

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.

Step 1: On your AWS console, click on ‘S3’ in the services section and click on ‘Create bucket’. Give your bucket a name.

Create bucket [Info](#)

Buckets are containers for data stored in S3.

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](#)

expad12

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

Uncheck the ‘Block all public access’ box.

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.

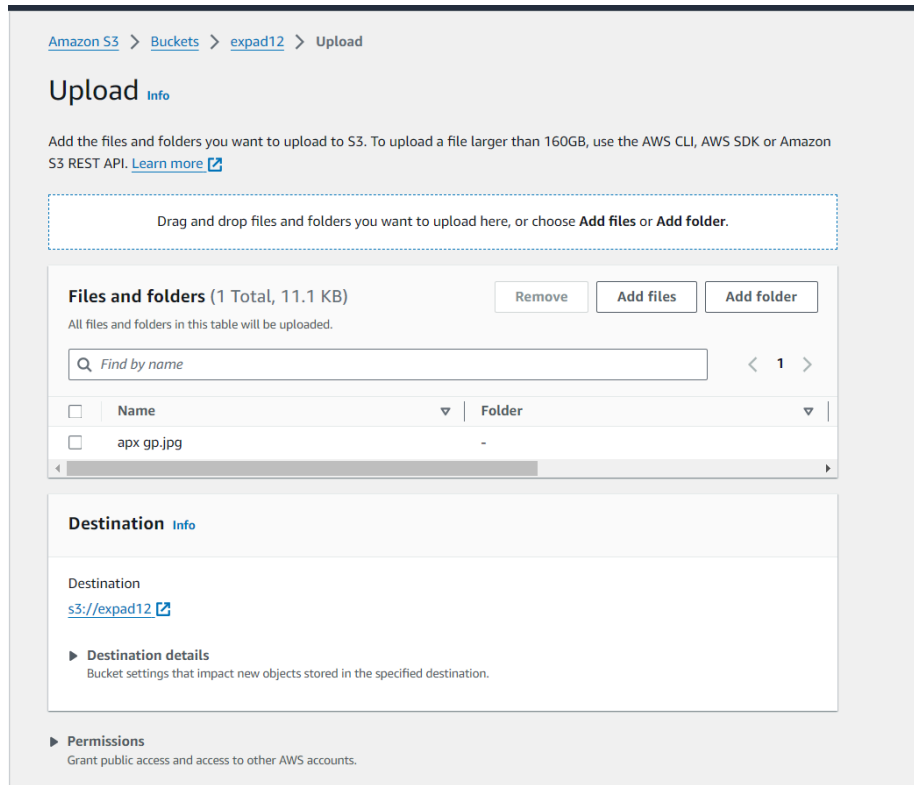
☐ Block public access to buckets and objects granted through *new* public bucket or access point policies
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☐ 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.

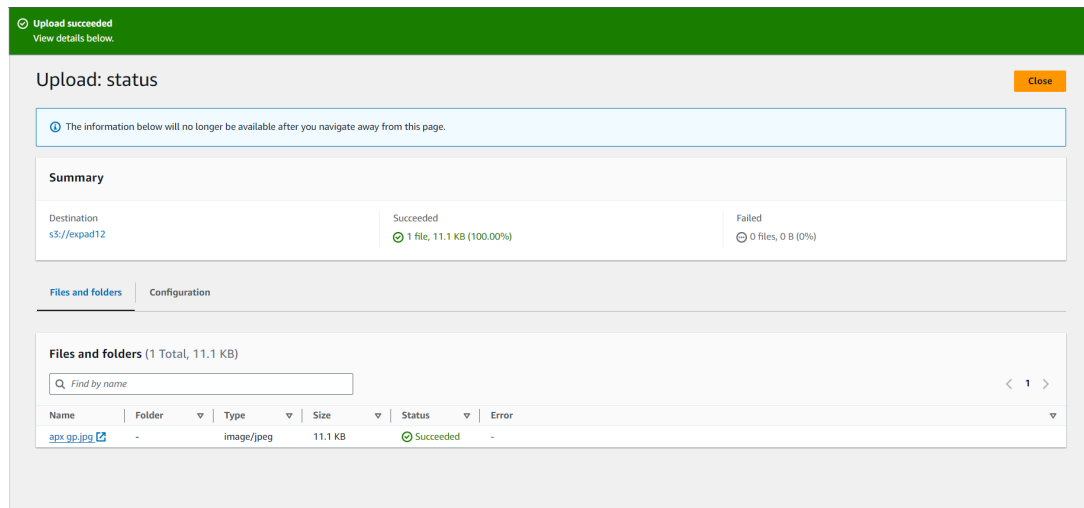
 **Turning off block all public access might result in this bucket and the objects within becoming public**
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

☒ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Step 2: Upload an image onto your S3 bucket by clicking on your S3 bucket, clicking on 'Upload', clicking on 'Add files', navigating to your image and selecting it.



Your image gets uploaded onto the S3 bucket



Step 3: Navigate to the AWS Lambda console using the 'Services' section. Click on 'Create function'.

Lambda > Functions

Functions (6) Last fetched 4 seconds ago Actions Create function

Filter by tags and attributes or search by keyword

<input type="checkbox"/>	Function name	Description	Package type	Runtime	Last modified
<input type="checkbox"/>	RedshiftOverwatch	Deletes Redshift Cluster if the count is more than 2.	Zip	Python 3.8	2 months ago
<input type="checkbox"/>	RoleCreationFunction	Create SLR if absent	Zip	Python 3.8	2 months ago
<input type="checkbox"/>	RedshiftEventSubscription	Create Redshift event subscription to SNS Topic.	Zip	Python 3.8	2 months ago
<input type="checkbox"/>	MainMonitoringFunction	-	Zip	Python 3.8	2 months ago
<input type="checkbox"/>	ModLabRole	updates LabRole to allow it to assume itself	Zip	Python 3.8	2 months ago
<input type="checkbox"/>	veydant	-	Zip	Node.js 20.x	1 day ago

Step 4: Give your function a name and keep other settings as default

aws Services Search [Alt+S]

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

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](#).

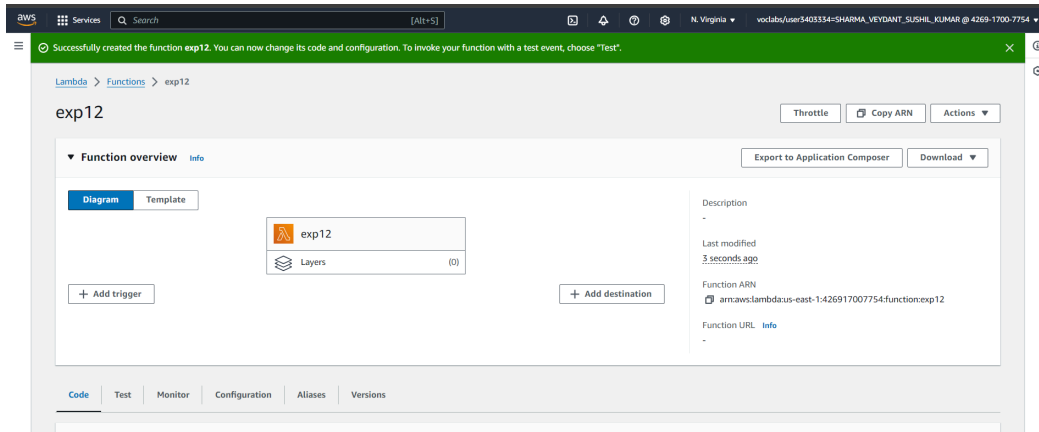
☐ Create a new role with basic Lambda permissions
☒ Use an existing role
☐ Create a new role from AWS policy templates

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.

[View the LabRole role](#) on the IAM console.


Step 5: On the page of the function you created, click on 'Add trigger'

Step 6: Choose 'Trigger configuration' as S3 and select the name of your bucket in the dropdown box below it. Keep other options as default and click on 'Add'.



The trigger gets successfully added to your function

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.

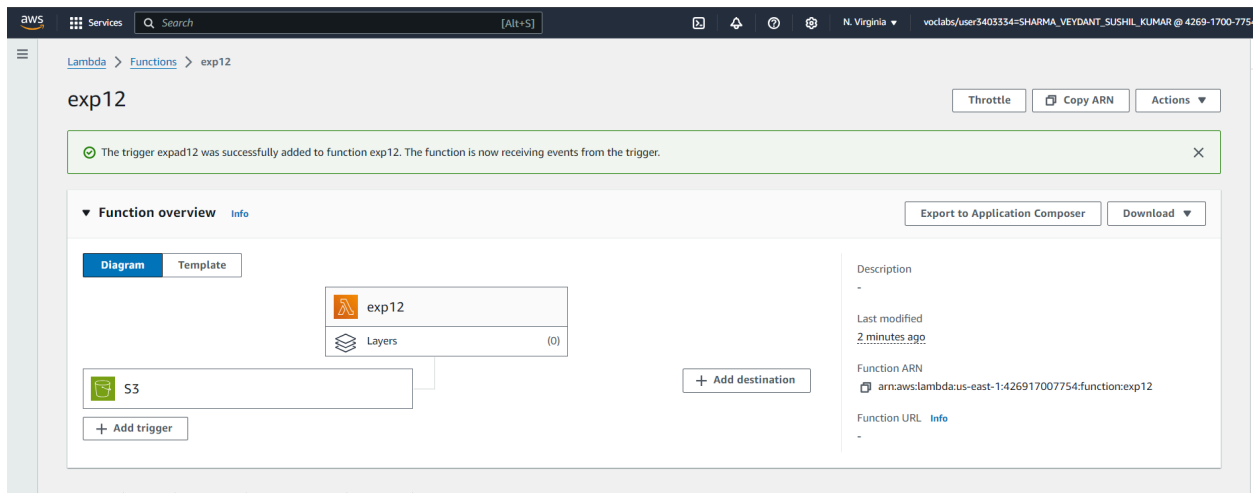
Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any [special characters](#) must be URL encoded.

Recursive invocation
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

Step 7: In the 'Code source' section of your function, paste the following javascript code instead of the existing code:-

```
export const handler = async (event) => { if (!event.Records || event.Records.length === 0) {  
  console.error("No records found in the event."); return { statusCode: 400, body:  
    JSON.stringify('No records found in the event') }; } // Extract bucket name and object key from  
the event const record = event.Records[0]; const bucketName = record.s3.bucket.name; const  
objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle encoded  
keys console.log(` An image has been added to the bucket ${bucketName}: ${objectKey}`);  
console.log(` Event Source: ${record.eventSource}`); console.log(` Event Source:  
${record.eventSource}`); console.log(` Event Source: ${record.eventSource}`);  
console.log(` Event Source: ${record.eventSource}`); return { statusCode: 200, body:  
  JSON.stringify('Log entry created successfully!') }; };
```



Step 8: Click on the arrow next to the 'Test' button and click on 'Configure test event'. In the popup box that appears, if you have an existing event, enter the name of your event or create a new event and add the code.

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, configure the JSON event, then choose Test.

Test event action

Create new event

Edit saved event

Event name

lambdaevent

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

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    }
16  ]
17 }
```

Cancel

Invoke

Save

Step 9: Response generated after testing the code logs that the image was uploaded successfully.

lambdaevent	
<div>Response</div> <div><pre>{ "statusCode": 200, "body": "\"Log entry created successfully!\"" }</pre></div>	
<div>Function Logs</div> <div>START RequestId: 6466cb07-530d-4873-b7d9-c5d656525014 Version: \$LATEST</div> <div>2024-10-10T20:01:27.106Z 6466cb07-530d-4873-b7d9-c5d656525014 INFO An image has been added to the bucket example-bucket: test/key</div> <div>2024-10-10T20:01:27.129Z 6466cb07-530d-4873-b7d9-c5d656525014 INFO Event Source: aws:s3</div> <div>2024-10-10T20:01:27.130Z 6466cb07-530d-4873-b7d9-c5d656525014 INFO Event Source: aws:s3</div> <div>2024-10-10T20:01:27.130Z 6466cb07-530d-4873-b7d9-c5d656525014 INFO Event Source: aws:s3</div> <div>END RequestId: 6466cb07-530d-4873-b7d9-c5d656525014</div> <div>REPORT RequestId: 6466cb07-530d-4873-b7d9-c5d656525014 Duration: 45.26 ms Billed Duration: 46 ms Memory Size: 128 MB Max Memory Used: 64 MB Init Duration: 143.23 ms</div>	
<div>Request ID</div> <div>6466cb07-530d-4873-b7d9-c5d656525014</div>	

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
2024-10-10T15:09:08.226Z	INIT_START Runtime Version: nodejs128.v39 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:ad9b28ae2310fc4c3325e183024ccb4d90e1aa14796d98295f898140841242f7
2024-10-10T15:09:08.362Z	START RequestId: 52f08781-87fe-4178-9fa9-0d39b07cc62f Version: \$LATEST
2024-10-10T15:09:08.363Z	2024-10-10T15:09:08.363Z 52f08781-87fe-4178-9fa9-0d39b07cc62f INFO An image has been added to the bucket example-bucket: test/key
2024-10-10T15:09:08.377Z	2024-10-10T15:09:08.377Z 52f08781-87fe-4178-9fa9-0d39b07cc62f INFO Event Source: aws:s3
2024-10-10T15:09:08.394Z	2024-10-10T15:09:08.394Z 52f08781-87fe-4178-9fa9-0d39b07cc62f INFO Event Source: aws:s3
2024-10-10T15:09:08.394Z	2024-10-10T15:09:08.394Z 52f08781-87fe-4178-9fa9-0d39b07cc62f INFO Event Source: aws:s3
2024-10-10T15:09:08.394Z	2024-10-10T15:09:08.394Z 52f08781-87fe-4178-9fa9-0d39b07cc62f INFO Event Source: aws:s3
2024-10-10T15:09:08.416Z	END RequestId: 52f08781-87fe-4178-9fa9-0d39b07cc62f
2024-10-10T15:09:08.416Z	REPORT RequestId: 52f08781-87fe-4178-9fa9-0d39b07cc62f Duration: 53.40 ms Billed Duration: 54 ms Memory Size: 128 MB Max Memory Used: 64 MB Init Duration: 133.92 ms
	No newer events at this moment. Auto retry paused. Resume

Conclusion

- In this experiment we created Lambda function with S3 bucket that uploads our image.
- We configured the 'Code section' of our Lambda function and a test event for our Lambda function.
- A log of our function was also created showing successful operation.