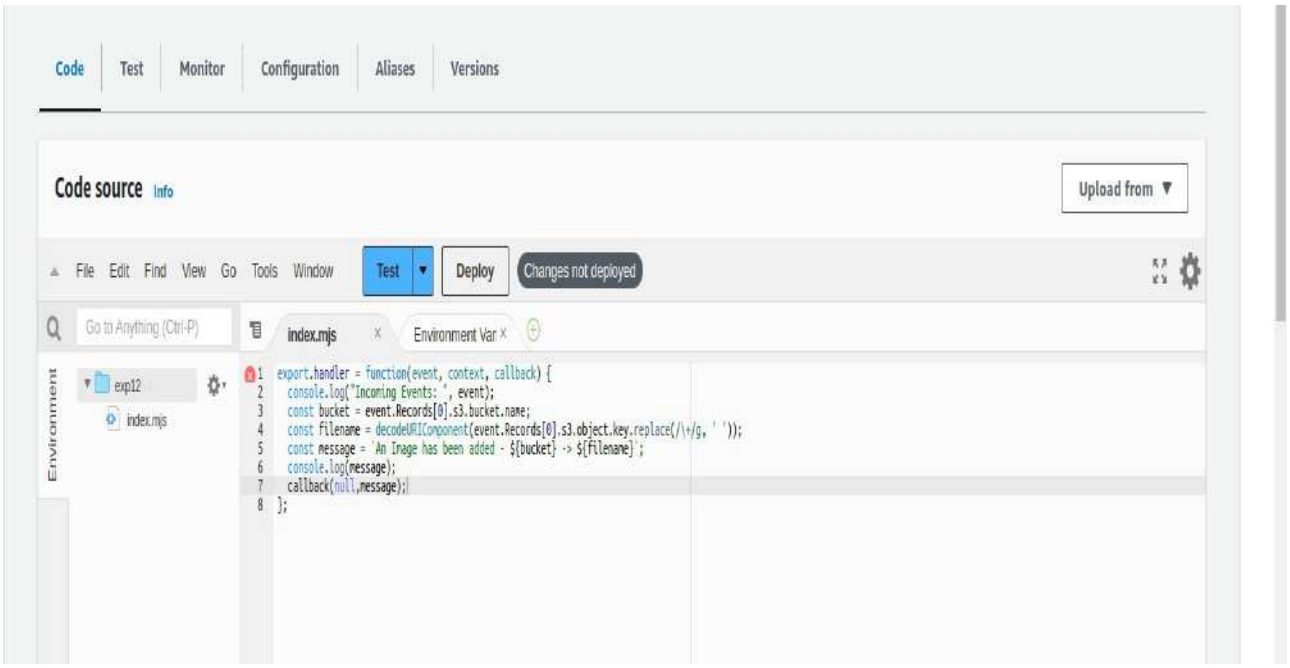
Sample

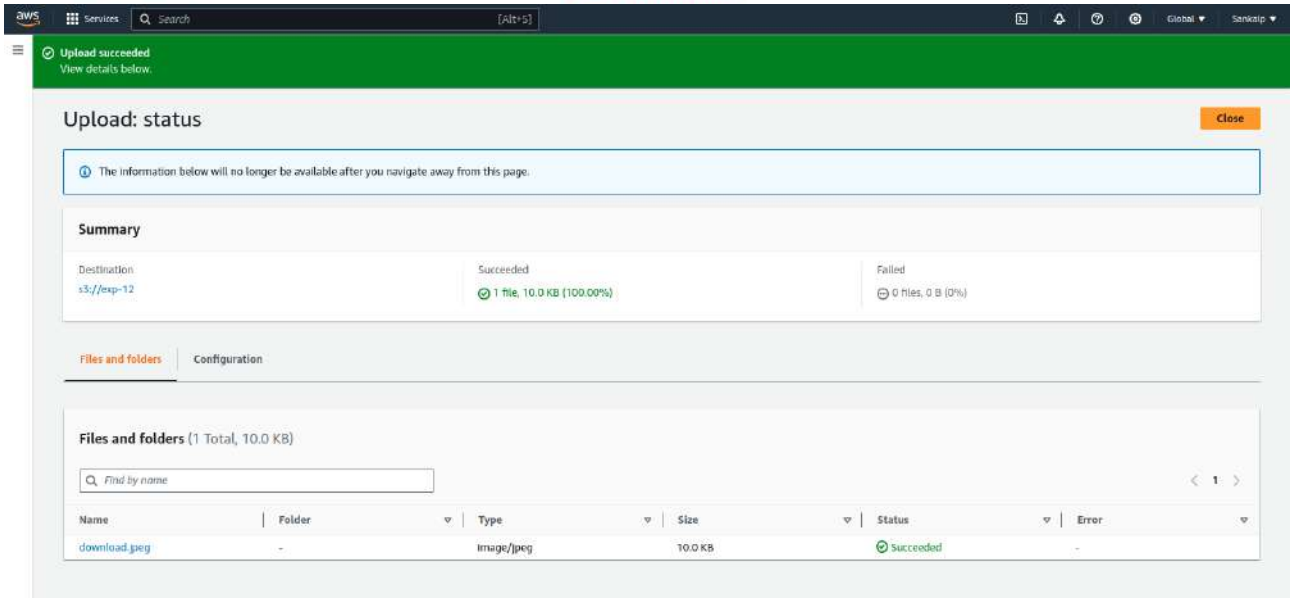
View the Sample role on the IAM console.

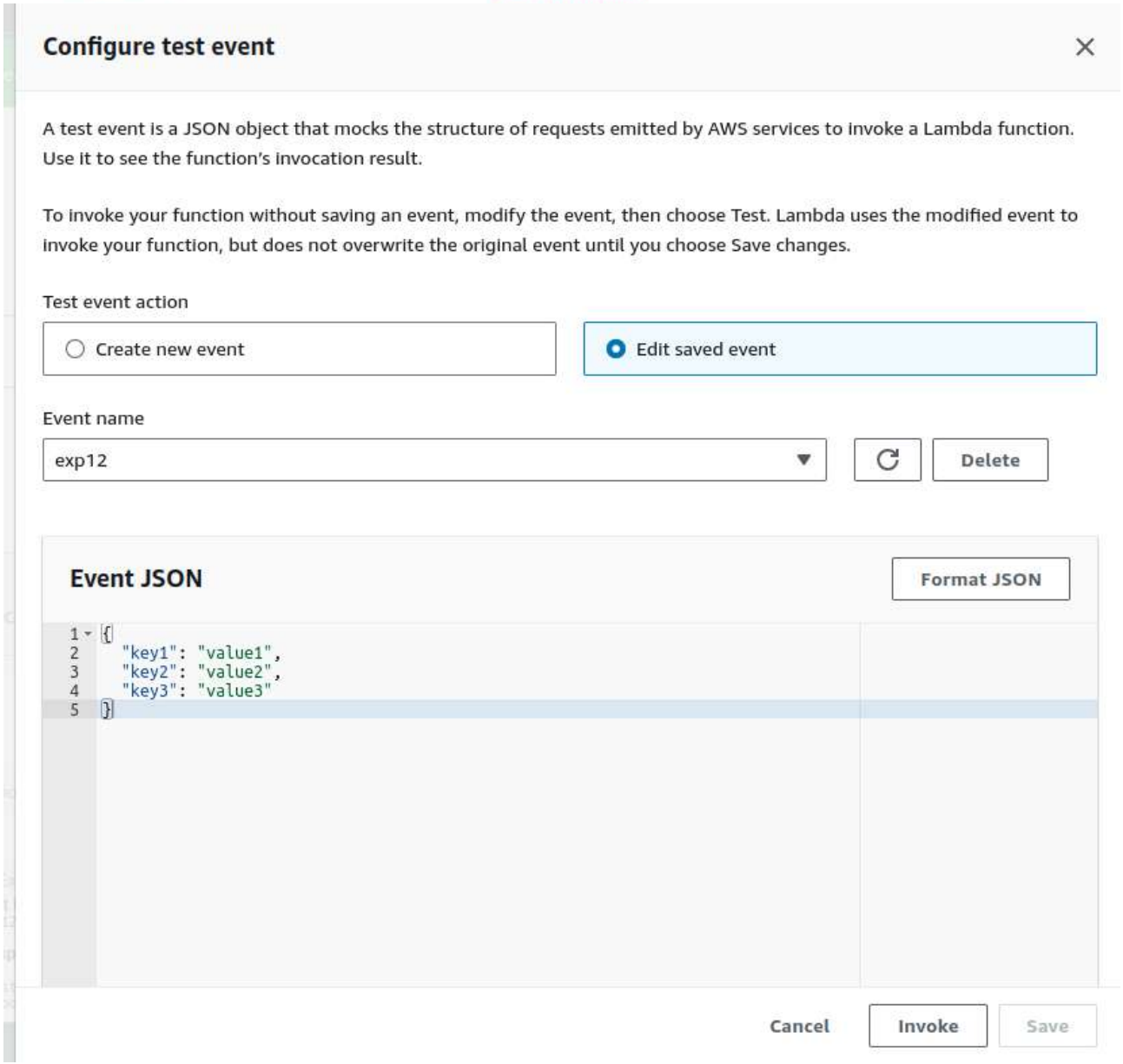


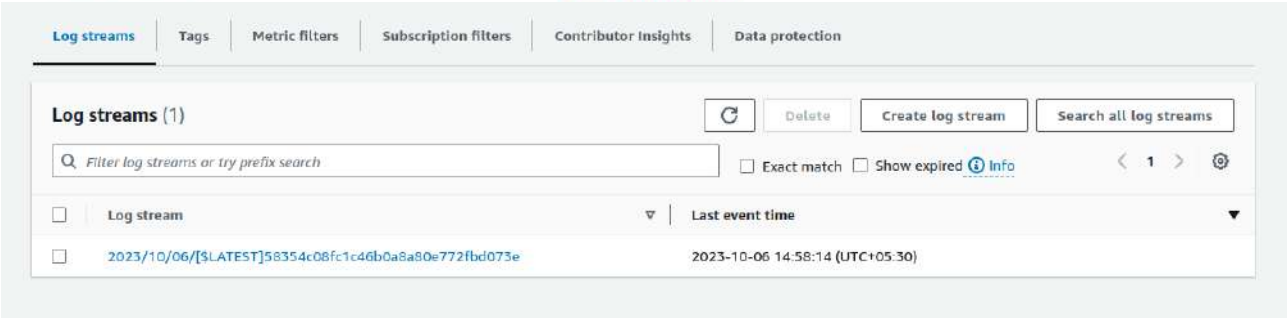

Lambda > Add trigger
Add trigger
Trigger configuration Info
S3
aws asynchronous storage
Bucket
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.
s3/exp-12
Bucket region: us-east-1
Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
All object create events
Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.
e.g. images/
Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.
e.g. .jpg
Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. Learn more
I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.
Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.
Cancel
Add

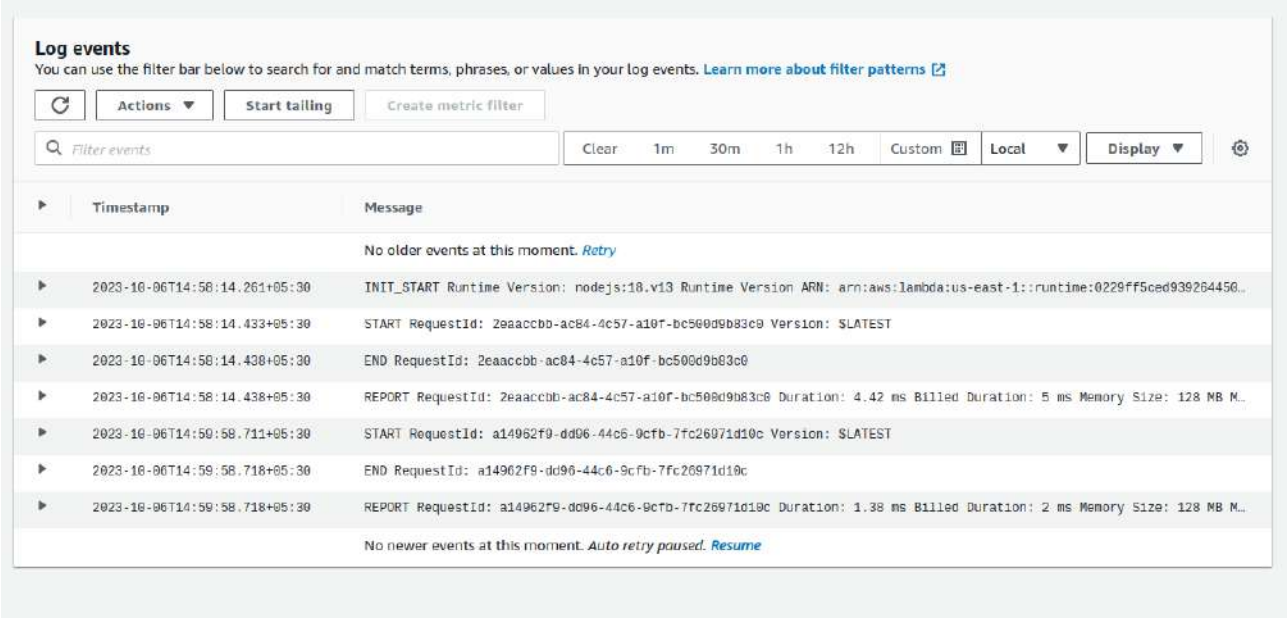

Lambda > Functions > exp12
exp12
Throttle
Copy ARN
Actions
The trigger exp-12 was successfully added to function exp12. The function is now receiving events from the trigger.
Function overview Info
exp12
Layers (0)
S3
Add trigger
Add destination
Description
-
Last modified
3 minutes ago
Function ARN
arn:aws:lambda:us-east-1:258978217732:function:exp12
Function URL Info
-


Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Changes not deployed
Go to Anything (Ctrl-P)
index.mjs
Environment Var
Environment
exp12
index.mjs
export.handler = function(event, context, callback) {
console.log("Incoming Events: ", event);
const bucket = event.Records[0].s3.bucket.name;
const filename = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));
const message = `An Image has been added - ${bucket} -> ${filename}`;
console.log(message);
callback(null,message);
};


Services
Search
[Alt+S]
Global
Sankalp
Upload succeeded
View details below.
Upload: status
Close
The information below will no longer be available after you navigate away from this page.
Summary
Destination
s3://exp-12
Succeeded
1 file, 10.0 KB (100.00%)
Failed
0 files, 0 B (0%)
Files and folders
Configuration
Files and folders (1 Total, 10.0 KB)
Find by name
1
Name
Folder
Type
Size
Status
Error
download.jpeg
-
image/jpeg
10.0 KB
Succeeded
-


Configure test event
A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.
To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes.
Test event action
Create new event
Edit saved event
Event name
exp12
Delete
Event JSON
Format JSON
{
"key1": "value1",
"key2": "value2",
"key3": "value3"
}
Cancel
Invoke
Save




**Conclusion: Hence we have studied and implemented to create a Lambda function which will log "An Image has been added" once you add an object to a specific bucket in S3.**