

Adv DevOps Exp-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:

AWS Lambda and S3 Integration:

AWS Lambda allows you to execute code in response to various events, including those triggered by Amazon S3. When an object is added to an S3 bucket, it can trigger a Lambda function to execute, allowing for event-driven processing without managing servers.

Workflow:

1. Create an S3 Bucket:

- First, create an S3 bucket that will store the objects. This bucket will act as the trigger source for the Lambda function.

2. Create the Lambda Function:

- Set up a new Lambda function using AWS Lambda's console. You can choose a runtime environment like Python, Node.js, or Java.
- Write code that logs a message like “An Image has been added” when triggered.

3. Set Up Permissions:

- Ensure that the Lambda function has the necessary permissions to access S3. You can do this by attaching an IAM role with policies that allow reading from the bucket and writing logs to CloudWatch.

4. Configure S3 Trigger:

- Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).

5. Test the Setup:

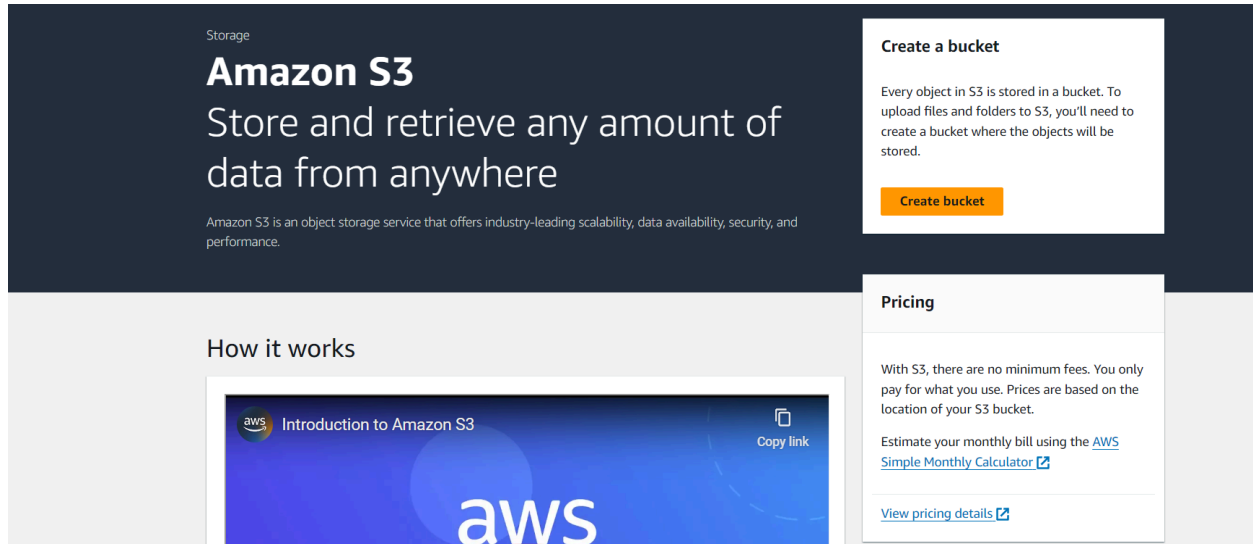
- Upload an object (e.g., an image) to the S3 bucket to test the trigger. The Lambda function should execute and log the message “An Image has been added” in AWS CloudWatch Logs.

Prerequisites: AWS Personal Account

Prerequisites: AWS Personal Account

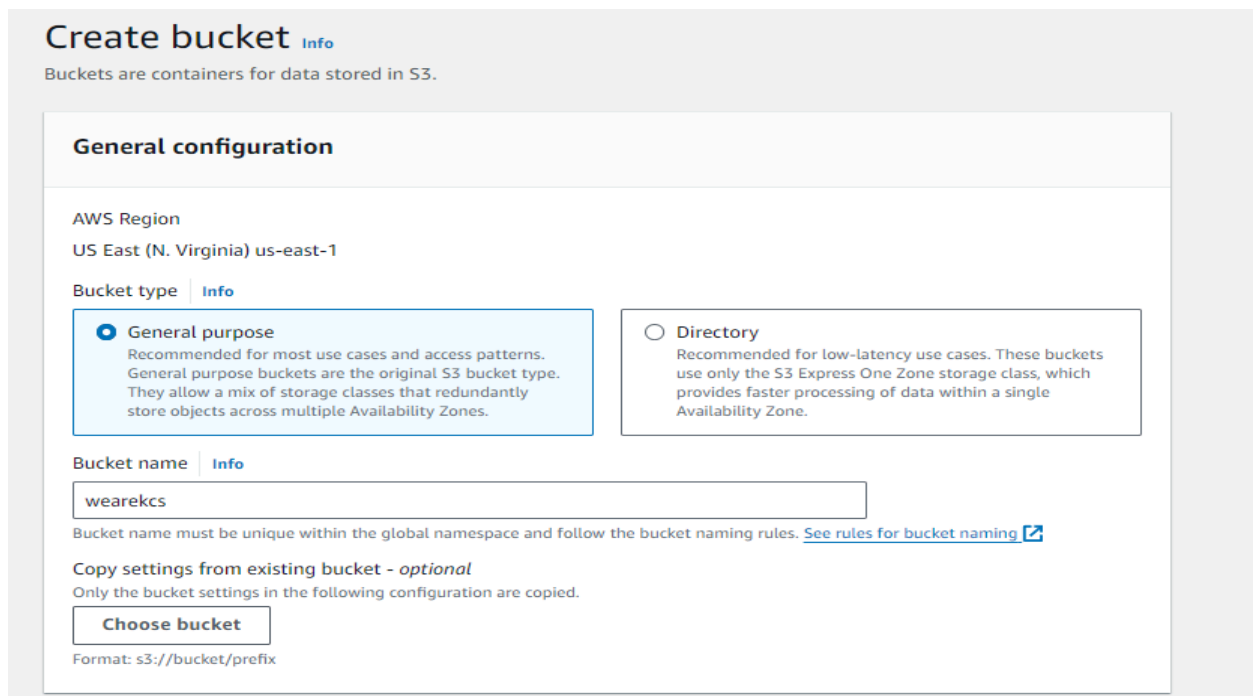
Steps To create the lambda function:

Step 1: Login to your AWS Personal account. Now open S3 from services and click on create S3 bucket.



The image shows the Amazon S3 landing page. On the left, there's a dark blue header with the text "Storage" and "Amazon S3 Store and retrieve any amount of data from anywhere". Below this, it says "Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance." On the right, there's a white box titled "Create a bucket" with the text "Every object in S3 is stored in a bucket. To upload files and folders to S3, you'll need to create a bucket where the objects will be stored." and a "Create bucket" button. Below that, there's a "Pricing" section with text about no minimum fees and a link to the "Simple Monthly Calculator". At the bottom, there's a "How it works" section with a video player titled "Introduction to Amazon S3" and a "Copy link" button.

Step 2: Now Give a name to the Bucket, select general purpose project and deselect the Block public access and keep other this to default.



The image shows the "Create bucket" form in the AWS console. The form has a title "Create bucket" and a subtitle "Buckets are containers for data stored in S3." Below this, there's a "General configuration" section. It includes a "Bucket type" dropdown menu with two options: "General purpose" (selected) and "Directory". The "General purpose" option is described as "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." The "Directory" option is described as "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." Below the bucket type, there's a "Bucket name" field with the text "wearekcs" and a "Choose bucket" button. The form also includes a "Copy settings from existing bucket - optional" section with a "Choose bucket" button. At the bottom, there's a "Format: s3://bucket/prefix" label.

Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

☒ ACLs disabled (recommended)

All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

☐ ACLs enabled

Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership
Bucket owner enforced

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

☐ Block *all* public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☐ Block public access to buckets and objects granted through *new* access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☐ Block public access to buckets and objects granted through *any* access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

☐ Block public access to buckets and objects granted through *new* public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☐ Block public and cross-account access to buckets and objects through *any* public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.



Turning off block all public access might result in this bucket and the objects within becoming

Successfully created bucket "wearekcs"
To upload files and folders, or to configure additional bucket settings, choose [View details](#).

[View details](#)

Amazon S3 > Buckets

Account snapshot - updated every 24 hours [All AWS Regions](#)
Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

[View Storage Lens dashboard](#)

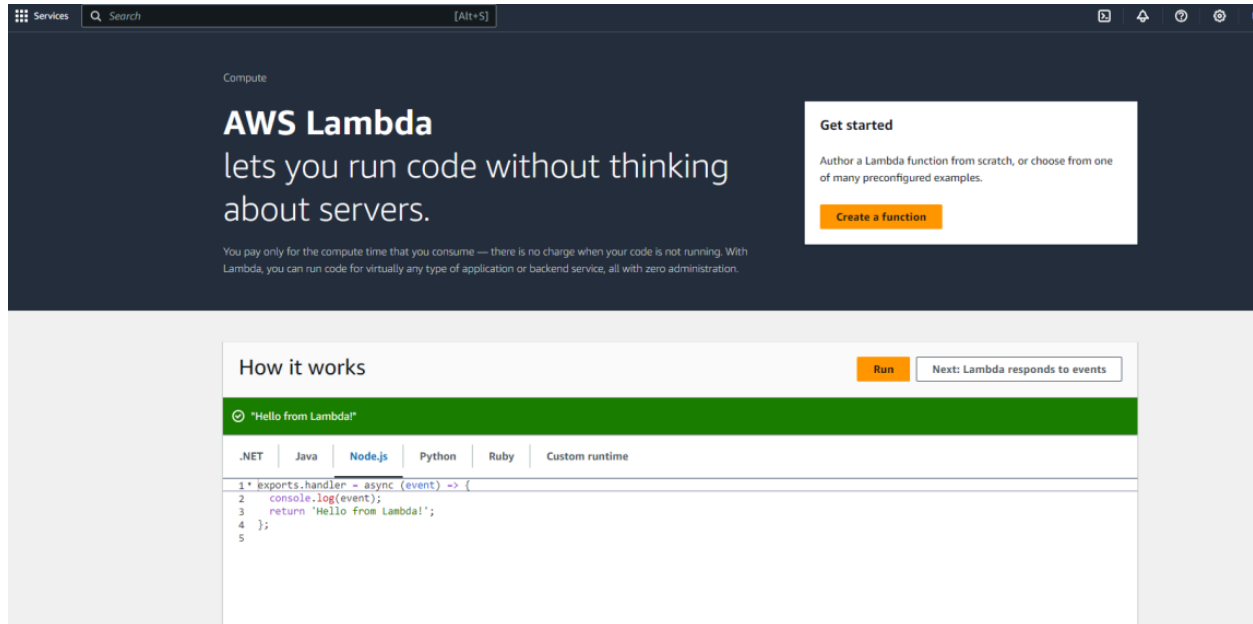
General purpose buckets | Directory buckets

General purpose buckets (1) [Info](#) [All AWS Regions](#)

Buckets are containers for data stored in S3.

[Copy](#) [Copy ARN](#) [Empty](#) [Delete](#) [Create bucket](#)

Name	AWS Region	IAM Access Analyzer	Creation date
<input type="radio"/> wearekcs	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 1, 2024, 13:40:40 (UTC+05:30)

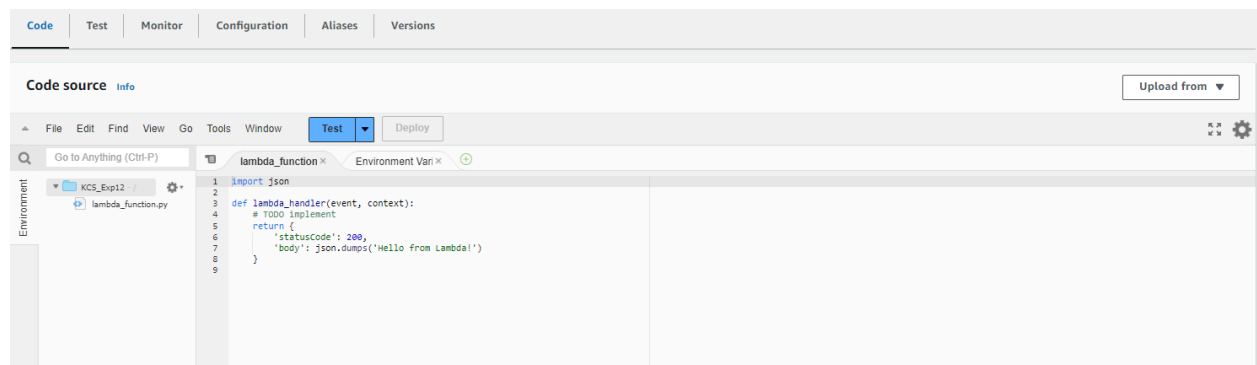
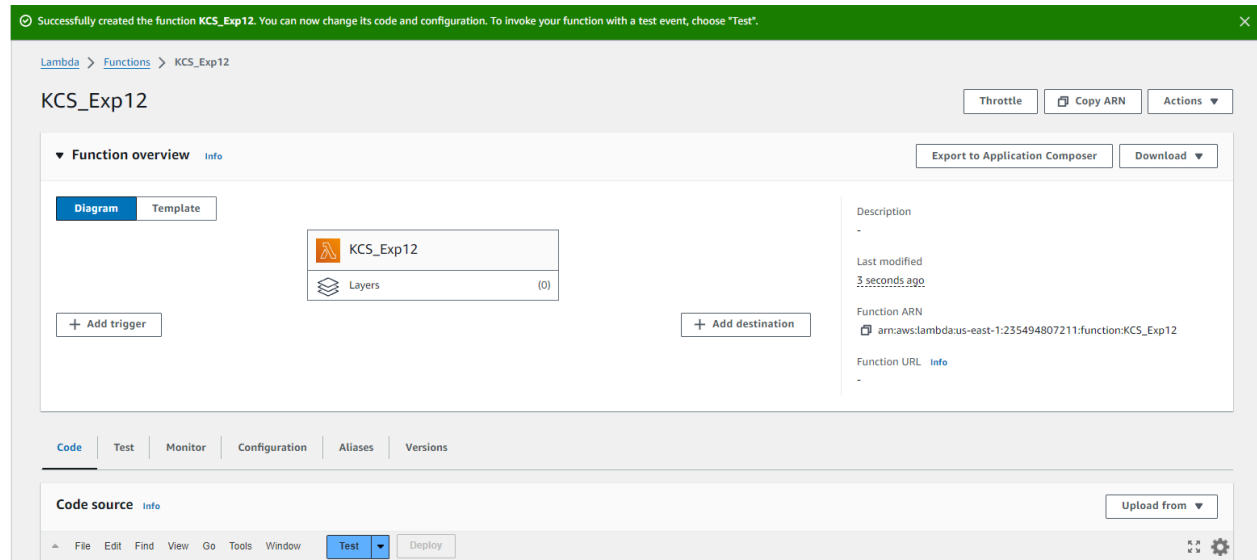
Step 3: Open lambda console and click on create function button

Step 4: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12 , Architecture as x86, and Execution role to Create a new role with basic Lambda permissions.

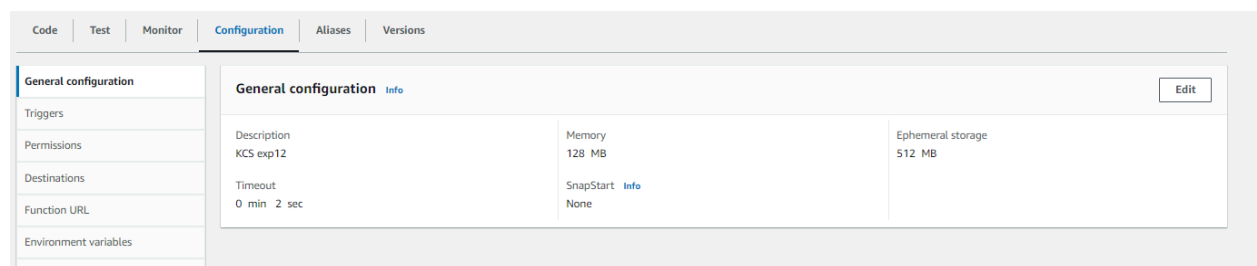
The screenshot shows the 'Create function' page in the AWS Lambda console. The page has a breadcrumb trail: 'Lambda > Functions > Create function'. The main heading is 'Create function' with an 'Info' link. Below the heading, there are three options to create a function: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. The 'Author from scratch' option is highlighted with a blue border. Below the options, there is a 'Basic information' section. It contains the following fields and options:

- Function name:** A text input field with the value 'KCS_Exp12'. Below the field, it says 'Use only letters, numbers, hyphens, or underscores with no spaces.'
- Runtime:** A dropdown menu with 'Python 3.12' selected. To the right of the dropdown is a refresh button. Below the dropdown, it says 'Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.'
- Architecture:** Two radio buttons: 'x86_64' (selected) and 'arm64'. Below the radio buttons, it says 'Choose the instruction set architecture you want for your function code.'
- Permissions:** A section with an 'Info' link. It says '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.'

At the bottom of the 'Basic information' section, there are two buttons: 'Change default execution role' and 'Advanced settings'.



To See or Edit the basic settings go to configuration then click on edit general setting



Change any setting of your choice. Here I have set a timeout of 2 secs. Then save changes

[Lambda](#) > [Functions](#) > [KCS_Exp12](#) > Edit basic settings

Edit basic settings

Basic settings [Info](#)

Description - optional

Memory [Info](#)

Your function is allocated CPU proportional to the memory configured.

 MB

Set memory to between 128 MB and 10240 MB

Ephemeral storage [Info](#)

You can configure up to 10 GB of ephemeral storage (/tmp) for your function. [View pricing](#)

 MB

Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart [Info](#)

Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the [SnapStart compatibility considerations](#).

None

Supported runtimes: Java 11, Java 17, Java 21.

Timeout

 min sec

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☒ 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.

service-role/KCS_Exp12-role-0q6h1t4r

↻

[View the KCS_Exp12-role-0q6h1t4r role](#) on the IAM console.

Step 5: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select s3 put template.

Code **Test** Monitor Configuration Aliases Versions

Test event [Info](#) Save Test

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

☒ Create new event ☐ Edit saved event

Event name

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

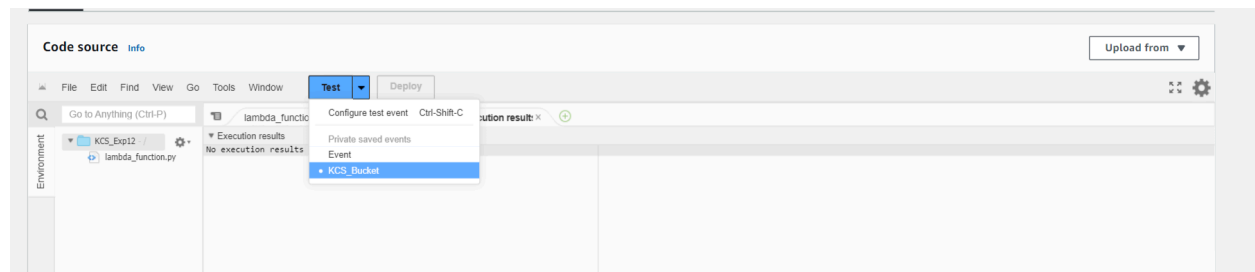
s3-put

Event JSON Format JSON

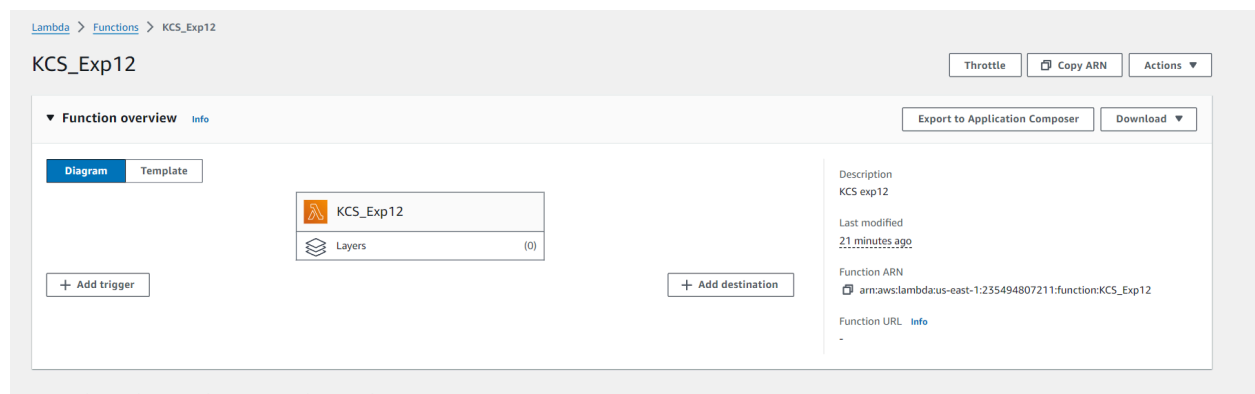
```
Event JSON
Format JSON

1 {
2   "Records": [
3     {
4       "eventVersion": "2.0",
5       "eventSource": "aws:s3",
6       "awsRegion": "us-east-1",
7       "eventTime": "1970-01-01T00:00:00.000Z",
8       "eventName": "ObjectCreated:Put",
9       "userIdentity": {
10        "principalId": "EXAMPLE"
11      },
12      "requestParameters": {
13        "sourceIPAddress": "127.0.0.1"
14      },
15      "responseElements": {
16        "x-amz-request-id": "EXAMPLE123456789",
17        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
18      },
19      "s3": {
20        "s3SchemaVersion": "1.0",
21        "configurationId": "testConfigRule",
22        "bucket": {
23          "name": "example-bucket",
24          "ownerIdentity": {
25            "principalId": "EXAMPLE"
26          },
27          "arn": "arn:aws:s3:::example-bucket"
28        },
29        "object": {
30          "key": "test2fkey",
```

Step 6: Now In the Code section select the created event from the dropdown .



Step 7: Now In the Lambda function click on add trigger




Now select the source as S3 then select the bucket name from the dropdown, keep other things to default and also you can add prefix to image.

[Lambda](#) > Add triggers

Add trigger

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.

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. Any [special characters](#) must be URL encoded.

Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any [special characters](#) must be URL encoded.

Recursive invocation



KCS_Exp12


 The trigger wearekcs was successfully added to function KCS_Exp12. The function is now receiving events from the trigger.

Function overview [Info](#)

Diagram


Template

 **KCS_Exp12**
 Layers (0)

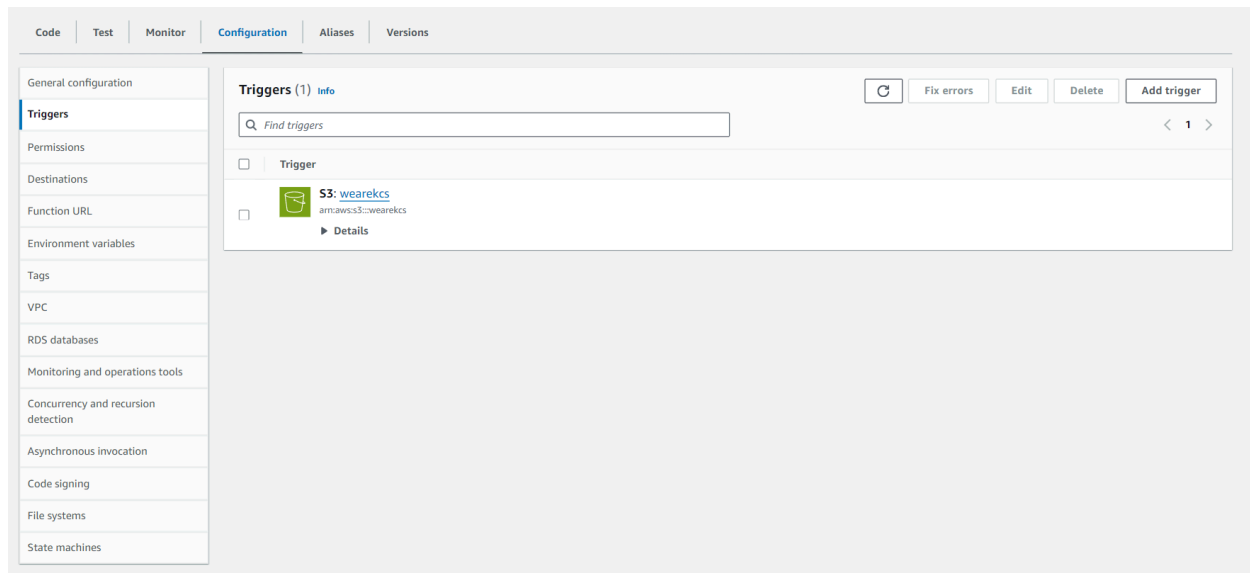
 **S3**

Description
KCS exp12

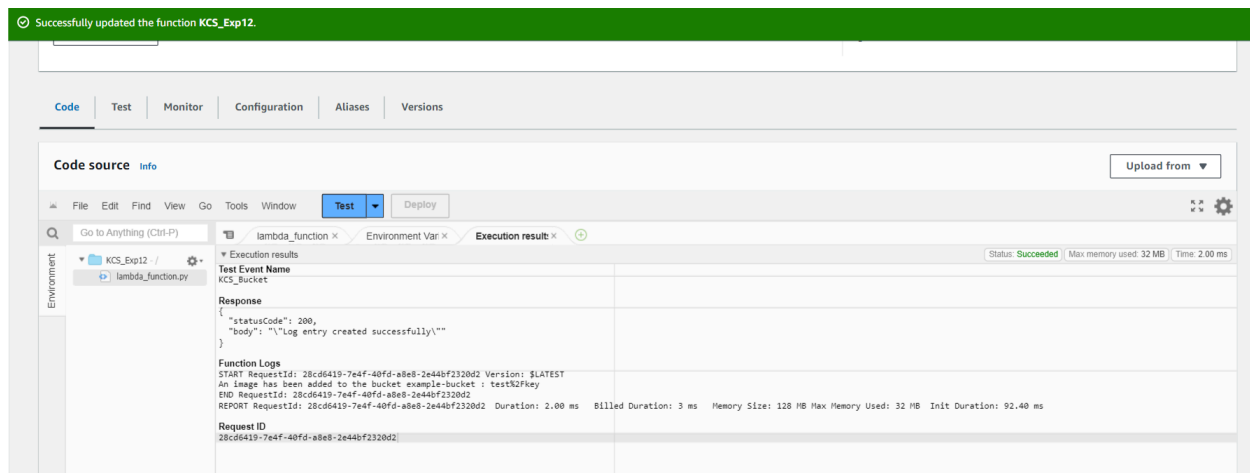
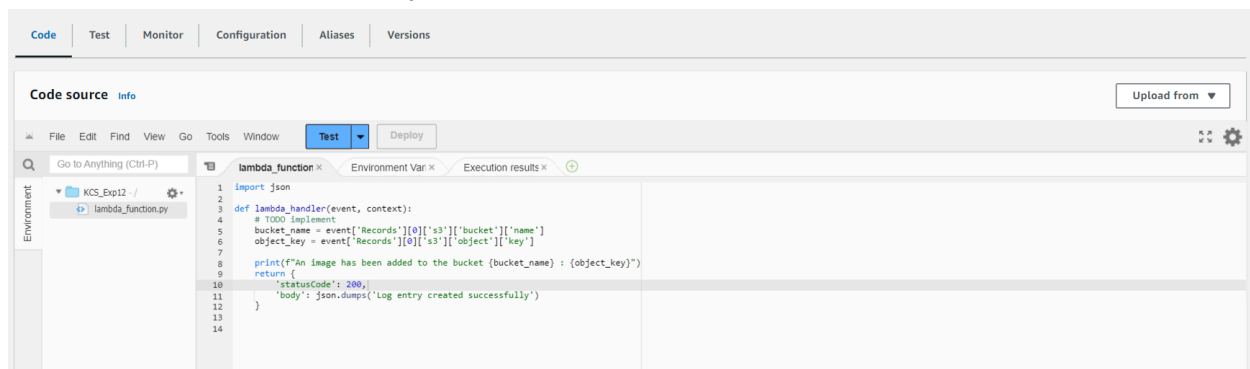
Last modified
26 minutes ago

Function ARN
 arnaws:lambda:us-east-1:235494807211:function:KCS_Exp12

Function URL [Info](#)



Step 8: Now Write code that logs a message like “An Image has been added” when triggered. Save the file and click on deploy.



Step 9: Now upload any image to the bucket.

[Amazon S3](#) > [Buckets](#) > [wearekcs](#) > Upload

Upload [Info](#)

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (1 Total, 957.0 KB)

[Remove](#)[Add files](#)[Add folder](#)

All files and folders in this table will be uploaded.

< 1 >

<input type="checkbox"/>	Name	Folder
<input type="checkbox"/>	F_iOUxsXgAAXB2s.jpg	-

Destination [Info](#)

Destination

[s3://wearekcs](#)

► Destination details

Bucket settings that impact new objects stored in the specified destination.

► Permissions

Grant public access and access to other AWS accounts.

► Properties

Specify storage class, encryption settings, tags, and more.

[Cancel](#)[Upload](#)

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://wearekcs	Succeeded 1 file, 957.0 KB (100.00%)	Failed 0 files, 0 B (0%)
------------------------------	---	-----------------------------

Files and folders (1 Total, 957.0 KB)

Find by name

Name	Folder	Type	Size	Status	Error
F_10UoXgA...	-	image/jpeg	957.0 KB	Succeeded	-

Step 10: Now to click on test in lambda to check whether it is giving log when image is added to S3

Code Test Monitor Configuration Aliases Versions

Code source Info Upload from

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl-P)

Environment

- KCS_Exp12
- lambda_function.py

Execution results

Test Event Name
KCS_Bucket

Response

```
{
  "statusCode": 200,
  "body": "\\Log entry created successfully\\n"
}
```

Function Logs

START RequestId: ba624cc5-6862-4d62-84ca-6a1bf867d831 Version: \$LATEST

An image has been added to the bucket example-bucket : test%2Fkey

END RequestId: ba624cc5-6862-4d62-84ca-6a1bf867d831

REPORT RequestId: ba624cc5-6862-4d62-84ca-6a1bf867d831 Duration: 1.88 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 32 MB

Request ID
ba624cc5-6862-4d62-84ca-6a1bf867d831

Step 11: Now Lets see the log on Cloud watch.To see it go to monitor section and then click on view cloudwatch logs.

CloudWatch > Log groups > /aws/lambda/KCS_Exp12 > 2024/10/01/[\$LATEST]b93d5fc4cf3b4bf8802c5b106ff03bde

Log events Actions Start tailing Create metric filter

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](#)

Filter events - press enter to search

Clear 1m 30m 1h 12h Custom UTC timezone Display

Timestamp	Message
No older events at this moment. Retry	
2024-10-01T08:55:09.068Z	INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:188d9ca2e2714ff5637bd2bbe...
2024-10-01T08:55:09.163Z	START RequestId: 28cd6419-7e4f-40fd-a8e8-2e44bf2320d2 Version: \$LATEST
2024-10-01T08:55:09.164Z	An image has been added to the bucket example-bucket : test%2Fkey
2024-10-01T08:55:09.174Z	END RequestId: 28cd6419-7e4f-40fd-a8e8-2e44bf2320d2
2024-10-01T08:55:09.174Z	REPORT RequestId: 28cd6419-7e4f-40fd-a8e8-2e44bf2320d2 Duration: 2.00 ms Billed Duration: 3 ms Memory Size: 128 MB Max Memor...
2024-10-01T08:59:18.675Z	START RequestId: ba624cc5-6862-4d62-84ca-6a1bf867d831 Version: \$LATEST
2024-10-01T08:59:18.676Z	An image has been added to the bucket example-bucket : test%2Fkey
2024-10-01T08:59:18.678Z	END RequestId: ba624cc5-6862-4d62-84ca-6a1bf867d831
2024-10-01T08:59:18.678Z	REPORT RequestId: ba624cc5-6862-4d62-84ca-6a1bf867d831 Duration: 1.88 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memor...
No newer events at this moment. Auto retry paused Resume	