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:-

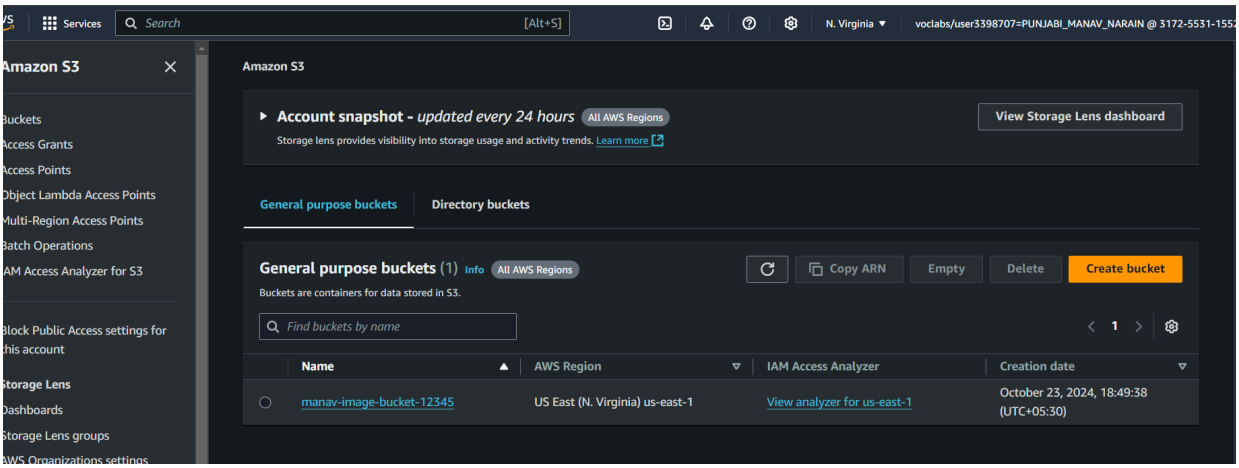
- **Creation of an S3 bucket in AWS to serve as storage for uploaded images.**
- **Configuration of an AWS Lambda function that triggers upon image uploads to the S3 bucket**
- **Development of a Python script within the Lambda function to log relevant events.**
- **Uploading test images to S3 and validating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

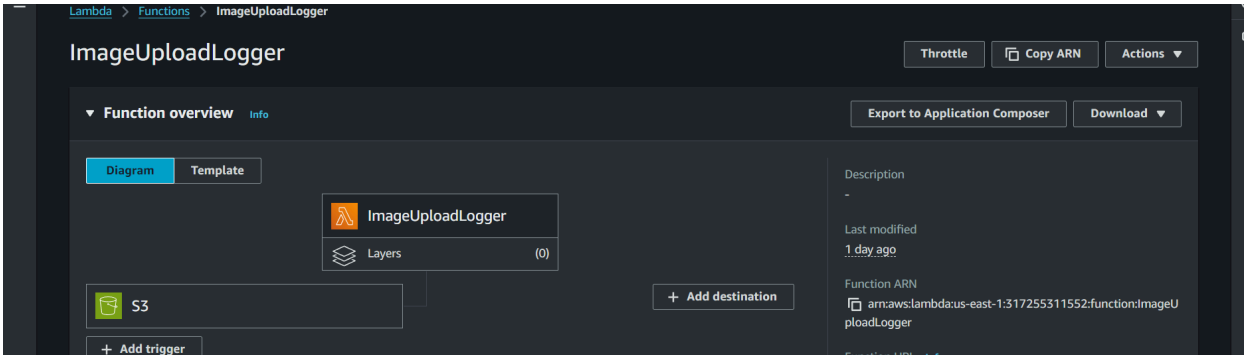
To start, an S3 bucket is created using the AWS Management Console. This bucket will store images that will trigger the Lambda function upon upload.



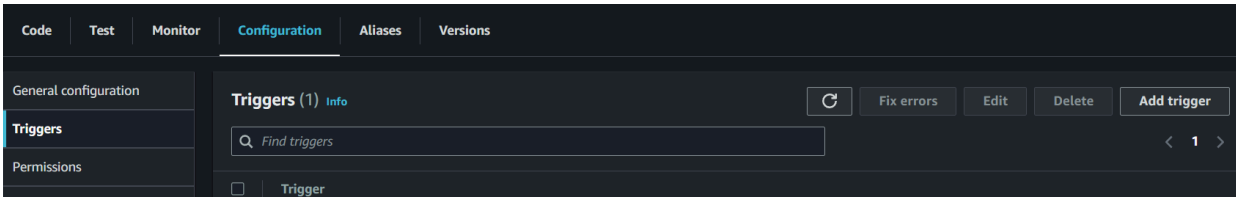
CREATION OF LAMBDA FUNCTION:-

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.



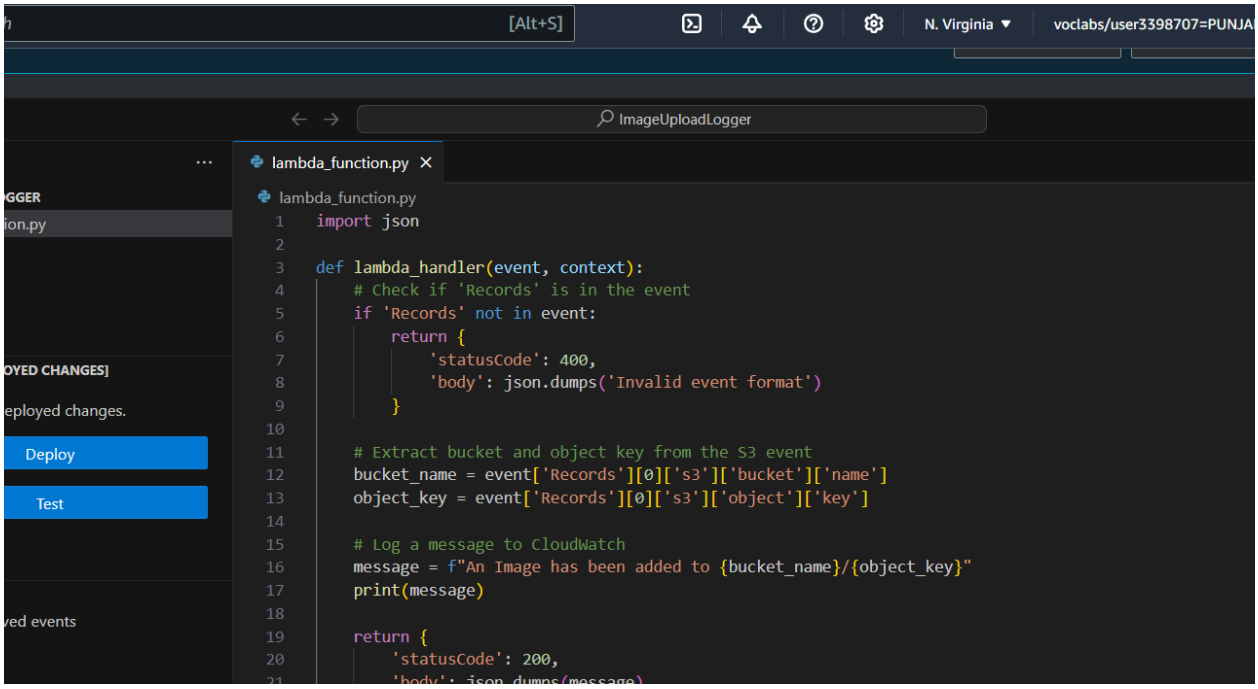
ADD AN S3 TRIGGER TO GET UPDATED WHEN A NEW IMAGE IS UPLOADED IN THE S3 BUCKET



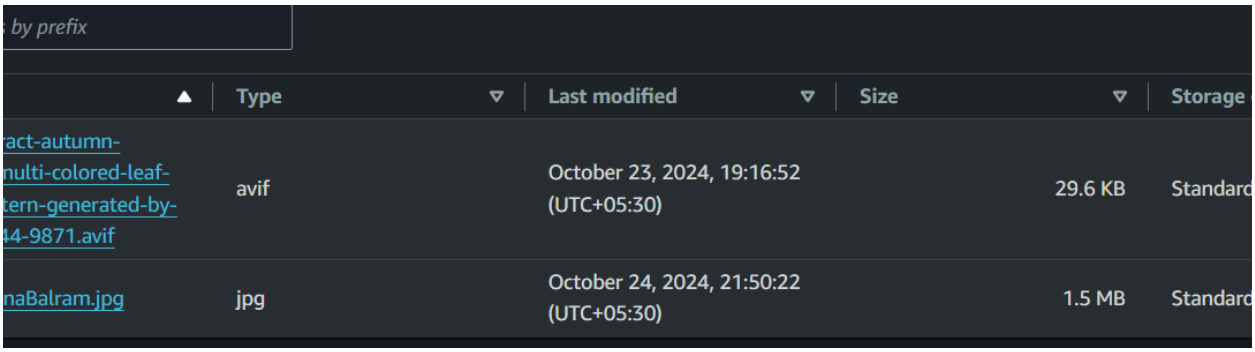
UPDATE THE PYTHON SCRIPT IN THE CODE SOURCE TO WRITE A CODE THAT PRINTS “AN IMAGE IS UPLOADED” WHEN THE LAMBDA FUNCTION IS TRIGGERED BY S3.

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!')

    }
```



1)UPLOAD AN IMAGE IN THE S3 BUCKET:



NOW GO BACK TO LAMBDA FUNCTION AND

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

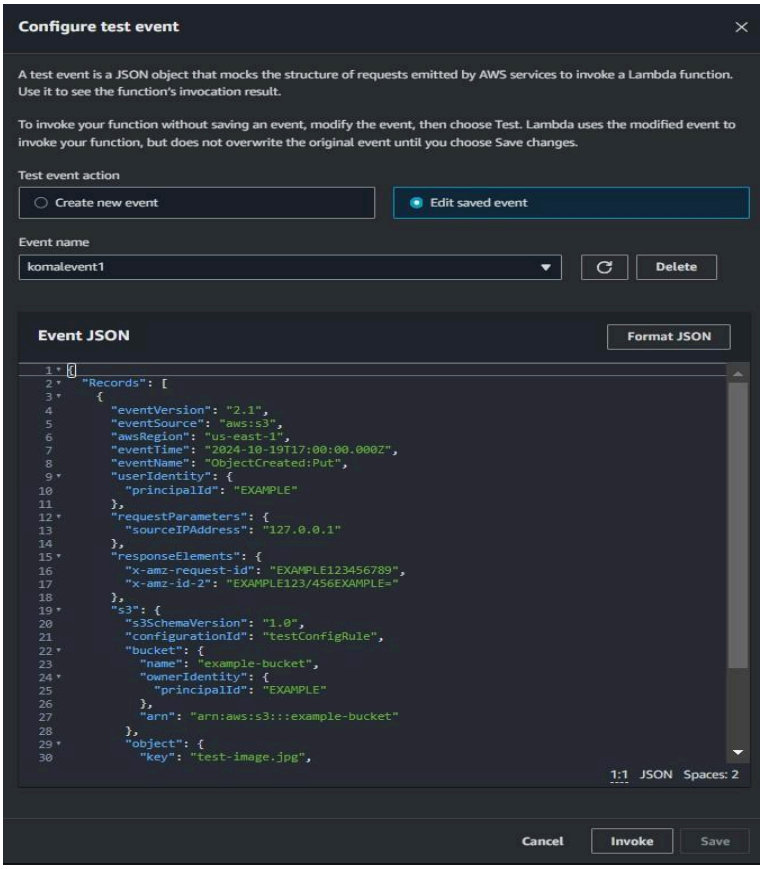
-USE THE SUITABLE EVENT FOR YOUR CODE.

THE EVENT SHOULD INCLUDE BELOW

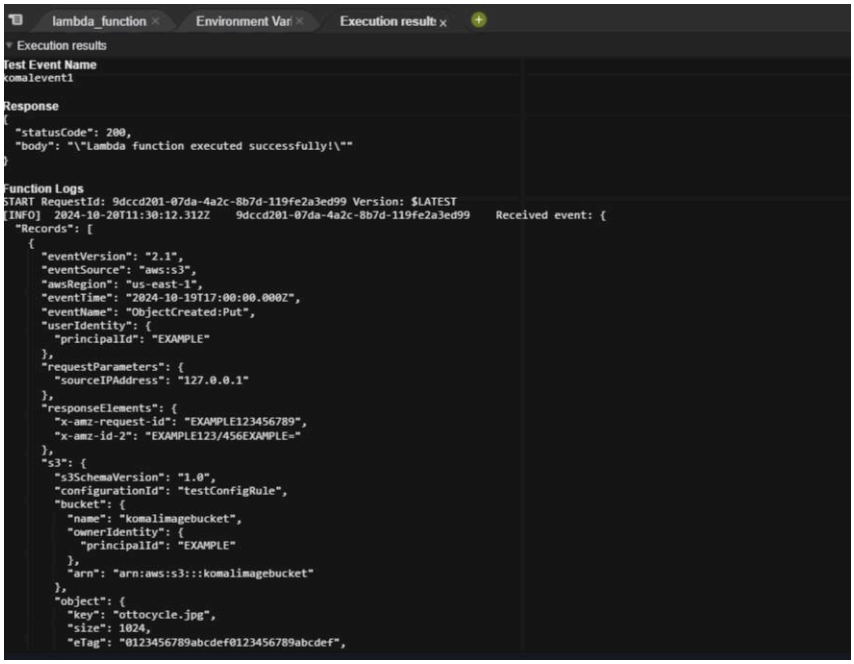
SCRIPT.

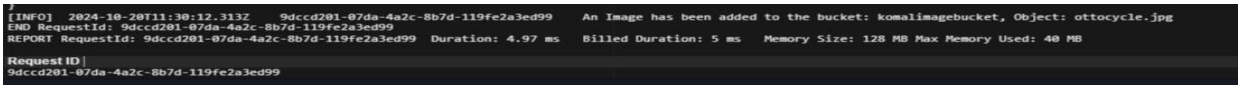

```
{
  "Records": [
    {
      "eventVersion": "2.1",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "2024-10-19T17:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "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"
```

```
    },
    "object": {
      "key": "test-image.jpg",
      "size": 1024,
      "eTag": "0123456789abcdef0123456789abcdef",
      "sequencer": "0A1B2C3D4E5F678901"
    }
   }
  }
]}
```



THE RESULT OF EXECUTION WILL LOOK LIKE:



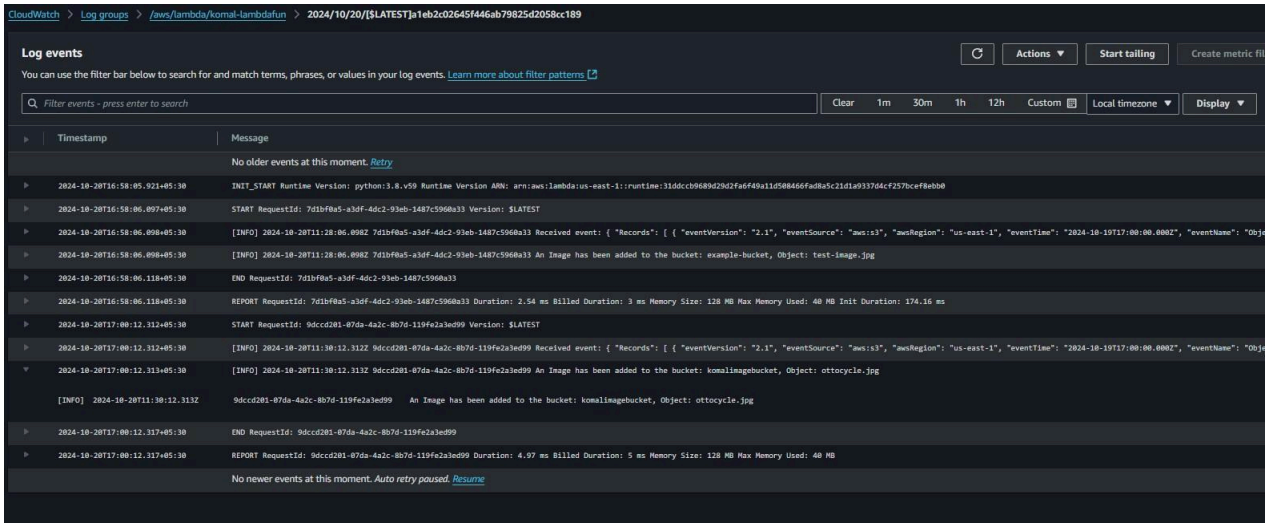


-*the output of this code is of before the image fourstroke.jpg was uploaded.*

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



AFTER UPLOADING ONE MORE IMAGE:

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
```

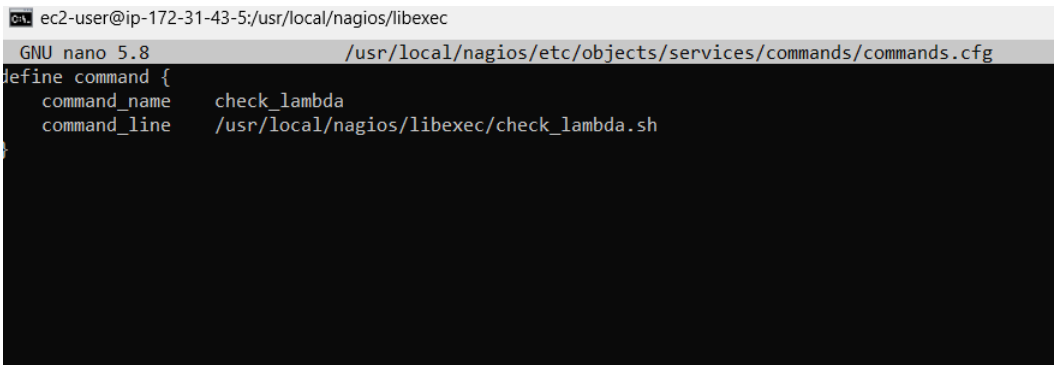
```
    echo "No log events found for Lambda function execution after image upload."
    exit 3
else
    echo "Log events from Lambda function triggered by S3 upload:"
    echo "$LOG_EVENTS"
    exit 0
fi
```

-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
}
```



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="ASIASWU6DPVCRK2RKYT7"
ec2-user@ip-172-31-43-5 libexec]$   export AWS_SESSION_TOKEN="IQoJb3JpZ2luX2VjEAcaCXVzLXdlc3QtMiJIMEYCIQCd+3eoNv+R4fYRdMXGn8EJZybHYYtYLCkcgsy+368WbQIhAJo76ofF9/13D7vdWLAMgjs0DaO9wtDvrUjhOm
2bgnVKrUCCHAQARoMMTg2MDg4OTE0MjQ1IgwL6krnu8dc2j4+R0cqkgIzb4mKEu3h7CcxacQ/K4ttwsRkWxcB23W8utmDOMKntu1oMT9oSPglenSAEGjUm56TbNo7jpMk3cJZ9h0t2aGS8DUgLO66Swg+ce9EQRPJsrnbYiGnFTR9yDu579BPgfipI8u
3V9iW+Y/N75f1VEbT3yLIevmQAXoX59RurOdMFMf1rfu1MN8JkgcAf8JeJr5BJIeMfnNtZcKPMGtoTopBMir3QVjplZZ+ABmJES+9CtAwitbfVopQILFAw2uXqRGPNtTv4eBrW0ZWbA2Zfn0L8UnEpFr+jOg4i+EqE7YEVniWN2cnpJtqIDVoCnQNJ+E
9G5S5G9zYLjlmSKpH5MHojDRmcFs649qwJ+SnaaJhp80MPjO0rgGOpw8Nni6lfZKV5IUd4/EtBl5dgLI9eFTTTICYcdaPnCD0F2WUf3VzVa1ibCdl00rSbwTtbEXsXx41PBmTj2GmZ4qwmjR2ftaXkF9uwSMxQKZkrebOOv7c6II/v85OpJQAEOLC3ald
WQt/AeS9+2OkDrlI6LHdJv+seEqCmNhQ3DEoKDZ1YXR93wHUQy3T6Dfv4IHC+vcQYzafvIzpks"TZUKk0XeF1qMYWojXAeDtlCjQQdoB%2BlhGP1awBsSln1tqA1CzLXC%2FxK2TQxG%2Fw29L1FGvzq%2BTREzcgLPEXz0BdwwmrGuwB8r2%2B0gMC5
ec2-user@ip-172-31-43-5 libexec]$ oEb50M7IzTZrAq%2FE9XWcn6sFvQ4d9KEZsr7Ep64GYz1a4JU6EWpaLYQQA1Xn10utbCbVoH6VZOR9knGhBxIFScLxDZ%2BoXs8Nycsai7jtazNvWF7sIUHmsNC8zRXl0%2Bw450hK%2Fkb%2FaSmHZ%2F
ec2-user@ip-172-31-43-5 libexec]$ ^CuhL96Sq8wGlovqK0mFYv3SeExWCrl4febQrCxM9Lk%2BEH554XksEIQKAk%2B2Lcmvh35rK98VUY3V3MzrQC4%2F3ueNB0x1VPInqEkuWNuCT6KIDGs0lrnpmBa%2BkI85TrFW6hF8iDxFP27OW11bIk
ec2-user@ip-172-31-43-5 libexec]$ ^C2EhbTG3GiSMYQSwv%2F0XTgI3UlphHmPDsN3pyjmcwsZTTUm6H%2Bz8RlWpjnlP%2FDzzcocdgnleJUAvif%2FD08tWUA4z4ZAcsxEKXtbo9Ic4yVZH8FcL%2FOeD790kK3i3zZAIYMX6yjBMYFs4uFj
```

```
ec2-user@ip-172-31-43-5 libexec]$ export AWS_SECRET_ACCESS_KEY="uzjnupR3BdeJnPktvBZxxxzc4GHiCdmWJ9T+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.
```

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.