Expert Cloud Consulting

Enhance Optimise & Scale

ASCP GPUonCLOUD Pvt Ltd

“Expert Cloud Consulting” -

Documentation for Automating S3 and Lambda Integration Using AWS CloudFormation [Title,18, Arial]

15.May.2025 [Subtitle,14, Arial]

version 1.0

—

Contributed by Yogiraj Deshpande [Normal text,14, Arial]
Approved by Akshay Shinde(In Review)
Expert Cloud Consulting
Office #811, Gera Imperium Rise,
Hinjewadi Phase-II Rd, Pune, India – 411057

Week 5: Introduction to Infrastructure as Code (IaC)

Topics :

● Basics of Terraform and CloudFormation.

● Provisioning infrastructure on AWS.

Assignments:

● Use Terraform to create a multi-tier architecture:

● VPC, subnets, security groups.

● EC2 instances in private subnets.

● An internet-facing load balancer.

● Write a CloudFormation template to automate:

● Creation of an S3 bucket with versioning

● A Lambda function triggered by S3 events.

Resources:

● Getting Started with CloudFormation

● GitLab CI/CD: GitLab CI/CD Tutorial

“Expert Cloud Consulting”
Documentation for Automating S3 and Lambda Integration Using AWS CloudFormation[Title,18, Arial]

1.0: Document Overview:

This document describes the implementation of a CloudFormation template that provisions an AWS Lambda function triggered by events in an existing S3 bucket. It automates the creation of IAM roles, Lambda function configuration, and Lambda invoke permissions. Manual configuration of the S3 bucket notification is required post-deployment to complete the event trigger setup.

2.0: Objective:

- Provision infrastructure as code using AWS CloudFormation for consistent deployment.
- Create a Lambda function using code stored in S3.
- Assign necessary IAM roles and permissions to allow logging and S3 access.
- Allow the S3 bucket to invoke the Lambda function.
- Manually configure the S3 bucket to trigger the Lambda on object creation.

3.0: Prerequisites:

1. An AWS account with IAM permissions to use CloudFormation, Lambda, S3, and IAM services.
2. CloudFormation CLI or AWS Management Console access.
3. An existing S3 bucket (e.g., `my-ecc-lambda-bucket`) with a deployment package (`function.zip`) already uploaded.
4. Python-based Lambda function handler file: `lambda_function.py`.
5. Basic understanding of AWS Lambda, S3 event notifications, and IAM policies

4.0: Step-by-Step Implementation:

4.1: CloudFormation Template Overview:

File: `s3_lambda_template.yaml`:
Defines the following resources:

- IAM Role for Lambda execution (`AWS::IAM::Role`)
- Lambda function (`AWS::Lambda::Function`)
- Lambda invoke permission for S3 (`AWS::Lambda::Permission`)

4.2: IAM Role Definition:

```yaml
LambdaExecutionRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: s3-lambda-execution-role
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
      - Effect: Allow
        Principal:
          Service: lambda.amazonaws.com
        Action: sts:AssumeRole
    Policies:
    - PolicyName: LambdaS3LogsPolicy
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: Allow
          Action:
          - logs:CreateLogGroup
          - logs:CreateLogStream
          - logs:PutLogEvents
          - s3:GetObject
          Resource: "*"
```

Purpose: Allows Lambda to write logs to CloudWatch and read objects from S3.

4.3: Lambda Function Definition:

```
LambdaFunction:
  Type: AWS::Lambda::Function
  Properties:
    FunctionName: s3-event-handler-function
    Handler: lambda_function.lambda_handler
    Role: !GetAtt LambdaExecutionRole.Arn
    Runtime: python3.12
    Code:
      S3Bucket: my-ecc-lambda-bucket
      S3Key: function.zip
```

Purpose: Creates the Lambda function using a Python 3.12 runtime and code from the specified S3 bucket.

4.4: Lambda Invoke Permission:

```
LambdaInvokePermission:
  Type: AWS::Lambda::Permission
  Properties:
    FunctionName: !Ref LambdaFunction
    Action: lambda:InvokeFunction
    Principal: s3.amazonaws.com
    SourceArn: arn:aws:s3:::my-ecc-lambda-bucket
```

Purpose: Grants permission to S3 to invoke the Lambda function when an event occurs.

4.5: Outputs:

```
Outputs:

  LambdaFunctionName:

    Value: !Ref LambdaFunction
```

Purpose: Returns the name of the Lambda function as a CloudFormation output.

5.0: Adding `function.zip` to S3:

5.1: Create Lambda Code File:

- Create `lambda_function.py`:

```
import json

def lambda_handler(event, context):

    print("Received event:", json.dumps(event, indent=2))

    return {

        'statusCode': 200,

        'body': json.dumps('S3 trigger Lambda executed!')

    }
```

5.2: Zip the File:

Run the following command in your terminal:

```
zip function.zip lambda_function.py
```

5.3: Upload to S3:

```
aws s3 cp function.zip s3://my-ecc-lambda-bucket/
```

Make sure the bucket exists and the `S3Bucket` and `S3Key` in your CloudFormation template match.

6.0: Post-Deployment Manual Step:

S3 bucket notifications are not configurable via CloudFormation for existing buckets.

Manually configure S3 to trigger the Lambda:

- Go to S3 Console > my-ecc-lambda-bucket.
- Open Properties > Event Notifications.
- Click Create event notification.
- Name: `TriggerLambdaOnObjectCreate`.
- Event type: `All object create events`.

- Destination: Lambda Function → `s3-event-handler-function`.
- Save.

7.0: Deployment Steps:

7.1: Deploy Using AWS CLI:

```
aws cloudformation deploy \
  --template-file s3_lambda_template.yaml \
  --stack-name s3-lambda-stack \
  --capabilities CAPABILITY_NAMED_IAM
```

7.2: Validation Steps:

- Upload a file to `my-ecc-lambda-bucket`.
- Go to **CloudWatch Logs > Log Groups**.
- Check for logs under `/aws/lambda/s3-event-handler-function`.

8.0: Cleanup:

To delete the stack:

```
aws cloudformation delete-stack \
  --stack-name s3-lambda-stack
```

Ensure all manually attached resources (like S3 event notifications) are removed if the deletion is stuck.

9.0: Troubleshooting:

Issue	Cause	Resolution
DELETE_FAILED	Stack has external dependencies (e.g., S3 trigger not removed)	Remove S3 event notification manually
AccessDenied	Missing IAM permission for deletion	Attach `cloudformation:DeleteStack` permission to the user
Lambda not triggered	No S3 event notification	Add it manually as explained above

10.0: Conclusion:

This CloudFormation template offers a streamlined, repeatable approach to deploying AWS Lambda functions triggered by S3 events. By defining resources such as IAM roles, the Lambda function, and invoke permissions, it reduces manual setup and ensures consistency across environments.

While CloudFormation cannot directly manage event notifications for existing S3 buckets, this solution integrates well by requiring only a minor manual step post-deployment. This trade-off still allows for robust automation, especially useful in CI/CD pipelines or DevOps workflows.