

Case Study Report

Case Study : 05 Serverless Application With Monitoring

2022.teena.waishy@ves.ac.in

Name :Teena Waishy Div:D15B Roll No: 65

Advance DevOps Case Study

Date : 24 -10 -24

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."

Concepts Used: AWS Lambda, S3, and Nagios.

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:

This project involves setting up a **Serverless Application with Monitoring** using **AWS Lambda, S3, and Nagios**. The goal is to create an automated system where an AWS Lambda function logs an event every time an image is uploaded to a specific S3 bucket. In parallel, Nagios will monitor the Lambda function's execution and S3 bucket status, ensuring that both components are working properly. This solution combines serverless architecture for scalability and cost efficiency with traditional monitoring tools for reliability.

Components Used:

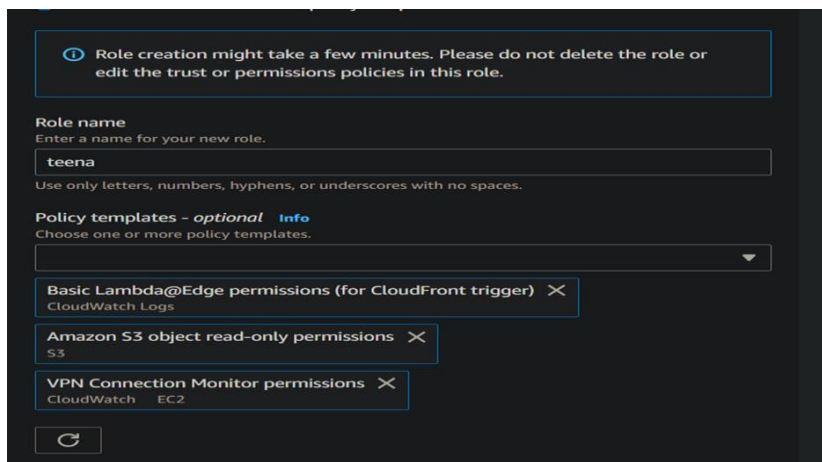
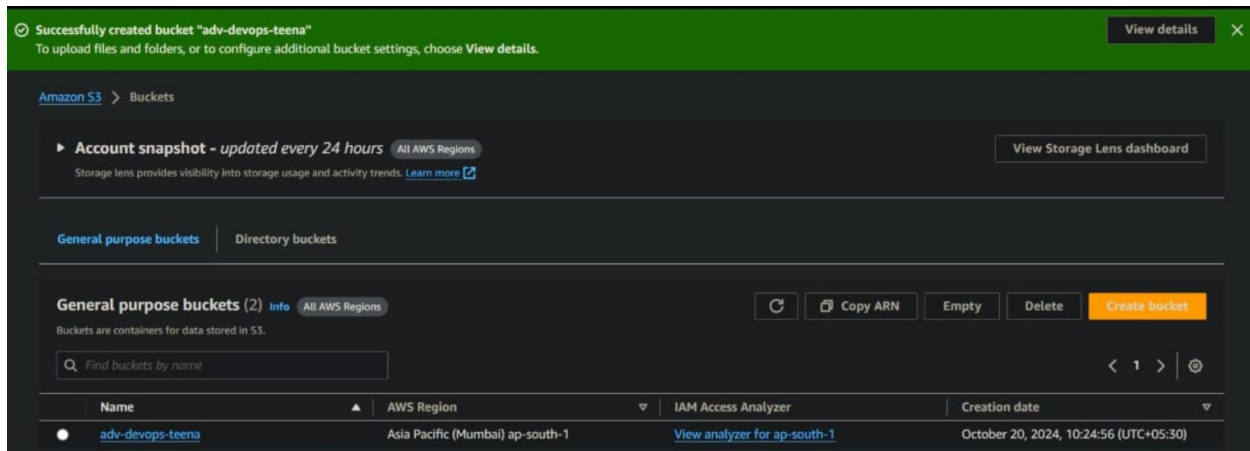
1. AWS Lambda: A serverless compute service that runs code in response to events
2. Amazon S3 (Simple Storage Service): An object storage service that stores data in the form of objects (files)
3. AWS CloudWatch: A monitoring and observability service used to collect and visualize logs.

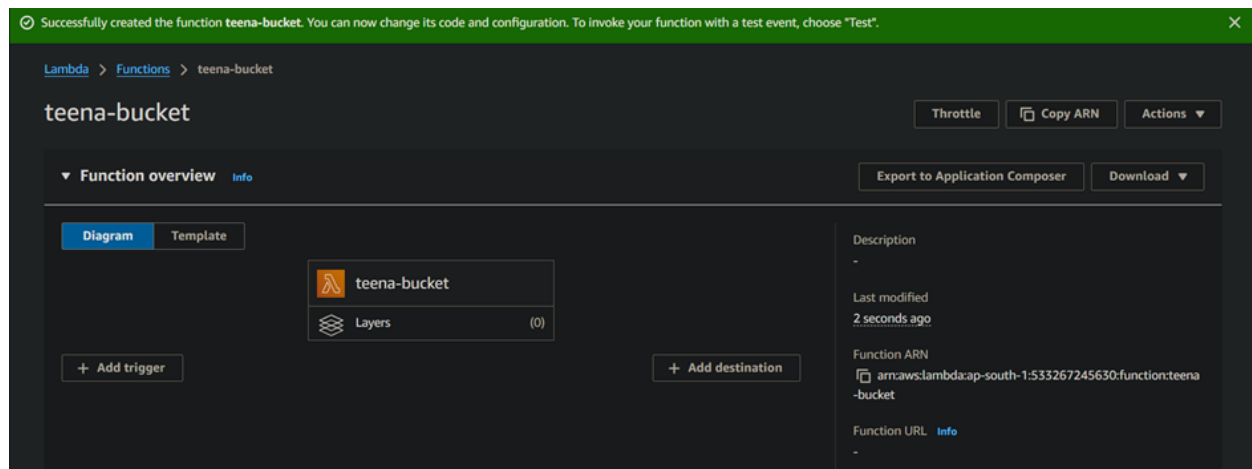
4. Nagios : An open-source monitoring system that provides alerts on the status of IT infrastructure.

Step-by-Step Implementation

Step 1: Creating the S3 Bucket

1. Login to AWS Console: Access the AWS Management Console.
2. Navigate to S3: Search for and select "S3" service.
3. Create an S3 Bucket:
 - Click "Create bucket".
 - Enter a unique Bucket name.
 - Choose the desired AWS Region.
 - Leave the rest as default and click "Create bucket".





Step 2: Creating the Lambda Function

1. Navigate to Lambda Service:

- Search for and select "Lambda" service in the AWS Console.
- Click "Create function".
- Choose "Author from scratch".

2. Configure Function:

- Function name: LogImageUpload
- Runtime: **Python 3.x** (Choose the latest version available)
- Click "Create function".

3. Lambda Function Code:

- Scroll to the "Function code" section.
- Replace the template code with the following Python code:
- Click "Deploy" to save the function.

The image shows two parts of the AWS Lambda console. The top part is the 'Code source' view, displaying a Python file named `lambda_function.py` with the following code:

```
1 import json
2 import logging
3
4 logger = logging.getLogger()
5 logger.setLevel(logging.INFO)
6
7 def lambda_handler(event, context):
8     # Parse the S3 event
9     for record in event['Records']:
10         s3_bucket = record['s3']['bucket']['name']
11         s3_object = record['s3']['object']['key']
12
13         # Log the image upload event
14         logger.info(f"An Image has been added to bucket: {s3_bucket}, object key: {s3_object}")
15
16     return {
17         'statusCode': 200,
18         'body': json.dumps('Image upload logged')
19     }
20
```

The bottom part is the 'Add trigger' configuration page. It shows the 'Trigger configuration' section with the following settings:

- Source:** S3 (aws asynchronous storage)
- Bucket:** s3/adv-devops-teena (Bucket region: ap-south-1)
- Event types:** All object create events (selected), PUT (selected)
- Prefix - optional:** e.g. images/

Step 3: Configuring S3 to Trigger Lambda Function

Set Up S3 Trigger:

- Scroll down to the "Triggers" section of the Lambda function.
- Click "Add trigger".
- Choose **S3** as the trigger source.
- Configure the following:
 - **Bucket:** Select the bucket created.
 - **Event type:** Choose "All object create events".

Click "Add" to save the trigger.

Lambda > Functions > teena-bucket

teena-bucket

Throttle Copy ARN Actions

✓ The trigger adv-devops-teena was successfully added to function teena-bucket. The function is now receiving events from the trigger.

▼ Function overview Info Export to Application Composer Download

Diagram Template

teena-bucket

Layers (0)

S3

+ Add trigger

+ Add destination

Description

Last modified 2 minutes ago

Function ARN arn:aws:lambda:ap-south-1:533267245630:function:teena-bucket

Function URL Info

Code Test Monitor Configuration Aliases Versions

✓ Upload succeeded View details below.

Upload: status

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

Summary

Destination	Succeeded	Failed
s3://adv-devops-teena	✓ 1 file, 56.0 KB (100.00%)	⊖ 0 files, 0 B (0%)

Files and folders Configuration

Files and folders (1 Total, 56.0 KB)

Find by name

Name	Folder	Type	Size	Status	Error
51QLAOUcVAL_SL1000.jpg	-	image/jpeg	56.0 KB	✓ Succeeded	-

Step 4: Uploading a Test Image to S3

1. Upload an Image:

- Go back to the S3 bucket
- Click "Upload".
- Select a sample image from your local machine and upload it.

2. Verify Lambda Logs:

- Navigate to **CloudWatch Logs** to check Lambda execution logs.
- Search for the log group associated with your Lambda function
- Ensure the log entries show "An Image has been added."

CloudWatch > Log groups

Log groups (2) [Refresh](#) [Actions](#) [View in Logs Insights](#) [Start tailing](#) [Create log group](#)

By default, we only load up to 10000 log groups.

☒ Exact match [< 1 >](#) [Settings](#)

Log group	Log class	Anomaly d...	Data p...	Sensit...	Retenti...	Metr
/aws/lambda/S3ImageLogger	Standard	Configure	-	-	Never expire	-
/aws/lambda/teena-bucket	Standard	Configure	-	-	Never expire	-

CloudWatch > Log groups > [/aws/lambda/teena-bucket](#) > 2024/10/20/[\$LATEST]2a0e9e32d9164e13b57ceacd9d09447d

Log events [Refresh](#) [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](#) [UTC timezone](#) [Display](#) [Settings](#)

Timestamp	Message
No older events at this moment. Retry	
2024-10-20T05:10:09.162Z	INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:ap-south-1::runtime:188d9ca2e2714ff5637bd2bb...
2024-10-20T05:10:09.261Z	START RequestId: 9031b915-4416-4d56-95be-7f2405323c15 Version: \$LATEST
2024-10-20T05:10:09.262Z	[INFO] 2024-10-20T05:10:09.261Z 9031b915-4416-4d56-95be-7f2405323c15 An Image has been added to bucket: adv-devops-teena, ob...
2024-10-20T05:10:09.266Z	END RequestId: 9031b915-4416-4d56-95be-7f2405323c15
2024-10-20T05:10:09.266Z	REPORT RequestId: 9031b915-4416-4d56-95be-7f2405323c15 Duration: 4.80 ms Billed Duration: 5 ms Memory Size: 128 MB Max Memor...
No newer events at this moment. Auto retry paused. Resume	

Step 5: Setting Up Nagios to Monitor AWS Environment

1. Set Up a Nagios Server:

- Follow Nagios installation instructions for your OS. Use resources like Nagios Downloads.
- Install the necessary plugins and configure the server.

2. Install CloudWatch Plugin for Nagios:

- On your Nagios server, download and install a plugin like `check-cloudwatch` from the community repository.
- Configure your **AWS credentials** on the Nagios server to allow access to CloudWatch.

3. Configure Nagios to Monitor CloudWatch Logs:

- Create a command definition in your Nagios configuration file
- Create a service definition in Nagios to monitor Lambda execution

4. Monitor S3 Bucket:

- Use a plugin like `check_s3` to monitor your S3 bucket.
- Add a new command in `commands.cfg` for the S3 bucket status

Create a service definition in Nagios for S3 monitoring

```
A newer release of "Amazon Linux" is available.  
Version 2023.6.20241010:  
Run "/usr/bin/dnf check-release-update" for full release and version update info  
  
#  
~\##### Amazon Linux 2023  
~~~\#####\  
~~~\####|  
~~~\#/ https://aws.amazon.com/linux/amazon-linux-2023  
~~~v~' '~>  
~~~~  
~~~~.  
~~~~/  
~~~~/m/'
```

Last login: Sat Oct 12 13:01:16 2024 from 49.36.110.11
[ec2-user@ip-172-31-5-147 ~]\$

i-0d62d6e9163a3f579 (nagios-host)

PublicIPs: 15.207.19.46 PrivateIPs: 172.31.5.147

```
ec2-user@ip-172-31-5-147 ~$ pip3 install boto3
Defaulting to user installation because normal site-packages is not writeable
Collecting boto3
  Downloading boto3-1.35.44-py3-none-any.whl (139 kB)
    [REDACTED] 139 kB 6.0 MB/s
Collecting s3transfer<0.11.0,>=0.10.0
  Downloading s3transfer-0.10.3-py3-none-any.whl (82 kB)
    [REDACTED] 82 kB 2.2 MB/s
Collecting botocore<1.36.0,>=1.35.44
  Downloading botocore-1.35.44-py3-none-any.whl (12.6 MB)
    [REDACTED] 12.6 MB 35.0 MB/s
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3.9/site-packages (from boto3) (0.10.0)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3.9/site-packages (from botocore<1.36.0,>=1.35.44->boto3) (1.25.10)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.36.0,>=1.35.44->boto3) (2.8.1)
Requirement already satisfied: six>=1.5 in /usr/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.36.0,>=1.35.44->boto3) (1.15.0)
Installing collected packages: botocore, s3transfer, boto3
Successfully installed botocore-1.35.44 boto3-1.35.44 s3transfer-0.10.3
ec2-user@ip-172-31-5-147 ~$
```

```

[ec2-user@ip-172-31-5-147 libexec]$ ls
check_apt      check_dhcp      check_file_age  check_ifoperstatus  check_mailq      check_ntp      check_ping      check_smtp      check_ups      utils.pm
check_breeze   check_dig       check_flexlm    check_ifstatus      check_mrtg       check_ntp_peer  check_pop       check_ssh      check_uptime  utils.sh
check_by_ssh   check_disk     check_ftp       check_imap          check_mrtgtraf   check_ntp_time  check_procs    check_swap     check_users
check_cimind   check_disk_smb  check_http     check_ircd         check_nagios     check_nwstat   check_real     check_tcp      check_wave
check_cloudwatch_logs.py  check_dns      check_icmp      check_load         check_nntp       check_oracle   check_rpc      check_time     negate
check_cluster  check_dummy    check_ide_smart  check_log          check_nt         check_overcr   check_sensors  check_udp      urlize
[ec2-user@ip-172-31-5-147 libexec]$

```

Permissions policies (1/1248)

Choose one or more policies to attach to your new user.

Filter by Type		7 matches	
<input type="text" value="CloudWatchLogs"/>		All types	
<input type="checkbox"/>	Policy name ↗	Type	Attached entities
<input type="checkbox"/>	AmazonAPIGatewayPushToCloudWatchLogs	AWS managed	0
<input type="checkbox"/>	AmazonDMSCloudWatchLogsRole	AWS managed	0
<input type="checkbox"/>	AWSAppSyncPushToCloudWatchLogs	AWS managed	0
<input type="checkbox"/>	AWSOpsWorksCloudWatchLogs	AWS managed	0
<input type="checkbox"/>	CloudWatchLogsCrossAccountSharingConfiguration	AWS managed	0
<input type="checkbox"/>	CloudWatchLogsFullAccess	AWS managed	0
<input checked="" type="checkbox"/>	CloudWatchLogsReadOnlyAccess	AWS managed	0

```

aws Services Search [Alt+S]

import boto3
import sys

def check_cloudwatch_logs(log_group_name):
    client = boto3.client('logs', region_name='ap-south-1') # Set your region here
    try:
        response = client.describe_log_streams(
            logGroupName=log_group_name,
            orderBy='LastEventTime',
            descending=True,
            limit=1
        )
        if response['logStreams']:
            latest_stream = response['logStreams'][0]
            log_stream_name = latest_stream['logStreamName']

            # Get the latest log events
            events = client.get_log_events(
                logGroupName=log_group_name,
                logStreamName=log_stream_name,
                limit=10
            )

            # Check if there are any new events
            if events['events']:
                print(f"Found {len(events['events'])} new log events:")
                for event in events['events']:
                    print(f"Log Event: {event['message']}")

-- INSERT --

```


Access key created

This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

IAM > Users > teena > Create access key

Step 1

Access key best practices & alternatives

Step 2 - optional

Set description tag

Step 3

Retrieve access keys

Retrieve access keys [info](#)

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIAXYKJUNI7DPCQZF4K	7V3ISpaPUX2lFmC3I4pyN/vdA4a4HwU5Fd8GRZgR Hide

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

Download .csv file

Done

aws

Services

Search [Alt+S]

```
[ec2-user@ip-172-31-5-147 libexec]$ aws configure
AWS Access Key ID [None]: AKIAXYKJUNI7DPCQZF4K
AWS Secret Access Key [None]: 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
Default region name [None]: ap-south-1
Default output format [None]: json
[ec2-user@ip-172-31-5-147 libexec]$
```

```
define command {
    command_name    check_cloudwatch_logs
    command_line    /usr/bin/python3 /usr/local/nagios/libexec/check_cloudwatch_logs.py $ARG1$
}


```

aws

Services

Search [Alt+S]

```
[ec2-user@ip-172-31-5-147 libexec]$ sudo vim /usr/local/nagios/etc/objects/services.cfg
```

Step 6: Testing the Monitoring Setup

1. Upload another Test Image to S3:
 - Upload another image to [image-upload-bucket](#).
2. Verify Lambda Log in Nagios:
 - Check the status in Nagios.

- Ensure that the Lambda function logs are being captured.
- Confirm that any alerts are visible if issues are detected

The screenshot displays the Nagios web interface. At the top, there's a navigation bar with the AWS logo, a 'Services' menu, a search bar, and an '[Alt+S]' button. Below this, a code editor shows the configuration for a service named 'generic-service' on 'localhost'. The configuration includes a description 'Check Lambda Logs', a command 'check_cloudwatch_logs!/aws/lambda/teena-bucket', and various intervals and retry settings.

The main dashboard shows the 'Current Network Status' as 'Up' with 1 host, 1 down, 0 unreachable, and 0 pending. The 'Service Status Totals' show 10 OK, 1 warning, 0 unknown, 4 critical, and 0 pending. The 'Service Status Details For All Hosts' table lists services for 'linuxserver' and 'localhost'.

Host	Service	Status	Last Check	Duration	Attempt	Status Information
linuxserver	Current Load	OK	10-20-2024 06:27:20	7d 16h 44m 39s	1/4	OK - load average: 0.00, 0.00, 0.00
	Current Users	OK	10-20-2024 06:28:00	7d 16h 43m 59s	1/4	USERS OK - 1 users currently logged in
	HTTP	CRITICAL	10-20-2024 06:28:40	0d 0h 56m 10s	4/4	CRITICAL - Socket timeout after 10 seconds
	Root Partition	OK	10-20-2024 06:29:20	7d 16h 42m 39s	1/4	DISK OK - free space: / 5892 MB (72% inode=98%)
	SSH	CRITICAL	10-20-2024 06:25:00	0d 0h 54m 50s	4/4	CRITICAL - Socket timeout after 10 seconds
	Swap Usage	CRITICAL	10-20-2024 06:25:40	7d 16h 38m 19s	4/4	SWAP CRITICAL - 0% free (0 MB out of 0 MB) - Swap is either disabled, not present, or of zero size.
localhost	Total Processes	OK	10-20-2024 06:26:20	7d 16h 40m 39s	1/4	PROCS OK: 37 processes with STATE = RSZDT
	Current Load	OK	10-20-2024 06:26:40	7d 17h 19m 49s	1/4	OK - load average: 0.00, 0.00, 0.00
	Current Users	OK	10-20-2024 06:27:40	7d 17h 19m 11s	1/4	USERS OK - 1 users currently logged in
	HTTP	WARNING	10-20-2024 06:28:20	0d 0h 46m 30s	4/4	HTTP WARNING: HTTP/1.1 403 Forbidden - 319 bytes in 0.000 second response time
	PING	OK	10-20-2024 06:29:01	7d 17h 17m 56s	1/4	PING OK - Packet loss = 0%, RTA = 0.03 ms
	Root Partition	OK	10-20-2024 06:29:40	7d 17h 17m 19s	1/4	DISK OK - free space: / 5892 MB (72% inode=98%)

Step 7: The final step involves verifying the successful execution of the Lambda function and the monitoring setup using AWS CloudWatch and Nagios:

Lambda Log Verification:

- In the provided screenshot, the CloudWatch log output clearly shows that the AWS Lambda function was triggered successfully when an image was uploaded to the S3 bucket. The log entry, "An Image has been added to bucket: adv-devops-teena, object key: SLI000...jpg", confirms that the function executed and accurately captured the event details, including the specific bucket name and object key.

- This output indicates that the Lambda function is functioning as expected, logging each new image upload event correctly.

```
Found 5 new log events:
Log Event: INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:ap-south-1::runtime:188d9ca2e2714ff5637bd2bbe06ceb81ec3bc408a0f277dab1
04c14cd814b081
Log Event: START RequestId: 9031b915-4416-4d56-95be-7f2405323c15 Version: $LATEST
Log Event: [INFO] 2024-10-20T05:10:09.261Z 9031b915-4416-4d56-95be-7f2405323c15 An Image has been added to bucket: adv-devops-teena, object key: 5
1QLA0UcVAL_SL1000_.jpg
Log Event: END RequestId: 9031b915-4416-4d56-95be-7f2405323c15
Log Event: REPORT RequestId: 9031b915-4416-4d56-95be-7f2405323c15 Duration: 4.80 ms Billed Duration: 5 ms Memory Size: 128 MB Max Memory Used: 3
2 MB Init Duration: 96.52 ms
[ec2-user@ip-172-31-5-147 libexec]$
```

Conclusion

The serverless application setup was successfully implemented, demonstrating the power of AWS Lambda for event-driven automation and the effectiveness of Nagios for real-time monitoring. Each image upload to the S3 bucket triggered the Lambda function, which logged the event as expected with the message "An Image has been added." Additionally, Nagios successfully monitored the Lambda function's execution and the status of the S3 bucket, ensuring both are operating smoothly. This project showcases the seamless integration of serverless architecture with traditional monitoring tools, providing a scalable, cost-effective, and reliable solution for event tracking and infrastructure monitoring.