Name: Hrishikesh Kumbhar

Div: D15A

Roll no: 32

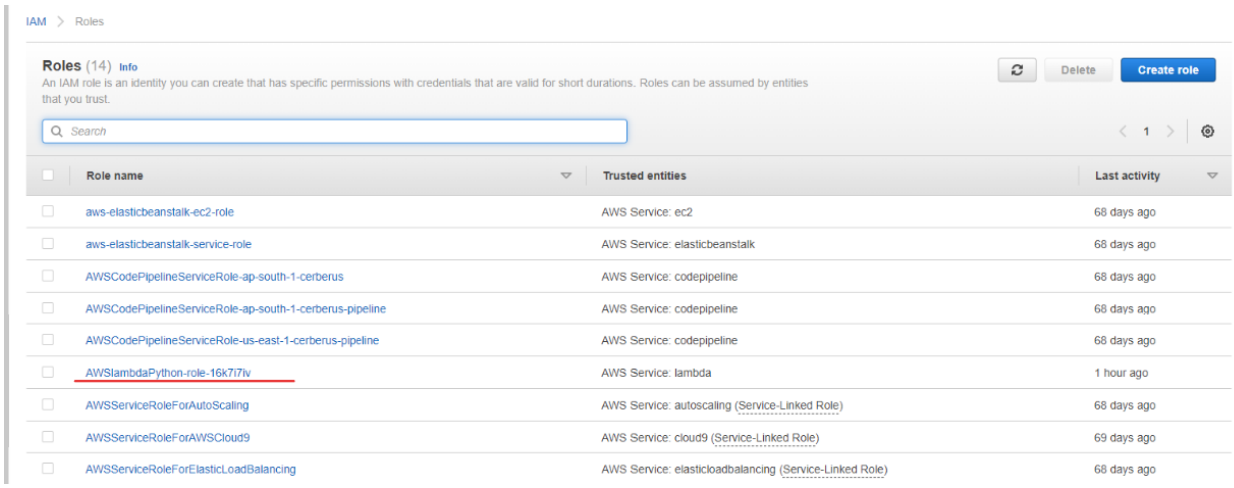
Sub: Advanced DevOps

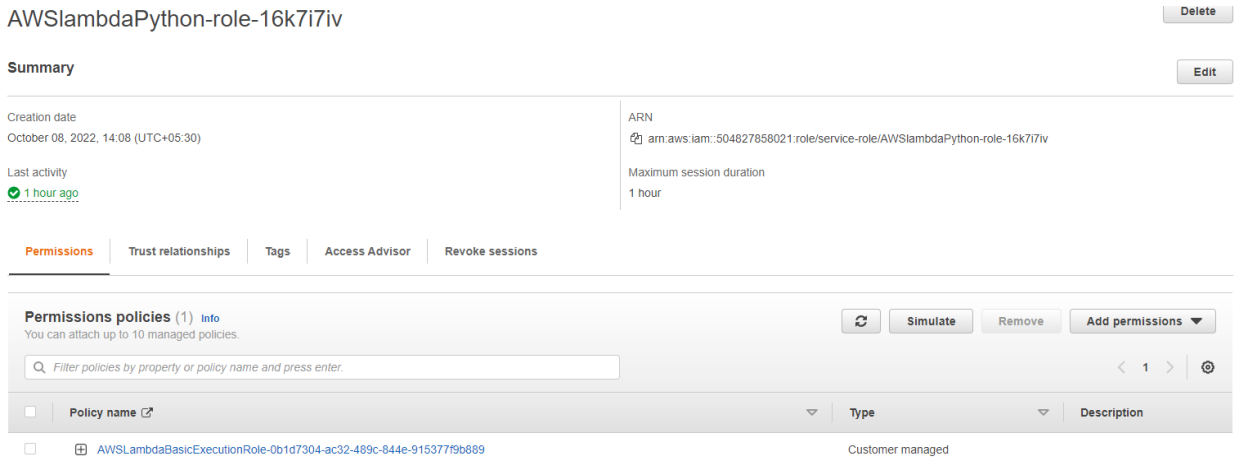
Experiment No: 12

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

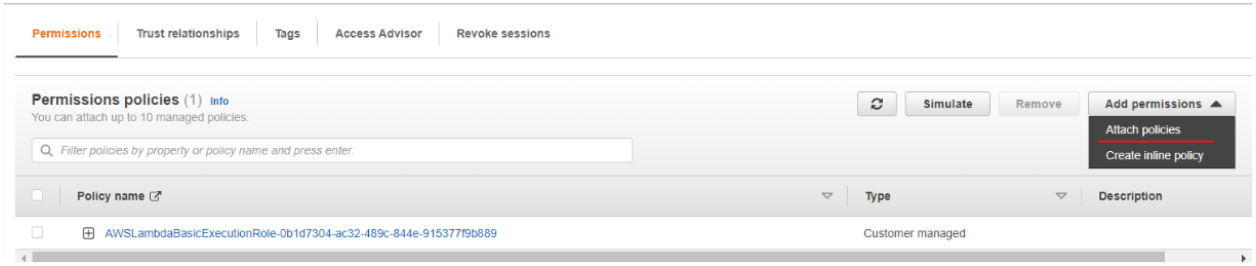
Steps to create a Lambda function that reacts to uploads in an S3 Bucket:

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

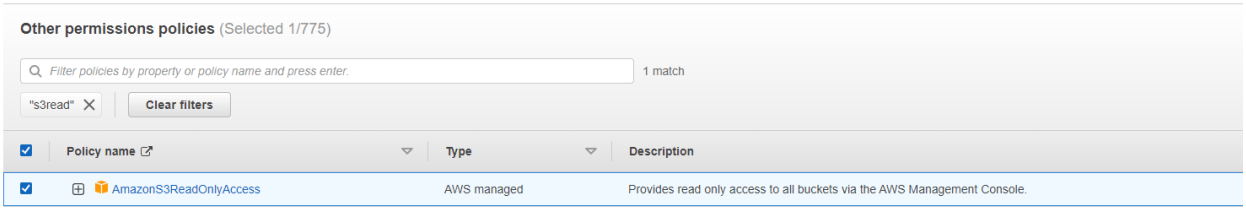




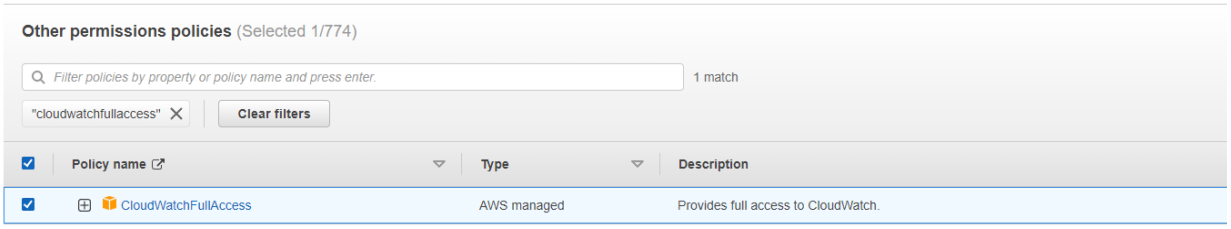
Under Attach Policies, add S3-ReadOnly and CloudWatchFull permissions to this role.

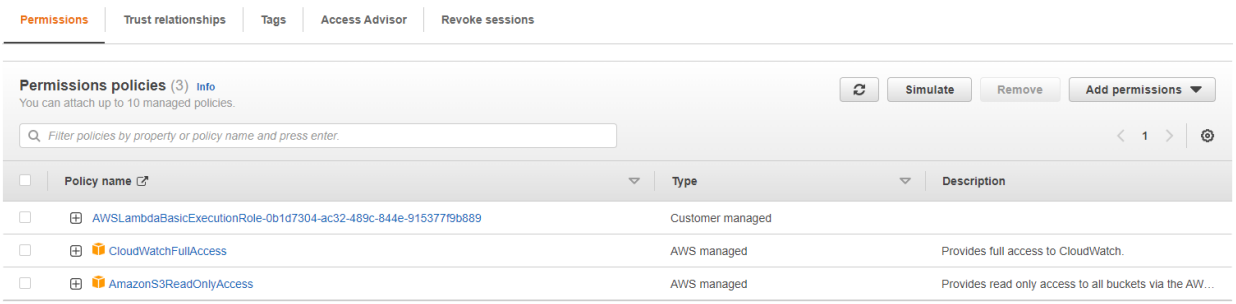


Adding AmazonS3ReadOnlyAccess

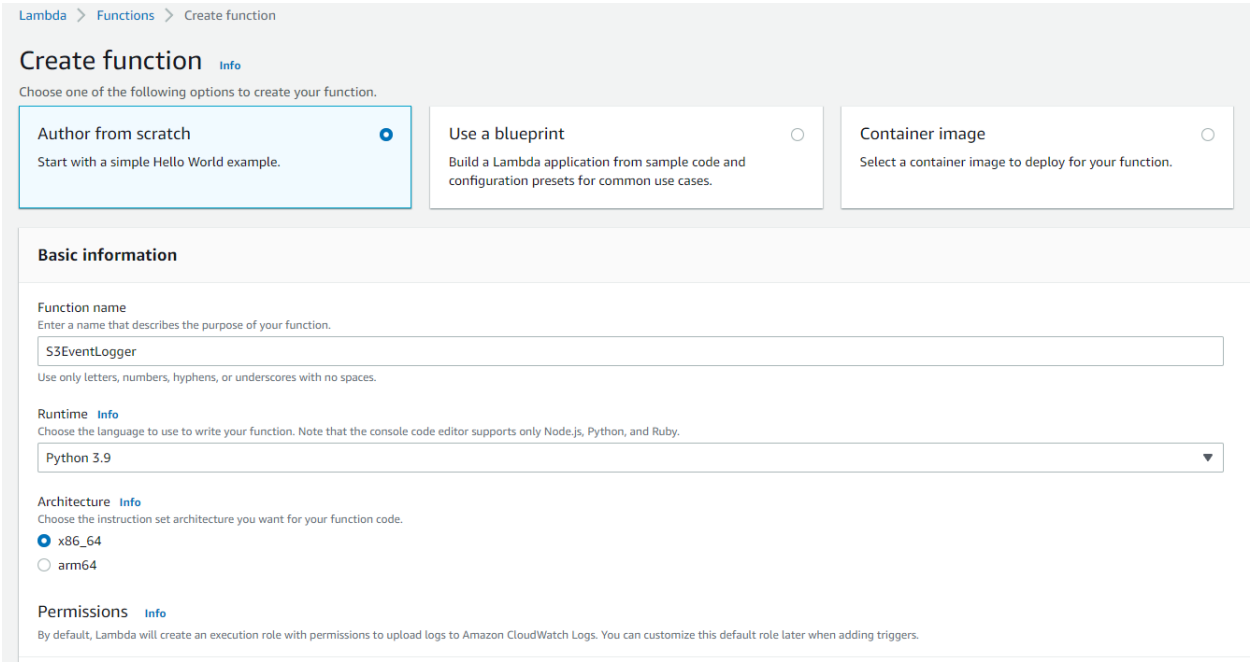


Adding CloudWatchfullaccess

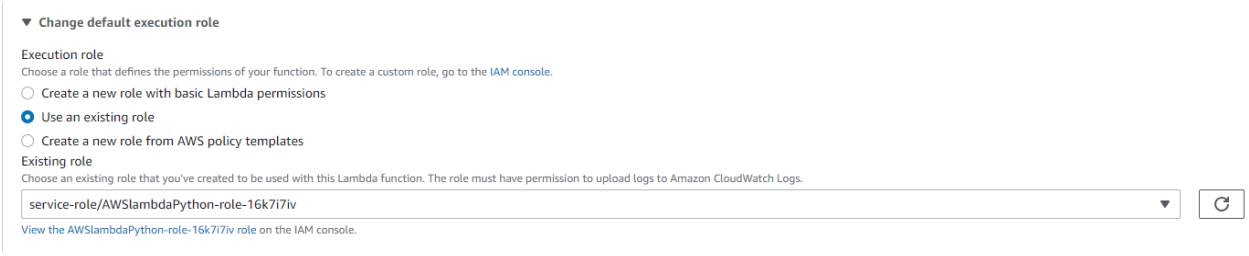




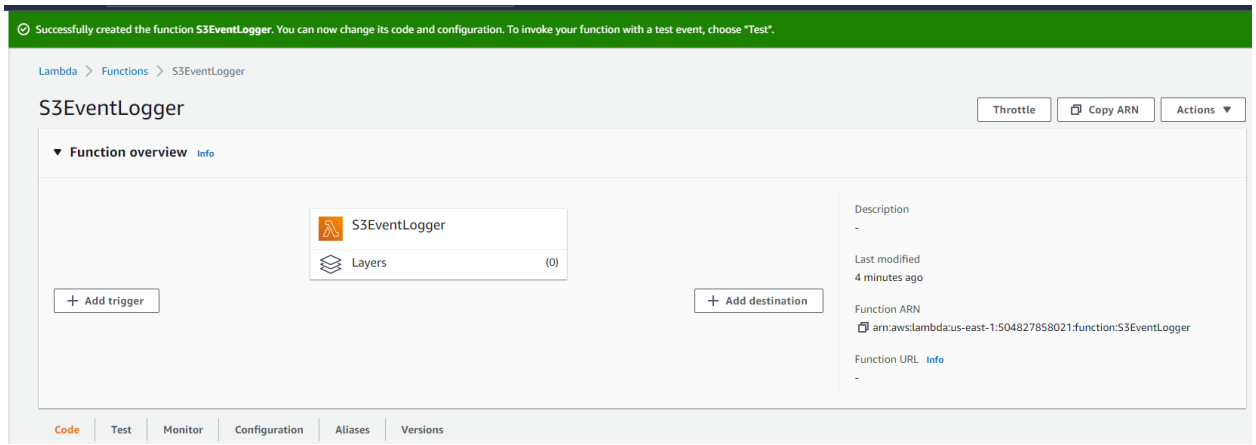
Step 2: Open up AWS Lambda and create a new Python function



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

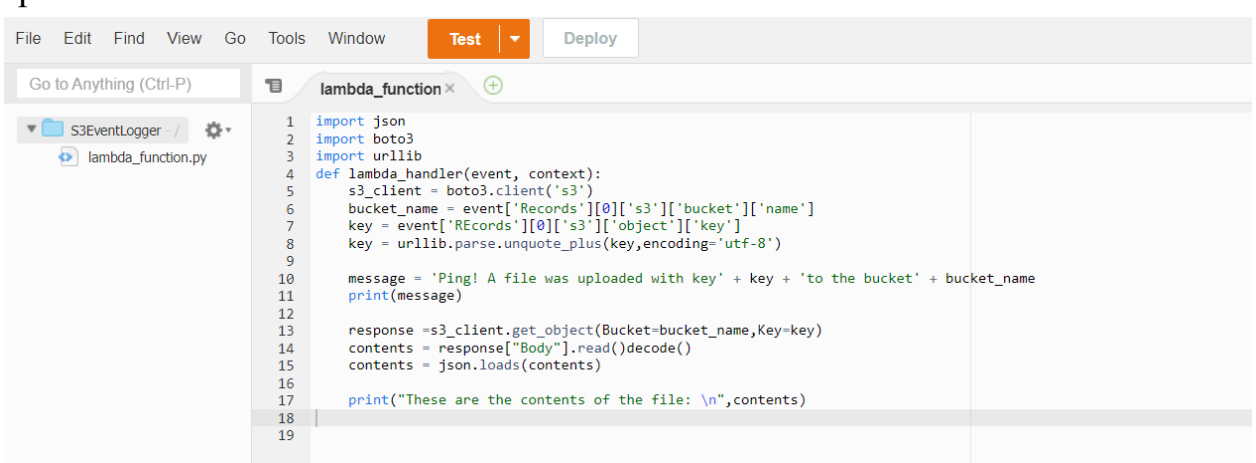


Step 3: The function is up and running



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

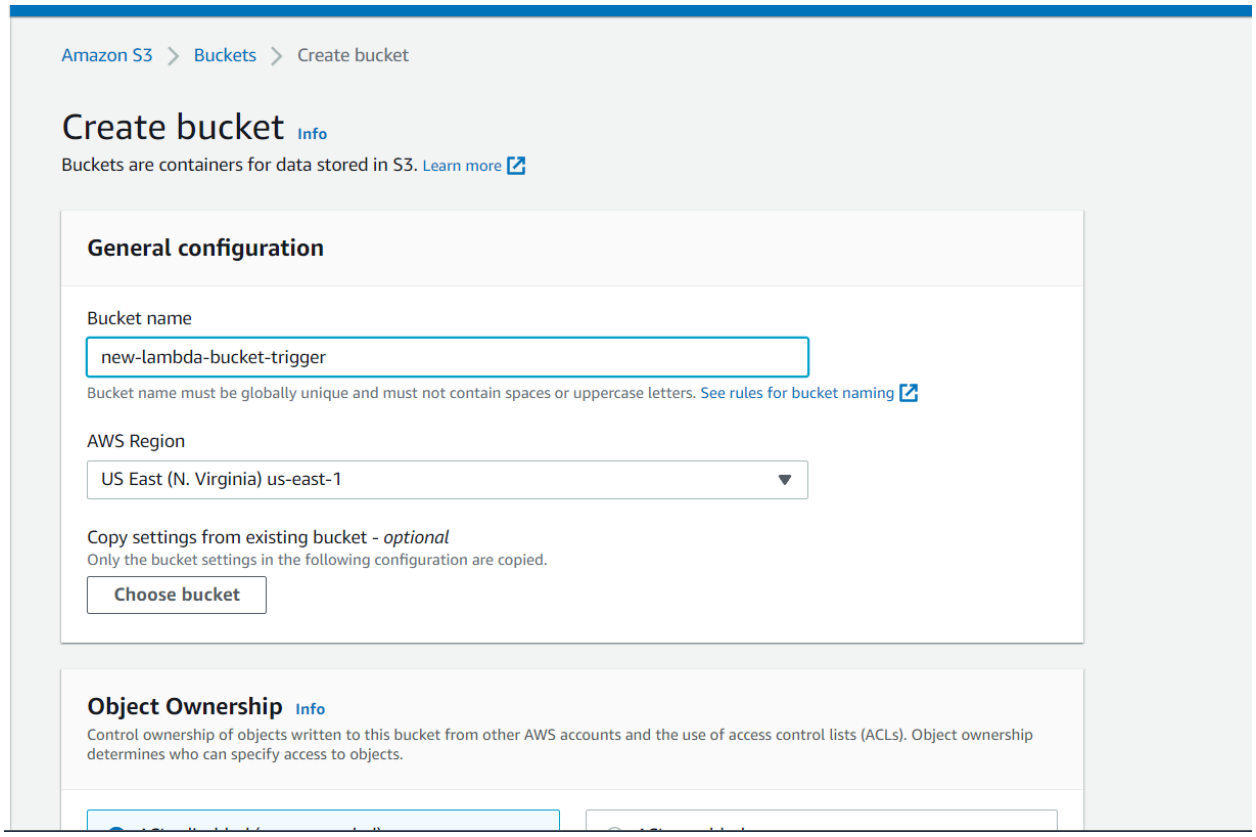
Format JSON


```
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": {
```

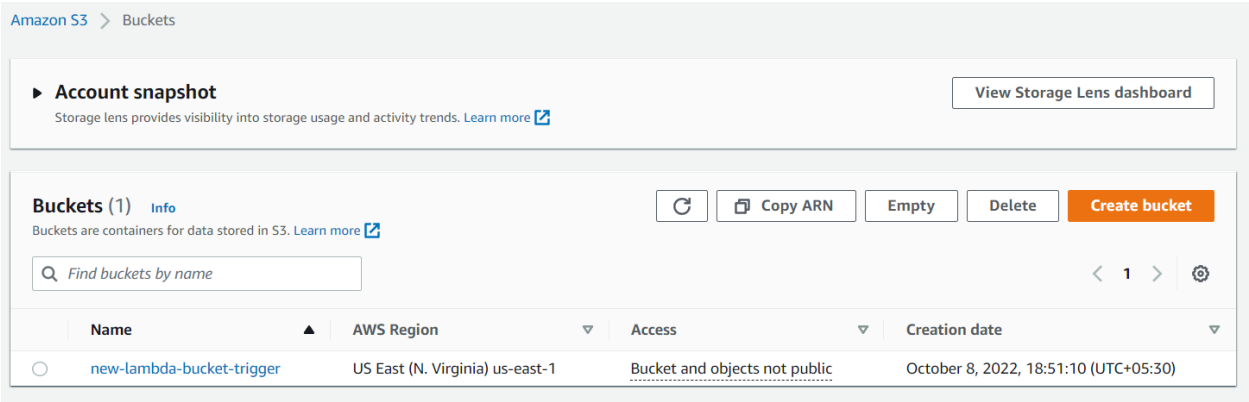

Here, inside the region, you may want to change the region to the AZ in which you've created your function and bucket. This doesn't seem mandatory but you might as well do it.

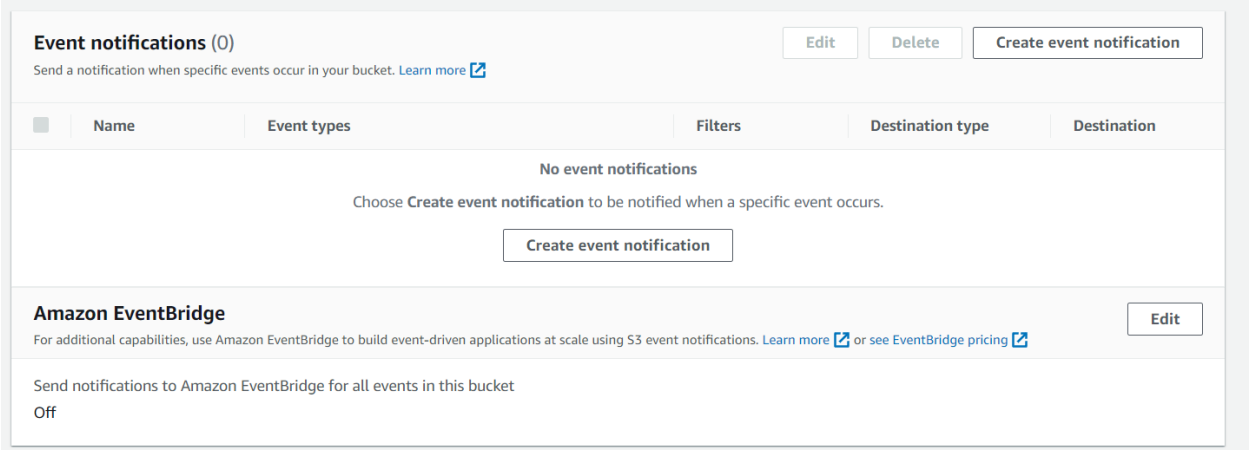
Step 6: Open up the S3 Console and create a new bucket

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



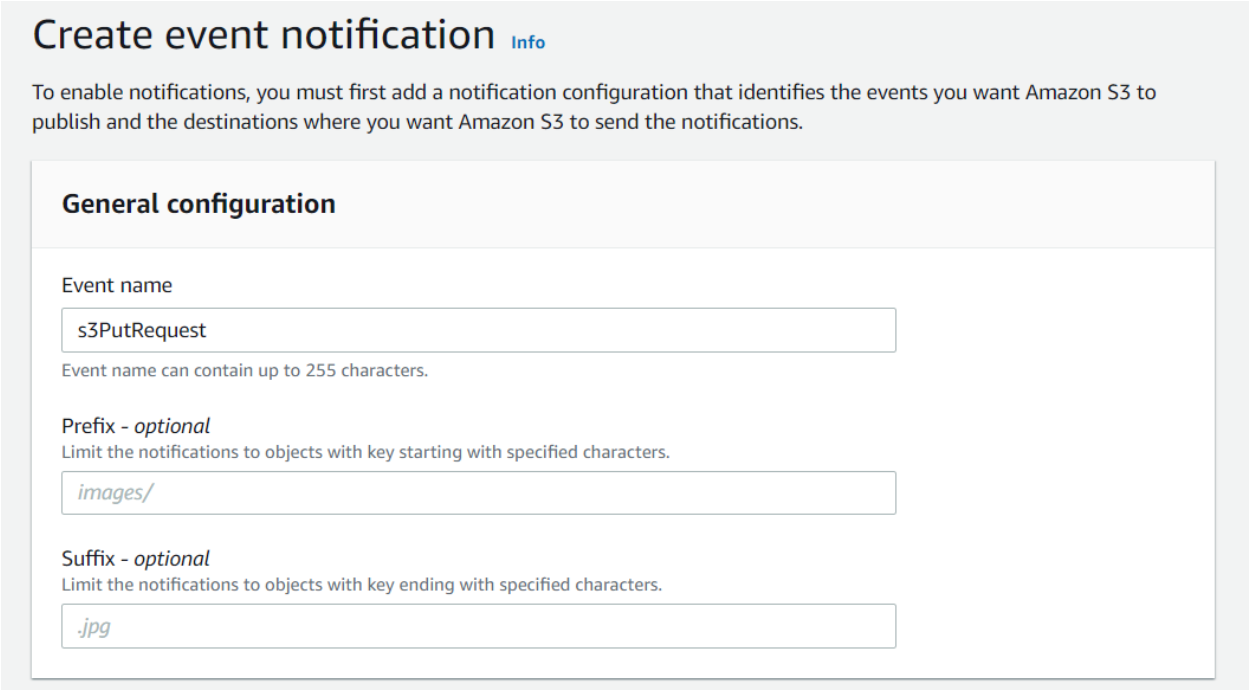
Step 8. Click on the created bucket and under properties, look for events.





Click on Create Event Notification.

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



Object creation

- [ ] All object create events
  s3:ObjectCreated:*
- [x] Put
  s3:ObjectCreated:Put
- [ ] Post
  s3:ObjectCreated:Post
- [ ] Copy
  s3:ObjectCreated:Copy
- [ ] Multipart upload completed
  s3:ObjectCreated:CompleteMultipartUpload

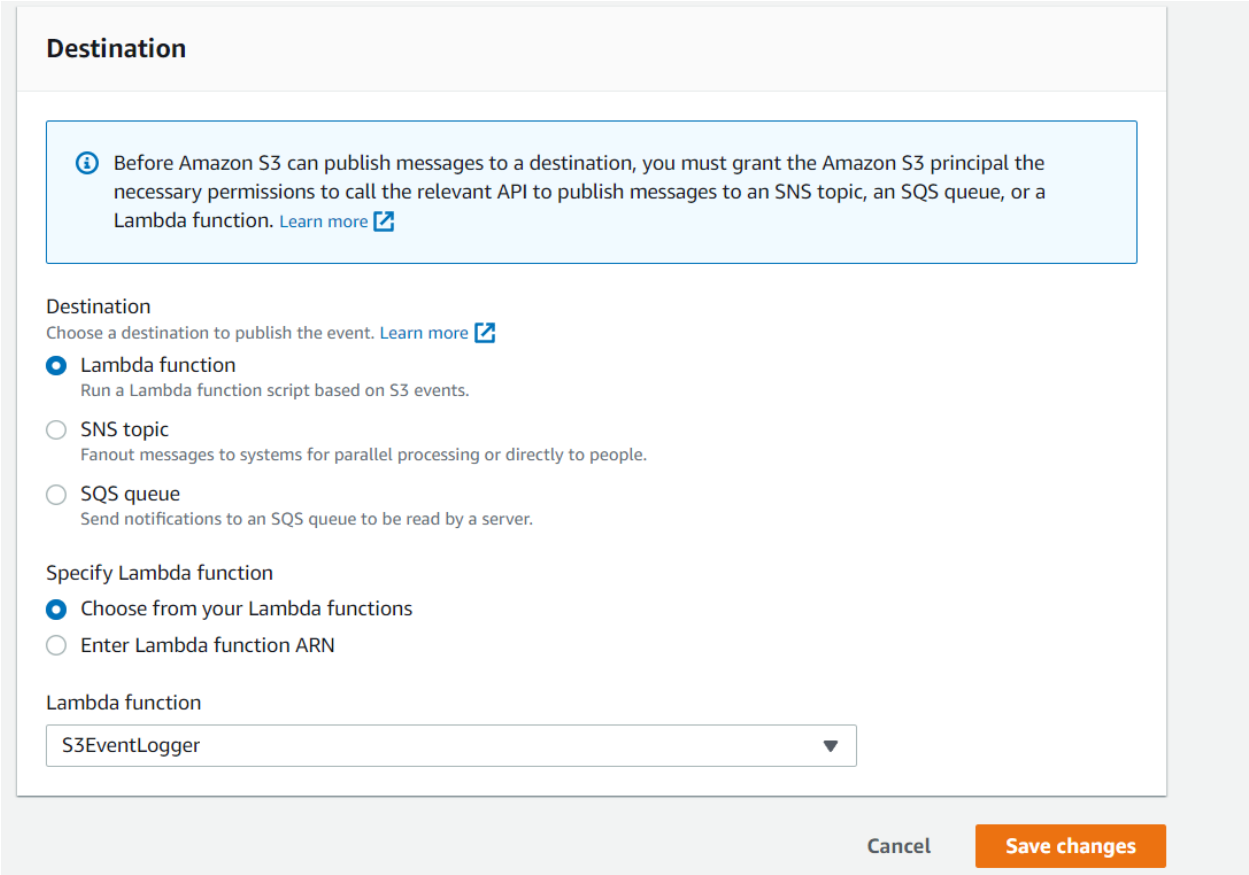
Object removal

- [ ] All object removal events
  s3:ObjectRemoved:*
- [ ] Permanently deleted
  s3:ObjectRemoved:Delete
- [ ] Delete marker created
  s3:ObjectRemoved:DeleteMarkerCreated

Object restore

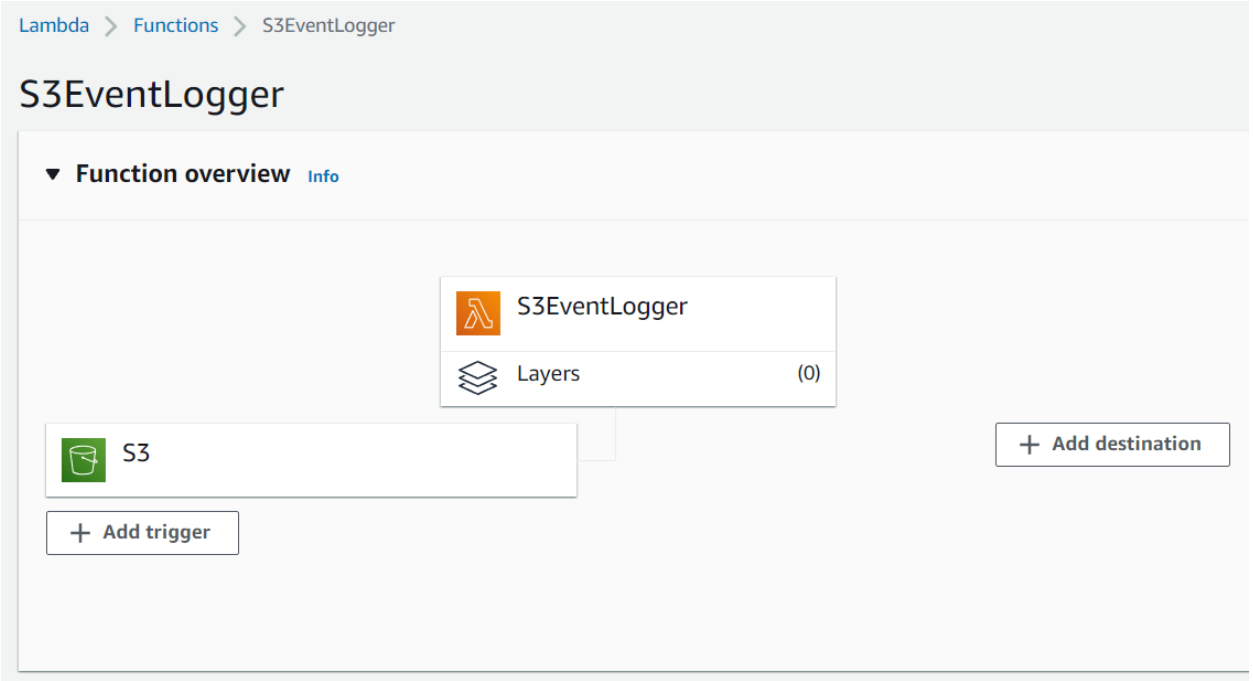
- [ ] All restore object events
  s3:ObjectRestore:*
- [ ] Restore initiated
  s3:ObjectRestore:Post
- [ ] Restore completed
  s3:ObjectRestore:Completed

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



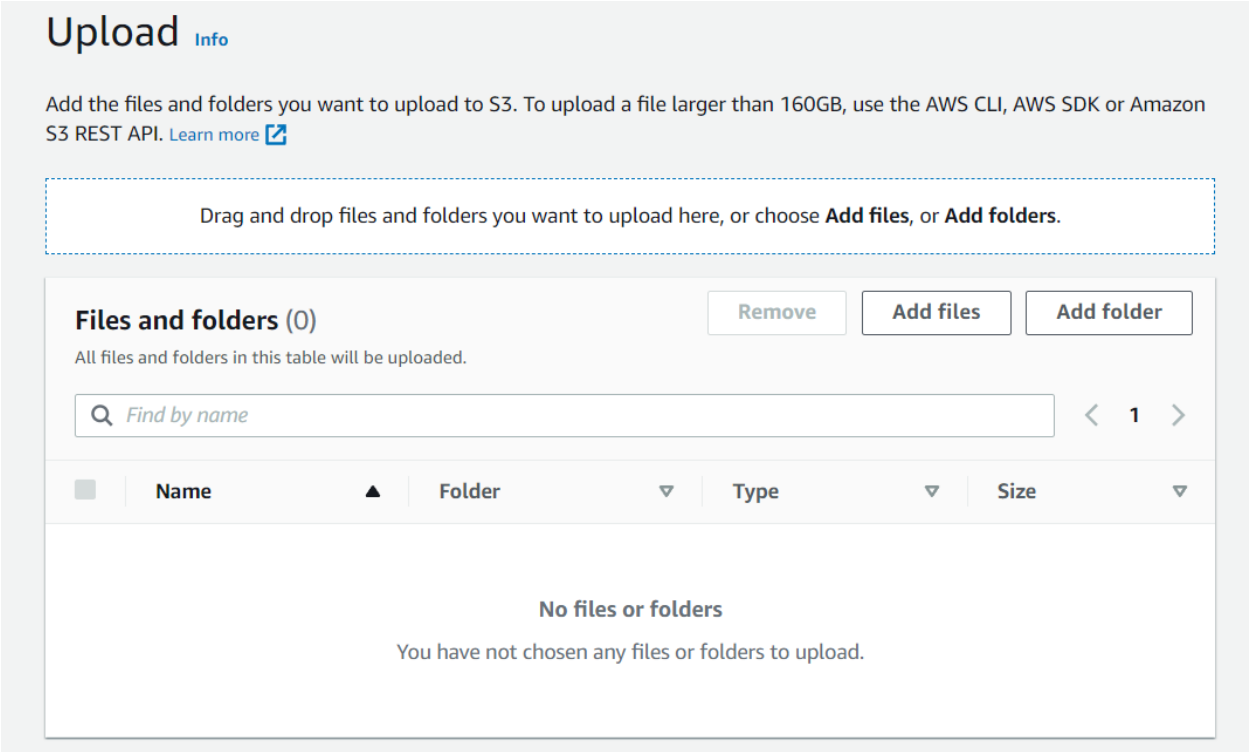
Choose Lambda function as destination and choose your lambda function and save the changes.

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

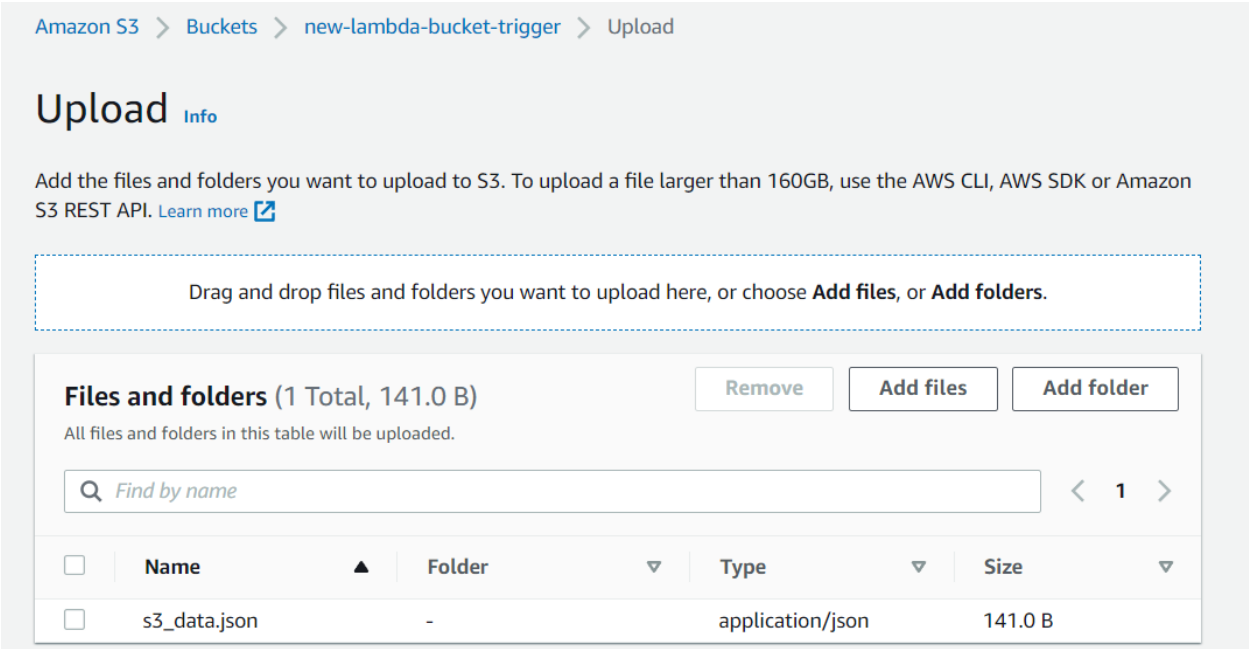


Step 11: Now, create a dummy JSON file locally.

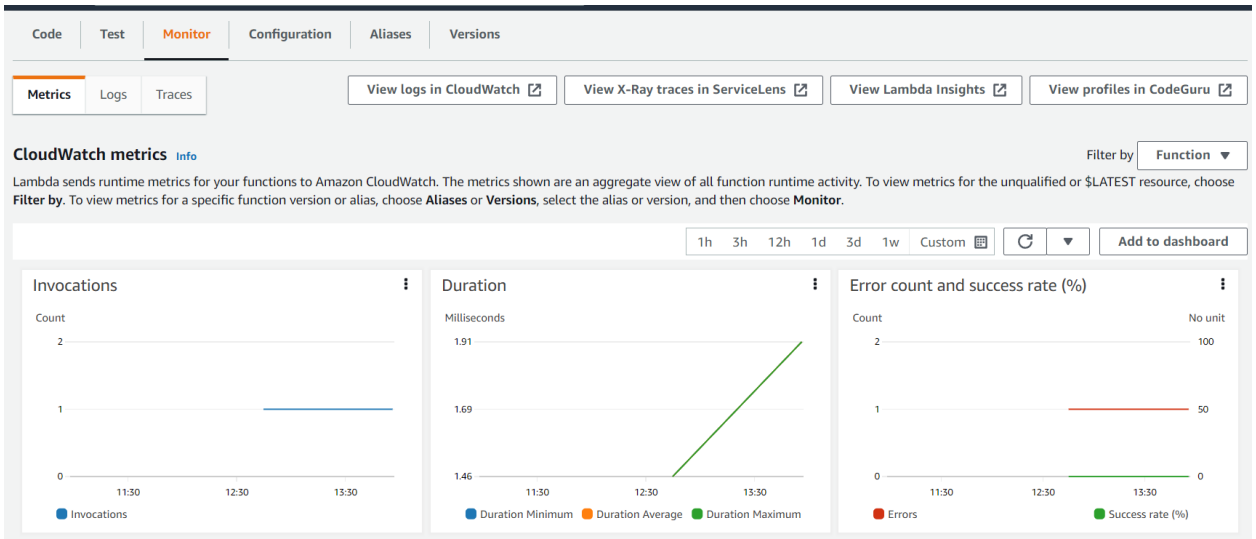
Step 12: Go back to your S3 Bucket and click on Add Files to upload a new file.



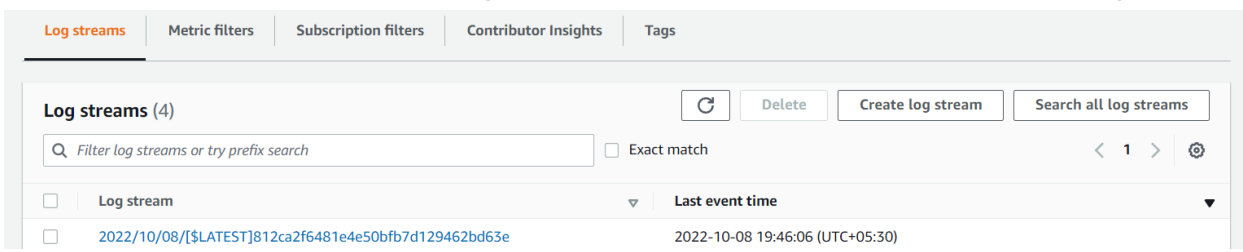
Step 13: Select the dummy data file from your computer and click Upload.

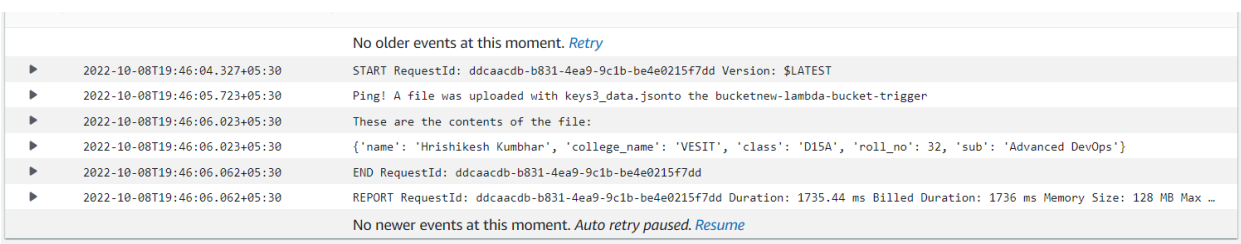


Step 14: Go back to your Lambda function and check the Monitor tab.



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

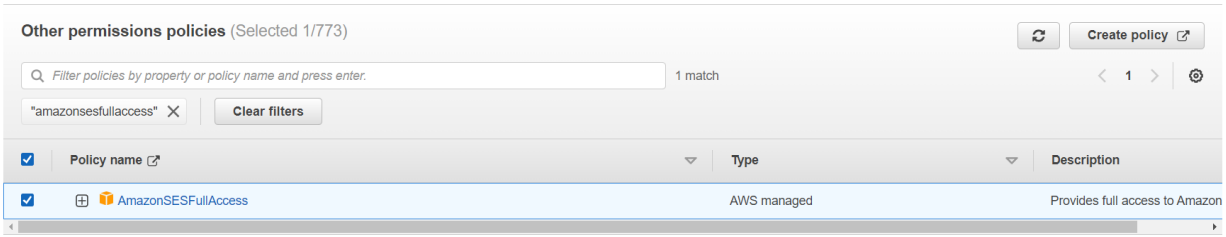


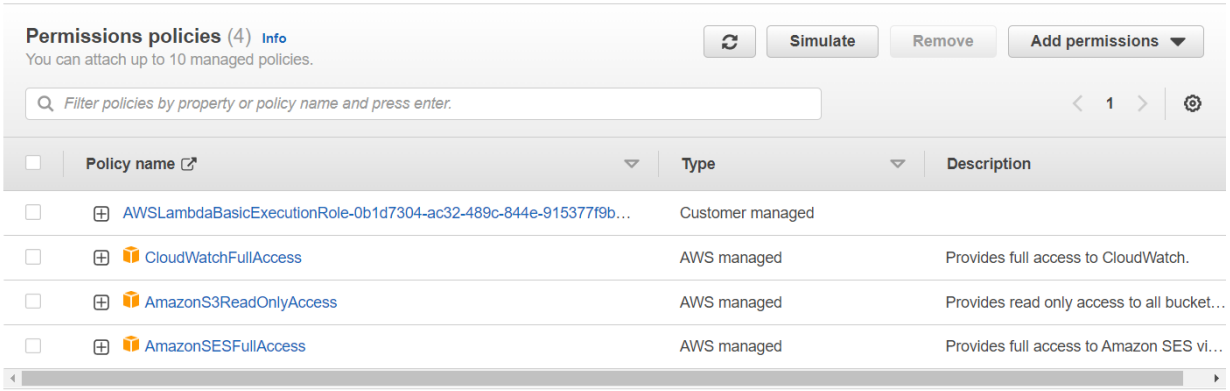


As you can see, our function logged that a file was uploaded with its file name and the bucket to which it was uploaded. It also mentions the contents inside the file as our function was defined to. Hence, we have successfully created a Python function inside AWS Lambda which logs every time an object is uploaded to an S3 Bucket.

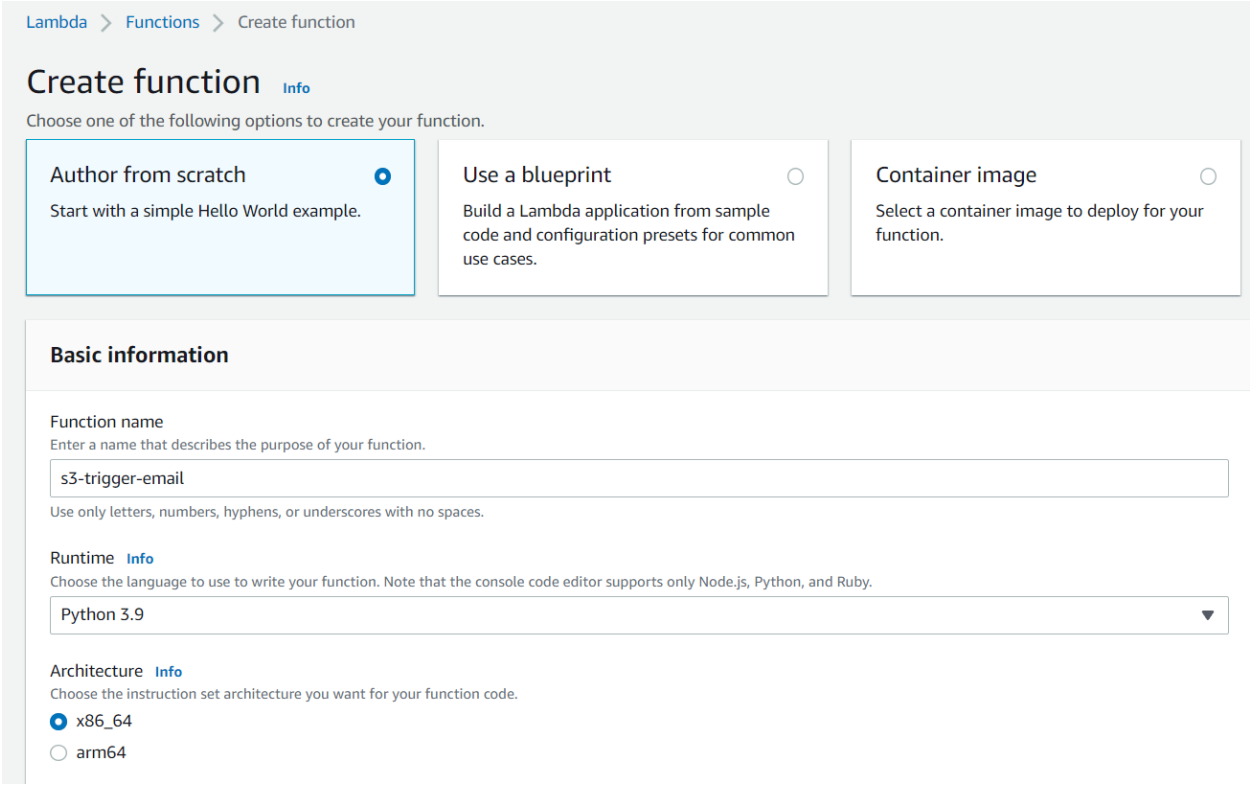
Sending an Email on Bucket additions to Bucket

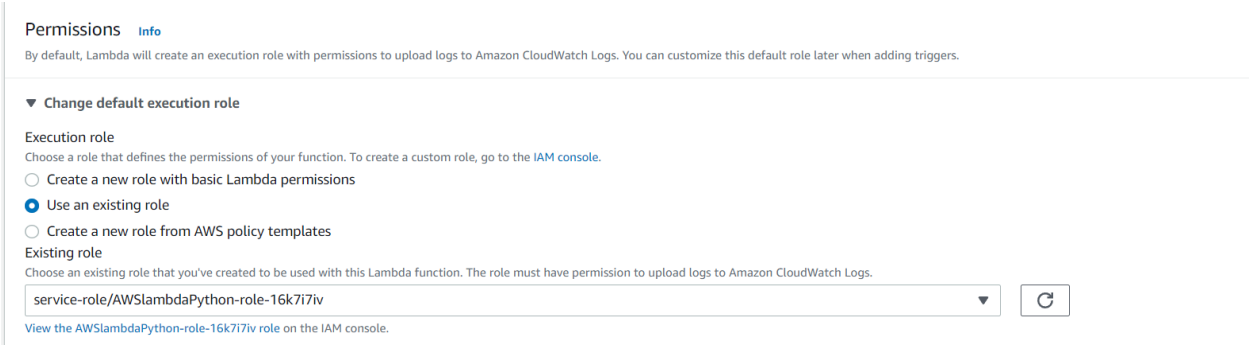
Step 1: Go to the IAM console and edit the same Lambda Role. This time, add SESFullAccess Permission to the role.





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





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.

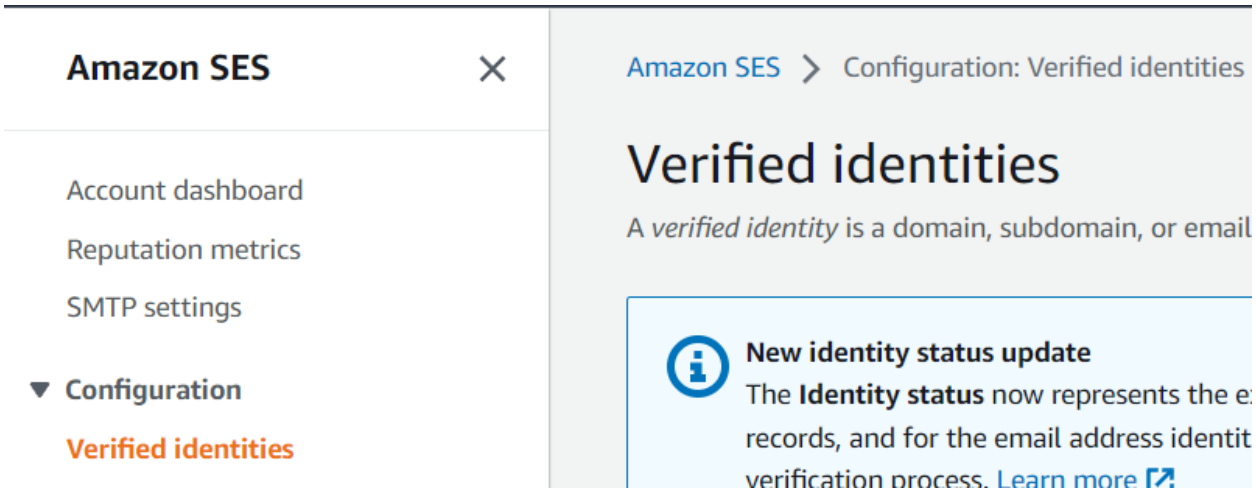
lambda_function

```python
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"
                ]
        },
```

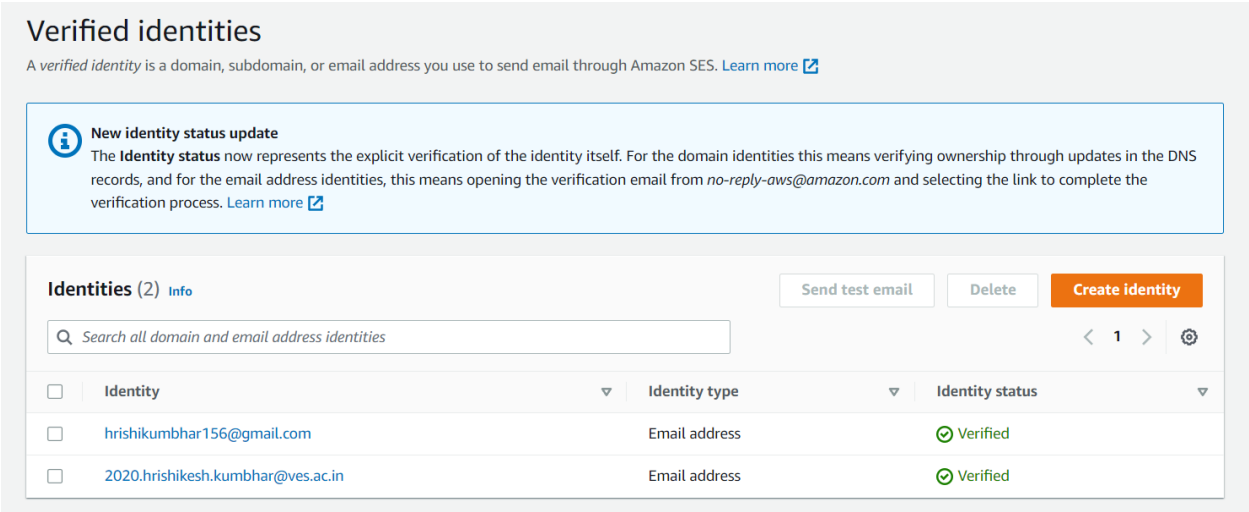
lambda_function.py

```
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.

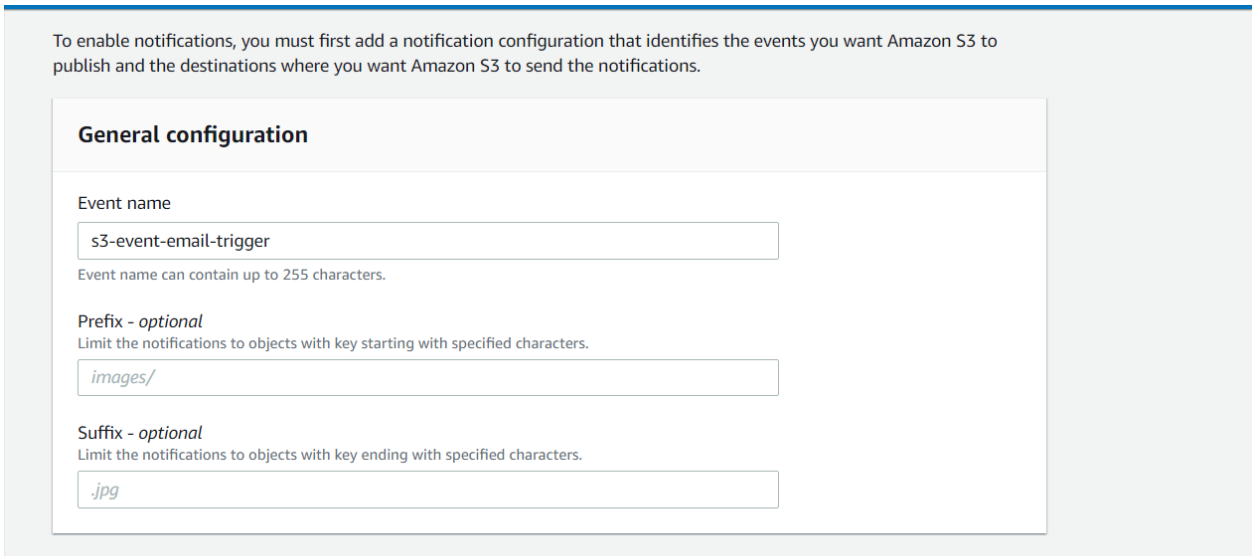


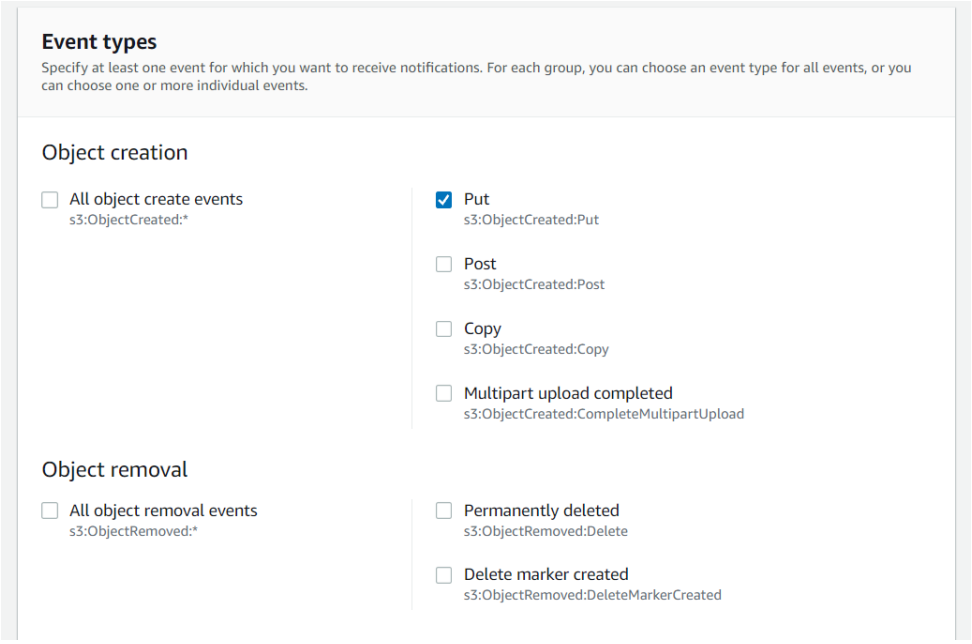
Step 5: Choose Verify Email Address and verify both sender and receiver email addresses

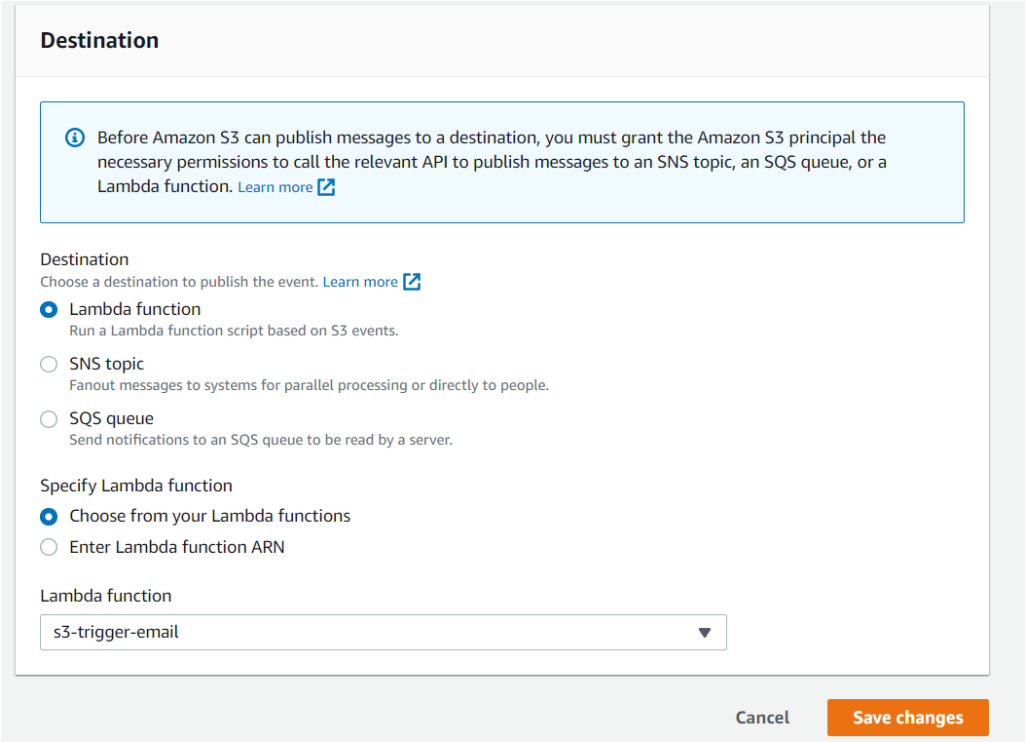


Click on the verification links you are sent and verify the emails.

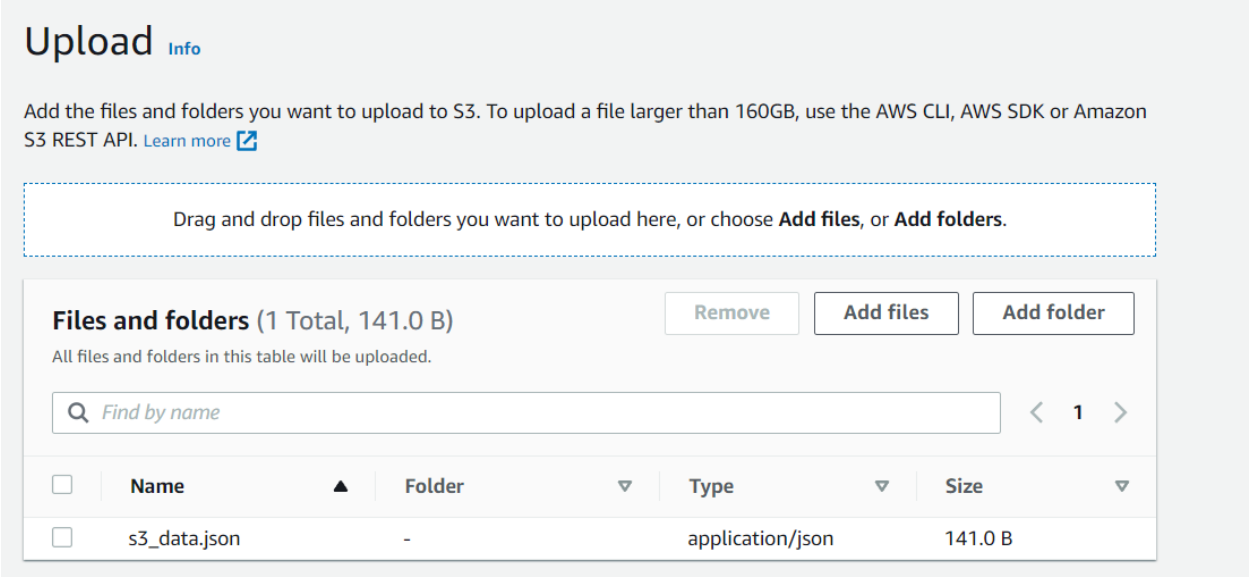
Step 6: Now, open up the S3 Console, create a new bucket as you did previously and add an event notification inside events and attach it to your Lambda function.



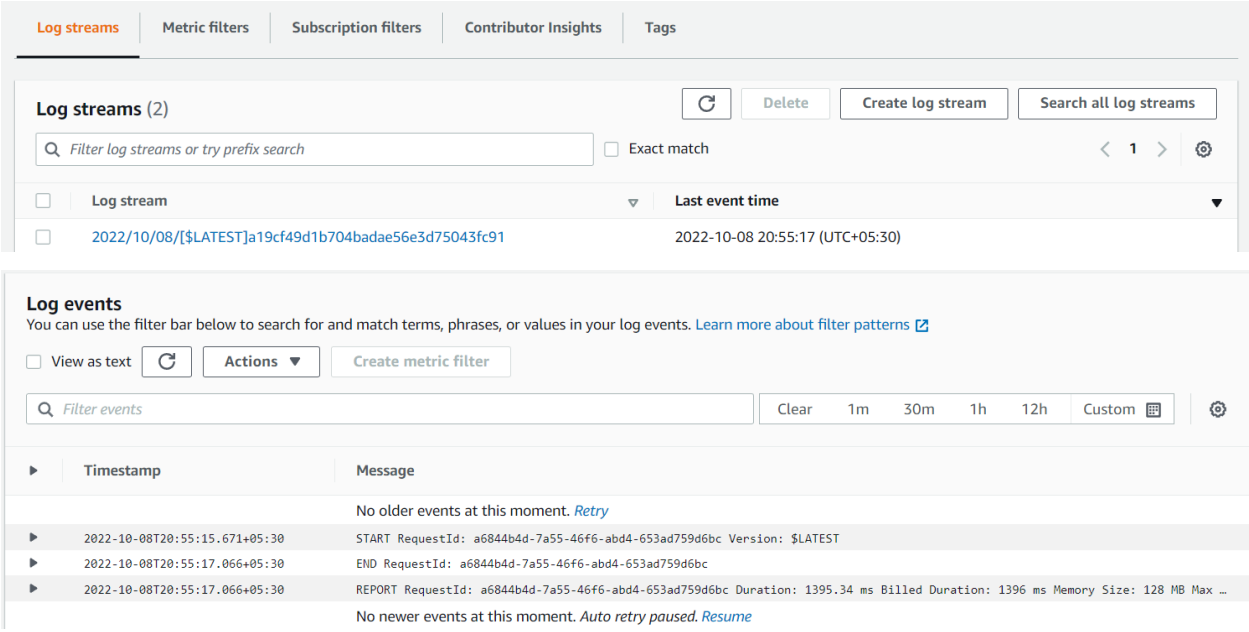


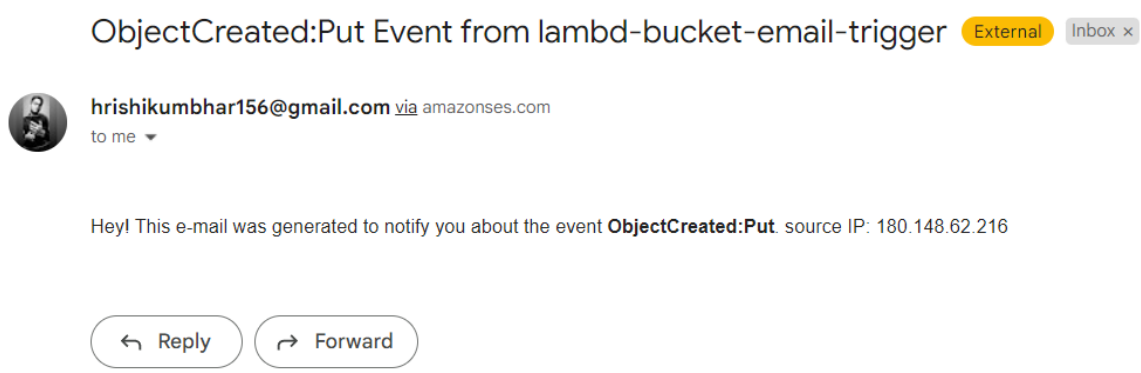


Step 7: Once that's done, upload any file to your S3 Bucket. I'll upload the same dummy JSON file again.



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





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.