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:

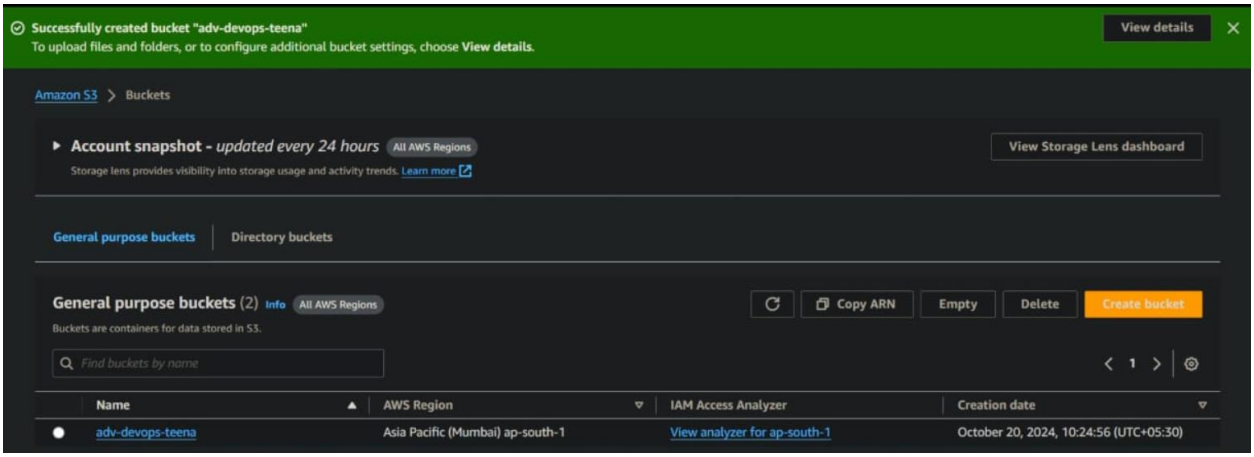
1. AWS Lambda: A serverless compute service that runs code in response to events
2. Amazon S3 (Simple Storage Service): An object storage service that stores data in the form of objects (files)
3. AWS CloudWatch: A monitoring and observability service used to collect and visualize logs.

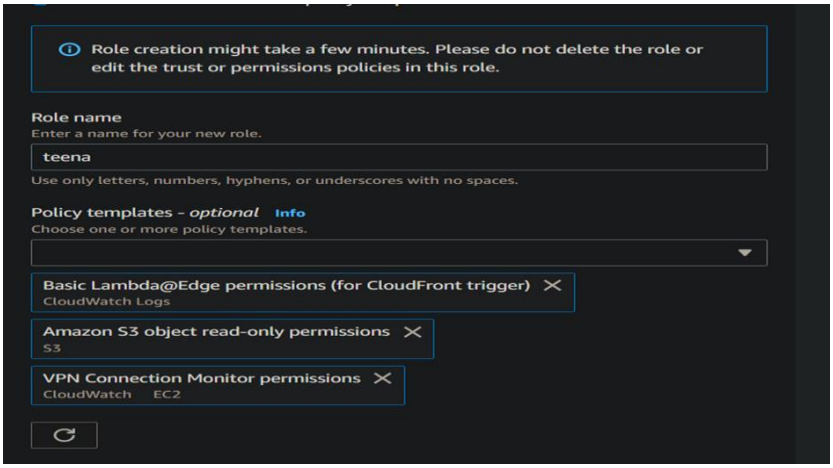
4. Nagios : An open-source monitoring system that provides alerts on the status of IT infrastructure.

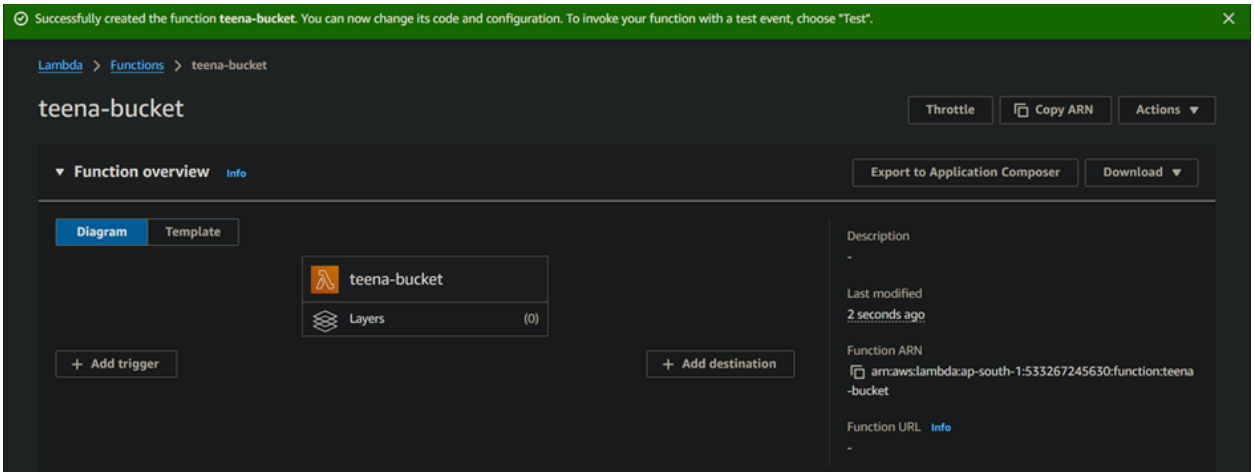
Step-by-Step Implementation

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".







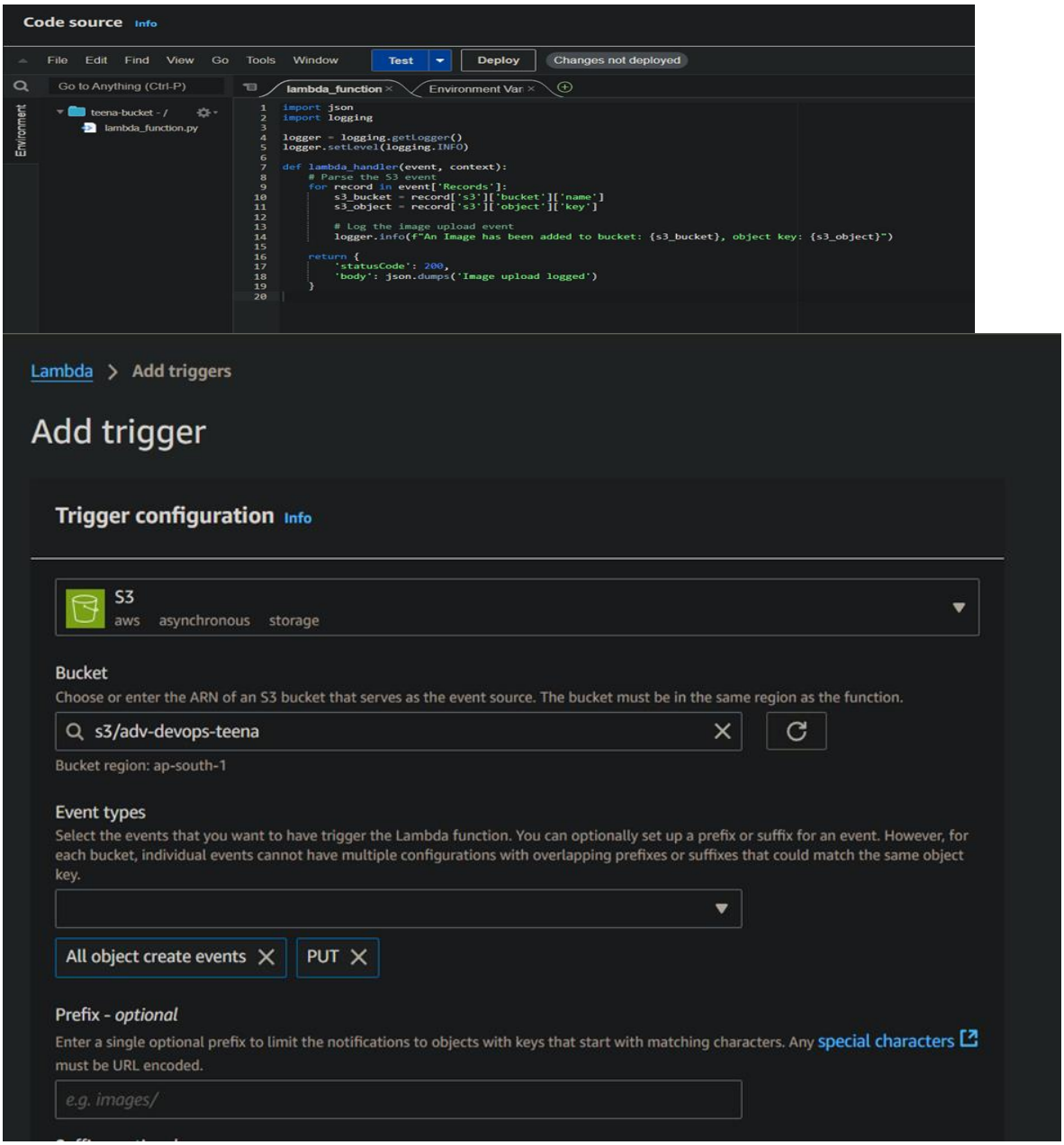
Step 2: Creating the Lambda Function

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.

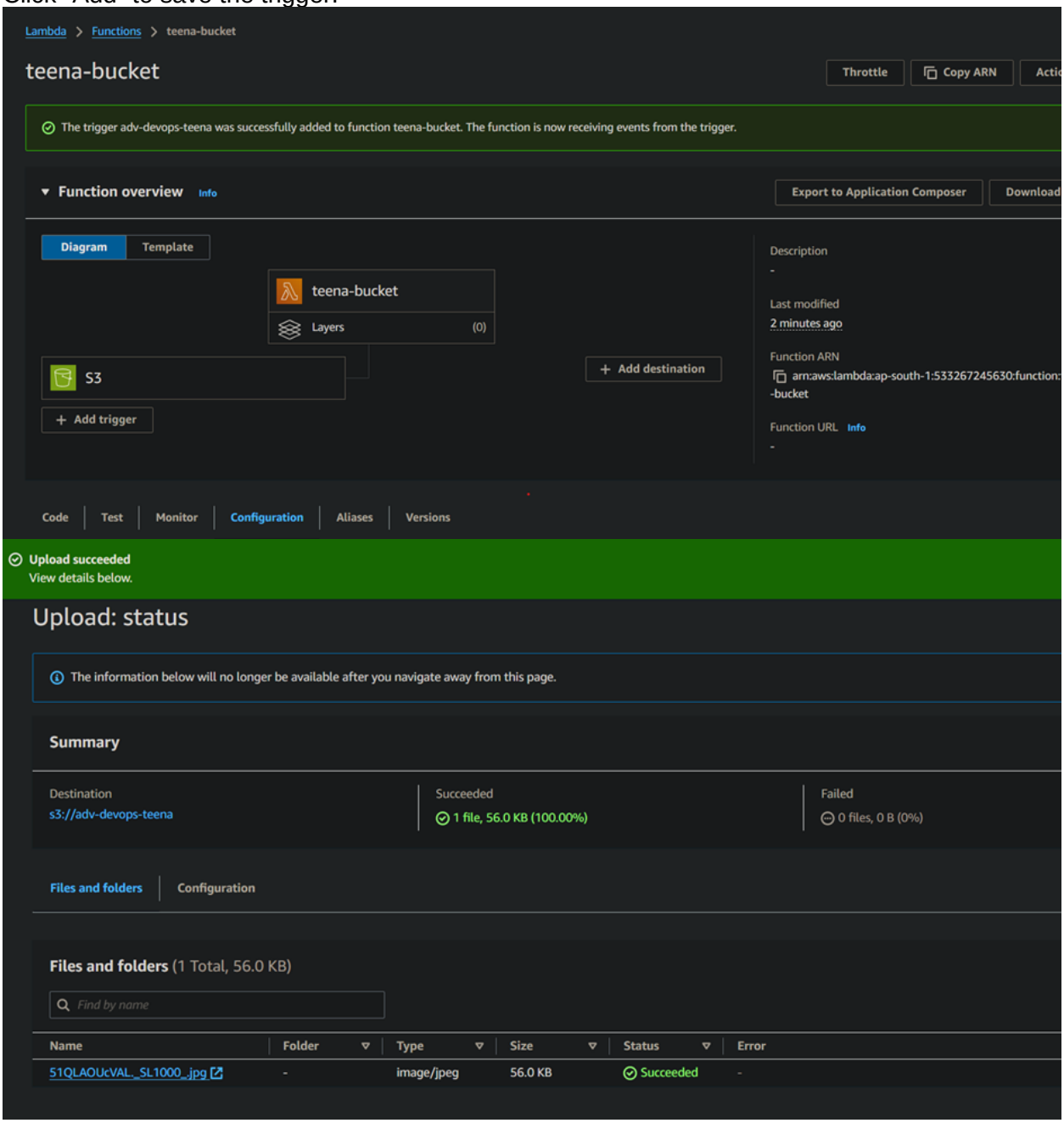


Step 3: Configuring S3 to Trigger Lambda Function

Set Up S3 Trigger:

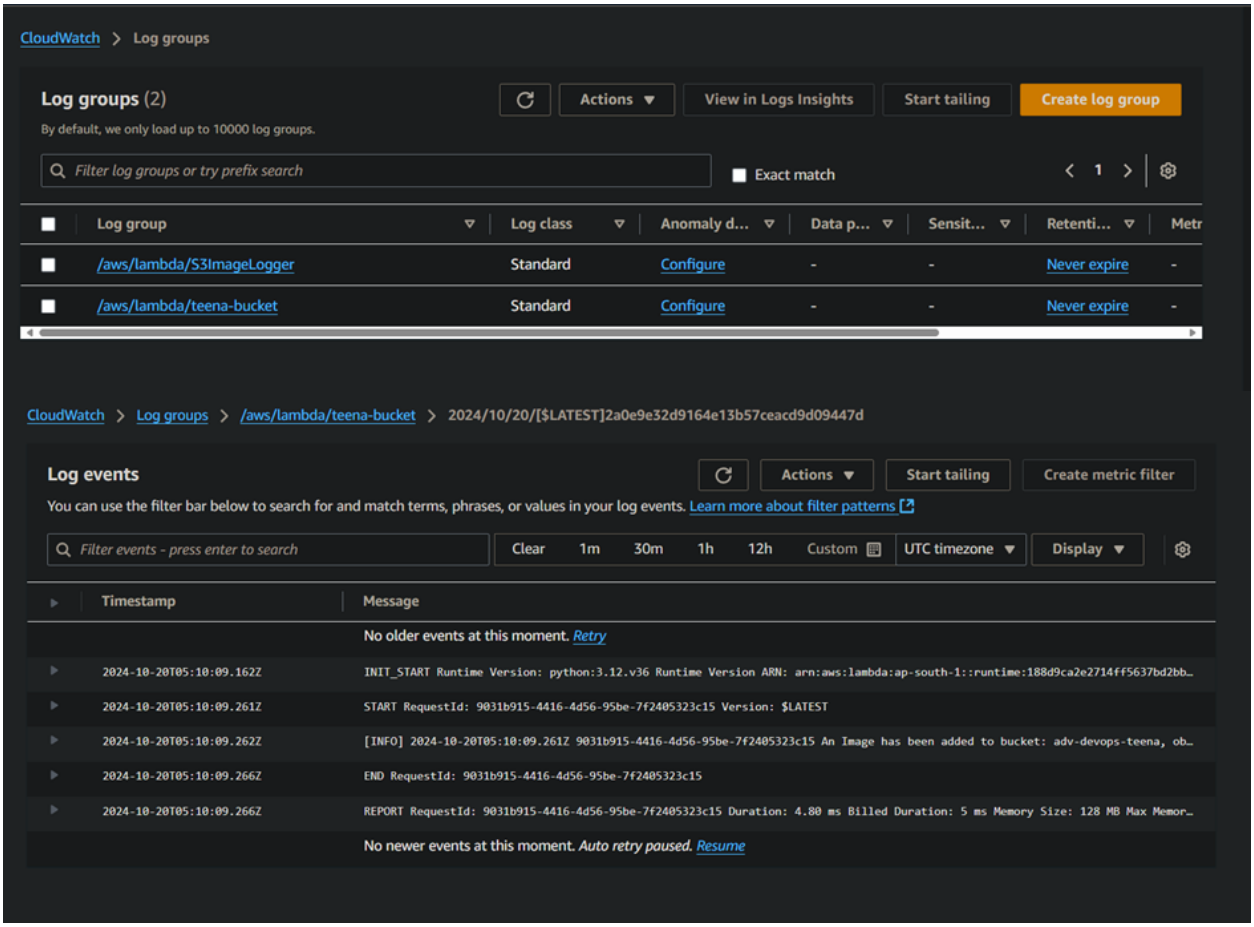
- Scroll down to the "Triggers" section of the Lambda function.
- Click "Add trigger".
- Choose **S3** as the trigger source.
- Configure the following:
  - **Bucket**: Select the bucket created.
  - **Event type**: Choose "All object create events".

Click "Add" to save the trigger.



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."



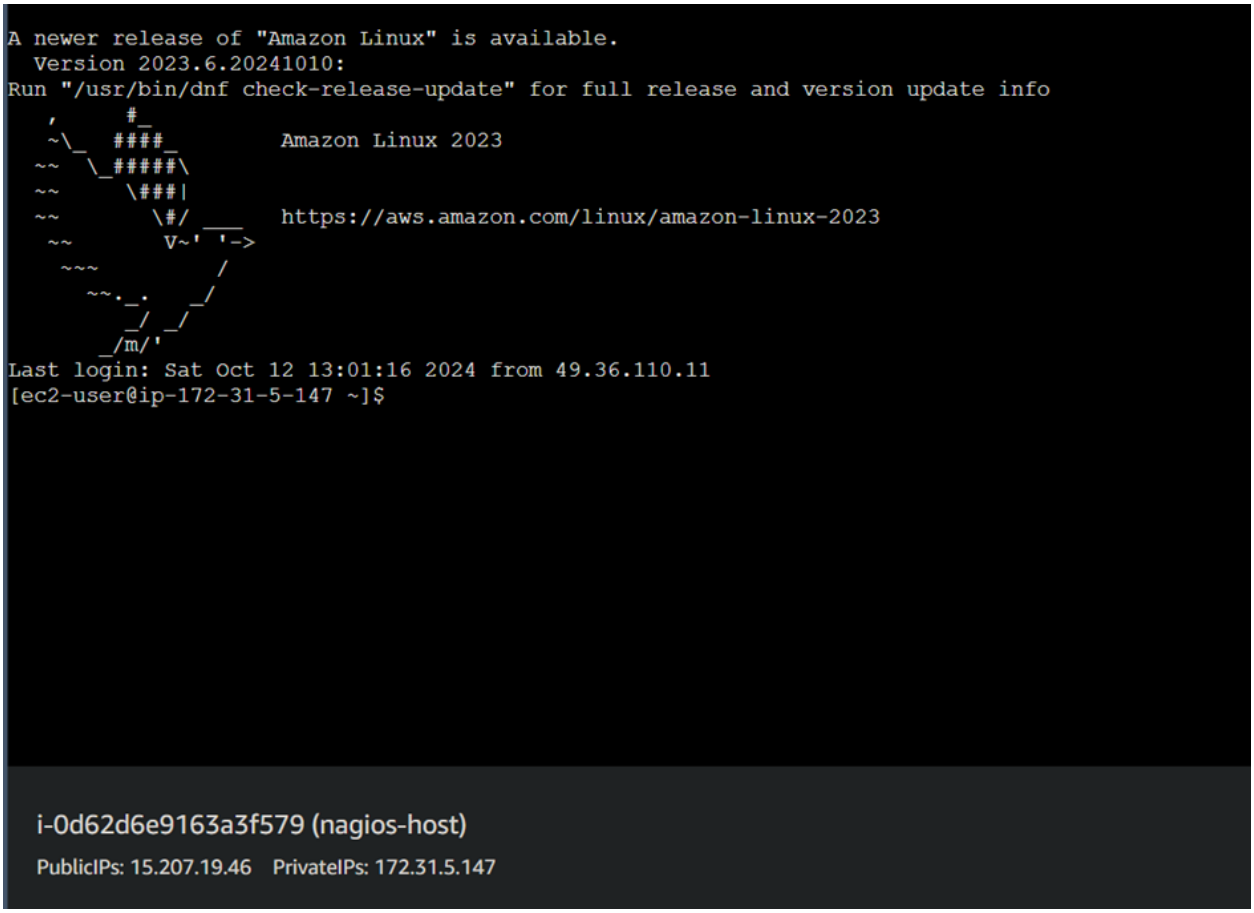
Step 5: Setting Up Nagios to Monitor AWS Environment

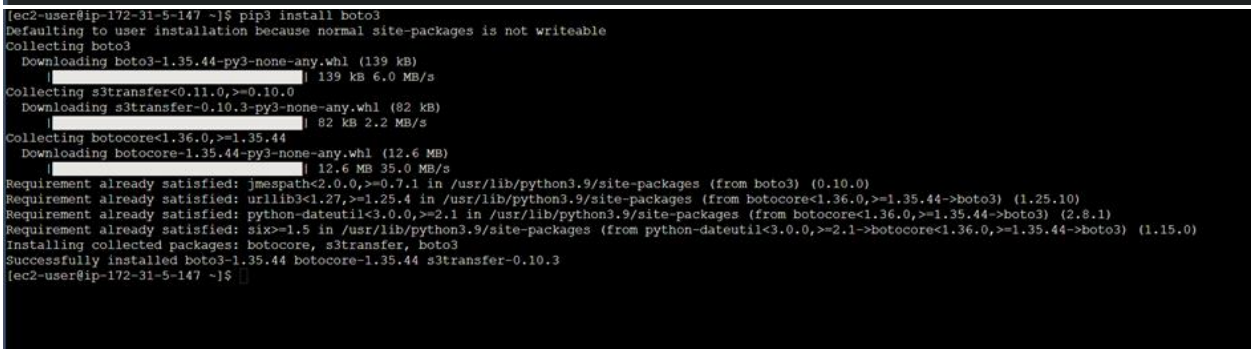
1. **Set Up a Nagios Server**:
    - Follow Nagios installation instructions for your OS. Use resources like Nagios Downloads.
    - Install the necessary plugins and configure the server.
2. **Install** CloudWatch **Plugin for Nagios**:
    - On your Nagios server, download and install a plugin like `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





```
[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_clamd            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.

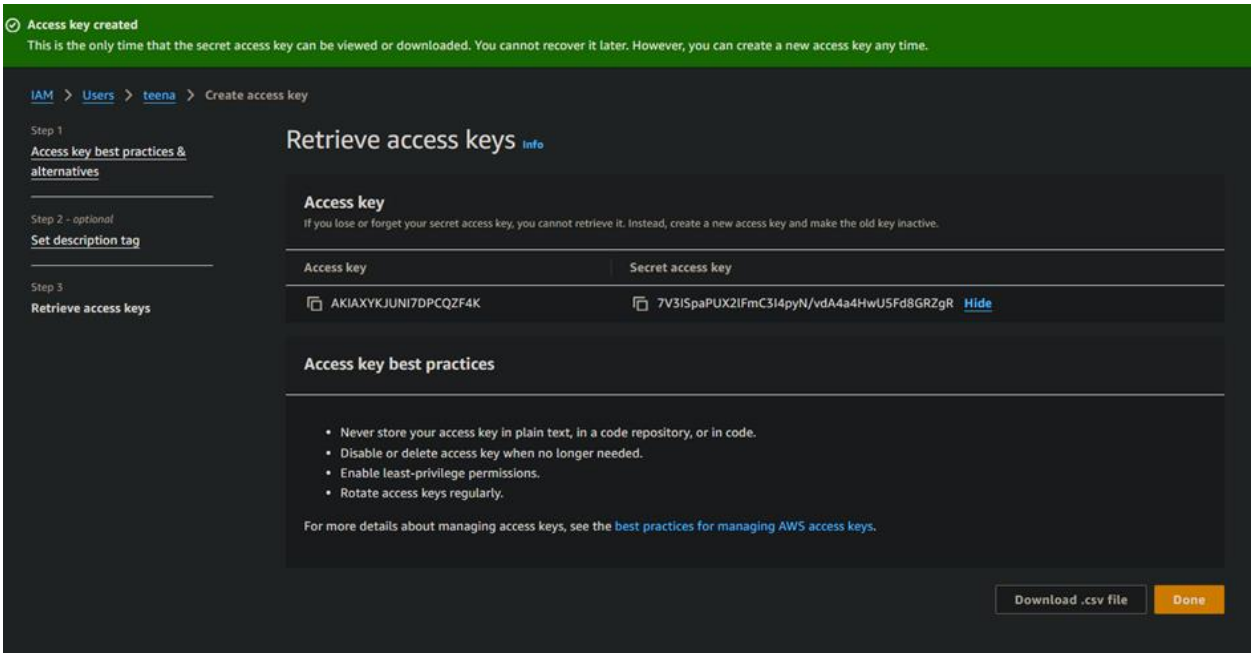
Filter by Type

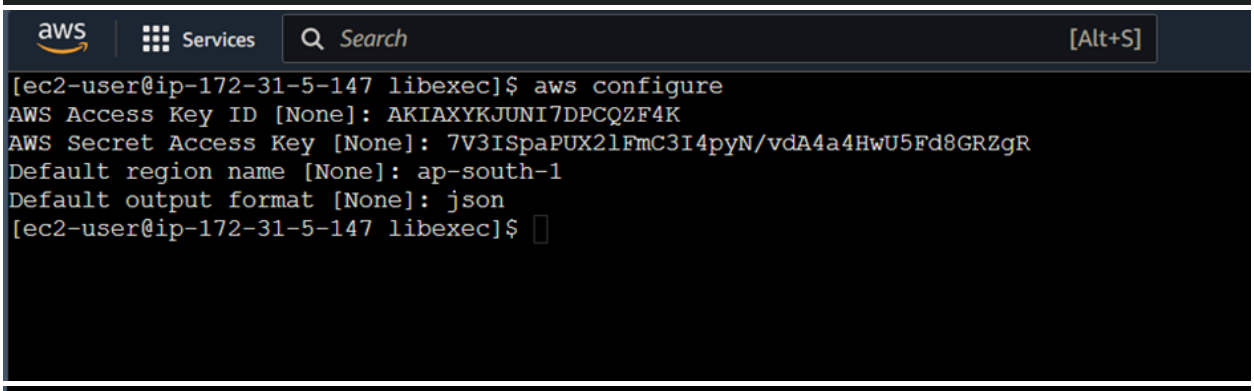
CloudWatchLogs | All types | 7 matches

Policy name	Type	Attached entities
AmazonAPIGatewayPushToCloudWatchLogs	AWS managed	0
AmazonDMSCloudWatchLogsRole	AWS managed	0
AWSAppSyncPushToCloudWatchLogs	AWS managed	0
AWSOpsWorksCloudWatchLogs	AWS managed	0
CloudWatchLogsCrossAccountSharingConfiguration	AWS managed	0
CloudWatchLogsFullAccess	AWS managed	0
CloudWatchLogsReadOnlyAccess	AWS managed	0

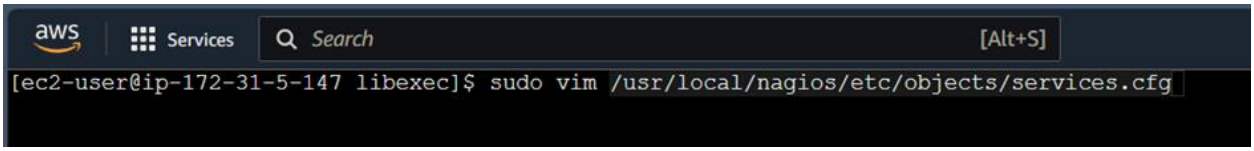
aws | Services | Search [Alt+S]

```python
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 --
```





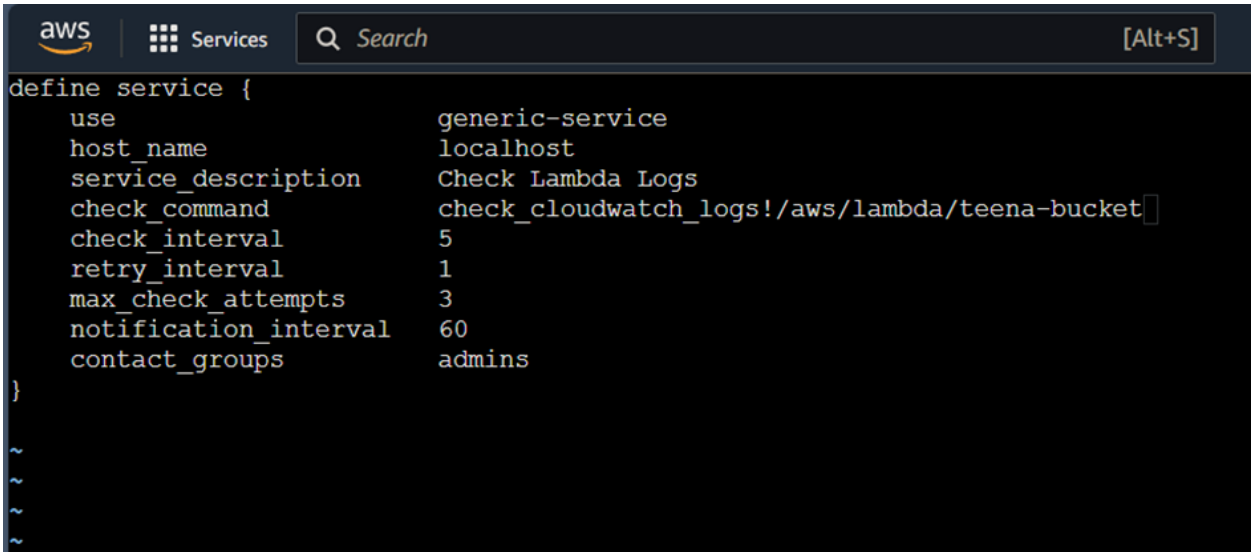

```
define command {
    command_name    check_cloudwatch_logs
    command_line    /usr/bin/python3 /usr/local/nagios/libexec/check_cloudwatch_logs.py $ARG1$
}
```



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



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.

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.