Lambda Python

- Navigate to AWS Lambda Service and select `Create Function`
- Create a Lambda function with default function code using AWS Console
- Open the Functions page on the Lambda console.
- Choose Create function.
- Under Basic information, do the following:
  - For Function name, enter `my-lambda-function`.
  - For Runtime, confirm that `Python 3.8` is selected.
- Choose `Create function`.
- Lambda creates a Python function and an `execution role` that grants the function permission to upload logs to CloudWatch Log Group.
- Cloudwatch Log Group Format will be : `/aws/lambda/<LAMBDA_FUNCTION_NAME>`
- The Lambda function assumes the execution role when you invoke your function, and uses the execution role to create credentials for the AWS SDK and to read data from event sources.

Use the designer

- The Designer shows an overview of your function along and its upstream and downstream resources. You can use it to configure triggers, layers, and destinations.

1) AWS Lambda Function Handler in Python

- At the time you create a Lambda function, you specify a handler, which is a function in your code, that AWS Lambda can invoke when the service executes your code.

Invoke the Lambda function

- To invoke a function, In the upper right corner, choose `Test`.
- In the Configure test event dialog box, choose `Create new test event`.
- In Event template, leave the default Hello World option. Enter an Event name and note the following sample event template:

```
{
    "key1": "value1",
    "key2": "value2",
    "key3": "value3"
  }
```

- This is the Event passed to the Lambda Function when it is executed.

Handler

- The Lambda function handler name specified at the time you create a Lambda function is derived from the following:
    - the name of the Python file in which the Lambda handler function is located
    - the name of the Python handler function

> `lambda_function.lambda_handler`

> `filename.functionName`

General Syntax :

```
def handler_name(event, context):
    return some_value
```

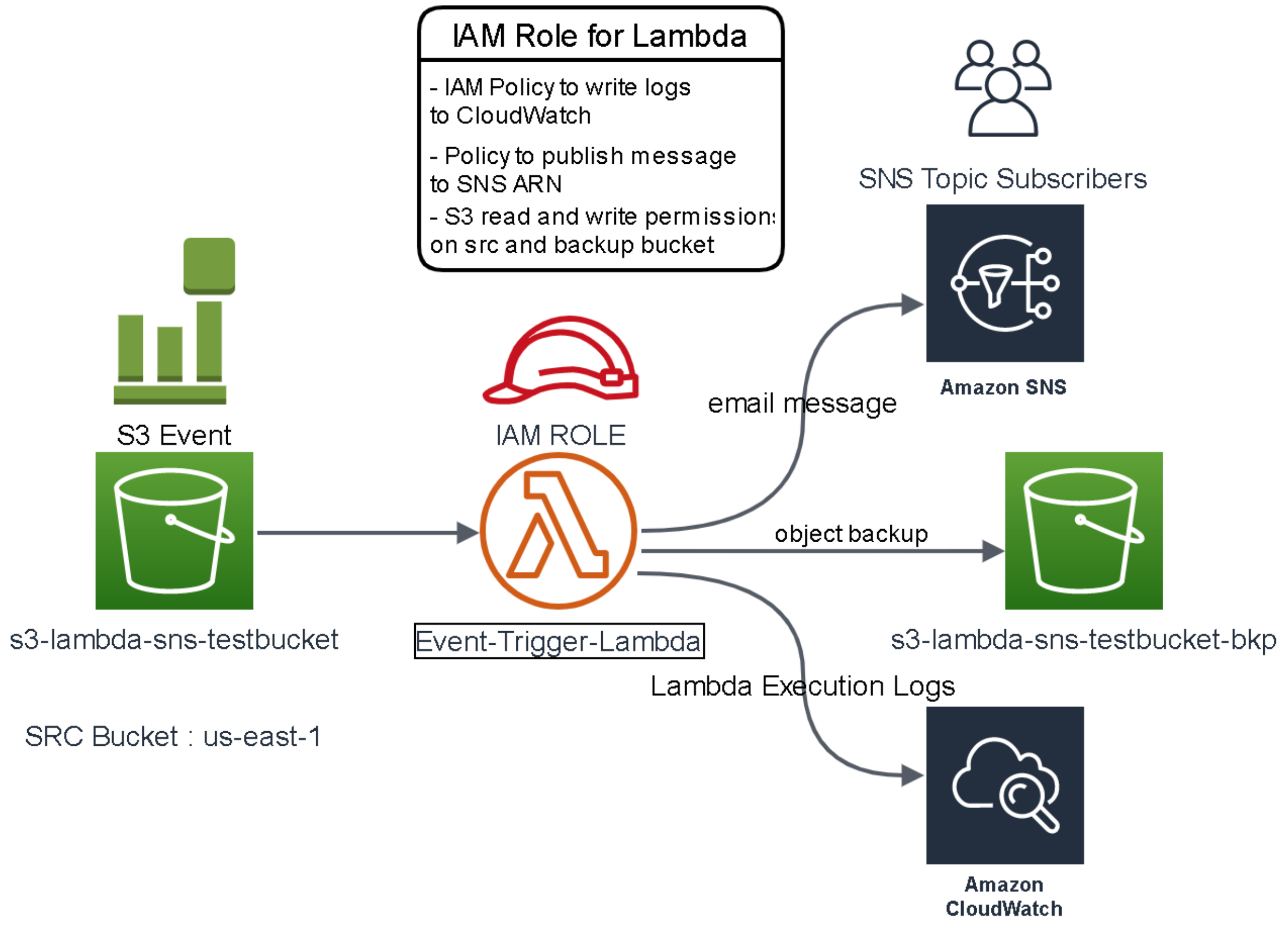
AWS Lambda Function `event` in Python

- `event` – AWS Lambda uses this parameter to pass in event data to the handler. This parameter is usually of the Python `dict` type.
- When a Lambda Function is invoked, we can determine the content and structure of the event.
- When an AWS service invokes your function, the event structure varies by service.
- If a Lambda Function is invoked by S3 Service, the event data will be as below:

```
key_name = event['Records'][0]['s3']['object']['key']
{
    'Records': [{
        'eventVersion': '2.1',
        'eventSource': 'aws:s3',
        'awsRegion': 'us-east-1',
        'eventTime': '2019-12-14T01:54:29.721Z',
        'eventName': 'ObjectCreated:Put',
        'userIdentity': {
            'principalId': 'A3HLTU27MENXZ1'
        },
        'requestParameters': {
            'sourceIPAddress': '111.225.11.82'
        },
        'responseElements': {
            'x-amz-request-id': 'BB4FFADF2A079713',
            'x-amz-id-2':
'QUopUd3JXVpE6W9UAD6cCa5ql9CYbQt3Y7TyuDJhFPPeKGV7l7hO7602QSq9vxVntsQ9XxMfj0Y='
        },
        's3': {
            's3SchemaVersion': '1.0',
            'configurationId': 'ad5d40d4-8e21-4f13-afcd-4cdc65d1d157',
            'bucket': {
                'name': 's3bkt',
                'ownerIdentity': {
                    'principalId': 'A3HLTU27MENXZ1'
                },
                'arn': 'arn:aws:s3:::s3bkt'
            },
            'object': {
                'key': 'AWS_KeyPairs/aws-NV.ppk',
                'size': 1464,
                'eTag': '809d9725c217af3ab4f70f52a0d5e08f',
                'versionId': 'DoSaRg91K2aGVesfp_ttE0QCa8ggnCyZ',
                'sequencer': '005DF440D599CFB897'
            }
        }
    }]
}
```

> Problem Statement: Create an S3 event Trigger to invoke Lambda Function to trigger SNS email and Copy the Uploaded object into a Backup Bucket. Use Case : To get an email when a file is uploaded in a particular bucket for every file upload with below details : Source Bucket Name , AWS Region Name, File name and file size, Upload IP Address, Event Time

S3 Event Custom Email Notification



Follow Below Steps :

1. Setup a S3 event trigger for PUT Object only on a bucket to trigger the Lambda :
2. Create an SNS topic and add subscribers
3. Write below Lambda Code :
4. Navigate to `Configuration` > choose `Environment variables` > `Edit`. Here, Create two Environment variables that will be used in Python Code

- Under Key enter `BACKUP_BUCKET_NAME` , under Value enter `<BUCKET_NAME_VALUE>`
- Under Key enter `SNS_TOPIC_ARN` , under Value enter `<SNS_TOPIC_ARN_VALUE>`

> Make sure to enter Value fields for above Environment Variables as per your AWS Account. Change value of SNS Topic ARN and Backup Bucket Name as per your account.

```
import os,json,boto3

s3 = boto3.client("s3")
sns = boto3.client("sns")

def lambda_handler(event,context):
    print(event)
    source_bucket = event['Records'][0]['s3']['bucket']['name']
    aws_region = event['Records'][0]['awsRegion']
    key_val = event['Records'][0]['s3']['object']['key']
    size_val = event['Records'][0]['s3']['object']['size']/1024
    ipAddress = event['Records'][0]['requestParameters']['sourceIPAddress']
    event_time = event['Records'][0]['eventTime']

    message = "Hi, \n The Event time is : " + event_time + "Hi, \nYou are
receiving this email because you are subscribed to this S3event. \nThe Source
Bucket is : " + source_bucket + "\nThe AWS Region is :" + aws_region + "\nThe
Uploaded Filename is : " + key_val + " having Size : " + str(size_val) + " KB" +
"\nThe Object is upload from IP Address: " + ipAddress
    #Below are the variables for copy_object function parameters
    #Provide below the target bucket name where your object needs to be copied
    backupBucket = os.environ['BACKUP_BUCKET_NAME']
    snsArn = os.environ['SNS_TOPIC_ARN']
    emailSubject = "S3EventTrigger-Notification"
    copy_source = {'Bucket' : source_bucket, 'Key' : key_val}
    sns_response =
sns.publish(TopicArn=snsArn,Message=message,Subject=emailSubject)
    print(sns_response)
    try:
        print("Copying the object from source to destination")
        s3.copy_object(Bucket=backupBucket,Key=key_val, CopySource=copy_source)
    except Exception as e:
        print(e)
        print("Error getting object")
        raise e
```

6. Add the below inline policy and add to lambda role, below policy should have only Publish access to specific topic and S3 Permissions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SnsPolicy",
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource": "arn:aws:sns:us-east-1:ACCOUNT_ID:SNSDemoTopic"
        },
        {
            "Sid": "S3Access",
            "Action": "s3:*",
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::*"
        }
    ]
}
```

7. Upload any file to the bucket

8. Verify Lambda Trigger from Cloudwatch Log Group with Lambda Name `/aws/lambda/<LambdaFuntionName>`
9. Verify if the uploaded file is copied to the Backup Bucket.
10. Verify if email notification is trigger from above lambda code.

- For details, see Using AWS Lambda with Other Services visit AWS Lambda with Other Services

1b) AWS Lambda `context` Object in Python

- `context` – AWS Lambda uses this parameter to provide runtime information to your handler.
- When Lambda runs your function, it passes a context object to the handler. This object provides methods and properties that provide information about the invocation, function, and execution environment.

Context Properties

- `function_name` – The name of the Lambda function.
- `function_version` – The version of the function. `invoked_function_arn` – The Amazon Resource Name (ARN) that's used to invoke the function. Indicates if the invoker specified a version number or alias. `memory_limit_in_mb` – The amount of memory that's allocated for - the function.
- `aws_request_id` – The identifier of the invocation request.
- `log_group_name` – The log group for the function.
- `log_stream_name` – The log stream for the function instance.

```
import json,time

def lambda_handler(event, context):
    print("Lambda Function Version:", context.function_version)
    print("Resoource ARN that invoked the current Function
:",context.invoked_function_arn)
    print("Memory allocated to this Lambda Function is :",
context.memory_limit_in_mb)
    print("Invokation request identifier :", context.aws_request_id)
    print("Log Group Name:", context.log_group_name)
    print("Log stream name:", context.log_stream_name)
```

Reference:

- s3.copy_object()
- sns.publish()