

Name - miki lalwani

015B/37

Advance DevOps Lab

Experiment 12.

Aim -

To create a lambda function which will log "An image has been added" once you add an object to specific S3 bucket.

Theory -

AWS Lambda is a serverless, event-driven computing platform.

Working -

- 1) Write code in supported language and upload to AWS Lambda.
- 2) Enter event details.
- 3) Lambda code executes only when certain conditions are met.

AWS Lambda charges users only when executed.

eg Email sending, hosting of website etc.

Steps-

1. Open up the IAM Console and under Roles, choose the Role we previously created for the Python Lambda Function.

The screenshot shows the AWS IAM console interface. On the left, a sidebar navigation includes 'Identity and Access Management (IAM)', 'Dashboard', 'Access management' (with 'User groups', 'Users', and 'Roles' sub-options), 'Policies', 'Identity providers', and 'Account settings'. The main content area has two tabs: 'Roles' and 'Policies'. The 'Roles' tab displays a list of roles with a search bar containing 'lambda'. The 'Policies' tab displays a list of policies with a search bar containing 's3read'. A context menu is open over the 'AmazonS3ReadOnlyAccess' policy, showing options: 'Actions' (with 'Attach', 'Detach', and 'Delete'), 'Copy', 'Delete', and 'Create policy'.

Roles (14) Info
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
python_lambda-role-8robb3f1	AWS Service: lambda	-
python_lambda-role-zzx8jux7	AWS Service: lambda	29 minutes ago
py_lambda-role-3kz1ppyb	AWS Service: lambda	-

Policies (1/977) Info
A policy is an object in AWS that defines permissions.

Policy name	Type	Used as
AmazonS3ReadOnlyAccess	AWS managed	None

Policies (1/977) Info
A policy is an object in AWS that defines permissions.

Policy name	Type	Used as
CloudWatchFullAccess	AWS managed	None

Attach policy

Attach the policy to users, groups, or roles in your account

The screenshot shows the 'Attach policy' interface in the AWS IAM console. At the top, there's a search bar and a filter dropdown set to 'Filter'. Below that is a table with columns for 'Name' and 'Type'. The table lists several roles and one group:

Name	Type
AWSCodePipelineServiceRole-us-east-1-PHP_web_lambda	Role
AWSCodePipelineServiceRole-us-west-1-test-server	Role
AWSCodePipelineServiceRole-us-west-1-testserver2	Role
AWSCodePipelineServiceRole-us-west-1-testserver3	Role
<input checked="" type="checkbox"/> py_lambda-role-3kz1ppyb	Role
<input type="checkbox"/> terraform	Group

At the bottom right of the table area are 'Cancel' and 'Attach policy' buttons. Below the table, there are tabs for 'Permissions', 'Trust relationships', 'Tags', 'Access Advisor', and 'Revoke sessions'. The 'Permissions' tab is selected.

Permissions policies (2)
You can attach up to 10 managed policies.

Filter policies by property or policy name and press enter

Policy name	Type	Description
AWSLambdaBasicExecutionRole-d16747d4-50a3-4166-8b00-3d2ecef72bd	Customer managed	
CloudWatchFullAccess	AWS managed	Provides full acce

Buttons for 'Simulate', 'Remove', and 'Add permissions' are also visible.

Under Attach Policies, add S3-ReadOnly and CloudWatchFull permissions to this role.

2. Open up AWS Lambda and create a new Python function

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

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.
Python 3.9

Architecture Info
Choose the instruction set architecture you want for your function code.
x86_64

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.

service-role/py_lambda-role-3kz1ppyb

[View the py_lambda-role-3kz1ppyb role on the IAM console](#) C

Under Execution Role, choose the existing role, the one which was previously created and to which we just added permissions.

3. The function is up and running.

Lambda > Functions > newpyfunclambda

newpyfunclambda Throttle Copy ARN Actions ▾

▼ Function overview Info

 newpyfunclambda	Description -
 Layers (0)	Last modified 11 seconds ago

4. Make the following changes to the function and click on the deploy button.

This code basically logs a message and logs the contents of a JSON file which is uploaded to an S3 Bucket.

The screenshot shows the AWS Lambda function editor. The tab bar at the top has 'Tools', 'Window', 'Test' (which is selected), and 'Deploy'. Below the tabs is a file list with 'lambda_function.x' and a '+' icon. The main area contains the following Python code:

```
1 import json
2 import boto3
3 import urllib
4
5 def lambda_handler(event, context):
6     s3_client=boto3.client('s3')
7     bucket_name=event["Records"][0]['s3']['bucket']['name']
8     key=event["Records"][0]['s3']['object']['key']
9     key=urllib.parse.unquote_plus(key,encoding='utf-8')
10
11     message='ping! file was uploaded with key' + key + 'to bucket ' + bucket_name
12     print(message)
13
14     response=s3_client.getobject(Bucket=bucket_name,Key=key)
15
16     contents=response["Body"].read().decode()
17     contents=json.loads(contents)
18
19     print("These are contents of the file:\n",contents)
```

5. Open up the S3 Console and create a new bucket

The screenshot shows the 'General configuration' step of the AWS S3 bucket creation wizard. It includes fields for 'Bucket name' (set to 'lambda-trigger-bucket37'), 'AWS Region' (set to 'US East (N. Virginia) us-east-1'), and a 'Copy settings from existing bucket - optional' section with a 'Choose bucket' button. Below this is the 'Object Ownership' section, which notes that control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). It shows two options: 'ACLs disabled (recommended)' (selected) and 'ACLs enabled'.

6. With all general settings, create the bucket in the same region as the function.

7. Click on the created bucket and under properties, look for events.

⌚ Successfully created bucket "lambda-trigger-bucket37"
To upload files and folders, or to configure additional bucket settings choose [View details](#).

Amazon S3 > Buckets

▶ Account snapshot
Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

Buckets (1) [Info](#)
Buckets are containers for data stored in S3. [Learn more](#)

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

Find buckets by name < 1 > ⚙️

Name	AWS Region	Access	Creation date
lambda-trigger-bucket37	US East (N. Virginia) us-east-1	Bucket and objects not public	September 12, 2022, 14:02:49 (UTC+05:30)

Click on Create Event Notification.

Event notifications (0)
Send a notification when specific events occur in your bucket. [Learn more](#)

[Edit](#) [Delete](#) [Create event notification](#)

Name	Event types	Filters	Destination type	Destination
No event notifications				
Choose Create event notification to be notified when a specific event occurs.				
Create event notification				

Amazon EventBridge
For additional capabilities, use Amazon EventBridge to build event-driven applications at scale using S3 event notifications. [Learn more](#) or see [EventBridge pricing](#)

[Edit](#)

Send notifications to Amazon EventBridge for all events in this bucket
Off

Transfer acceleration
Use an accelerated endpoint for faster data transfers. [Learn more](#)

[Edit](#)

8. Mention an event name and check Put under event types.

General configuration

Event name

Event name can contain up to 255 characters.

Prefix - optional

Limit the notifications to objects with key starting with specified characters.

Suffix - optional

Limit the notifications to objects with key ending with specified characters.

Event types

Specify at least one event for which you want to receive notifications. For each group, you can choose an event type for all events, or you can choose one or more individual events.

Object creation

<input type="checkbox"/> All object create events s3:ObjectCreated:*	<input checked="" type="checkbox"/> Put s3:ObjectCreated:Put
---	---

Destination

Before Amazon S3 can publish messages to a destination, you must grant the Amazon S3 principal the necessary permissions to call the relevant API to publish messages to an SNS topic, an SQS queue, or a Lambda function. [Learn more](#)

Destination

Choose a destination to publish the event. [Learn more](#)

- Lambda function**
Run a Lambda function script based on S3 events.
- SNS topic**
Send notifications to email, SMS, or an HTTP endpoint.
- SQS queue**
Send notifications to an SQS queue to be read by a server.

Specify Lambda function

- Choose from your Lambda functions**
- Enter Lambda function ARN**

Lambda function

You can optionally choose .json under the suffix since the code only accepts JSON.

Choose the Lambda function as the destination and choose your lambda function and save the changes.

9. Refresh the Lambda function console and you should be able to see an S3 Trigger in the overview.

The screenshot shows the AWS Lambda Function Overview page for a function named "newpyfunclambda". The function has no layers. It has one trigger, "S3", which was added 11 seconds ago. There is a button to "Add destination". The Function ARN is listed as "arn:aws:lambda:us-east-1:689179663899:function:newpyfunclambda". The Function URL is listed as "-". The bottom navigation bar includes tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. The "Code" tab is currently selected.

10. Now, create a dummy JSON file locally.

```
Dummy.json
{
    "id":49,
    "name":"Kajal Jewani",
    "Designation":"Assistant Professor",
    "Publications":40
}
```

11. Go back to your S3 Bucket and click on Add Files to upload a new file.

lambda-trigger-bucket37 [Info](#)

Objects Properties Permissions Metrics Management Access Points

Objects (0)
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](#)

[Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions ▾](#) [Create folder](#) [Upload](#)

Find objects by prefix

No objects
You don't have any objects in this bucket.

[Upload](#)

12. Select the dummy data file from your computer and click Upload.

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

Files and folders (1 Total, 91.0 B)
All files and folders in this table will be uploaded.

Find by name

<input type="checkbox"/>	Name	Folder	Type	Size
<input type="checkbox"/>	dummy.json	-	application/json	91.0 B

Destination

Destination
<s3://lambda-trigger-bucket37>

▶ **Destination details**
Bucket settings that impact new objects stored in the specified destination.

▶ **Permissions**
Grant public access and access to other AWS accounts.

Upload succeeded
View details below.

Upload: status

The information below will no longer be available after you navigate away from this page.

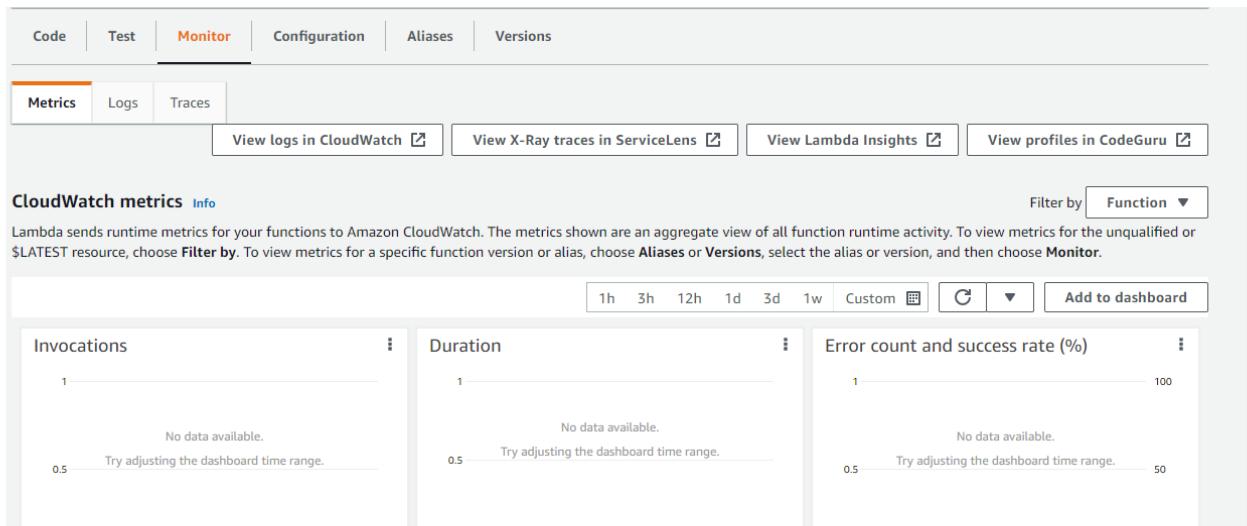
Summary

Destination	Succeeded	Failed
s3://lambda-trigger-bucket37	1 file, 91.0 B (100.00%)	0 files, 0 B (0%)

Files and folders | Configuration

Files and folders (1 Total, 91.0 B)

13. Go back to your Lambda function and check the Monitor tab.



Under Metrics, click on View logs in Cloudwatch to check the Function logs

The screenshot shows the AWS Lambda function configuration page. At the top, there are several settings: Retention (Never expire), Creation time (Now), Subscription filters (0), KMS key ID (-), Metric filters (0), Contributor Insights rules (-), and Stored bytes (-). Below this, the ARN is listed as arn:aws:logs:us-east-1:689179663899:log-group:/aws/lambda/newpyfunclambda:*. Below the configuration, there is a navigation bar with tabs: Log streams (selected), Metric filters, Subscription filters (highlighted in orange), Contributor Insights, and Tags. Under the Log streams tab, it says 'Log streams (1)'. There is a search bar with 'Filter log streams or try prefix search' and an 'Exact match' checkbox. A table lists one log stream: '2022/09/12/[\$LATEST]457d22508a504773aa66d9651ee6a2ec' with a timestamp of '2022-09-12 14:08:47 (UTC+05:30)'.

14. Click on this log Stream that was created to view what was logged by your function.

As you can see, our function logged that a file was uploaded with its file name and the bucket to which it was uploaded. It also mentions the contents inside the file as our function was defined to.

Hence, we have successfully created a Python function inside AWS Lambda which logs every time an object is uploaded to an S3 Bucket.

Part 2

Sending an Email on Bucket additions to Bucket

1. Go to the IAM console and edit the same Lambda Role. This time, add SESFullAccess Permission to the role.

Other permissions policies (Selected 1/772)			
<input type="text"/> Filter policies by property or policy name and press enter		11 matches Create policy	
<input type="checkbox"/> "ses" Clear filters		Type	Description
Policy name		Type	Description
<input type="checkbox"/> + AmazonSESReadOnlyAccess		AWS managed	Provides read only access to the Amazon SES service.
<input checked="" type="checkbox"/> + AmazonSESFullAccess		AWS managed	Provides full access to the Amazon SES service.
<input type="checkbox"/> + AWSVendorInsightsAssessorFullAccess		AWS managed	Provides full access to the AWS Vendor Insights Assessor service.
<input type="checkbox"/> + AwsGlueSessionUserRestrictedNotebookPolicy		AWS managed	Provides permissions to run a notebook session.
<input type="checkbox"/> + AmazonSageMakerServiceCatalogProductsFirehoseServiceRolePolicy		AWS managed	Service role policy used by the Amazon SageMaker Service Catalog Product.
<input type="checkbox"/> + AWSOpsWorksRegisterCLI_OnPremises		AWS managed	Policy to enable registration of On-Premises instances.
<input type="checkbox"/> + AwsGlueSessionUserRestrictedNotebookServiceRole		AWS managed	Provides full access to the AWS Glue Session User Restricted Notebook Service Role.
<input type="checkbox"/> + ElementalActivationsGenerateLicenses		AWS managed	Access to view purchased Elemental activations.
<input type="checkbox"/> + AmazonCloudWatchLogsMetricsFullAccess		AWS managed	Provides permissions to publish metrics to CloudWatch Metrics.

2. Create a new Lambda function in a Python environment. Use the existing role which was previously created.

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.

Browse serverless app repository

Deploy a sample Lambda application from the AWS Serverless Application Repository.

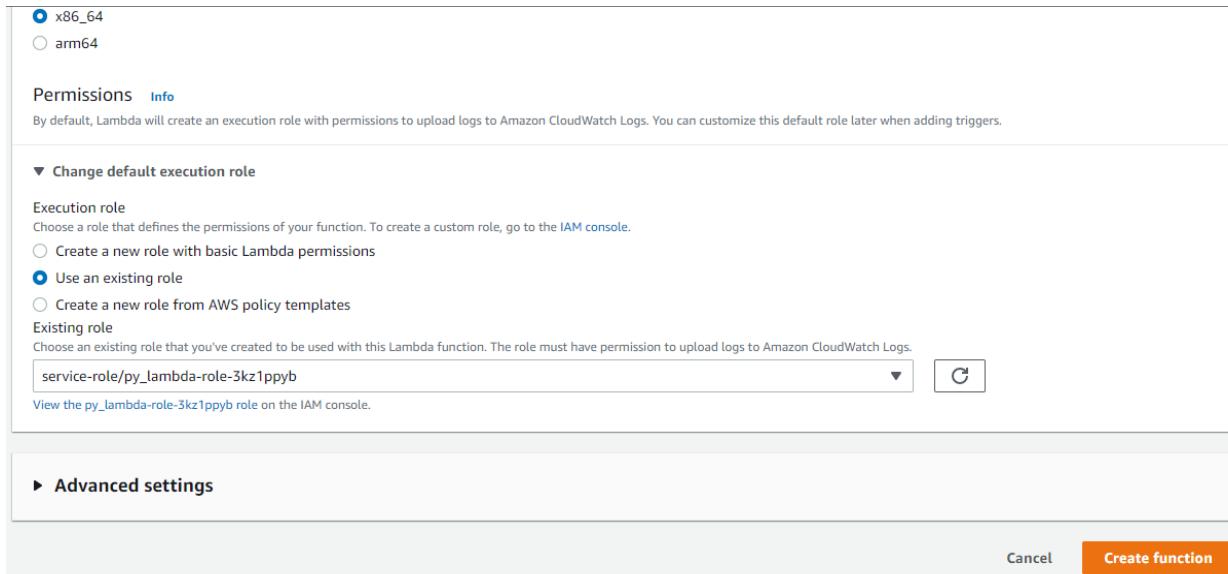
Basic information

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

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

Runtime Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Architecture Choose the instruction set architecture you want for your function code.
 x86_64



3. In this function, above the default hello-world TODO, add the following code.

This code is basically to send an email on the creation of an object in the attached S3 Bucket. It sends the bucket name, event, and source IP address. In this code, modify the Source and Destination ToAddresses to your sender and receiver email addresses. Once done, deploy the function.

```

Tools Window Test Deploy
lambda_function +
5 def lambda_handler(event, context):
6     for a in event["Records"]:
7         action = a["eventName"]
8         ip = a["requestParameters"]["sourceIPAddress"]
9         bucket_name = a["s3"]["bucket"]["name"]
10        object = a["s3"]["object"]["key"]
11        client = boto3.client("ses")
12        body = str(action) + " Event From " + bucket_name
13        subject = ""
14        <br>
15        <p>
16        Hey! This e-mail was generated to notify you about the event <strong>{}</strong>.
17        Source IP: {}
18        </p>
19        """.format(
20            action, ip
21        )
22        message = {"Subject": {"Data": subject}, "Body": {"Html": {"Data": body}}}
23        response = client.send_email(
24            Source="mikil.lalwani03@gmail.com",
25            Destination={"ToAddresses": ["2020.mikil.lalwani@ves.ac.in"]},
26            Message=message,
27        )
28        return {
29            'statusCode': 200,
30            'body': json.dumps('Hello from Lambda!')
31        }
32

```

4. Open up the SES Console and click on Manage Email Addresses.

5. Choose Verify Email Address and verify both sender and receiver email addresses.

Click on the verification links you are sent and verify the emails.

Create identity

A *verified identity* is a domain, subdomain, or email address you use to send email through Amazon SES. Identity verification at the domain level extends to all email addresses under one verified domain identity.

Identity details [Info](#)

Identity type

Domain
To verify ownership of a domain, you must have access to its DNS settings to add the necessary records.

Email address
To verify ownership of an email address, you must have access to its inbox to open the verification email.

Email address

mikil.lalwani03@gmail.com

Email address can contain up to 320 characters, including plus signs (+), equals signs (=) and underscores (_).

Assign a default configuration set
Enabling this option ensures that the assigned configuration set is applied to messages sent from this identity by default whenever a configuration set isn't specified at the time of sending.

Amazon SES > Configuration: Verified identities > mikil.lalwani03@gmail.com

mikil.lalwani03@gmail.com [Delete](#) [Send test email](#)

Legacy TXT records

Domain verification in Amazon SES is now based on *DomainKeys Identified Mail (DKIM)*, an email authentication standard that receiving mail servers use to validate an email's authenticity. Configuring DKIM in your domain's DNS settings confirms to SES that you're the identity owner, eliminating the need for TXT records. Domain identities that were verified using TXT records do not need to be reverified; however, we still recommend enabling DKIM signatures to enhance the deliverability of your mail with DKIM-compliant email providers. [To access your legacy TXT records, download Legacy TXT record set as .csv](#).

Summary for mikil.lalwani03@gmail.com

Identity status	Amazon Resource Name (ARN)	AWS Region
<input checked="" type="radio"/> Verified	arn:aws:ses:us-east-1:689179663899:identity/mikil.lalwani03@gmail.com	US East (N. Virginia)

6. Now, open up the S3 Console, create a new bucket as you did previously and add an event notification inside events and attach it to your Lambda function.

Buckets (2) Info
Buckets are containers for data stored in S3. [Learn more](#)

Name	AWS Region	Access	Creation date
bucketforlambdaemail	US East (N. Virginia) us-east-1	Bucket and objects not public	September 12, 2022, 14:33:31 (UTC+05:30)
lambda-trigger-bucket37	US East (N. Virginia) us-east-1	Bucket and objects not public	September 12, 2022, 14:02:49 (UTC+05:30)

Event notifications (1)
Send a notification when specific events occur in your bucket. [Learn more](#)

Name	Event types	Filters	Destination type	Destination
s3emailtrigger	Put	-	Lambda function	func1

Amazon EventBridge
For additional capabilities, use Amazon EventBridge to build event-driven applications at scale using S3 event notifications. [Learn more](#) or see [EventBridge pricing](#)

Send notifications to Amazon EventBridge for all events in this bucket
Off

7. Once that's done, upload any file to your S3 Bucket. I'll upload the same dummy JSON file again.

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 folders**.

Files and folders (1 Total, 91.0 B)
All files and folders in this table will be uploaded.

Name	Folder	Type	Size
dummy.json	-	application/json	91.0 B

Destination

Destination
`s3://bucketforlambdaemail`

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

⌚ 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	Succeeded <code>s3://bucketforlambdaemail</code>	Failed <code>0 files, 0 B (0%)</code>
	<input checked="" type="checkbox"/> 1 file, 91.0 B (100.00%)	

8. Check your ToAddress email. You'll receive an email from the Source Address via Amazon SES.

ObjectCreated:PutEvent from bucketforlambdaemail External Inbox x

 mikil.lalwani03@gmail.com via [amazoneses.com](#)
to me ▾

Mon, Sep 12, 2:47 PM (8 days ago) star forward more

Hey! This e-mail was generated to notify you about the event ObjectCreated:Put . Source IP: 45.114.195.254

Reply Forward

In this way, we successfully created a function in AWS Lambda that sends an email on uploading an object to an S3 Bucket using Amazon SES.

Recommended Cleanup

Once done with the experiment, it is recommended to delete all resources which have been created and used by us to avoid charges in AWS.

Here is a list of things you may delete:

1. AWS Lambda Function
2. Amazon S3 Storage Bucket
3. Amazon SES Verified Emails
4. AWS Cloudwatch Logs (Optional, won't affect bills)
5. AWS IAM Role (the one which was created for the function, again, won't affect bills)

Conclusion -

In this experiment, we learned how to create an AWS Lambda function to log everytime an image is uploaded to an s3 bucket.