

Name:-Manav Punjabi

Roll no:-45

Class:-D15A

Case Study:-Serverless Application With Monitoring

- **Concepts Used: AWS Lambda, S3, and Nagios,**
- **Problem Statement: "Create an AWS Lambda function that logs an event when an image is uploaded to a specific S3 bucket. Set up Nagios to monitor the Lambda function's execution status and S3 bucket"**

Tasks:-

- **Create a Lambda function in Python that logs 'An Image has been added when an object is uploaded to an S3 bucket.'**
- **Configure Nagios to monitor the Lambda function's logs.**
- **Upload a test image to the S3 bucket and verify that the function logs the event and Nagios captures the status.**

Introduction:-

In the realm of modern application development, serverless computing has emerged as a game-changer, allowing developers to build applications without worrying about infrastructure management. AWS Lambda, a key player in serverless technology, enables the execution of code in response to various events, such as uploads to an S3 bucket. This case study will outline the creation of a serverless application utilizing AWS S3 and Lambda, along with setting up Nagios for monitoring.

Key Points:-

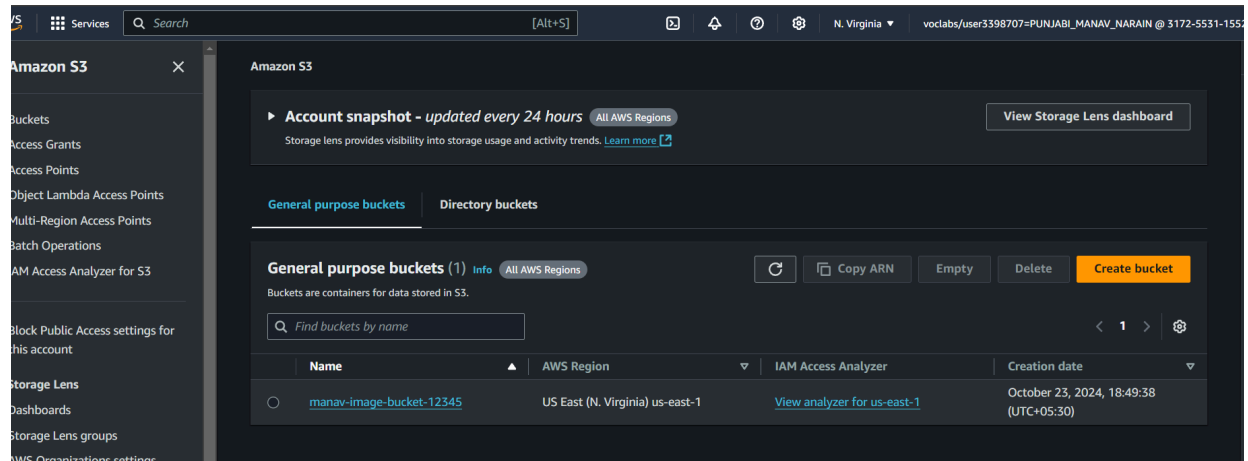
- **Creation of an S3 bucket in AWS to serve as storage for uploaded images.**
- **Configuration of an AWS Lambda function that triggers upon image uploads to the S3 bucket**
- **Development of a Python script within the Lambda function to log relevant events.**
- **Uploading test images to S3 and validating Lambda function execution through AWS CloudWatch**
- **Implementation of Nagios to monitor the health of the Lambda function and log changes**

in the S3 bucket

Application of Serverless and Monitoring

CREATION OF S3 BUCKET:

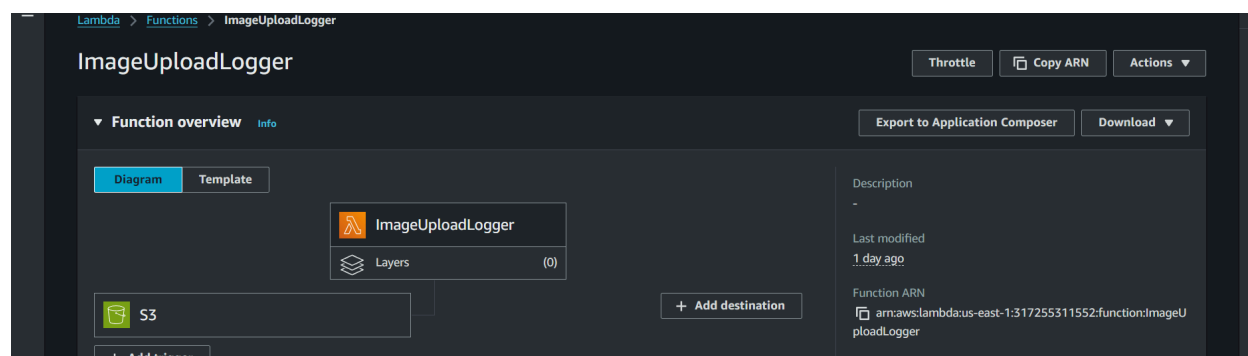
To start, an S3 bucket is created using the AWS Management Console. This bucket will store images that will trigger the Lambda function upon upload.



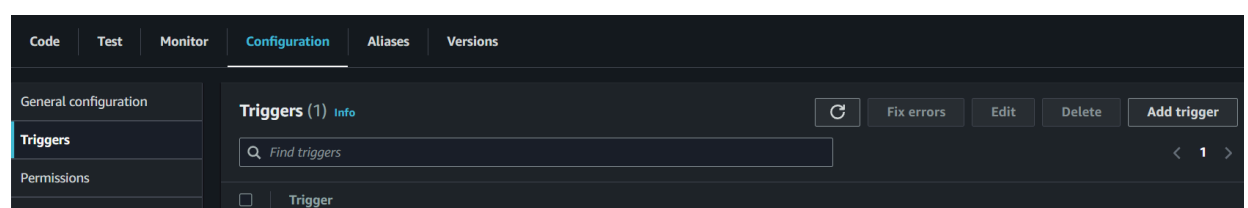
CREATION OF LAMBDA FUNCTION:-

A Lambda function is established to respond to S3 upload events. A Python script is integrated within the function to log incoming events. The creation steps include:

Accessing the AWS Lambda console. Selecting "Create function" and choosing Python as the runtime Assigning necessary roles and permissions to the function.



ADD AN S3 TRIGGER TO GET UPDATED WHEN A NEW IMAGE IS UPLOADED IN THE S3 BUCKET



UPDATE THE PYTHON SCRIPT IN THE CODE SOURCE TO WRITE A CODE THAT PRINTS “AN IMAGE IS UPLOADED” WHEN THE LAMBDA FUNCTION IS TRIGGERED BY S3.

Write the below code in the script.

```
import json

import logging

# Set up

logging

logger = logging.getLogger()

logger.setLevel(logging.INFO)


def lambda_handler(event, context):

    # Log incoming event data

    logger.info('Received event: %s', json.dumps(event, indent=2))


    # Get bucket name and object key from the event

    bucket_name = event['Records'][0]['s3']['bucket']['name']

    object_key = event['Records'][0]['s3']['object']['key']


    # Check if the uploaded object is an image

    if object_key.lower().endswith(('.png', '.jpg', '.jpeg', '.gif')):

        logger.info('An Image has been added to the bucket: %s, Object: %s', bucket_name, object_key)

    else:

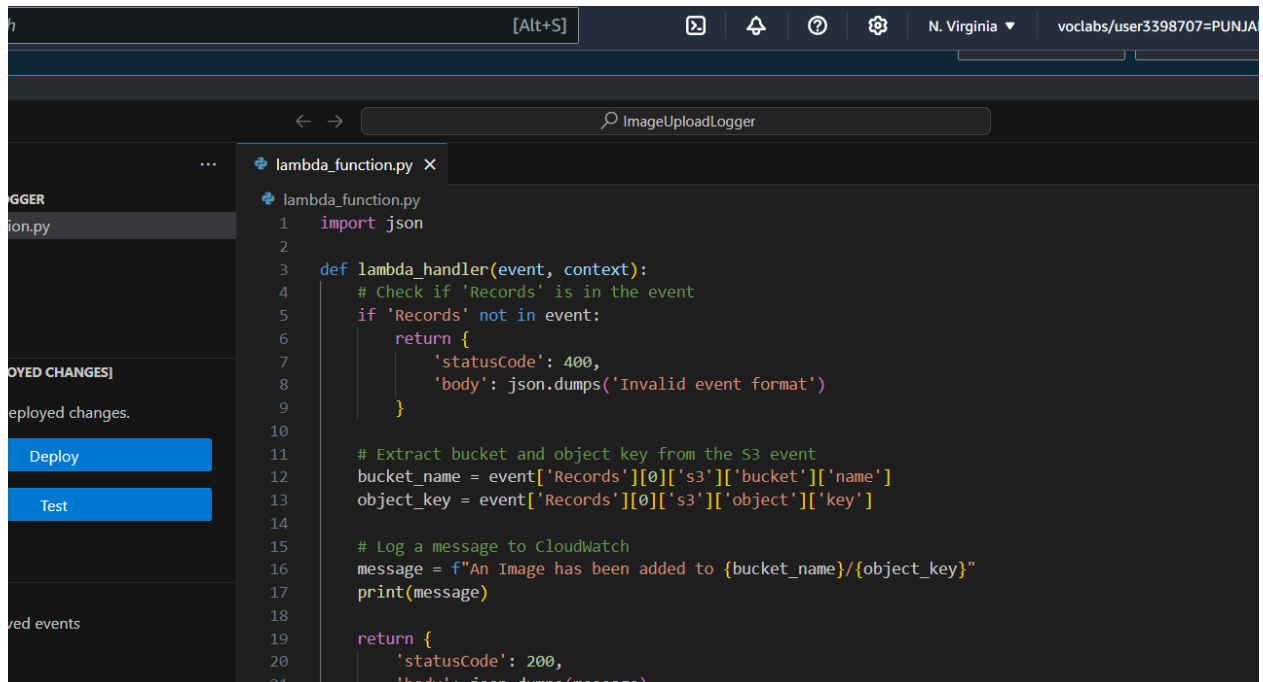
        logger.info('Non-image file uploaded to the bucket: %s, Object: %s', bucket_name, object_key)


    return {

        'statusCode': 200,

        'body': json.dumps('Lambda function executed successfully!')

    }
```



The screenshot shows a code editor with a file named `lambda_function.py`. The code is a Python lambda function that handles S3 events. It checks if the event contains 'Records'. If not, it returns a 400 status code with the message 'Invalid event format'. If it does, it extracts the bucket name and object key from the first record, constructs a log message, and prints it to CloudWatch. Finally, it returns a 200 status code with the message.

```
1 import json
2
3 def lambda_handler(event, context):
4     # Check if 'Records' is in the event
5     if 'Records' not in event:
6         return {
7             'statusCode': 400,
8             'body': json.dumps('Invalid event format')
9         }
10
11     # Extract bucket and object key from the S3 event
12     bucket_name = event['Records'][0]['s3']['bucket']['name']
13     object_key = event['Records'][0]['s3']['object']['key']
14
15     # Log a message to CloudWatch
16     message = f"An Image has been added to {bucket_name}/{object_key}"
17     print(message)
18
19     return {
20         'statusCode': 200,
21         'body': json.dumps(message)
```

1)UPLOAD AN IMAGE IN THE S3 BUCKET:

by prefix

▲	Type	▼	Last modified	▼	Size	▼	Storage
ract-autumn-multi-colored-leaf-tern-generated-by-14-9871.avif	avif		October 23, 2024, 19:16:52 (UTC+05:30)		29.6 KB		Standard
naBalram.jpg	jpg		October 24, 2024, 21:50:22 (UTC+05:30)		1.5 MB		Standard

NOW GO BACK TO LAMBDA FUNCTION AND

DEPLOY THE CODE BY CLICKING ON THE DEPLOY OPTION AND TEST IT.

-USE THE SUITABLE EVENT FOR YOUR CODE.

THE EVENT SHOULD INCLUDE BELOW

SCRIPT.

```
{
  "Records": [
    {
      "eventVersion": "2.1",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "2024-10-19T17:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/456EXAMPLE="
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "example-bucket",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::example-bucket"
        }
      }
    }
  ]
}
```

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, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes.

Test event action

☐ Create new event

☒ Edit saved event

Event name

komalevent1

Format JSON

```
1 * [
2 *   "Records": [
3 *     {
4 *       "eventVersion": "2.1",
5 *       "eventSource": "aws:s3",
6 *       "awsRegion": "us-east-1",
7 *       "eventTime": "2024-10-19T17:00:00.000Z",
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": "EXAMPLE123/456EXAMPLE="
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": "test-image.jpg",
```

1:1 JSON Spaces: 2

```
lambda_function x Environment Var x Execution results x +
Execution results
Test Event Name
komalevent1
Response
{
  "statusCode": 200,
  "body": "\\Lambda function executed successfully!\\\"
}
Function Logs
START RequestId: 9dccd201-07da-4a2c-8b7d-119fe2a3ed99 Version: $LATEST
[INFO] 2024-10-20T11:30:12.312Z 9dccd201-07da-4a2c-8b7d-119fe2a3ed99 Received event: {
  "Records": [
    {
      "eventVersion": "2.1",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "2024-10-19T17:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/456EXAMPLE="
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "komalimagebucket",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::komalimagebucket"
        },
        "object": {
          "key": "ottocycle.jpg",
          "size": 1024,
          "eTag": "0123456789abcdef0123456789abcdef",

```

```

}
[INFO] 2024-10-20T11:30:12.313Z 9dccd201-07da-4a2c-8b7d-119fe2a3ed99 An Image has been added to the bucket: komalimagebucket, Object: ottocycle.jpg
END RequestId: 9dccd201-07da-4a2c-8b7d-119fe2a3ed99
REPORT RequestId: 9dccd201-07da-4a2c-8b7d-119fe2a3ed99 Duration: 4.97 ms Billed Duration: 5 ms Memory Size: 128 MB Max Memory Used: 40 MB

Request ID |
9dccd201-07da-4a2c-8b7d-119fe2a3ed99
```

-the output of this code is of before the image fourstroke.jpg was uploaded.

2)CHECK THE CLOUDWATCH LOGS

Once the image is uploaded. The lambda function is triggered and it can be checked in the logs.

Go to cloudwatch>log group name>log stream

CloudWatch > Log groups > /aws/lambda/komal-lambdafun > 2024/10/20/[\$LATEST]a1eb2c02645f446ab79825d2058cc189

Log events 🔄 Actions ▼ Start tailing Create metric filter

You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

Clear 1m 30m 1h 12h Custom Local timezone ▼ Display ▼

Timestamp	Message
No older events at this moment. Retry	
2024-10-20T16:58:05.921+05:30	INIT_START Runtime Version: python:3.8.v59 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:31ddc9689d29d2f6f48a11d588466fad8a5c21d1a933704cf257bcefb8ebb0
2024-10-20T16:58:06.097+05:30	START RequestId: 7d1bf8a5-a3df-4dc2-93eb-1487c5968a33 Version: SLATEST
2024-10-20T16:58:06.098+05:30	[INFO] 2024-10-20T11:28:06.098Z 7d1bf8a5-a3df-4dc2-93eb-1487c5968a33 Received event: { "Records": [{ "eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "us-east-1", "eventTime": "2024-10-19T17:00:00.000Z", "eventName": "ObjectCreated", "bucket": "example-bucket", "object": "test-image.jpg" }
2024-10-20T16:58:06.098+05:30	[INFO] 2024-10-20T11:28:06.098Z 7d1bf8a5-a3df-4dc2-93eb-1487c5968a33 An Image has been added to the bucket: example-bucket, Object: test-image.jpg
2024-10-20T16:58:06.118+05:30	END RequestId: 7d1bf8a5-a3df-4dc2-93eb-1487c5968a33
2024-10-20T16:58:06.118+05:30	REPORT RequestId: 7d1bf8a5-a3df-4dc2-93eb-1487c5968a33 Duration: 2.54 ms Billed Duration: 3 ms Memory Size: 128 MB Max Memory Used: 48 MB Init Duration: 174.16 ms
2024-10-20T17:00:12.312+05:30	START RequestId: 9dccd281-07da-4a2c-8b7d-119fe2a3ed99 Version: SLATEST
2024-10-20T17:00:12.312+05:30	[INFO] 2024-10-20T11:30:12.312Z 9dccd281-07da-4a2c-8b7d-119fe2a3ed99 Received event: { "Records": [{ "eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "us-east-1", "eventTime": "2024-10-19T17:00:00.000Z", "eventName": "ObjectCreated", "bucket": "komalimagebucket", "object": "ottocycle.jpg" }
2024-10-20T17:00:12.313+05:30	[INFO] 2024-10-20T11:30:12.313Z 9dccd281-07da-4a2c-8b7d-119fe2a3ed99 An Image has been added to the bucket: komalimagebucket, Object: ottocycle.jpg
2024-10-20T17:00:12.313Z	[INFO] 2024-10-20T11:30:12.313Z 9dccd281-07da-4a2c-8b7d-119fe2a3ed99 An Image has been added to the bucket: komalimagebucket, Object: ottocycle.jpg
2024-10-20T17:00:12.317+05:30	END RequestId: 9dccd281-07da-4a2c-8b7d-119fe2a3ed99
2024-10-20T17:00:12.317+05:30	REPORT RequestId: 9dccd281-07da-4a2c-8b7d-119fe2a3ed99 Duration: 4.97 ms Billed Duration: 5 ms Memory Size: 128 MB Max Memory Used: 48 MB
No newer events at this moment. Auto retry paused . Resume	

▼ 2024-10-20T17:00:12.313+05:30 [INFO] 2024-10-20T11:30:12.313Z 9dccd281-07da-4a2c-8b7d-119fe2a3ed99 An Image has been added to the bucket: komalimagebucket, Object: ottocycle.jpg

[INFO] 2024-10-20T11:30:12.313Z 9dccd281-07da-4a2c-8b7d-119fe2a3ed99 An Image has been added to the bucket: komalimagebucket, Object: ottocycle.jpg

AFTER UPLOADING ONE MORE IMAGE:

▼ 2024-10-20T18:07:11.478+05:30 [INFO] 2024-10-20T18:37:11.478Z e1124b12-b1ee-437a-bf9b-3cf4a16e471d An Image has been added to the bucket: komalimagebucket, Object: fourstroke.jpg

[INFO] 2024-10-20T18:37:11.478Z e1124b12-b1ee-437a-bf9b-3cf4a16e471d An Image has been added to the bucket: komalimagebucket, Object: fourstroke.jpg

3)FOR MONITORING THE LAMBDA FUNCTION BY USING NAGIOS ON EC2:

PREREQUISITES: INSTALL AND CONFIGURE NAGIOS ON YOUR EC2 INSTANCE.

1)GO TO THE LIBEXEC DIRECTORY `cd /usr/local/nagios/libexec`

```
[ec2-user@ip-172-31-43-5 ~]$ cd /usr/local/nagios/libexec  
[ec2-user@ip-172-31-43-5 libexec]$ sudo chmod +x check_lambda.sh
```

2) RUN THE COMMAND `sudo nano check_lambda.sh`

3) WRITE THE BELOW CODE IN THE

SCRIPT `#!/bin/bash`

AWS Lambda function name

LAMBDA_FUNCTION_NAME="komal-lambdafun"

S3 bucket name (replace this with your actual bucket name)

S3_BUCKET_NAME="your-s3-bucket-name"

Log group for the Lambda function (CloudWatch logs)

LOG_GROUP_NAME="/aws/lambda/\$LAMBDA_FUNCTION_NAME"

Get the current time to filter logs from this point onward

CURRENT_TIME=\$(date -u +"%Y-%m-%dT%H:%M:%S.000Z")

Monitor Lambda for S3 trigger (image upload)

echo "Monitoring Lambda function '\$LAMBDA_FUNCTION_NAME' for image uploads to S3 bucket '\$S3_BUCKET_NAME'."

Monitor S3 bucket for any new object (image upload)

echo "Checking if there is a new image uploaded to the S3 bucket..."

```
LATEST_OBJECT=$(aws s3api list-objects --bucket $S3_BUCKET_NAME --query 'Contents[?contains(Key,`.jpg`)|contains(Key,`.png`)] | sort_by(@, &LastModified)[-1].Key' --output text)
```

```
if [ "$LATEST_OBJECT" == "None" ]; then
```

```
    echo "No image found in the bucket '$S3_BUCKET_NAME'."
```

```
    exit 1
```

```
else
```

```
    echo "Latest image uploaded: $LATEST_OBJECT"
```

```
fi
```

```
# Check the Lambda function's logs for recent activity (triggered by the S3 event)
```

```
echo "Checking Lambda function logs for the latest execution..."
```

```
# Fetch log streams (sorted by latest event time)
```

```
LATEST_LOG_STREAM=$(aws logs describe-log-streams --log-group-name $LOG_GROUP_NAME --order-by LastEventTime --descending --limit 1 --query 'logStreams[0].logStreamName' --output text)
```

```
if [ "$LATEST_LOG_STREAM" == "None" ]; then
```

```
    echo "No logs found for Lambda function '$LAMBDA_FUNCTION_NAME'."
```

```
    exit 2
```

```
else
```

```
    echo "Fetching logs from stream: $LATEST_LOG_STREAM"
```

```
fi
```

```
# Get logs for the latest Lambda execution
```

```
LOG_EVENTS=$(aws logs get-log-events --log-group-name $LOG_GROUP_NAME --log-stream-name $LATEST_LOG_STREAM --start-time $(date --date="$CURRENT_TIME" +%s%3N))
```

```
# Check if log events were found
```

```
if [ -z "$LOG_EVENTS" ]; then
```

```
    echo "No log events found for Lambda function execution after image upload."
    exit 3
else
    echo "Log events from Lambda function triggered by S3 upload:"
    echo "$LOG_EVENTS"
    exit 0
fi
```

-IT MONITORS YOUR LAMBDA FUNCTION AND THE S3 BUCKET'S LATEST CHANGES.

3) RUN *sudo chmod +x check_lambda.sh*

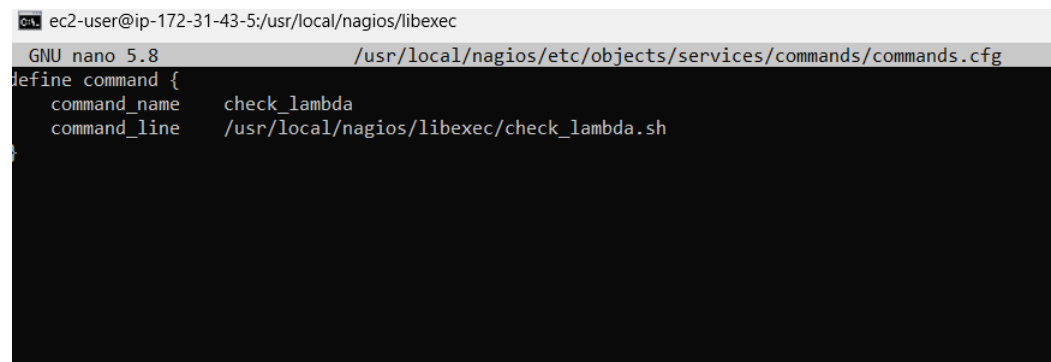
4) RUN THE FOLLOWING COMMAND TO ACCESS THE COMMANDS

SCRIPT *sudo nano*

/usr/local/nagios/etc/objects/services/commands/commands.cfg 5) WRITE

THE BELOW CODE IN THE SCRIPT

```
define command {
    command_name check_lambda
    command_line /usr/local/nagios/libexec/check_lambda.sh
}
```



```
ec2-user@ip-172-31-43-5:/usr/local/nagios/libexec
GNU nano 5.8 /usr/local/nagios/etc/objects/services/commands/commands.cfg
define command {
  command_name  check_lambda
  command_line  /usr/local/nagios/libexec/check_lambda.sh
}
```

6) RUN THE FOLLOWING COMMAND TO ACCESS THE SERVICES SCRIPT

sudo nano /usr/local/nagios/etc/objects/services/services.cfg

```
ec2-user@ip-172-31-43-5:/usr/local/nagios/libexec
GNU nano 5.8 /usr/local/nagios/etc/objects/services/services.cfg
define service {
    use                generic-service
    host_name          localhost
    service_description Check Lambda Function
    check_command       check_lambda
}
```

7)WRITE THE BELOW CODE IN THE SCRIPT

```
define service {

    use                generic-service

    host_name          localhost

    service_description Check Lambda Function

    check_command       check_lambda

}
```

8)SET THE AWS CREDENTIALS TO ACCESS AND MONITOR THE CREATED LAMBDA FUNCTION.

```
export AWS_SECRET_ACCESS_KEY="your_secret_key"

export AWS_SESSION_TOKEN="your_session_token"

export AWS_SESSION_TOKEN="your_session_token"
```

```
ec2-user@ip-172-31-43-5 libexec]$ export AWS_ACCESS_KEY_ID="ASIASWU6DPVCRK2RKY7"
ec2-user@ip-172-31-43-5 libexec]$ export AWS_SESSION_TOKEN="IqoJb3jpZ2lUx2VjEAcKXvzLxdlc3QtMlJlMEYClQCd+3e0Nv+R4FYRd0XGn8EJZybhYYTYLCKcgsy+368WbQ1hA3o76oFf9/13D7vdmLAmGjs0Da09wtDvrUjhOm
2bgnVKUCCHAQAROMMTg2PdG40TE0HjQ1IgwL6krnu8dc2j4+R0cqkg1zB4mKEu3h7CxcacQ/k4ttwSRkXxcB23W8utmDOMKntu1oM9oSPglenSAEGJUm56TbNo7jpMk3cJZ9h0t2aG58DUGL066Swg+ce9EQRPJsrnbY1GnFR9yDu579BPgfipIBu
3V91mY/N75f1VEBT3yLtevmQAXOX59RurOdMFMF1rFu1MNB3kgcAF83eJr58J1eMfnnMTZCKPMGtoIopBM1r3QVjpLZZ+ABWJES+9CTAwitbfVopQ1LFaw2uxqRGPNNTV4eBrW0ZmbA2Zfn0LBunepFr+J0g41+EqE7YEVn1MwZcnpJtqIDVoCnQNJ+E
9GSS569zYlJlmsKpH5MHoJ0RmcfS649qW+SnaaJhp88MPJ0BrgOpwBnniG1fZKVS1Ud4/EtB15dgl19eF1TTCYcdapNCD0F2MUF3Vzvai1bCd10rSbwTbEXSxx41P8mJj2Gm/24qwmJR2ftaXKF9uWShXQKZkreboov7c611/v8SopJQAEOLC3aId
hWJ/A6S9+20Kv11G1UJUV+seteqcmhQ2DEKXZ1VX89+HhQy3160FvdiHE+VCGZafvZtpkS"TZUKX0Kef1pRw0jXAND1CjQqde0901hgP1wBSS1nttqATcZLXC22+K2TQXG2Pw20L1f6Vzq32B1REZcglPEX200Bw+rcdaB8r2K2B0gK5
ec2-user@ip-172-31-43-5 libexec]$ oEB50W712TzAg2FE9XKc6sFVQ4d0KEZsr7Ep64Gy1a4JUG6mpalVQQAIXn10utbcBV0H6VZOR9KngHbX1f5sclXDZK2B0x58Nycsa17jtaZNVwF7sTUMsNC8zRX10F2Bw450hk32Fkb2FasMHZK2F
ec2-user@ip-172-31-43-5 libexec]$ ^Cuh1.965qBwGloVqK0MFYv3seEXKcr14f0bQrCpM9Lk32BEH54XksEIQKAK2B2Lcmvh35F938VUY3VMzrQC4%2F3uenB0x1VPInqEkuWuCTGKIDG50L7npmbA2k8KI85TfFW0hf81dxFP270M11b1k
ec2-user@ip-172-31-43-5 libexec]$ ^C2EhbtG3G1SMYQ5wv%2F0XTg13U1phmpDsn3pyjmcwsZTTUmG0X2B2BR1WpInLP22FDZ2ccodgnle3UAv1f%2FD081WUA4ZACsxEKXtbo9Ic4yVZHRfcl%2F0eD790Kk313zZATYXGvYjBMVFs4uFj
```

```
ec2-user@ip-172-31-43-5 libexec]$ export AWS_SECRET_ACCESS_KEY="uzjnupR3BdeJnPKtvBZxxxzc4GHicdmWJ9Tf+wgKL"
```

9) RESTART NAGIOS TO GET THE CHANGES UPDATED *sudo systemctl restart nagios*

10)RUN BELOW COMMAND TO CHECK THE MONITORING OF YOUR LAMBDA FUNCTION

./check_lambda.sh

```
Checking if there is a new image uploaded to the S3 bucket...
Latest image uploaded: ottocycle.jpg
Checking Lambda function logs for the latest execution...
Fetching logs from stream: 2024/10/19/[$LATEST]6fffb263855042b295ab4aabd7f7f3fb
Log events from Lambda function triggered by S3 upload:
{
    "events": [],
    "nextForwardToken": "f/38567525873567295377391290756440763726736881459105103872/s",
    "nextBackwardToken": "b/38567365242593073582827578416871117500402558849515520000/s"
}
```

ALSO TRY UPLOADING A NEW IMAGE IN THE S3 BUCKET TO CHECK IF THE MONITORING IS WORKING

-IT ALSO NOTIFIES YOU ABOUT SYSTEM BEING POWERED OFF BY THE HOST.

11) RUN `aws lambda get-function --function-name your_function_name` TO GET THE CONFIGURATION OF THE LAMBDA FUNCTION.

```
Checking if there is a new image uploaded to the S3 bucket...
Latest image uploaded: fourstroke.jpg
Checking Lambda function logs for the latest execution...
Fetching logs from stream: 2024/10/20/[$LATEST]2385120a48774ac5aac4a8c8a8f44d49
Log events from Lambda function triggered by S3 upload:
{
  "events": [],
  "nextForwardToken": "f/38567529563337092200275542639233030527771534431094833152/s",
  "nextBackwardToken": "b/38567368944516776538911019911800350760030568842264576000/s"
}
[ec2-user@ip-172-31-43-5 libexec]$
Broadcast message from root@localhost (Sun 2024-10-20 10:37:49 UTC):

The system will power off now!

Connection to ec2-98-83-23-82.compute-1.amazonaws.com closed by remote host.
Connection to ec2-98-83-23-82.compute-1.amazonaws.com closed.
```

Monitor

```
functionArn: "arn:aws:lambda:us-east-1:186088914245:function:komal-lambdafun",
runtime: "python3.8",
role: "arn:aws:iam:186088914245:role/LabRole",
handler: "lambda_function.lambda_handler",
codeSize: 557,
description: "",
timeout: 3,
memorySize: 128,
lastModified: "2024-10-19T17:57:43.000+0000",
codeSha256: "SUR6xl7B69bHk3QqX3UFhBHP1jj0cm2cuJjsk9wr94=",
version: "$LATEST",
tracingConfig: {
  mode: "PassThrough"
},
revisionId: "0376bc2b-e931-4909-b7cc-e09fcd5ea146",
state: "Active",
lastUpdateStatus: "Successful",
packageType: "Zip",
architectures: [
  "x86_64"
],
ephemeralStorage: {
  size: 512
},
snapStart: {
  applyOn: "None",
  optimizationStatus: "Off"
},
runtimeVersionConfig: {
  runtimeVersionArn: "arn:aws:lambda:us-east-1::runtime:31ddccb9609d29d2fa6f49a11d508466fad8a5c21d1a9337d4cf257bcef8ebbo"
},
loggingConfig: {
  logFormat: "Text",
  logGroup: "/aws/lambda/komal-lambdafun"
}
},
code: {
  repositoryType: "S3",
  location: "https://prod-04-2014-tasks.s3.us-east-1.amazonaws.com/snapshots/186088914245/komal-lambdafun-dbc2793b-3ebf-49f3-afe9-5f2cbaefcfc0?versionId=Q2bTAwhlgWg2vaW86p1Bcmwh1VoA86.t&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEAoACXVzLWVhc3QtMSJhMEUCIPV8uNF8s2bYXCQ%2BHZN84Nol.d2Trf7VER7dED3sA5uIAiEA91gyQ1NPv9TVI8Sk4AJt9s5dyHBCX2FVXTvtFXs5Z3PucqsgUICHAAGgw3NDk2NzgzSMD14Mzk
```

Conclusion:-

This case study demonstrates how to leverage AWS Lambda for processing S3 upload events, offering a suitable solution for tasks requiring real-time image processing. By integrating Nagios, the performance and health of the Lambda function can be effectively monitored, ensuring a reliable serverless application.