Muskan Chandiramani Roll No : 05 D15C

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 modern software architecture, serverless computing has revolutionized how we design and deploy applications. AWS Lambda, a serverless compute service, is a popular choice for executing code in response to various events such as S3 object uploads. Lambda is not only scalable and cost-effective but also integrates seamlessly with other AWS services, making it a go-to for developers seeking automation.

This report will guide you through the creation of a serverless application, highlighting the use of AWS S3 and Lambda, and showing how to monitor these components using Nagios on an EC2 instance. Monitoring plays a crucial role in ensuring the health and performance of applications, and Nagios provides robust capabilities for system and application monitoring.

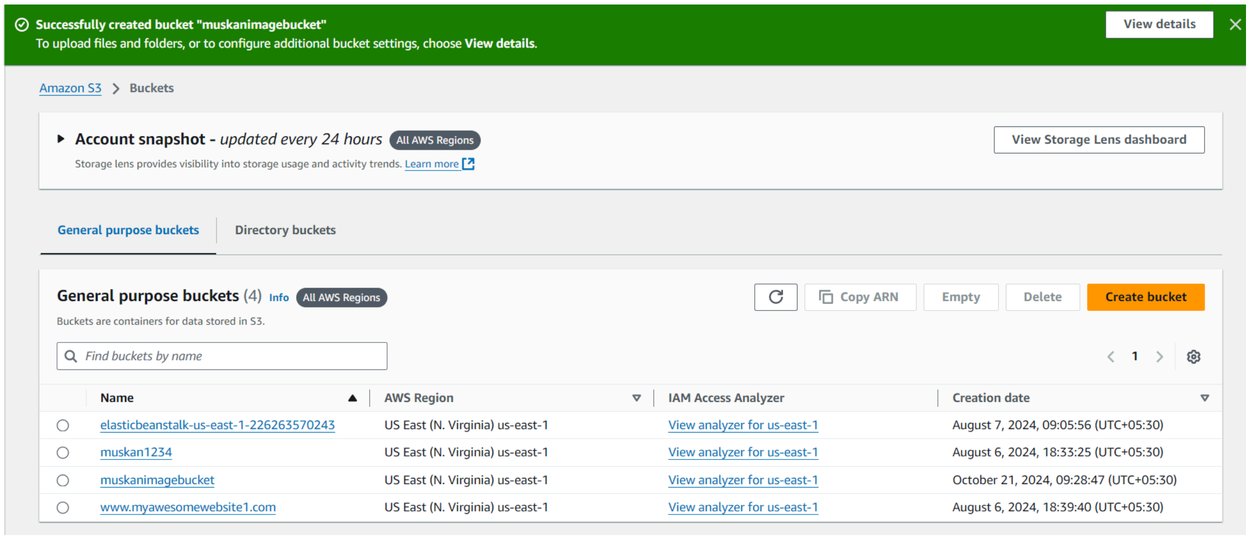
Key Points

- Creation of an S3 bucket in AWS to store files.
- Setting up an AWS Lambda function triggered by S3 uploads.
- Writing a Python script for Lambda that logs S3 events, specifically image uploads.
- Uploading images to S3 and monitoring Lambda execution through CloudWatch.
- Configuring Nagios to monitor Lambda functions and S3 bucket changes using a custom script.

Application of Serverless and Monitoring

1. Creation of an S3 Bucket

AWS S3 (Simple Storage Service) serves as scalable object storage, and it integrates well with Lambda. We begin by creating an S3 bucket in the AWS Management Console. The bucket will be used to upload images, and this will trigger the Lambda function we configure.



Successfully created bucket "muskanimagebucket"
To upload files and folders, or to configure additional bucket settings, choose **View details**.

Amazon S3 > Buckets

Account snapshot - *updated every 24 hours* All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets (4) Info All AWS Regions
Buckets are containers for data stored in S3.

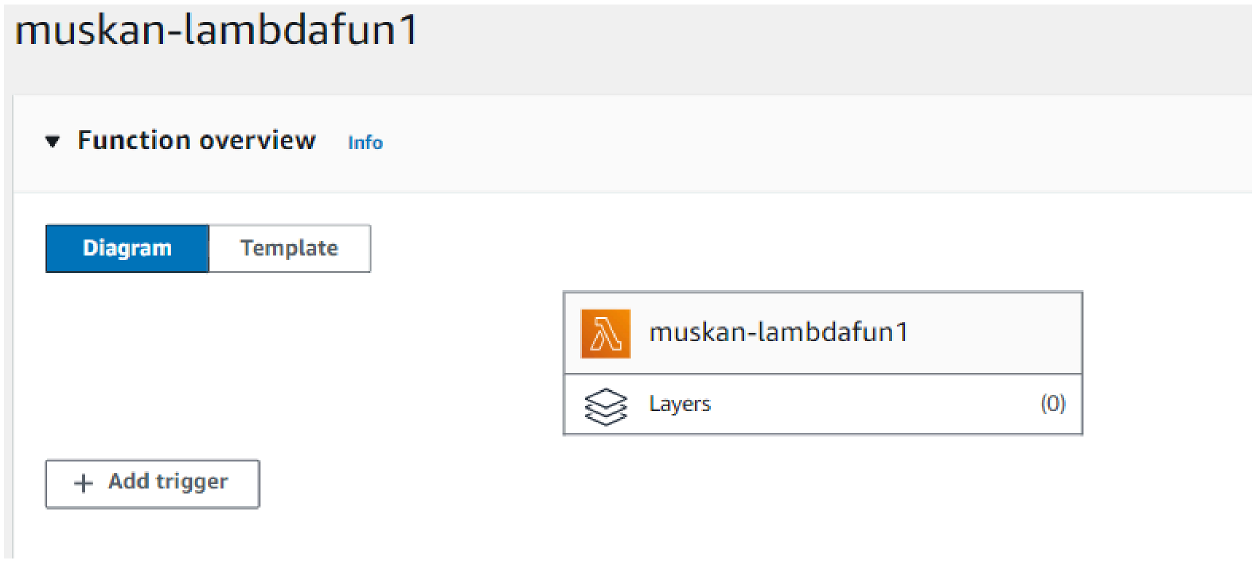
Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-226263570243	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 7, 2024, 09:05:56 (UTC+05:30)
muskan1234	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 6, 2024, 18:33:25 (UTC+05:30)
muskanimagebucket	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 21, 2024, 09:28:47 (UTC+05:30)
www.myawesomewebsite1.com	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 6, 2024, 18:39:40 (UTC+05:30)

2. Creation of Lambda Function

Next, we create a Lambda function and configure it to respond to events triggered by image uploads to the S3 bucket. A basic Python script is implemented within the Lambda function to log the events and verify if the uploaded file is an image.

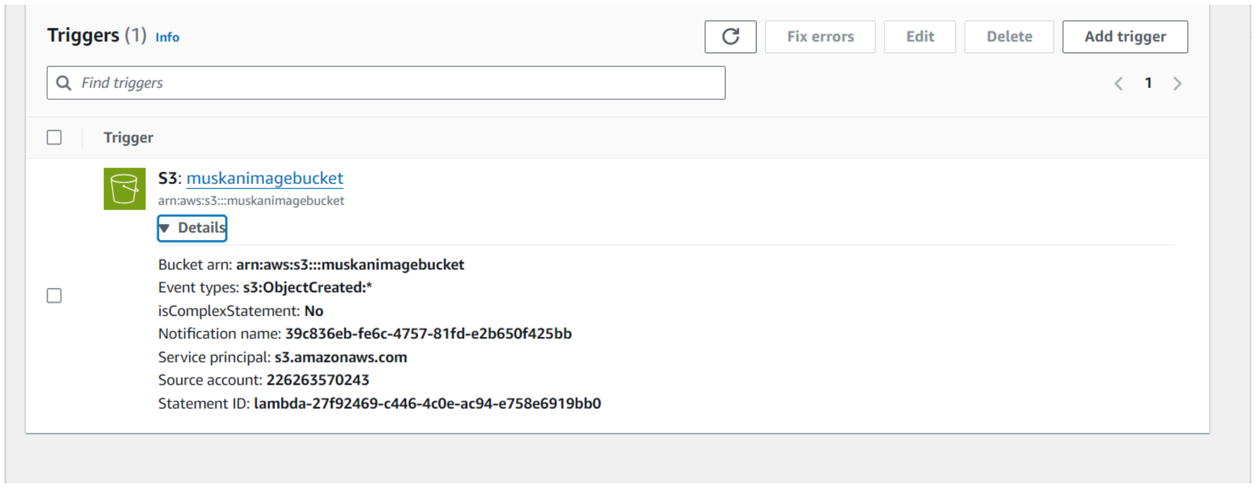
The Lambda function creation involves the following steps:

- Open the AWS Lambda console.
- Choose **Create function** and select the runtime as Python.
- Configure the function with necessary roles and permissions.



3. S3 Trigger for Lambda

In the S3 console, we add a trigger that notifies the Lambda function whenever a new image is uploaded. This is done by navigating to the S3 bucket, selecting "Properties," and adding a notification that sends S3 events to the Lambda function.



Bucket arn: arn:aws:s3:::muskanimagebucket
Event types: s3:ObjectCreated:*
isComplexStatement: No
Notification name: 39c836eb-fe6c-4757-81fd-e2b650f425bb
Service principal: s3.amazonaws.com
Source account: 226263570243
Statement ID: lambda-27f92469-c446-4c0e-ac94-e758e6919bb0

4. Lambda Python Script

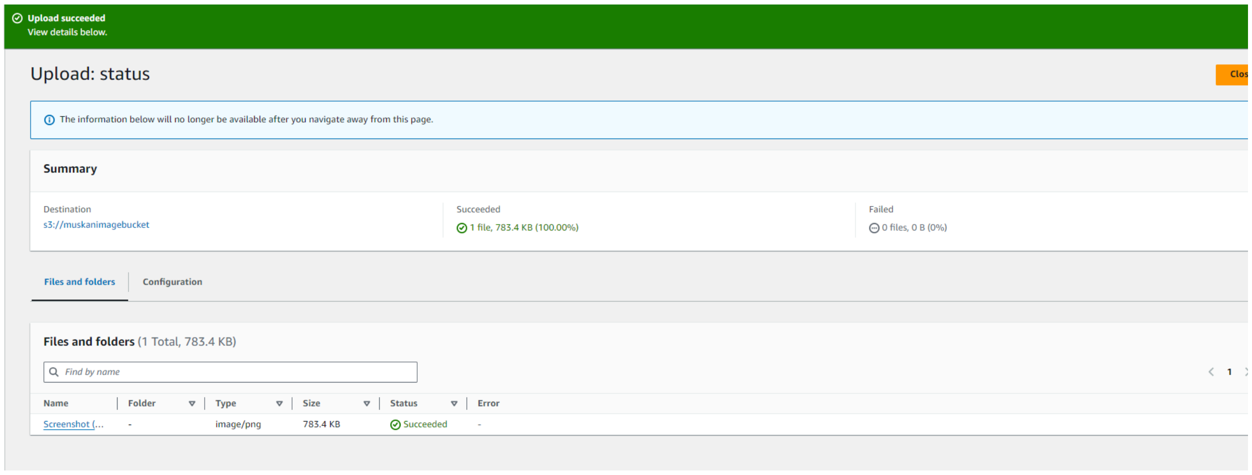
The Python script inside the Lambda function uses AWS SDK (Boto3) to capture S3 events. The script logs the name and key of the uploaded object and verifies if the uploaded file is an image. The script is as follows:

python
Copy code

```
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!')
    }
```

5. Upload Image to S3

Upload an image (e.g., `fourstroke.jpg`) to the S3 bucket. This triggers the Lambda function, which processes the event and logs the result.



Upload succeeded
View details below.

Upload: status

Clos

The information below will no longer be available after you navigate away from this page.

Summary

Destination	Succeeded	Failed
s3://muskanimagebucket	1 file, 783.4 KB (100.00%)	0 files, 0 B (0%)

Files and folders | Configuration

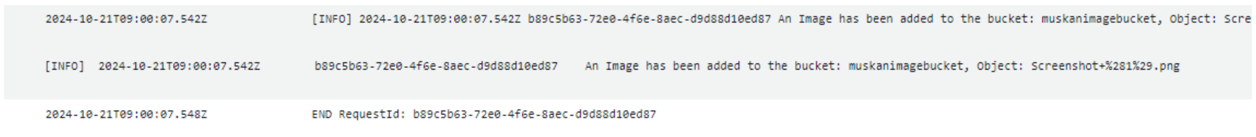
Files and folders (1 Total, 783.4 KB)

Find by name

Name	Folder	Type	Size	Status	Error
Screenshot (...	-	image/png	783.4 KB	Succeeded	-

6. CloudWatch Logs

Once the image is uploaded, the Lambda function’s execution is logged in AWS CloudWatch. To verify, navigate to CloudWatch, go to "Log Groups," and find the logs under the corresponding Lambda function's name. Here, you can see detailed logs indicating the success of the image upload event.



```
2024-10-21T09:00:07.542Z          [INFO] 2024-10-21T09:00:07.542Z b89c5b63-72e0-4f6e-8aec-d9d88d10ed87 An Image has been added to the bucket: muskanimagebucket, Object: Scre

[INFO]  2024-10-21T09:00:07.542Z     b89c5b63-72e0-4f6e-8aec-d9d88d10ed87    An Image has been added to the bucket: muskanimagebucket, Object: Screenshot+%281%29.png

2024-10-21T09:00:07.548Z          END RequestId: b89c5b63-72e0-4f6e-8aec-d9d88d10ed87
```

Monitoring Using Nagios on EC2

Nagios is a well-known open-source monitoring tool used to monitor applications and infrastructure. In this use case, Nagios is installed on an EC2 instance to monitor the Lambda function and S3 bucket.

1. Nagios Setup

First, ensure that Nagios is installed and running on your EC2 instance. The following steps outline how Nagios can be configured to monitor the Lambda function:

1. Navigate to the `libexec` directory where Nagios plugins are located.
2. Create a shell script named `check_lambda.sh` that checks for Lambda function execution triggered by S3 image uploads.

2. Nagios Monitoring Script

The following script monitors the Lambda function for S3 image uploads. It checks the latest object in the S3 bucket and fetches logs of the latest Lambda execution from CloudWatch:

bash
Copy code
```
#!/bin/bash

# AWS Lambda function name
LAMBDA_FUNCTION_NAME="muskan-lambdafun1"
S3_BUCKET_NAME="your-s3-bucket-name"
LOG_GROUP_NAME="/aws/lambda/$LAMBDA_FUNCTION_NAME"
CURRENT_TIME=$(date -u +"%Y-%m-%dT%H:%M:%S.000Z")

# Check for new image upload in 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

# Fetch logs of Lambda function execution
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

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))

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
```

3. Configure Nagios

After writing the script, add it to Nagios as a new command and service. Update the `commands.cfg` and `services.cfg` files to include the Lambda function monitoring. Restart Nagios for the changes to take effect.

4. Monitor Lambda Function

Run the following command to monitor the Lambda function and S3 bucket changes:

bash
Copy code
```
./check_lambda.sh
```

Theoretical Overview

AWS Lambda represents a shift towards serverless computing, where developers focus on business logic without worrying about infrastructure. Events from services like S3 can trigger Lambda, leading to reactive programming models. However, monitoring serverless functions is crucial, as failures or performance issues need to be detected early. Nagios provides a powerful solution by monitoring Lambda's interactions with S3 and logging these events for visibility.

Conclusion

This serverless architecture showcases how AWS Lambda can be efficiently used to process events from S3, making it a suitable solution for real-time processing tasks such as image uploads. By integrating Nagios, we can monitor Lambda’s performance and track critical events, ensuring a well-maintained and resilient application. Together, these tools offer a robust solution for developing scalable and monitored serverless applications.