Advanced DevOps Lab
Experiment 12

Roll No.	24
Name	Iyer Sreekesh Subramanian
Class	D15-A
Subject	Advanced DevOps Lab

Aim: To create a Lambda function that will log “An Image has been added” once you add an object to a specific bucket in S3.

Theory:

AWS Lambda

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS). Users of AWS Lambda create functions, self-contained applications written in one of the supported languages and runtimes, and upload them to AWS Lambda, which executes those functions in an efficient and flexible manner.

The Lambda functions can perform any kind of computing task, from serving web pages and processing streams of data to calling APIs and integrating with other AWS services.

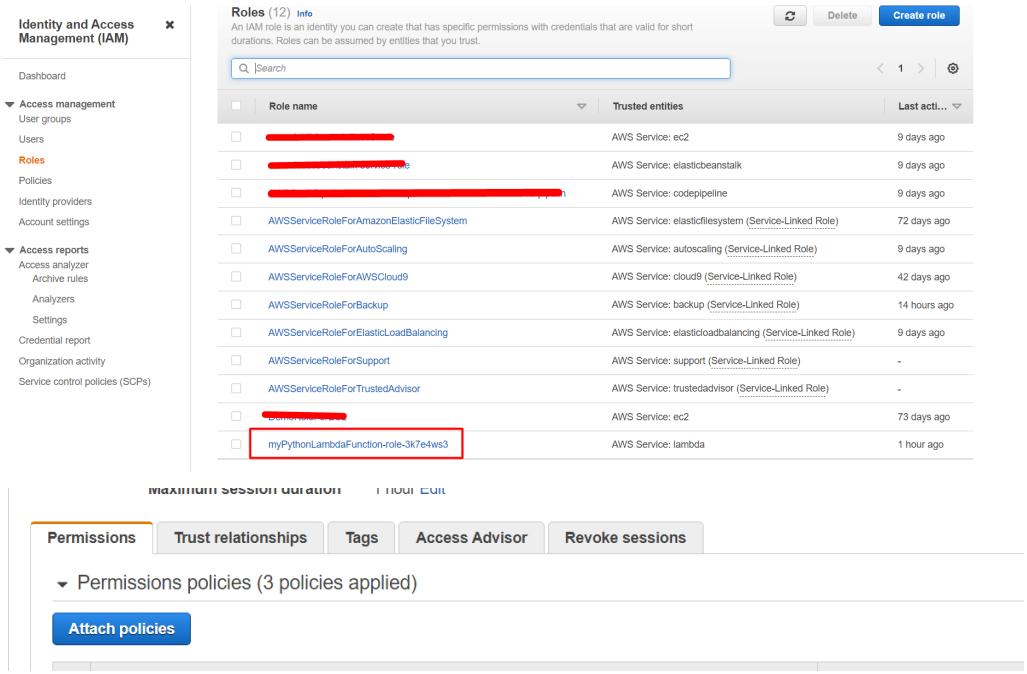
The concept of “serverless” computing refers to not needing to maintain your own servers to run these functions. AWS Lambda is a fully managed service that takes care of all the infrastructure for you. And so “serverless” doesn’t mean that there are no servers involved: it just means that the servers, the operating systems, the network layer and the rest of the infrastructure have already been taken care of so that you can focus on writing application code.

Features of AWS Lambda

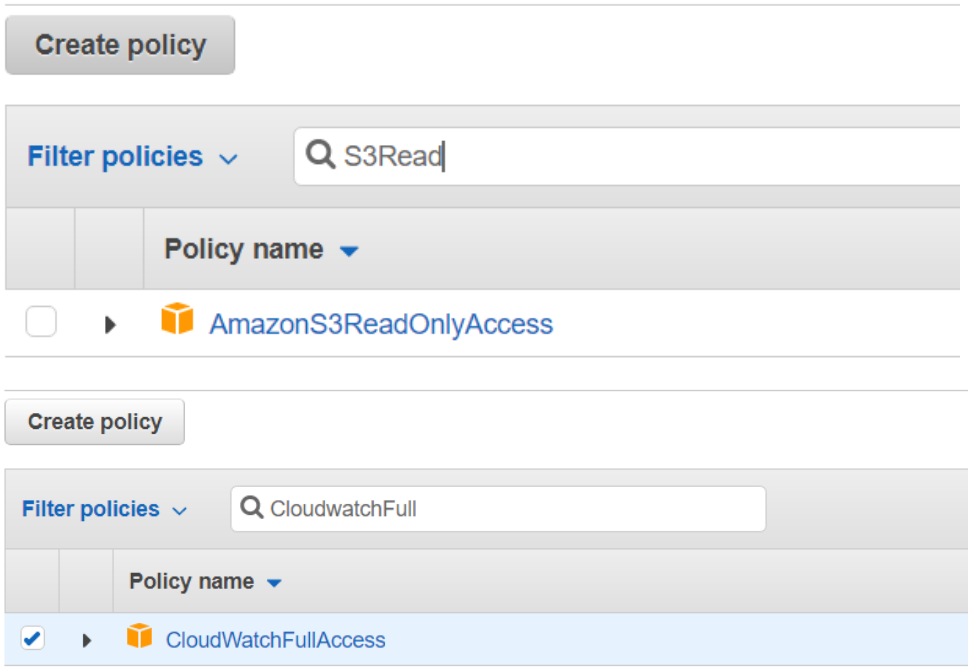
- AWS Lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved.
- It offers multiple options like AWS S3, CloudWatch, DynamoDB, API Gateway, Kinesis, CodeCommit, and many more to trigger an event.
- You don’t need to invest upfront. You pay only for the memory used by the lambda function and minimal cost on the number of requests hence cost-efficient.
- AWS Lambda is secure. It uses AWS IAM to define all the roles and security policies.
- It offers fault tolerance for both services running the code and the function. You do not have to worry about the application down.

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

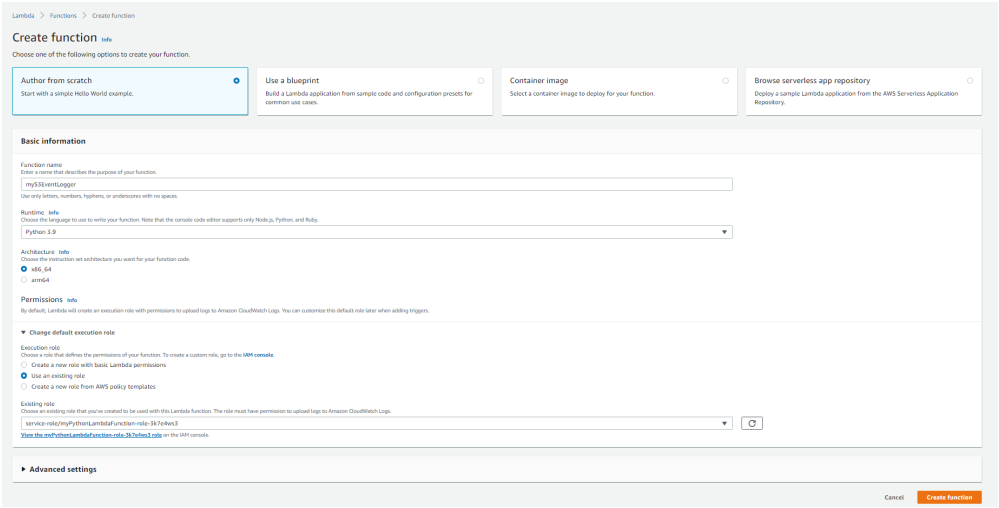
1. Open up the IAM Console and under Roles, choose the Role we previously created for the Python Lambda Function.



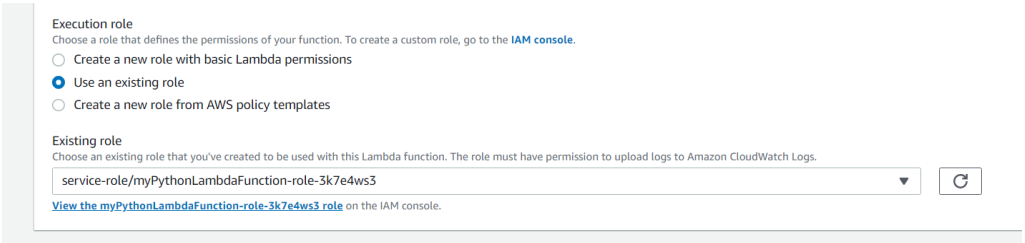
Under Attach Policies, add S3-ReadOnly and CloudWatchFull permissions to this role.



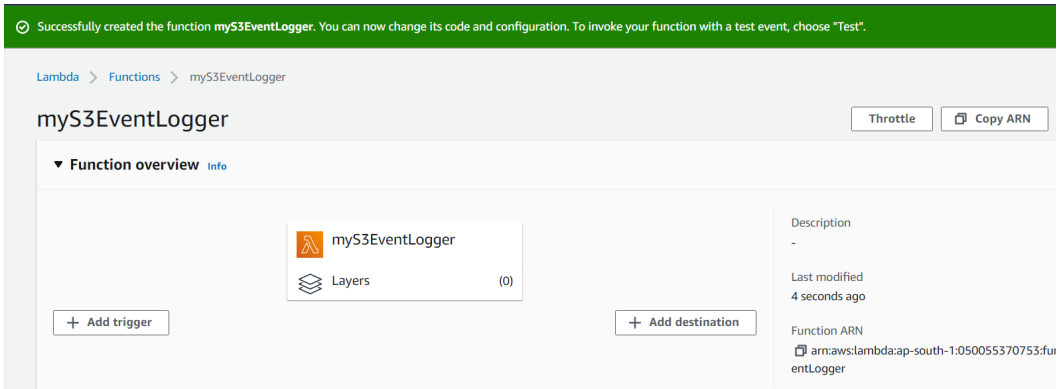
2. Open up AWS Lambda and create a new Python function



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

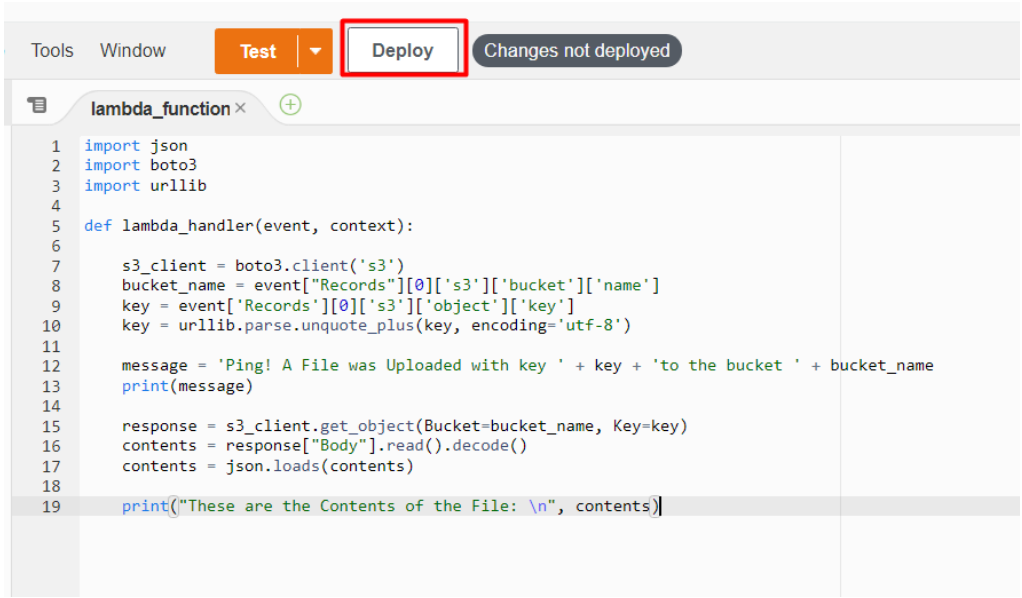


3. The function is up and running.



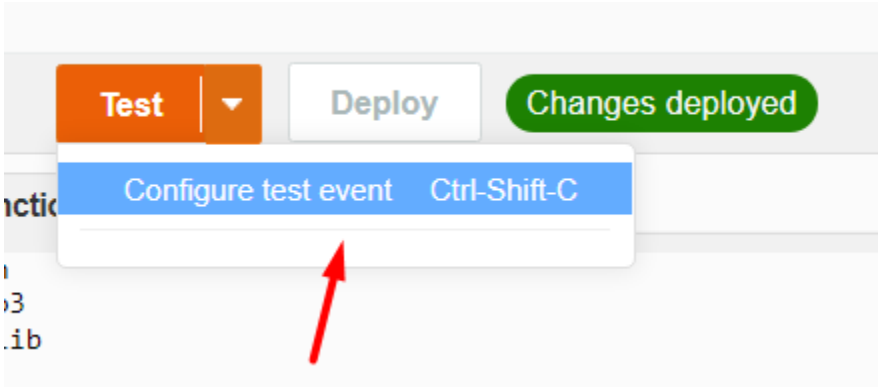
4. Make the following changes to the function and click on the deploy button.

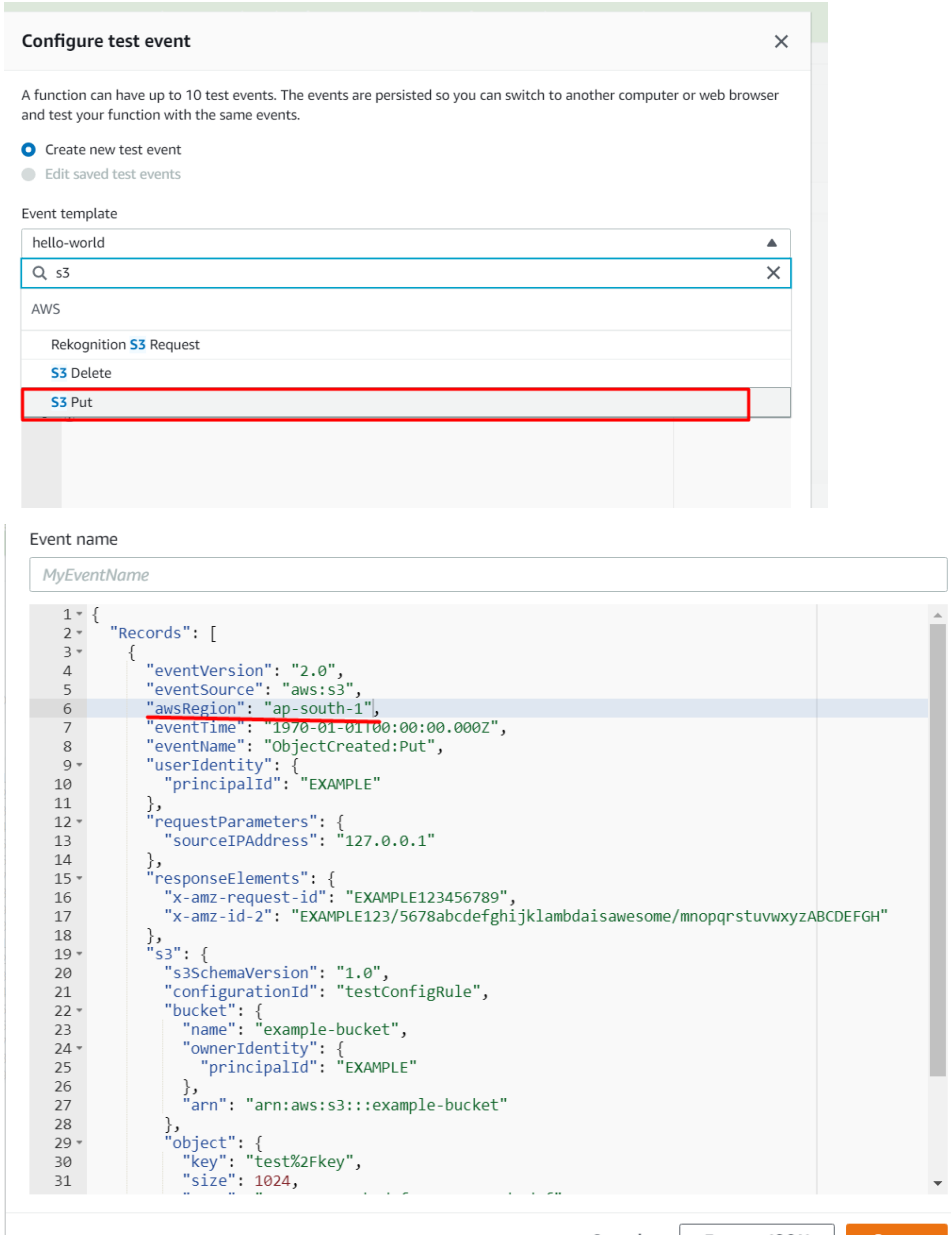
   This code basically logs a message and logs the contents of a JSON file which is uploaded to an S3 Bucket.




```python
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)
```

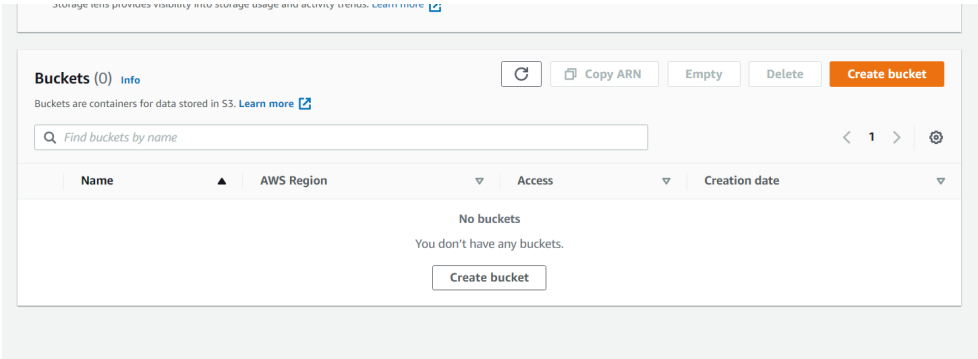
5. Click on Test and choose the ‘S3 Put’ Template.



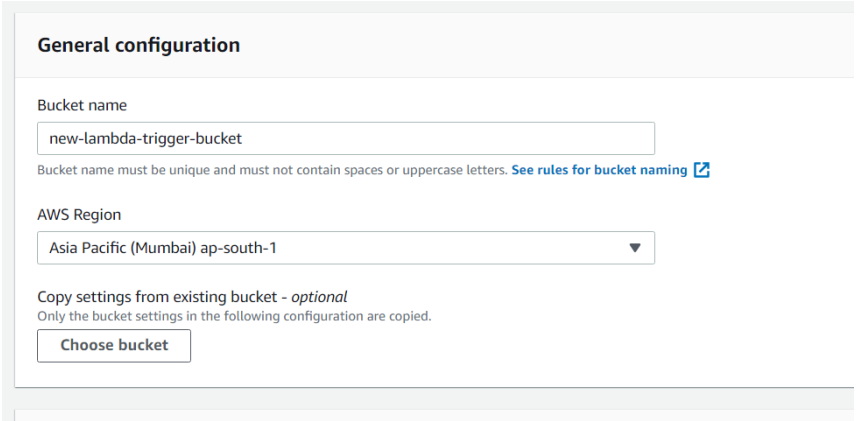


Here, inside the region, you may want to change the region to the AZ in which you've created your function and bucket. This doesn't seem mandatory but you might as well do it.

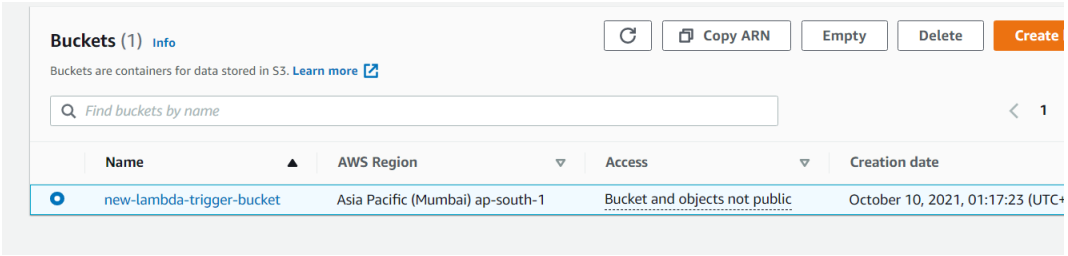
6. Open up the S3 Console and create a new bucket

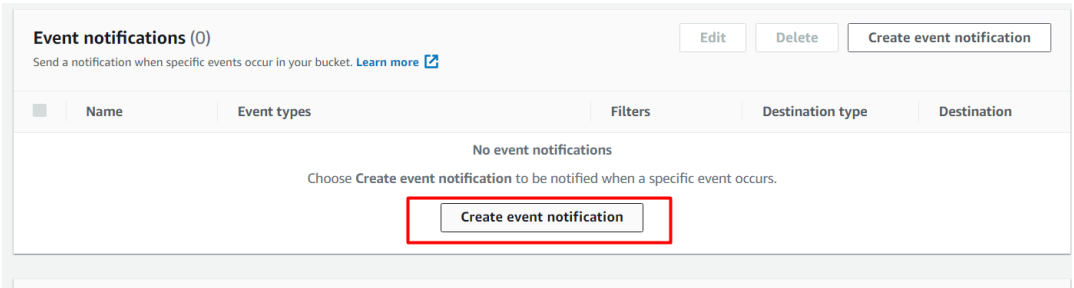


7. With all general settings, create the bucket in the same region as the function.



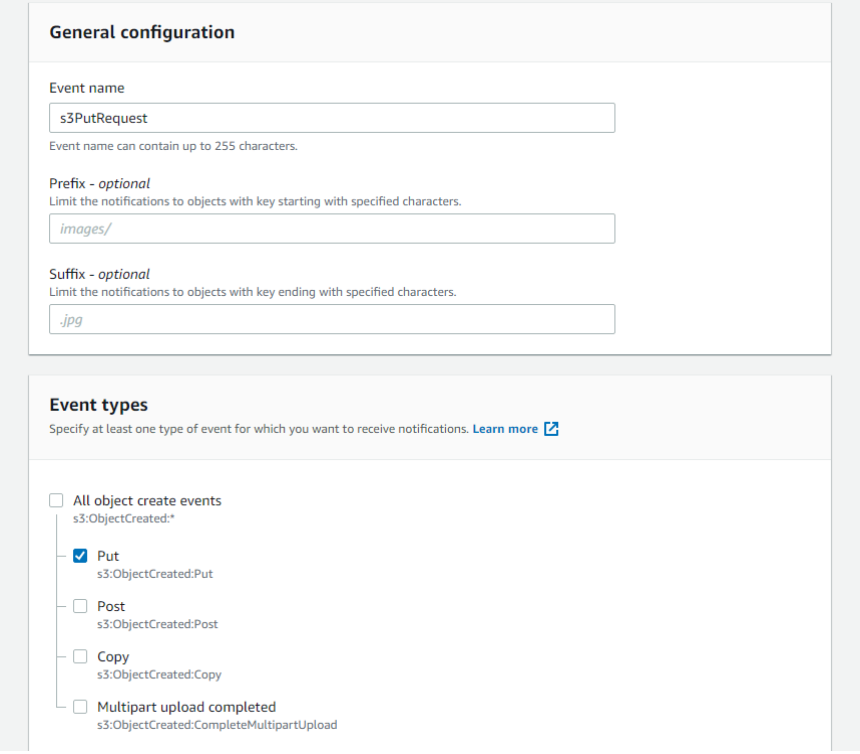
8. Click on the created bucket and under properties, look for events.



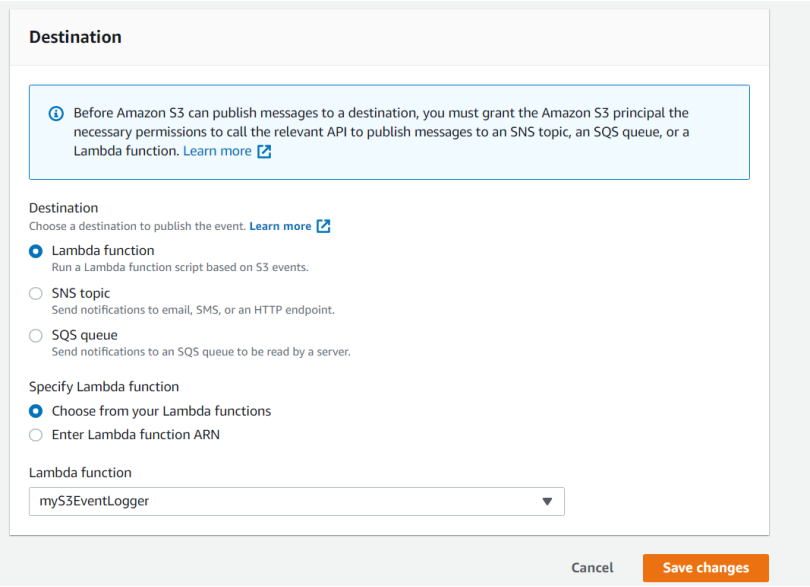


Click on Create Event Notification.

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

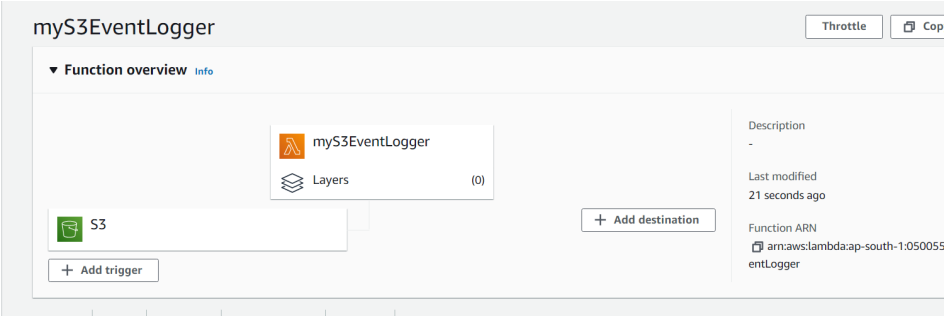


You can optionally choose .json under the suffix since the code only accepts JSON.



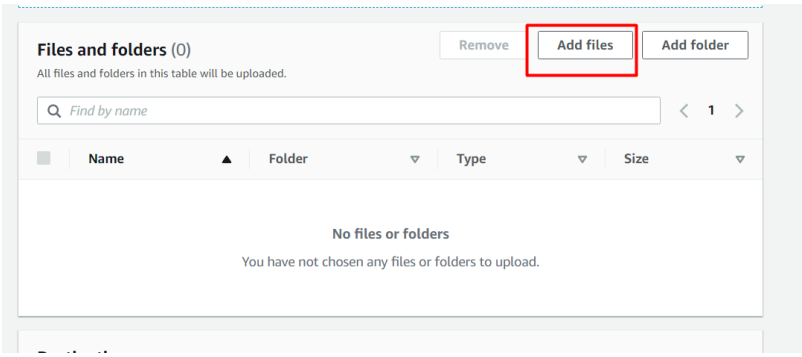
Choose Lambda function as destination and choose your lambda function and save the changes.

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

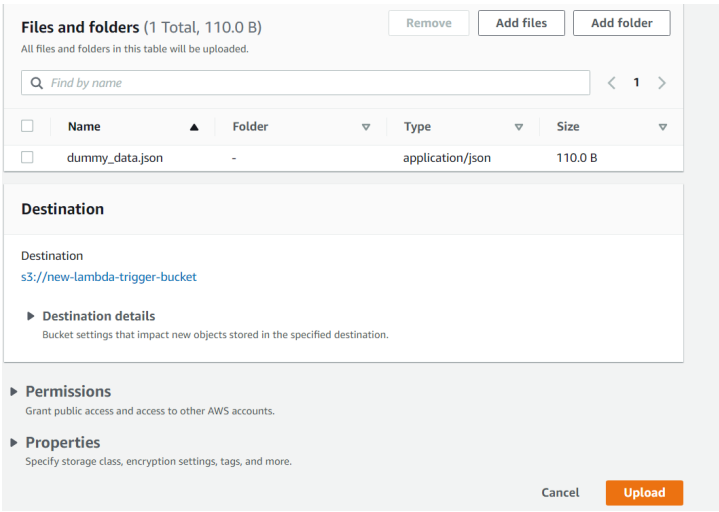


11. Now, create a dummy JSON file locally.

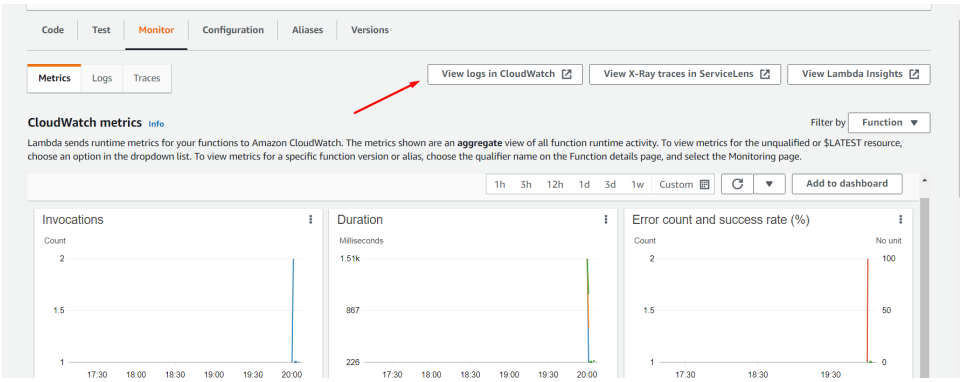
12. Go back to your S3 Bucket and click on Add Files to upload a new file.



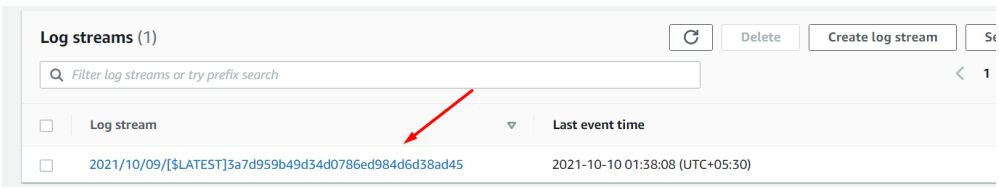
13. Select the dummy data file from your computer and click Upload.



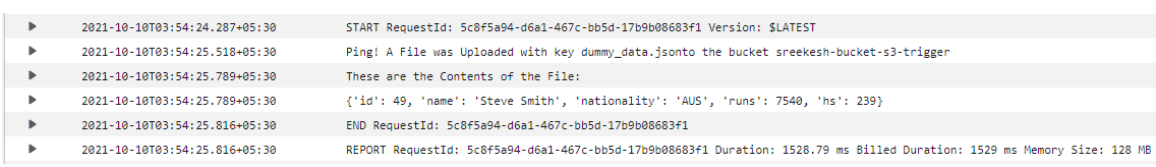
14. Go back to your Lambda function and check the Monitor tab.



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



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



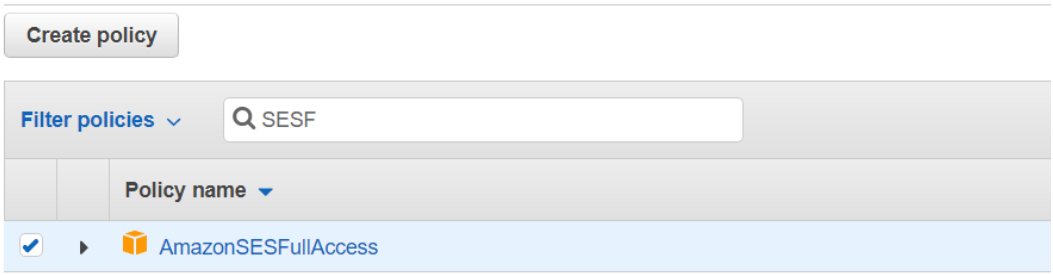
As you can see, our function logged that a file was uploaded with its file name and the bucket to which it was uploaded. It also mentions the contents inside the file as our function was defined to.

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

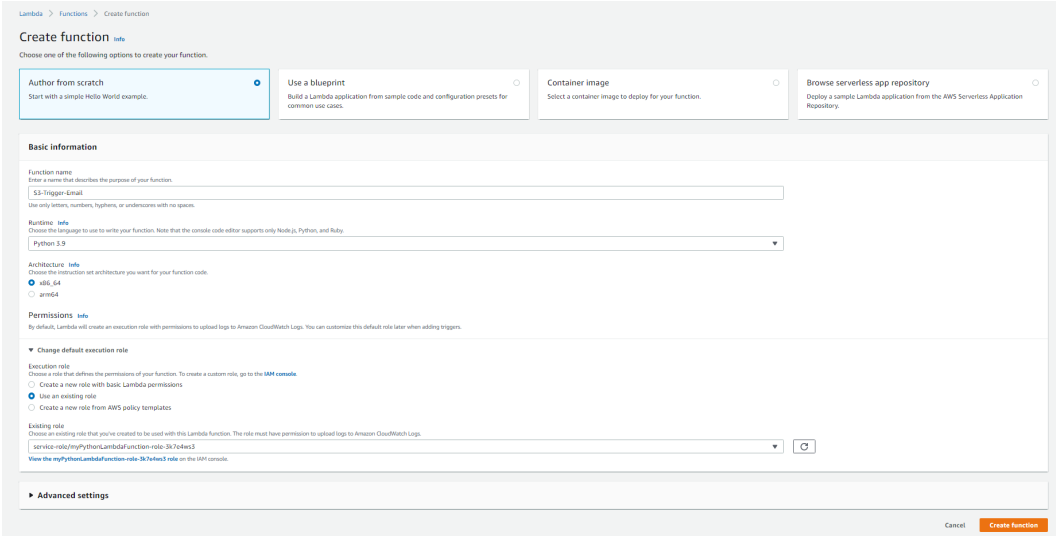
OPTIONAL

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.



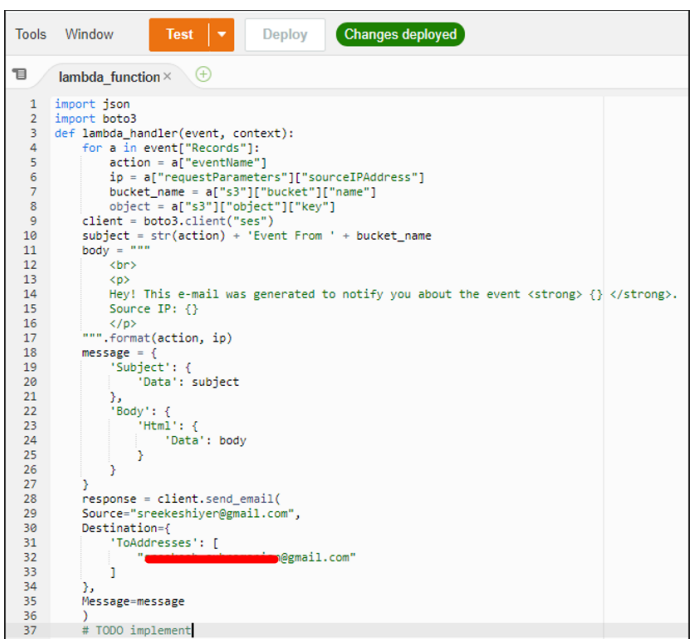
2. Create a new Lambda function in a Python environment. Use the existing role which was previously created.



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

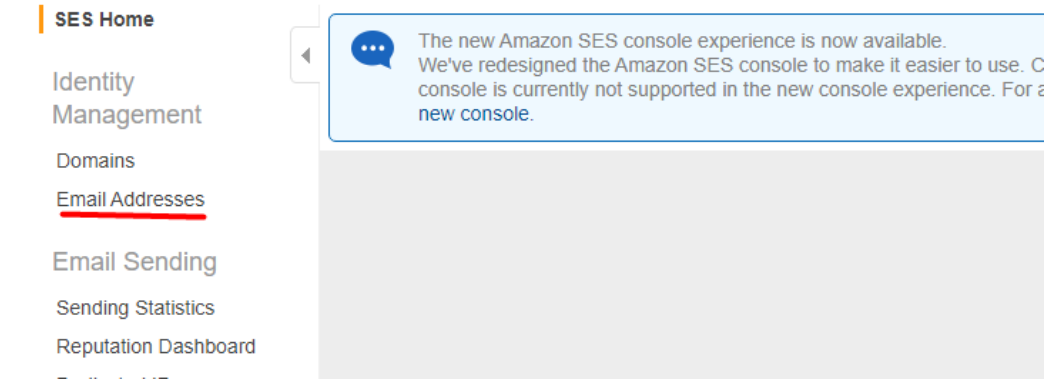
   This code is basically to send an email on the creation of an object in the attached S3 Bucket. It sends the bucket name, event and source IP address.

   In this code, modify the **Source** and **Destination ToAddresses** to your sender and receiver email addresses. Once done, deploy the function.

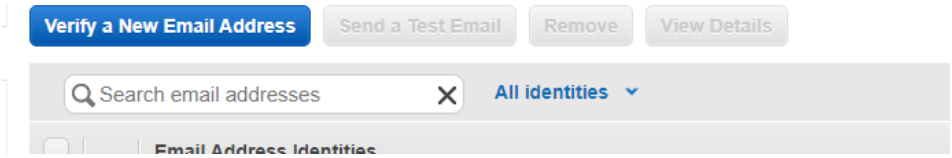


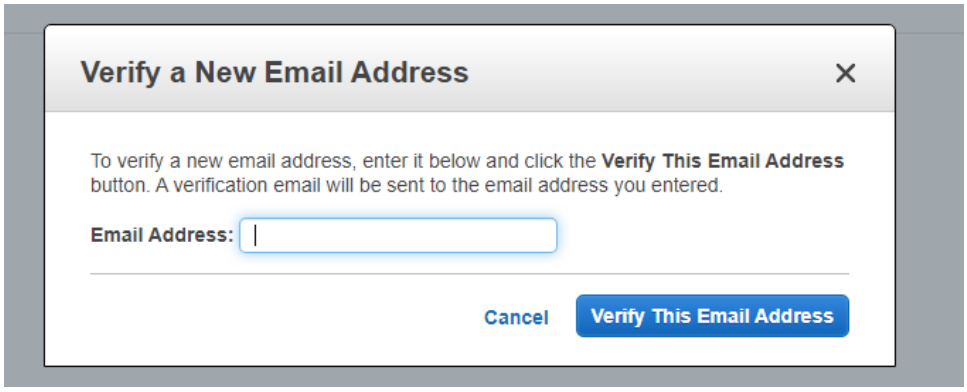
```python
import json
import boto3
def lambda_handler(event, context):
    for a in event["Records"]:
        action = a["eventName"]
        ip = a["requestParameters"]["sourceIPAddress"]
        bucket_name = a["s3"]["bucket"]["name"]
        object = a["s3"]["object"]["key"]
    client = boto3.client("ses")
    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="sreekeshiyer@gmail.com",
    Destination={
        'ToAddresses': [
            "[redacted]@gmail.com"
        ]
    },
    Message=message
    )
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hey there! Check your mail I guess..')
    }
```

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

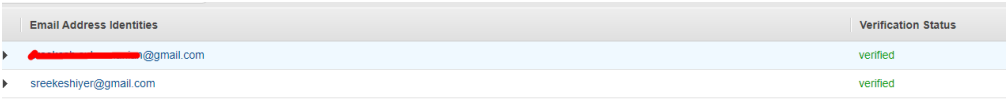


5. Choose Verify Email Address and verify both sender and receiver email addresses

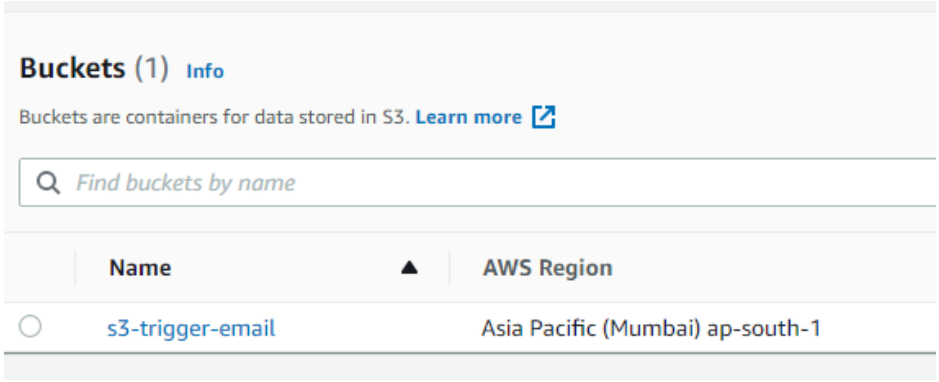


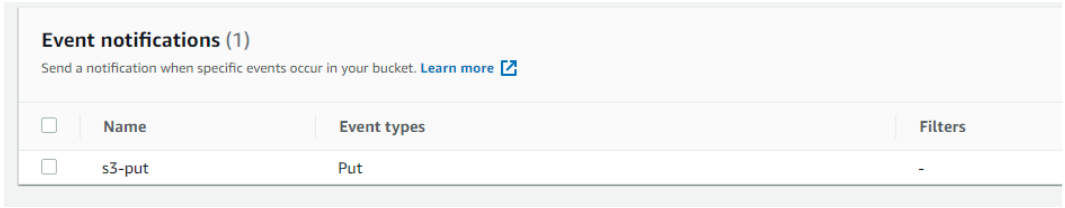


Click on the verification links you are sent and verify the emails.

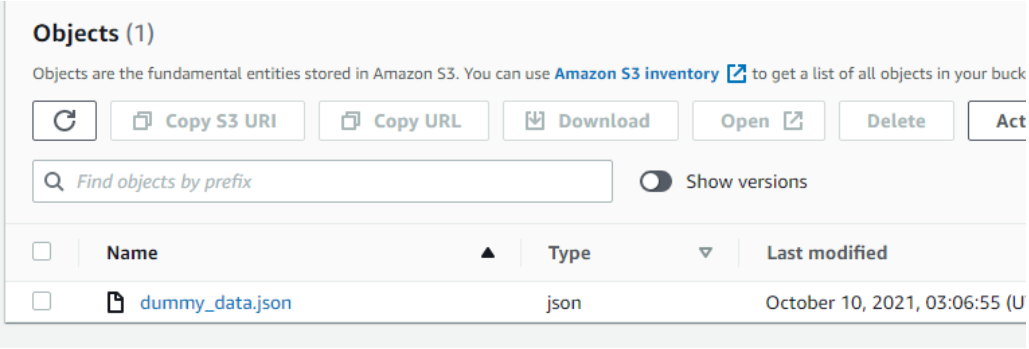


6. Now, open up the S3 Console, create a new bucket as you did previously and add an event notification inside events and attach it to your Lambda function.

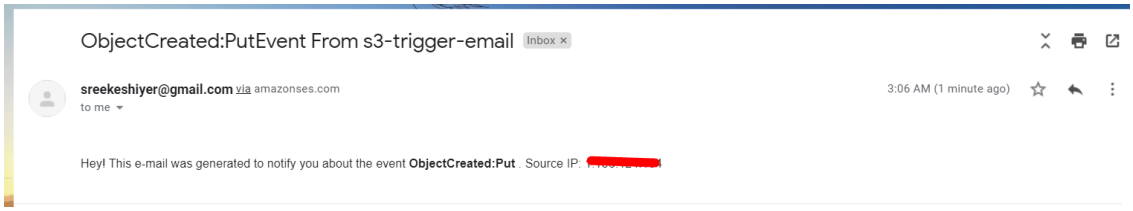




7. Once that's done, upload any file to your S3 Bucket. I'll upload the same dummy JSON file again.



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.