

Palak Chanchlani

Roll No : 05

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

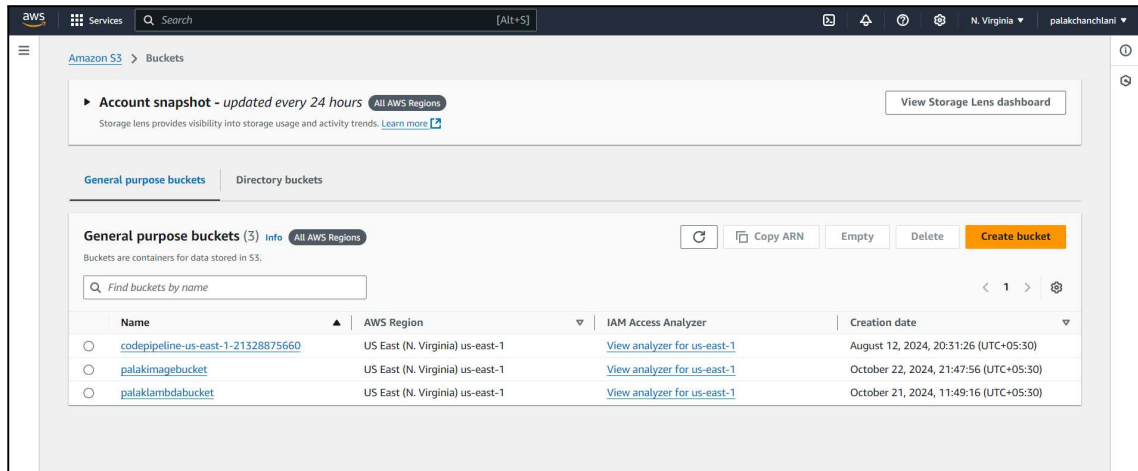
In the realm of modern application development, serverless computing has emerged as a game-changer, allowing developers to build applications without worrying about infrastructure management. AWS Lambda, a key player in serverless technology, enables the execution of code in response to various events, 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.

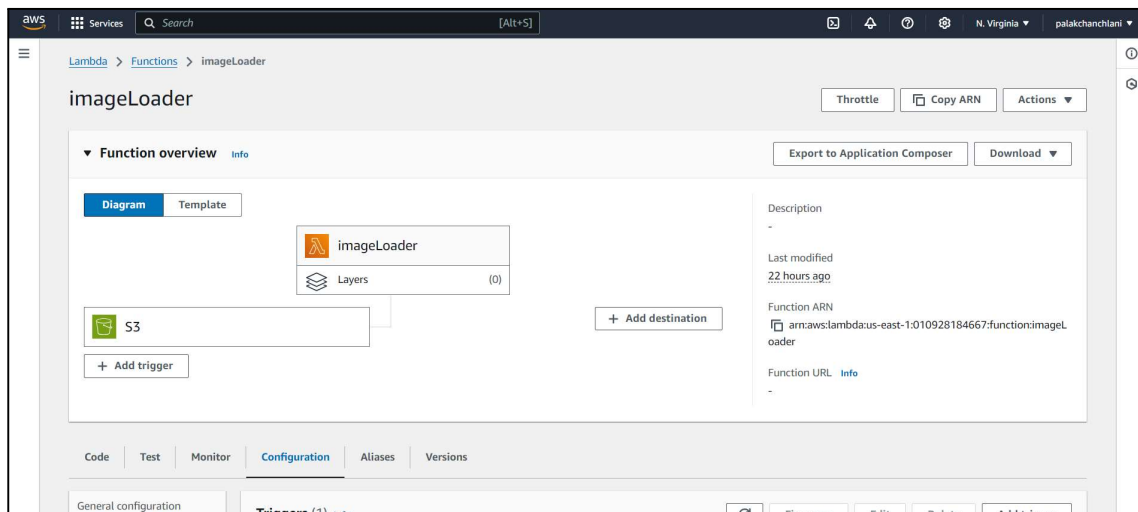
Application of Serverless and Monitoring

1. **Creation of an 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.

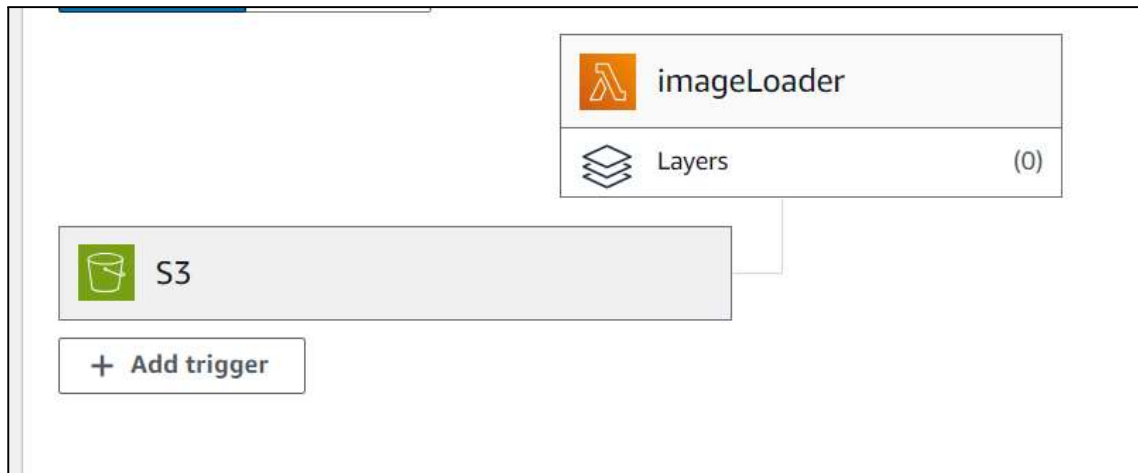


2. **Creation of Lambda Function** A Lambda function is established to respond to S3 upload events. A Python script is integrated within the function to log incoming events. The creation steps include:

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



3. **S3 Trigger for Lambda** The S3 bucket is configured to notify the Lambda function whenever a new image is uploaded. This involves navigating to the S3 bucket properties and adding a notification for Lambda event triggers.



Lambda Python Script The Lambda function's Python script utilizes the AWS SDK (Boto3) to capture S3 events. Below is the script that logs image uploads:

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)
```

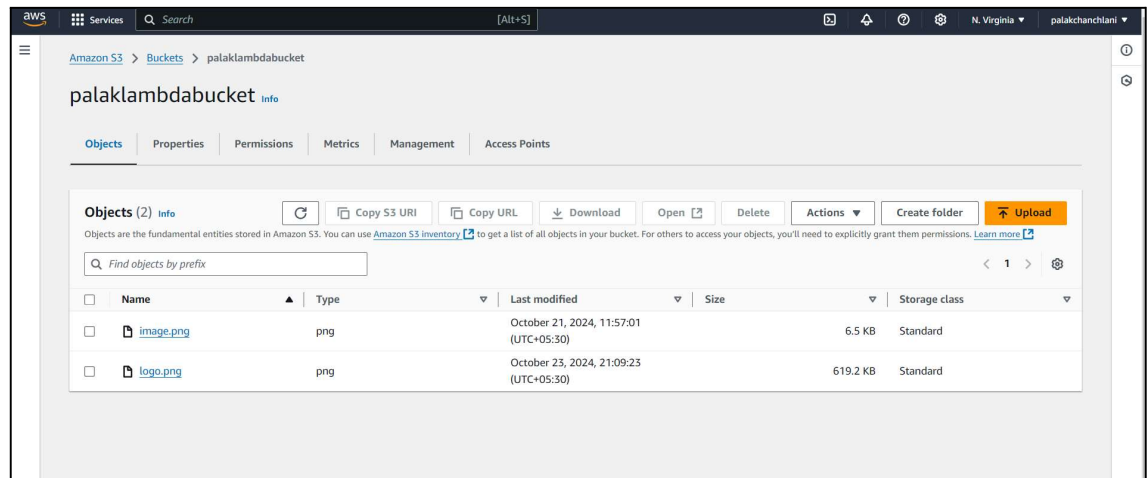
```
    return {
```

```
        'statusCode': 200,
```

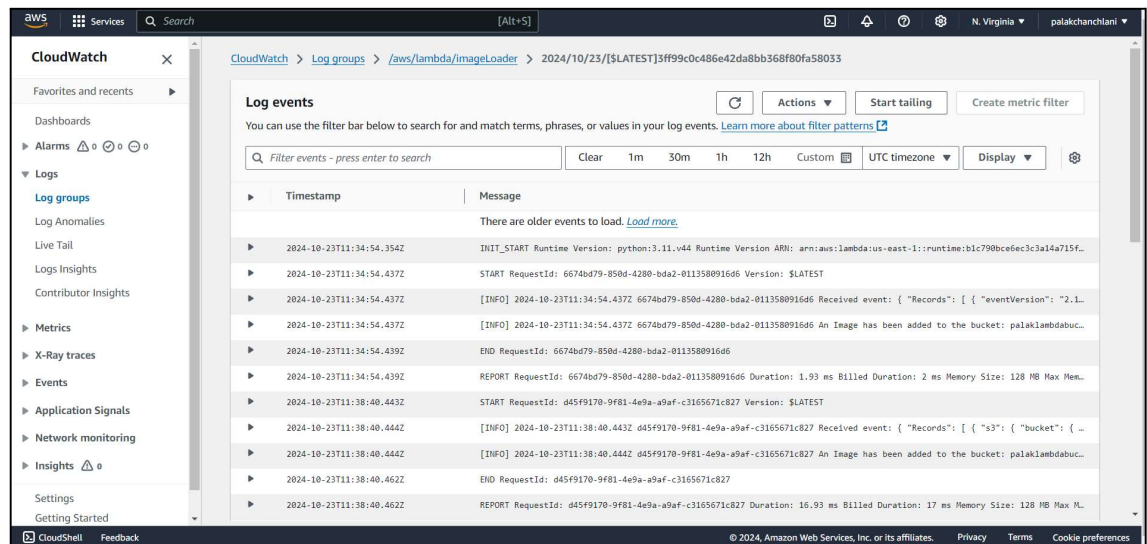
```
        'body': json.dumps('Lambda function executed successfully!')
```

```
    }
```

4. **Upload Image to S3** A test image (e.g., sample_image.jpg) is uploaded to the S3 bucket, triggering the Lambda function, which processes the event and logs the result.



5. **CloudWatch Logs** The Lambda function's execution is recorded in AWS CloudWatch. To verify, the CloudWatch console is accessed, and logs can be viewed under the appropriate log group corresponding to the Lambda function.



Monitoring Using Nagios on EC2

Nagios is a widely used open-source monitoring tool that provides monitoring capabilities for applications and infrastructure. This project involves setting up Nagios on an EC2 instance to monitor both the Lambda function and the S3 bucket.

1. **Nagios Setup** First, ensure that Nagios is properly installed and configured on the EC2 instance. This will involve verifying the installation and running the service.

Nagios Monitoring Script A shell script named `check_lambda.sh` is created in the Nagios `libexec` directory. This script checks the Lambda function's logs and verifies image uploads to the S3 bucket. Here is an example of what the script may look like:

```
#!/bin/bash
LAMBDA_FUNCTION_NAME="your-lambda-function-name"
S3_BUCKET_NAME="your-s3-bucket-name"
LOG_GROUP_NAME="/aws/lambda/$LAMBDA_FUNCTION_NAME"

# 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`)].Key' --output text | sort -r | head -n 1)

if [ -z "$LATEST_OBJECT" ]; then
    echo "No images found in the bucket."
    exit 1
else
    echo "Latest image uploaded: $LATEST_OBJECT"
fi

# Fetch logs of Lambda function execution
aws logs tail $LOG_GROUP_NAME --follow --max-items 10
```

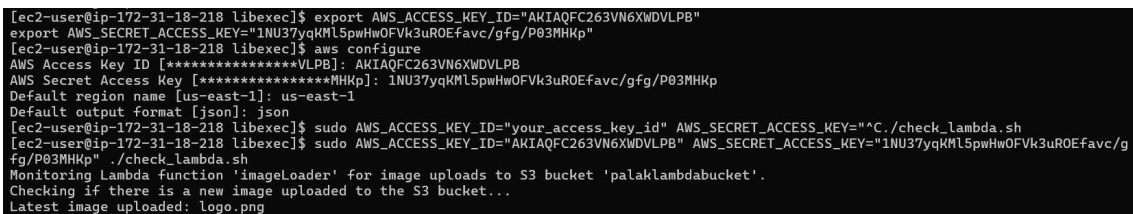
2. **Configure Nagios** After writing the script, it is added to Nagios as a new command and service. Modifications to the `commands.cfg` and `services.cfg` files are made to include monitoring for the Lambda function.

Monitor Lambda Function Run the monitoring script to check the status of the Lambda function and S3 bucket updates:

bash

Copy code

`./check_lambda.sh`



```
[ec2-user@ip-172-31-18-218 libexec]$ export AWS_ACCESS_KEY_ID="AKIAQFC263VN6XWDVLPB"
export AWS_SECRET_ACCESS_KEY="1NU37yqKM15pwHwOFV3uROEfavc/gfg/P03MHKp"
[ec2-user@ip-172-31-18-218 libexec]$ aws configure
AWS Access Key ID [*****]: AKIAQFC263VN6XWDVLPB
AWS Secret Access Key [*****]: 1NU37yqKM15pwHwOFV3uROEfavc/gfg/P03MHKp
Default region name [us-east-1]: us-east-1
Default output format [json]: json
[ec2-user@ip-172-31-18-218 libexec]$ sudo AWS_ACCESS_KEY_ID="your_access_key_id" AWS_SECRET_ACCESS_KEY=""^C./check_lambda.sh
[ec2-user@ip-172-31-18-218 libexec]$ sudo AWS_ACCESS_KEY_ID="AKIAQFC263VN6XWDVLPB" AWS_SECRET_ACCESS_KEY="1NU37yqKM15pwHwOFV3uROEfavc/gfg/P03MHKp" ./check_lambda.sh
Monitoring Lambda function 'imageLoader' for image uploads to S3 bucket 'palaklambabucket'.
Checking if there is a new image uploaded to the S3 bucket...
Latest image uploaded: logo.png
```

Theoretical Overview:

Serverless computing has transformed the way modern applications are developed and deployed, enabling developers to focus on writing code without the need to manage

underlying infrastructure. AWS Lambda is a key player in this serverless paradigm, providing a way to execute code in response to events such as HTTP requests, file uploads to S3, and more. When an image is uploaded to an S3 bucket, AWS Lambda can be triggered to execute a function that processes the event.

AWS Lambda: AWS Lambda is a serverless compute service that allows you to run code without provisioning or managing servers. It automatically scales your application by running code in response to events and only charges for the compute time consumed. This makes it ideal for use cases like event-driven image processing, data transformation, and automation tasks. By triggering a Lambda function with an S3 upload event, developers can automatically process uploaded images, analyze data, or update records.

AWS S3 (Simple Storage Service): S3 is a scalable, durable, and secure object storage service. It is commonly used to store files, such as images, videos, backups, and data logs. S3's event notification feature allows it to be integrated with Lambda, so that when a new object is added to an S3 bucket, an event is triggered, and Lambda can perform further actions like logging or processing the uploaded file.

CloudWatch for Logging: AWS CloudWatch is a monitoring and observability service that collects metrics and logs from AWS resources. In this case, CloudWatch captures the logs from the Lambda function's execution. It provides insights into the performance of the Lambda function, showing whether the function was triggered successfully and any errors or details from the code execution.

Nagios for Monitoring: Nagios is an open-source monitoring tool used to monitor system health, services, and applications. It provides a robust solution for alerting and monitoring, helping ensure that systems remain operational. In this use case, Nagios is configured to monitor the Lambda function's execution status and S3 bucket's activity, providing visibility into the performance and ensuring that the Lambda function responds appropriately to S3 events.

Integration and Monitoring Flow:

1. **Image Upload Trigger:** When an image file is uploaded to the S3 bucket, the event triggers the Lambda function.
2. **Lambda Execution:** The Lambda function executes a Python script that logs the event, checks if the uploaded object is an image, and records this information.
3. **CloudWatch Logging:** Logs from the Lambda execution are stored in AWS CloudWatch, providing a record of the event and the status of the Lambda function.
4. **Nagios Monitoring:** A custom script on Nagios monitors the S3 bucket for changes and checks the latest logs from CloudWatch to ensure that the Lambda function was triggered and executed correctly.
5. **Alerts and Insights:** Nagios generates alerts if any issues are detected, such as a failed Lambda execution or a delay in processing S3 uploads, enabling timely troubleshooting and ensuring system reliability.

Significance of Monitoring in Serverless Architecture: Monitoring is essential in serverless architectures because of the dynamic nature of event-driven workflows. With serverless, there are no servers to monitor directly, so it's crucial to keep an eye on function

invocations, response times, and error rates. This ensures that applications run smoothly and that potential issues are identified before they affect the end-user experience. Nagios complements AWS CloudWatch by providing an additional layer of monitoring, offering deeper insights and alerting capabilities, especially for complex workflows where multiple services interact.

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.