

Name: Hrishikesh Kumbhar

Div: D15A

Roll no: 32

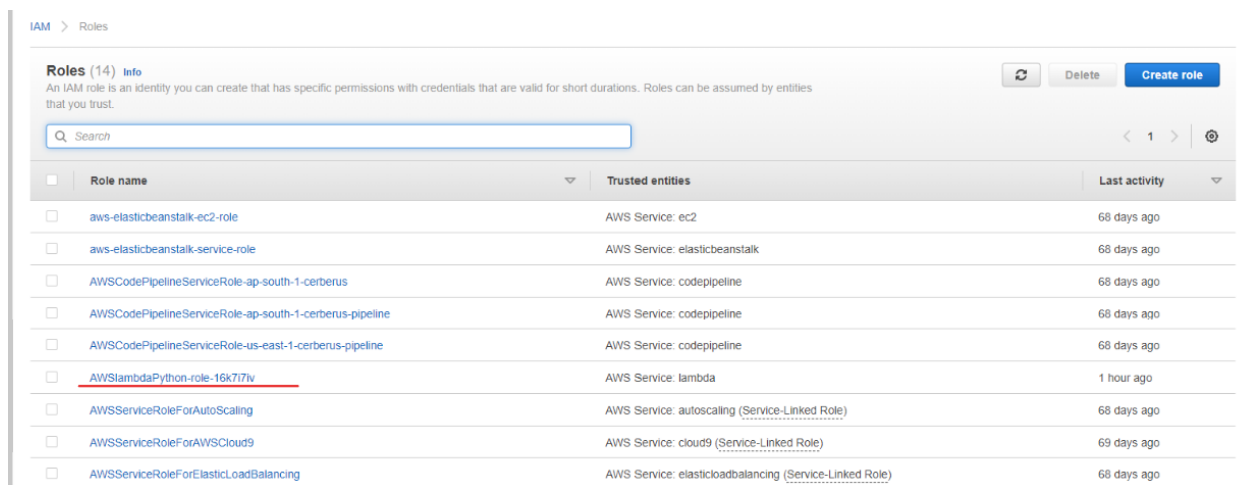
Sub: Advanced DevOps

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

Steps to create a Lambda function that reacts to uploads in an S3 Bucket:

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



| <input type="checkbox"/> | Role name | Trusted entities | Last activity |
|--------------------------|---|---|---------------|
| <input type="checkbox"/> | aws-elasticbeanstalk-ec2-role | AWS Service: ec2 | 68 days ago |
| <input type="checkbox"/> | aws-elasticbeanstalk-service-role | AWS Service: elasticbeanstalk | 68 days ago |
| <input type="checkbox"/> | AWSCodePipelineServiceRole-ap-south-1-cerberus | AWS Service: codepipeline | 68 days ago |
| <input type="checkbox"/> | AWSCodePipelineServiceRole-ap-south-1-cerberus-pipeline | AWS Service: codepipeline | 68 days ago |
| <input type="checkbox"/> | AWSCodePipelineServiceRole-us-east-1-cerberus-pipeline | AWS Service: codepipeline | 68 days ago |
| <input type="checkbox"/> | AWSLambdaPython-role-16k7i7iv | AWS Service: lambda | 1 hour ago |
| <input type="checkbox"/> | AWSServiceRoleForAutoScaling | AWS Service: autoscaling (Service-Linked Role) | 68 days ago |
| <input type="checkbox"/> | AWSServiceRoleForAWSCloud9 | AWS Service: cloud9 (Service-Linked Role) | 69 days ago |
| <input type="checkbox"/> | AWSServiceRoleForElasticLoadBalancing | AWS Service: elasticloadbalancing (Service-Linked Role) | 68 days ago |

AWSLambdaPython-role-16k7i7iv

Delete

Summary

Edit

Creation date
October 08, 2022, 14:08 (UTC+05:30)

Last activity
1 hour ago

ARN
arn:aws:iam::504827858021:role/service-role/AWSLambdaPython-role-16k7i7iv
Maximum session duration
1 hour

Permissions Trust relationships Tags Access Advisor Revoke sessions

Permissions policies (1) Info
You can attach up to 10 managed policies.

Simulate Remove Add permissions

Filter policies by property or policy name and press enter.

| <input type="checkbox"/> | Policy name | Type | Description |
|--------------------------|--|------------------|-------------|
| <input type="checkbox"/> | AWSLambdaBasicExecutionRole-0b1d7304-ac32-489c-844e-91537779b889 | Customer managed | |

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

Step 2: Open up AWS Lambda and create a new Python function

Lambda > Functions > Create function

Create function [Info](#)

Choose one of the following options to create your function.

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.

Basic information

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

S3EventLogger

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
☐ 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.

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

▼ 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](#).

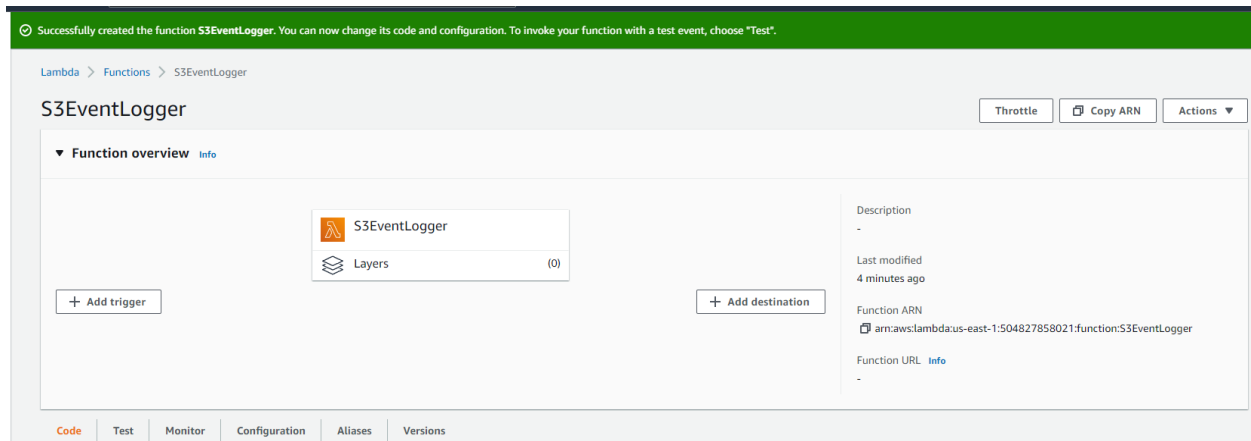
☐ Create a new role with basic Lambda permissions
☒ Use an existing role
☐ Create a new role from AWS policy templates

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/AWSLambdaPython-role-16k7i7iv ▼

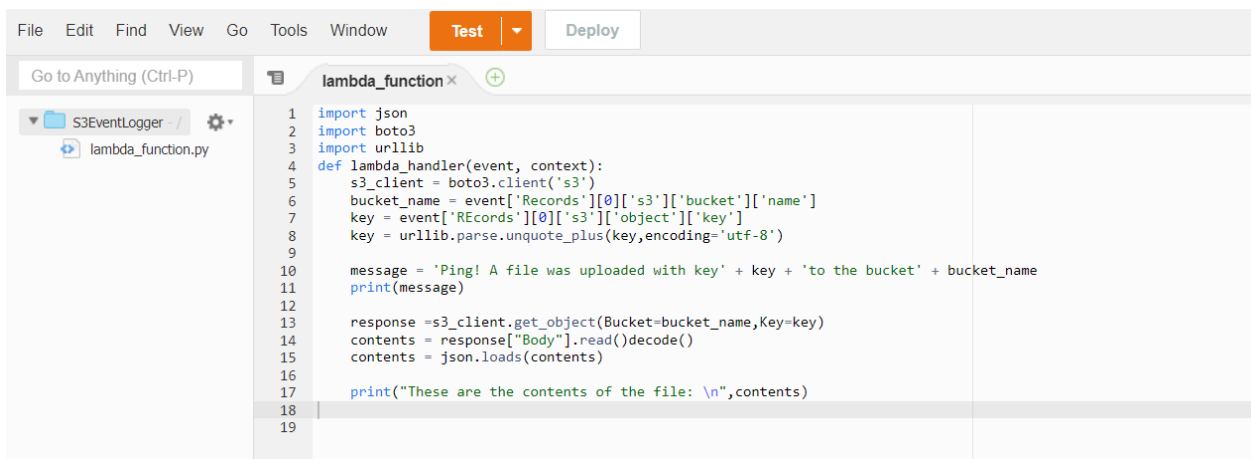
[View the AWSLambdaPython-role-16k7i7iv role on the IAM console.](#)

Step 3: The function is up and running



Step 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.



Lambda_function.py

```
import json
import boto3
import urllib

def lambda_handler(event, context):
    s3_client = boto3.client('s3')
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    key = event['Records'][0]['s3']['object']['key']
    key = urllib.parse.unquote_plus(key,encoding='utf-8')

    message = 'Ping! A file was uploaded with key' + key + 'to the bucket'
+ bucket_name
    print(message)

    response = s3_client.get_object(Bucket=bucket_name,Key=key)
    contents = response["Body"].read().decode()
    contents = json.loads(contents)

    print("These are the contents of the file: \n",contents)
```

Step 5. Click on Test and choose the ‘S3 Put’ Template.

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

s3Event

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 name

s3Event

↻

Delete

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": "example-key"
31        }
32      }
33    }
34  ]
35 }
```

Here, inside the region, you may want to change the region to the AZ in which you've created your function and bucket. This doesn't seem mandatory but you might as well do it.

Step 6: Open up the S3 Console and create a new bucket

Step 7: With all general settings, create the bucket in the same region as the function.

The screenshot shows the 'Create bucket' page in the Amazon S3 console. The breadcrumb navigation at the top reads 'Amazon S3 > Buckets > Create bucket'. The main heading is 'Create bucket' with an 'Info' link. Below this, a sub-header 'General configuration' is followed by a description: 'Buckets are containers for data stored in S3. [Learn more](#)'. The form contains three sections: 1. 'Bucket name' with a text input field containing 'new-lambda-bucket-trigger' and a note that the name must be globally unique and contain no spaces or uppercase letters. 2. 'AWS Region' with a dropdown menu set to 'US East (N. Virginia) us-east-1'. 3. 'Copy settings from existing bucket - optional' with a 'Choose bucket' button. Below the general configuration is the 'Object Ownership' section, which includes a description and a 'Choose' button.

Amazon S3 > Buckets > Create bucket

Create bucket [Info](#)

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

General configuration

Bucket name

Bucket name must be globally unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

AWS Region

US East (N. Virginia) us-east-1

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

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.

Step 8. Click on the created bucket and under properties, look for events.

The screenshot shows the 'Buckets' page in the Amazon S3 console. The breadcrumb navigation at the top reads 'Amazon S3 > Buckets'. There is an 'Account snapshot' section with a 'View Storage Lens dashboard' button. Below this is a 'Buckets (1)' section with a search bar and a table of buckets. The table has columns for Name, AWS Region, Access, and Creation date. The first row shows the bucket 'new-lambda-bucket-trigger' in the 'US East (N. Virginia) us-east-1' region, with 'Bucket and objects not public' access and a creation date of 'October 8, 2022, 18:51:10 (UTC+05:30)'. Above the table are buttons for 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'.

Amazon S3 > Buckets

► **Account snapshot** [View Storage Lens dashboard](#)

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](#)

[Find buckets by name](#)

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

| | Name | AWS Region | Access | Creation date |
|-----------------------|---------------------------|---------------------------------|--------------------------------------|---------------------------------------|
| <input type="radio"/> | new-lambda-bucket-trigger | US East (N. Virginia) us-east-1 | <u>Bucket and objects not public</u> | October 8, 2022, 18:51:10 (UTC+05:30) |

Event notifications (0)

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

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

| | Name | Event types | Filters | Destination type | Destination |
|---|------|-------------|---------|------------------|-------------|
| <p>No event notifications</p> <p>Choose Create event notification to be notified when a specific event occurs.</p> <div>Create event notification</div> | | | | | |

Amazon EventBridge

[Learn more](#) or [see EventBridge pricing](#)

[Edit](#)

Send notifications to Amazon EventBridge for all events in this bucket

Off

Click on Create Event Notification.

Step 9: Mention an event name and check Put under event types

Create event notification [Info](#)

To enable notifications, you must first add a notification configuration that identifies the events you want Amazon S3 to publish and the destinations where you want Amazon S3 to send the notifications.

General configuration

Event name

s3PutRequest

Event name can contain up to 255 characters.

Prefix - *optional*

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

images/

Suffix - *optional*

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

.jpg

Object creation

☐ All object create events
s3:ObjectCreated:*

- ☒ Put
s3:ObjectCreated:Put
- ☐ Post
s3:ObjectCreated:Post
- ☐ Copy
s3:ObjectCreated:Copy
- ☐ Multipart upload completed
s3:ObjectCreated:CompleteMultipartUpload

Object removal

☐ All object removal events
s3:ObjectRemoved:*

- ☐ Permanently deleted
s3:ObjectRemoved:Delete
- ☐ Delete marker created
s3:ObjectRemoved:DeleteMarkerCreated



Object restore

☐ All restore object events
s3:ObjectRestore:*


- ☐ Restore initiated
s3:ObjectRestore:Post
- ☐ Restore completed
s3:ObjectRestore:Completed

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

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**
Fanout messages to systems for parallel processing or directly to people.

☐ **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

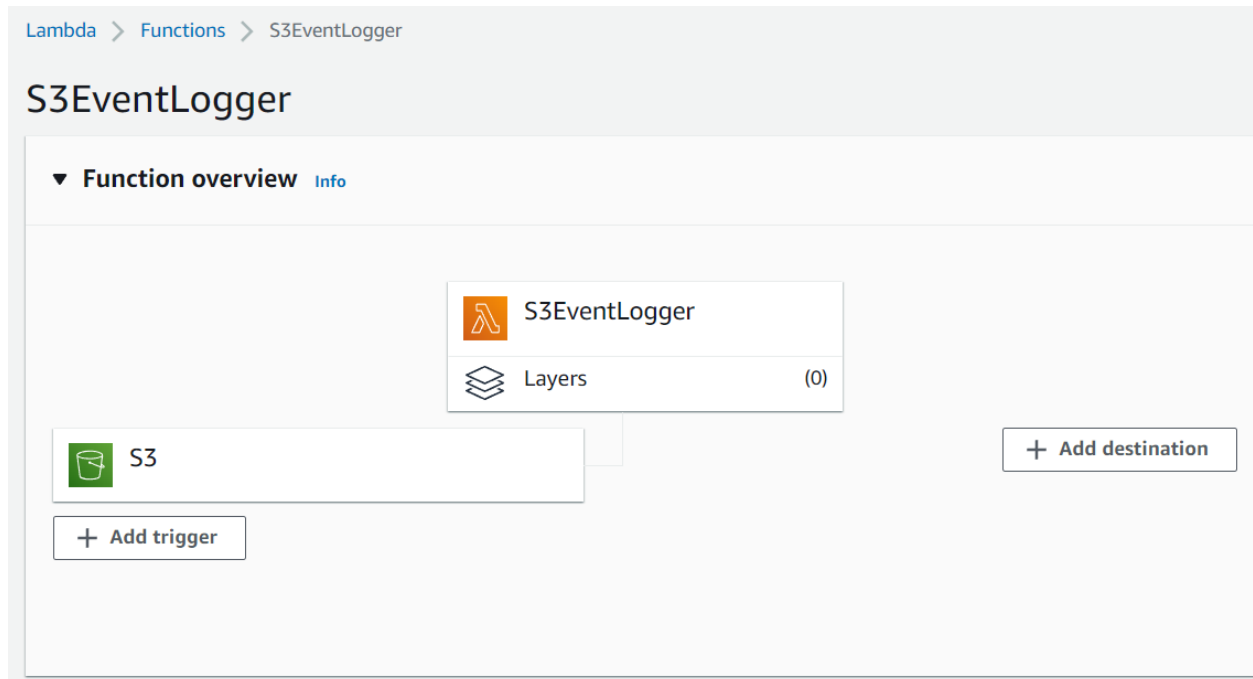
Lambda function

S3EventLogger ▼

[Cancel](#) [Save changes](#)

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

Step 10. Refresh the Lambda function console and you should be able to see an S3 Trigger in the overview.



Step 11: Now, create a dummy JSON file locally.

```
C: > Users > acer > Desktop > {} s3_data.json > ...  
1  {  
2    "name" : "Hrishikesh Kumbhar",  
3    "college_name": "VESIT",  
4    "class": "D15A",  
5    "roll_no": 32,  
6    "sub": "Advanced DevOps"  
7  }
```

Step 12: Go back to your S3 Bucket and click on Add Files to upload a new file.

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 folders**.

Files and folders (0)

Remove

Add files

Add folder

All files and folders in this table will be uploaded.

< 1 >

| <input type="checkbox"/> | Name | Folder | Type | Size |
|---|------|--------|------|------|
| No files or folders | | | | |
| You have not chosen any files or folders to upload. | | | | |

Step 13: Select the dummy data file from your computer and click Upload.

Amazon S3 > Buckets > new-lambda-bucket-trigger > 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 folders**.

Files and folders (1 Total, 141.0 B)

Remove

Add files

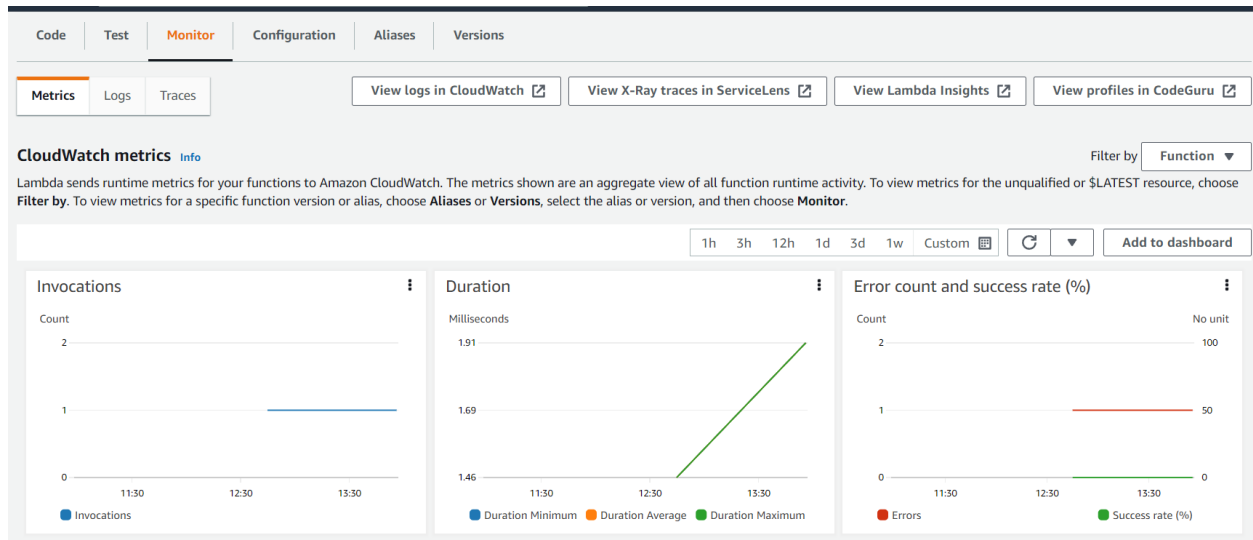
Add folder

All files and folders in this table will be uploaded.

< 1 >

| <input type="checkbox"/> | Name | Folder | Type | Size |
|--------------------------|--------------|--------|------------------|---------|
| <input type="checkbox"/> | s3_data.json | - | application/json | 141.0 B |

Step 14: 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 'Log streams' view in the AWS Lambda console. It lists log streams for the function. The first log stream is '2022/10/08/[LATEST]812ca2f6481e4e50bf7d129462bd63e' with a last event time of '2022-10-08 19:46:06 (UTC+05:30)'. There are buttons for 'Log streams (4)', 'Delete', 'Create log stream', and 'Search all log streams'. A search bar is also present.

| Log stream | Last event time |
|--|---------------------------------|
| 2022/10/08/[LATEST]812ca2f6481e4e50bf7d129462bd63e | 2022-10-08 19:46:06 (UTC+05:30) |

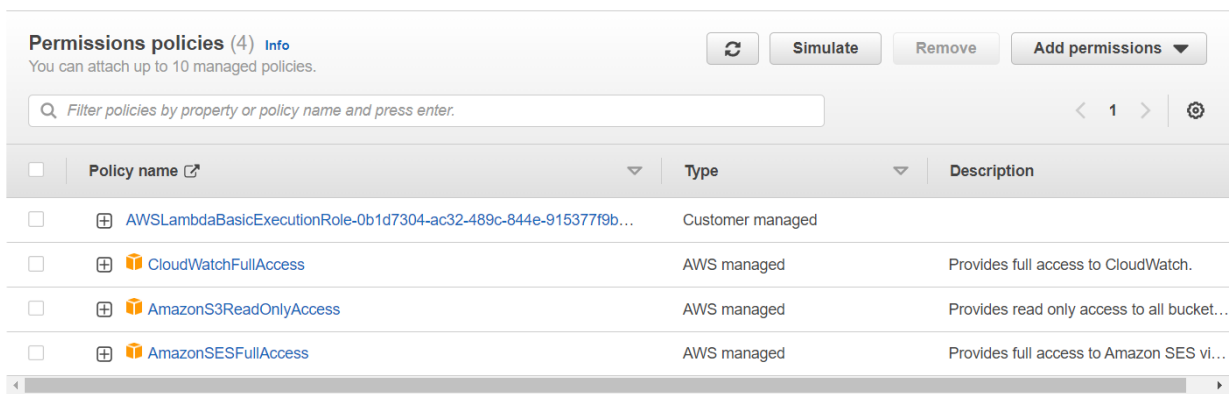
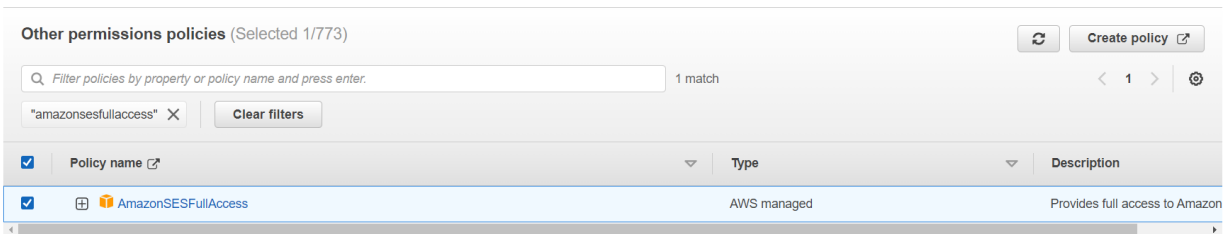
The screenshot shows the 'Log events' view in the AWS Lambda console. It displays a list of log events for the function. The events include a 'START' event, a 'Ping!' event, a file upload event, and a 'REPORT' event. The events are sorted by time, and there are buttons for 'No older events at this moment. Retry' and 'No newer events at this moment. Auto retry paused. Resume'.

| Time | Event |
|-------------------------------|--|
| 2022-10-08T19:46:04.327+05:30 | START RequestId: ddcaacdb-b831-4ea9-9c1b-be4e0215f7dd Version: \$LATEST |
| 2022-10-08T19:46:05.723+05:30 | Ping! A file was uploaded with keys3_data.js onto the bucketnew-lambda-bucket-trigger |
| 2022-10-08T19:46:06.023+05:30 | These are the contents of the file: |
| 2022-10-08T19:46:06.023+05:30 | {'name': 'Hrishikesh Kumbhar', 'college_name': 'VESIT', 'class': 'D15A', 'roll_no': 32, 'sub': 'Advanced DevOps'} |
| 2022-10-08T19:46:06.062+05:30 | END RequestId: ddcaacdb-b831-4ea9-9c1b-be4e0215f7dd |
| 2022-10-08T19:46:06.062+05:30 | REPORT RequestId: ddcaacdb-b831-4ea9-9c1b-be4e0215f7dd Duration: 1735.44 ms Billed Duration: 1736 ms Memory Size: 128 MB Max ... |

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.

Sending an Email on Bucket additions to Bucket

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



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

[Lambda](#) > [Functions](#) > Create function

Create function [Info](#)

Choose one of the following options to create your function.

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.

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 [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 AWSLambdaPython-role-16k7i7iv role on the IAM console.](#)

Step 3: In this function, 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.

```
lambda_function × (+)
1 import json
2 import boto3
3
4 def lambda_handler(event, context):
5     for a in event['Records']:
6         action = a['eventName']
7         ip = a['requestParameters']['sourceIPAddress']
8         bucket_name = a['s3']['bucket']['name']
9         object = a['s3']['object']['key']
10
11     client = boto3.client('ses')
12     subject = str(action) + ' Event from ' + bucket_name
13     body = """
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(action,ip)
20     message = {
21         'subject':{
22             'Data':subject
23         },
24         'Body':{
25             'Html':{
26                 'Data':body
27             }
28         }
29     }
30
31     response = client.send_email(
32         Source="hrishikumbhar156@gmail.com",
33         Destination={
34             'ToAddresses':[
35                 "2020.hrishikesh.kumbhar@ves.ac.in"
36             ]
37         },
```

lambda_function.py

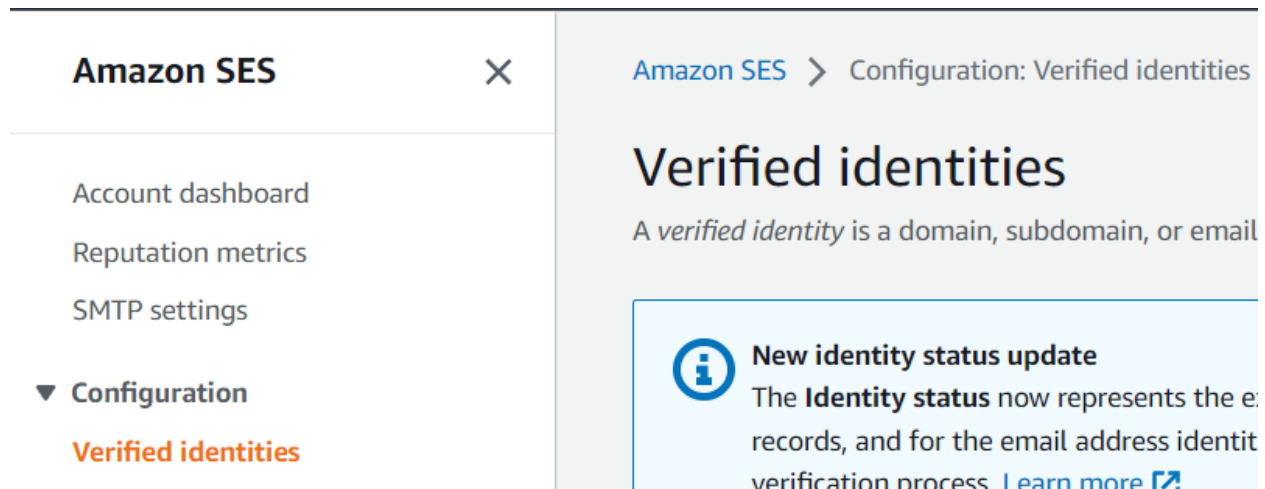
```
import json
import boto3

def Lambda_handler(event, context):
    for a in event['Records']:
        action = a['eventName']
        ip = a['requestParameters']['sourceIPAddress']
        bucket_name = a['s3']['bucket']['name']
        object = a['s3']['object']['key']

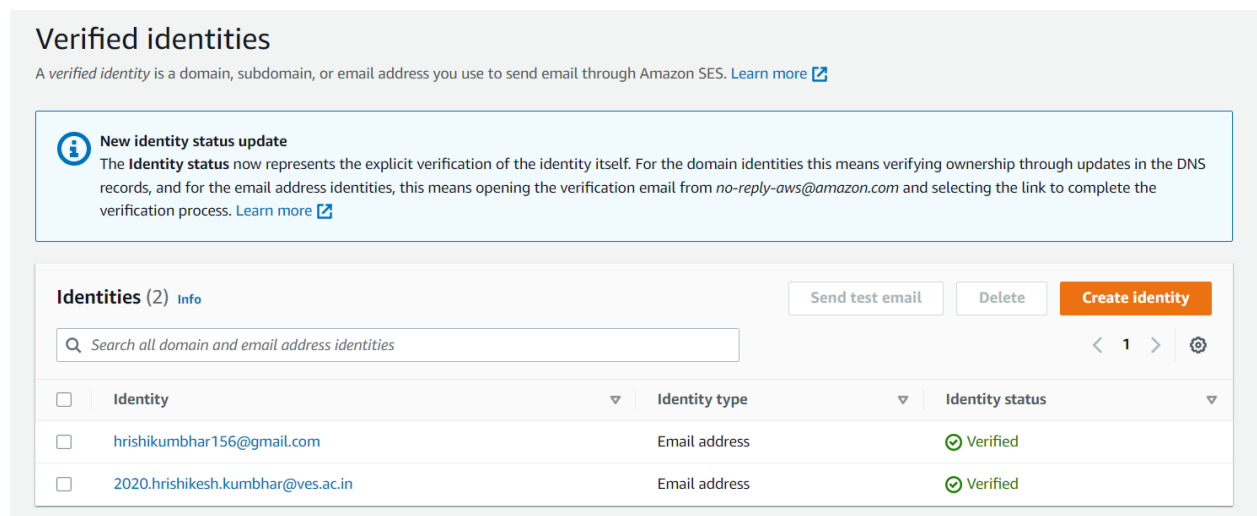
        client = boto3.client('ses')
        subject = str(action) + ' Event from ' + bucket_name
        body = """
        <br>
        <p>
        Hey! This e-mail was generated to notify you about the event
        <strong>{}</strong>.
        source IP: {}
        </p>
        """.format(action,ip)
        message = {
            'Subject':{
                'Data':subject
            },
            'Body':{
                "Html":{
                    'Data':body
                }
            }
        }

        response = client.send_email(
            Source="hrishikumbhar156@gmail.com",
            Destination={
                'ToAddresses':[
                    "2020.hrishikesh.kumbhar@ves.ac.in"
                ]
            },
            Message = message
        )
        return {
            'statusCode': 200,
            'body': json.dumps('Hello! Check your mail please!')
        }
```

Step 4 :Open up the SES Console and click on Verified Identities.



Step 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.

Step 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.

To enable notifications, you must first add a notification configuration that identifies the events you want Amazon S3 to publish and the destinations where you want Amazon S3 to send the notifications.

General configuration

Event name

s3-event-email-trigger

Event name can contain up to 255 characters.

Prefix - *optional*

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

images/

Suffix - *optional*

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

.jpg

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

☐ All object create events
s3:ObjectCreated:*

☒ Put

s3:ObjectCreated:Put

☐ Post

s3:ObjectCreated:Post

☐ Copy

s3:ObjectCreated:Copy

☐ Multipart upload completed

s3:ObjectCreated:CompleteMultipartUpload

Object removal

☐ All object removal events
s3:ObjectRemoved:*

☐ Permanently deleted

s3:ObjectRemoved:Delete

☐ Delete marker created

s3:ObjectRemoved:DeleteMarkerCreated

Destination

i 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**
Fanout messages to systems for parallel processing or directly to people.

☐ **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

s3-trigger-email ▼

Cancel Save changes

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

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 folders**.

Files and folders (1 Total, 141.0 B)

Remove Add files Add folder

All files and folders in this table will be uploaded.

< 1 >

| <input type="checkbox"/> | Name ▲ | Folder ▼ | Type ▼ | Size ▼ |
|--------------------------|--------------|----------|------------------|---------|
| <input type="checkbox"/> | s3_data.json | - | application/json | 141.0 B |

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

The screenshot displays the AWS CloudWatch console. At the top, there are tabs for 'Log streams', 'Metric filters', 'Subscription filters', 'Contributor Insights', and 'Tags'. The 'Log streams' tab is active, showing a list of log streams. Below the tabs, there's a search bar for log streams and a table with two columns: 'Log stream' and 'Last event time'. One log stream is listed: '2022/10/08/[\$LATEST]a19cf49d1b704badae56e3d75043fc91' with a last event time of '2022-10-08 20:55:17 (UTC+05:30)'. Below the log streams, the 'Log events' section is visible. It includes a search bar for events, a 'View as text' checkbox, and a table with two columns: 'Timestamp' and 'Message'. The table shows three events: a 'START' event, an 'END' event, and a 'REPORT' event, all with their respective timestamps and message details.

| Log stream | Last event time |
|---|---------------------------------|
| 2022/10/08/[\$LATEST]a19cf49d1b704badae56e3d75043fc91 | 2022-10-08 20:55:17 (UTC+05:30) |

| Timestamp | Message |
|-------------------------------|--|
| | No older events at this moment. Retry |
| 2022-10-08T20:55:15.671+05:30 | START RequestId: a6844b4d-7a55-46f6-abd4-653ad759d6bc Version: \$LATEST |
| 2022-10-08T20:55:17.066+05:30 | END RequestId: a6844b4d-7a55-46f6-abd4-653ad759d6bc |
| 2022-10-08T20:55:17.066+05:30 | REPORT RequestId: a6844b4d-7a55-46f6-abd4-653ad759d6bc Duration: 1395.34 ms Billed Duration: 1396 ms Memory Size: 128 MB Max ... |
| | No newer events at this moment. Auto retry paused. Resume |

ObjectCreated:Put Event from lambd-bucket-email-trigger External Inbox x



hrishikumbhar156@gmail.com via amazonses.com
to me ▾

Hey! This e-mail was generated to notify you about the event **ObjectCreated:Put** source IP: 180.148.62.216

↩ 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 every time an object is added to an S3 Bucket.