

Advance Devops Practical Examination: AWS Case Study Assignment

Topic : Serverless Image Processing Workflow

1. Introduction

Concepts Used: *AWS Lambda, S3, Codebuild, and CodePipeline.*

Problem Statement: "Create a serverless workflow that triggers an AWS Lambda function when a new image is uploaded to an S3 bucket. Use CodePipeline to automate the deployment of the Lambda function."

Tasks:

- Create a Lambda function in Python that logs and processes an image when uploaded to a specific S3 bucket.
- Set up AWS CodePipeline to automatically deploy updates to the Lambda function.
- Upload a sample image to S3 and verify that the Lambda function is triggered and logs the event.

Theory:

1. 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 triggers, such as changes in data or HTTP requests, without you needing to worry about infrastructure. Lambda supports various languages (e.g., Python, Node.js, Java, etc.).

- **Key Features:**
 - **Event-driven:** Triggered by events (e.g., S3 uploads, API Gateway requests).
 - **Auto-scaling:** Scales based on demand.
 - **Pay-as-you-go:** Charges based on the number of requests and the compute time used.

2. Amazon S3 (Simple Storage Service)

Amazon S3 is an **object storage service** that allows you to store and retrieve any amount of data from anywhere. It is widely used for storing large amounts of unstructured data like images, videos, backups, and logs.

- **Key Features:**
 - **Durability & Scalability:** Highly durable and scalable for storing large data sets.
 - **Access Control:** Offers fine-grained access controls using policies and permissions.
 - **Integration:** Integrates with other AWS services like Lambda, CloudFront, and Glacier for archiving.

3. AWS CodePipeline

AWS CodePipeline is a **continuous integration and continuous delivery (CI/CD) service** that automates the software release process. It defines a workflow (pipeline) for building, testing, and deploying your code automatically.

- **Key Features:**
 - **Automation:** Automates code deployment after every change.
 - **Multi-stage Pipeline:** Supports stages like source control, build, and deploy.
 - **Integration:** Integrates with tools like GitHub, Jenkins, and AWS services like CodeBuild and Lambda.

4. AWS IAM (Identity and Access Management)

AWS IAM is a service that helps you manage **access to AWS resources** securely. It allows you to create users, groups, and roles and assign permissions to control who can access what.

- **Key Features:**
 - **Granular Permissions:** Control access to AWS services and resources at a fine level.
 - **Multi-factor Authentication (MFA):** Adds an extra layer of security for users.
 - **Roles:** Allows temporary access for services or users to perform specific tasks.

5.AWS CloudWatch:

AWS CloudWatch is a monitoring service that provides data and actionable insights to monitor applications, understand system-wide performance changes, and optimize resource usage. In this experiment, CloudWatch was used to capture logs from the Lambda function, allowing us to track S3 events and troubleshoot any issues that occurred during image uploads and processing.

- **Key Features:**

- Logs: Collects and stores logs from AWS services like Lambda.
- Metrics: Provides metrics based on the logs and AWS services.
- Alerts: Configures alarms based on metrics to monitor resources.

6. AWS CodeBuild:

AWS CodeBuild is a fully managed build service in AWS that compiles source code, runs tests, and produces software packages. In this experiment, CodeBuild was used as the build stage of the pipeline to package the Lambda function code and deploy it automatically using AWS CLI.

- **Key Features:**

- Continuous Integration: Automates the build and test phases.
- Scalable: Scales automatically to handle multiple builds.
- Integration: Easily integrates with other AWS services like CodePipeline and Lambda.

7. S3 Event Notifications:

S3 Event Notifications enable you to receive notifications when specific events occur in an S3 bucket. In this case study, S3 Event Notifications were used to trigger the Lambda function when an image was uploaded, ensuring the function is only invoked when necessary.

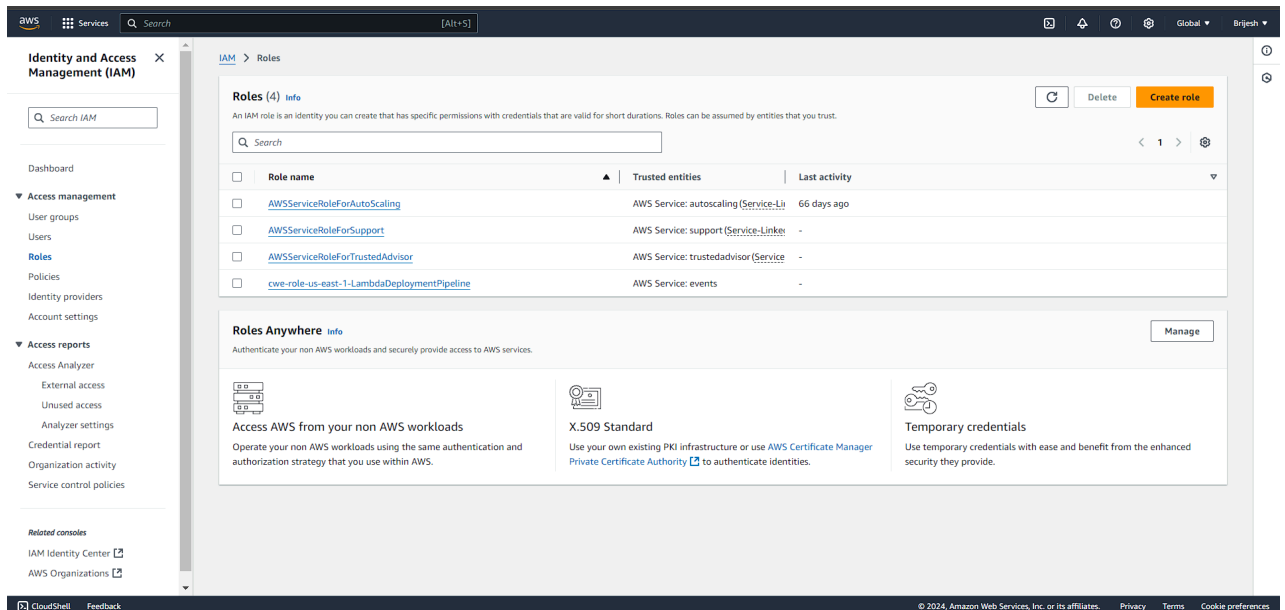
- **Key Features:**

- Event Triggers: Supports events like object creation, deletion, or replication.
- Destination: Can trigger Lambda functions, SNS, or SQS notifications.
- Granular Control: Can configure events to be triggered for specific objects or prefixes in the bucket.

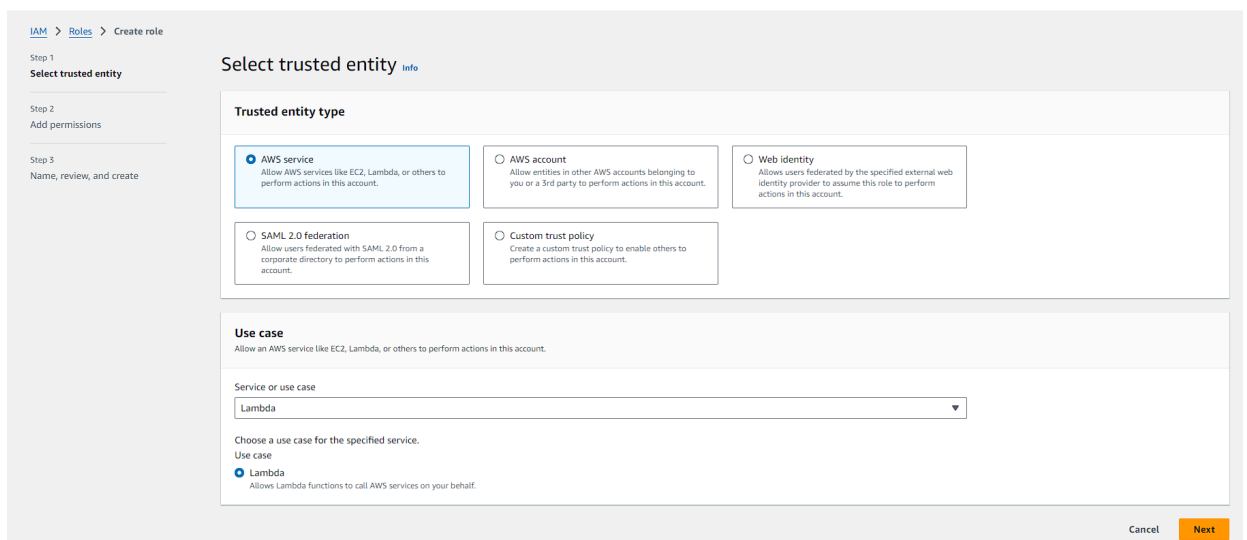
2. Step-by-Step Explanation

1. Create an IAM Role :

- Log in to the **AWS Management Console**.
- Go to the **IAM Console** in AWS.



- Click on **Roles** and choose **Create Role**.
- Select **AWS service** as the trusted entity type, and choose **Lambda** as the use case.



- Attach policies such as:
 - **AmazonS3FullAccess**

- **AWSCodeBuildAdminAccess**
- **AWSCodePipeline_FullAccess**
- **AWSLambda_FullAccess**
- **CloudWatchLogsFullAccess**
- Give the role a name: **MyCaseStudyRole48** and create it.

IAM > Roles > Create role

Step 1
[Select trusted entity](#)

Step 2
[Add permissions](#)

Step 3
Name, review, and create

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and "+-@_." characters.

Description
Add a short explanation for this role.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+~@-/\[\]\#\%*^&""

Step 1: Select trusted entities Edit

Trust policy

```

1- {
2-   "Version": "2012-10-17",
3-   "Statement": [
4-     {
5-       "Effect": "Allow",
6-       "Action": [
7-         "sts:AssumeRole"
8-       ],
9-       "Principal": {
10-        "Service": [
11-          "lambda.amazonaws.com"
12-        ]
13-      }
14-    ]
15-  }

```

Step 2: Add permissions Edit

Permissions policy summary

Policy name	Type	Attached as
AmazonS3FullAccess	AWS managed	Permissions policy
AWSCodeBuildAdminAccess	AWS managed	Permissions policy
AWSCodePipeline_FullAccess	AWS managed	Permissions policy
AWSLambda_FullAccess	AWS managed	Permissions policy
CloudWatchLogsFullAccess	AWS managed	Permissions policy

Step 3: Add tags

Add tags - optional [Info](#)

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel Previous Create role

- After creating the role, in Trust Relationship/Policy attach below policy in Statement.

```

{
  "Effect": "Allow",
  "Principal": {
    "Service": "codepipeline.amazonaws.com"
  }
}

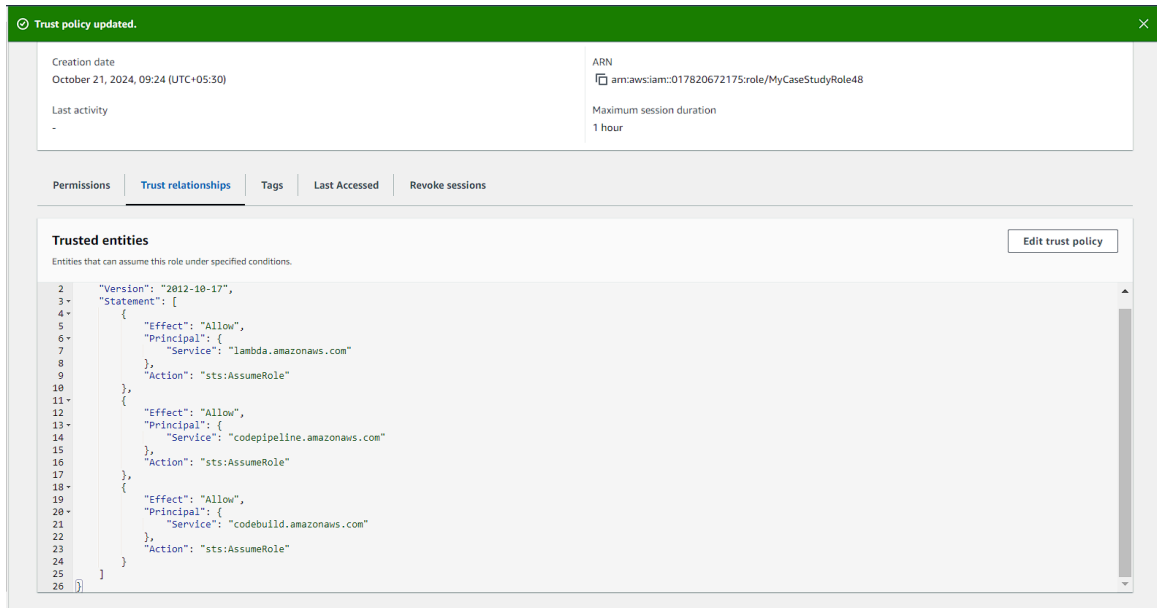
```

```
    },  
    "Action": "sts:AssumeRole"  
  },  
  {  
    "Effect": "Allow",  
    "Principal": {  
      "Service": "codebuild.amazonaws.com"  
    },  
    "Action": "sts:AssumeRole"  
  }  
}
```

[IAM](#) > [Roles](#) > [MyCaseStudyRole48](#) > Edit trust policy

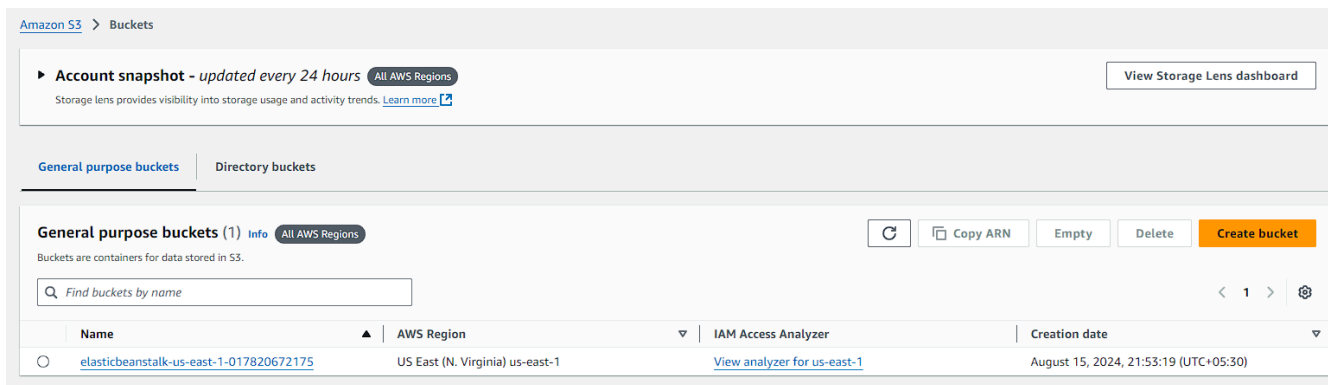
Edit trust policy

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Principal": {  
7         "Service": "lambda.amazonaws.com"  
8       },  
9       "Action": "sts:AssumeRole"  
10    },  
11    {  
12      "Effect": "Allow",  
13      "Principal": {  
14        "Service": "codepipeline.amazonaws.com"  
15      },  
16      "Action": "sts:AssumeRole"  
17    },  
18    {  
19      "Effect": "Allow",  
20      "Principal": {  
21        "Service": "codebuild.amazonaws.com"  
22      },  
23      "Action": "sts:AssumeRole"  
24    }  
25  ]  
26 }
```



2. Create an S3 Bucket:

- Navigate to the **S3** service.



- Click on **Create Bucket**.
 - Bucket Type: **General Purpose**.
 - Provide a unique bucket name. (**mycasestudybucket48**)
 - **Uncheck the Block all public access option**.
 - Keep Rest of the things to default.

Amazon S3

>

Buckets

>

Create bucket

Create bucket

Info

Buckets are containers for data stored in S3.

General configuration

AWS Region

US East (N. Virginia) us-east-1

Bucket type

Info

General purpose

Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory

Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name

Info

mycasestudybucket48

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - optional

Only the bucket settings in the following configuration are copied.

Choose bucket

Format: s3://bucket/prefix

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

Block public access to buckets and objects granted through new access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

Block public access to buckets and objects granted through any access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

Block public access to buckets and objects granted through new public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

Block public and cross-account access to buckets and objects through any public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Turning off block all public access might result in this bucket and the objects within becoming public

AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Successfully created bucket "mycasestudybucket48"

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

View Storage Lens dashboard

Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

General purpose buckets

Directory buckets

General purpose buckets (2)

Info

All AWS Regions

Copy ARN

Empty

Delete

Create bucket

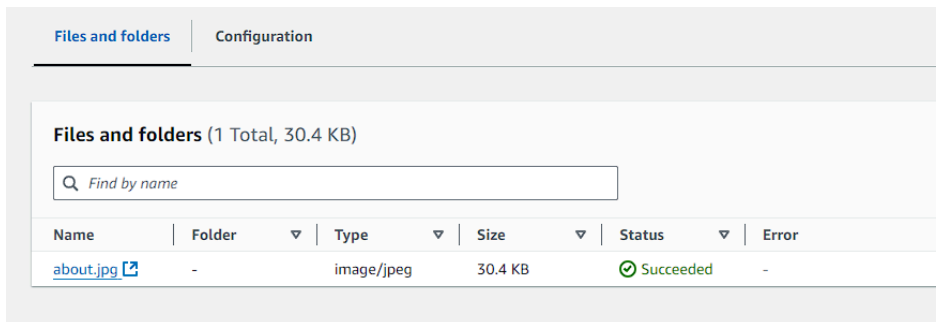
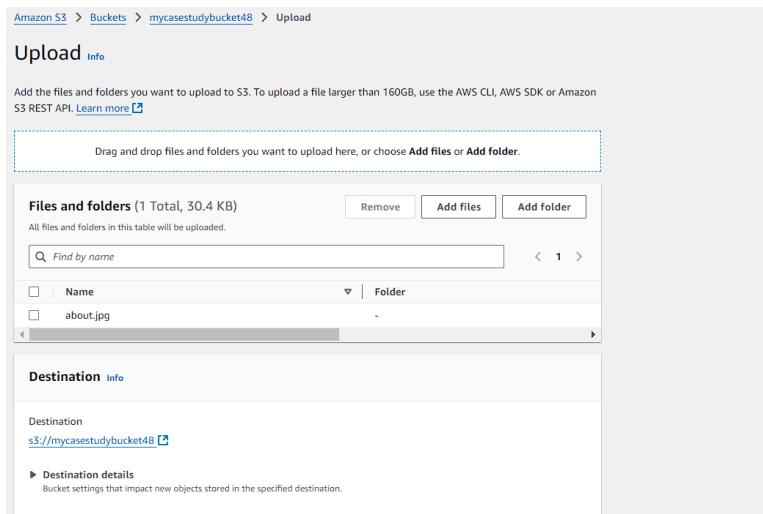
Buckets are containers for data stored in S3.

Find buckets by name

< 1 >

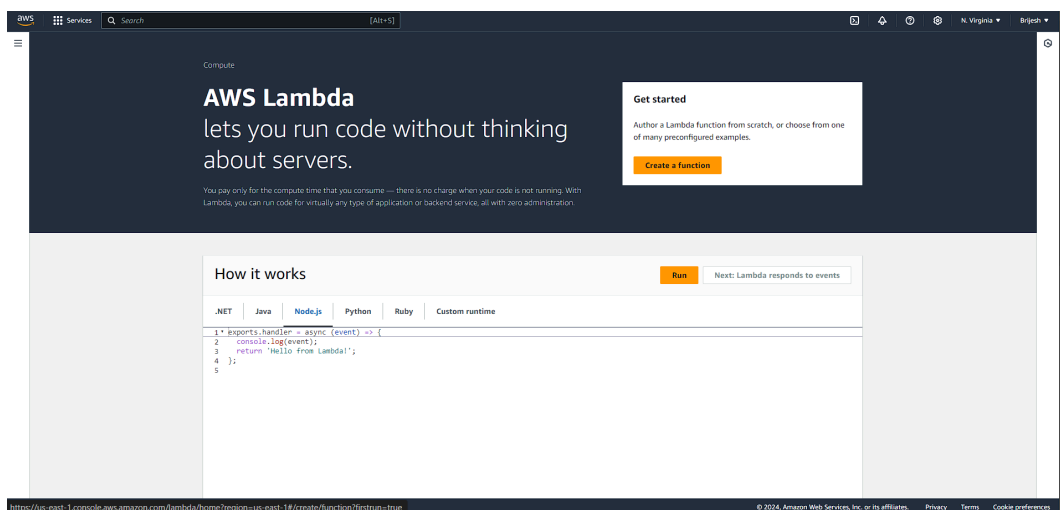
Name	AWS Region	IAM Access Analyzer	Creation date
<div><div></div><div>elasticbeanstalk-us-east-1-017820627175</div></div>	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 15, 2024, 21:53:19 (UTC+05:30)
<div><div></div><div>mycasestudybucket48</div></div>	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 21, 2024, 09:33:46 (UTC+05:30)

- You can upload a test image to ensure the bucket is working properly.



3. Create a Lambda Function:

- Navigate to the **Lambda Console**.



- Click **Create Function**.
 - Choose **Author from scratch**.
 - Provide a function name **ImageProcessorLambda**.
 - Choose a free-tier eligible runtime **Python 3.12**.
 - Assign the **IAM role** created earlier **MyCaseStudyRole48**.
 - Keep Rest of things to default.

Lambda > Functions > Create function

Create function [Info](#)

Choose one of the following options to create your function.

☒ **Author from scratch**
Start with a simple Hello World example.

☐ **Use a blueprint**
Build a Lambda application from sample code and configuration presets for common use cases.

☐ **Container image**
Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.

ImageProcessorLambda

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.12

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.

☒ x86_64
☐ arm64

Permissions [Info](#)

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ **Change default execution role**

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions
☒ Use an existing role
☐ Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

MyCaseStudyRole48

[View the MyCaseStudyRole48 role](#) on the IAM console.

► Additional Configurations

Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

Cancel Create function

- In the **Function Code** section, add the logic to log when the image is uploaded to S3.

Code:

```
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
        # Log the event details
        logger.info(f"Received event: {json.dumps(event)}")

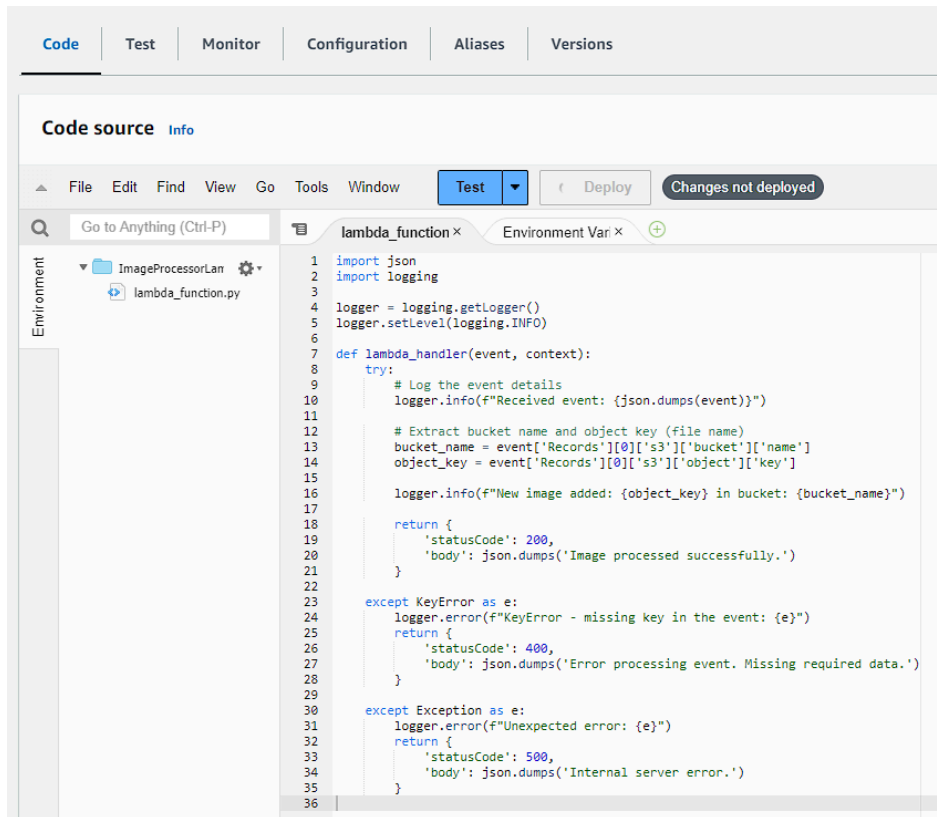
        # Extract bucket name and object key (file name)
        bucket_name = event['Records'][0]['s3']['bucket']['name']
        object_key = event['Records'][0]['s3']['object']['key']

        logger.info(f"New image added: {object_key} in bucket: {bucket_name}")

        return {
            'statusCode': 200,
            'body': json.dumps('Image processed successfully.')
        }

    except KeyError as e:
        logger.error(f"KeyError - missing key in the event: {e}")
        return {
            'statusCode': 400,
            'body': json.dumps('Error processing event. Missing required data.')
        }

    except Exception as e:
        logger.error(f"Unexpected error: {e}")
        return {
            'statusCode': 500,
            'body': json.dumps('Internal server error.')
        }
```



```
1 import json
2 import logging
3
4 logger = logging.getLogger()
5 logger.setLevel(logging.INFO)
6
7 def lambda_handler(event, context):
8     try:
9         # Log the event details
10        logger.info(f"Received event: {json.dumps(event)}")
11
12        # Extract bucket name and object key (file name)
13        bucket_name = event['Records'][0]['s3']['bucket']['name']
14        object_key = event['Records'][0]['s3']['object']['key']
15
16        logger.info(f"New image added: {object_key} in bucket: {bucket_name}")
17
18        return {
19            'statusCode': 200,
20            'body': json.dumps('Image processed successfully.')
21        }
22
23    except KeyError as e:
24        logger.error(f"KeyError - missing key in the event: {e}")
25        return {
26            'statusCode': 400,
27            'body': json.dumps('Error processing event. Missing required data.')
28        }
29
30    except Exception as e:
31        logger.error(f"Unexpected error: {e}")
32        return {
33            'statusCode': 500,
34            'body': json.dumps('Internal server error.')
35        }
36
```

- Now Add the **Trigger** (Click on Trigger).
 - In Trigger Configuration Select S3.
 - Select Bucket which we have created.(**mycasestudybucket48**)
 - Select Event Types
 - All object create events
 - PUT
 - POST
 - COPY
 - Multipart upload completed

Trigger configuration [Info](#)

S3

aws

asynchronous

storage

Bucket

Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.



s3/mycasestudybucket48



Bucket region: us-east-1

Event types

Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.



All object create events

PUT

POST

COPY

Multipart upload completed

Prefix - optional

Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any [special characters](#) must be URL encoded.

e.g. images/

Suffix - optional

Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any [special characters](#) must be URL encoded.

e.g. .jpg

Recursive invocation

If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)

☒ I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

ImageProcessorLambda

Throttle

Copy ARN

Actions

The trigger mycasestudybucket48 was successfully added to function ImageProcessorLambda. The function is now receiving events from the trigger.

Function overview [Info](#)

Export to Application Composer

Download

Diagram

Template



ImageProcessorLambda



Layers

(0)



S3

Add trigger

Add destination

Description

Last modified

12 minutes ago

Function ARN

arn:aws:lambda:us-east-1:017820672175:function:ImageProcessorLambda

Function URL [Info](#)

- Now Let's test that Lambda is properly working and check logs.
- Go to the bucket and upload your image.

Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (1 Total, 30.4 KB)

Remove

Add files

Add folder

All files and folders in this table will be uploaded.

< 1 >

<input type="checkbox"/>	Name	Folder
<input type="checkbox"/>	about.jpg	-

Destination Info

Destination

[s3://mycasestudybucket48](#)

► Destination details

Bucket settings that impact new objects stored in the specified destination.

► Permissions

Grant public access and access to other AWS accounts.

► Properties

Specify storage class, encryption settings, tags, and more.

Cancel

Upload

Summary											
Destination		Succeeded			Failed						
s3://mycasestudybucket48		✔ 1 file, 30.4 KB (100.00%)			☹ 0 files, 0 B (0%)						
Files and folders											
Configuration											
Files and folders (1 Total, 30.4 KB)											
<input type="text" value="Find by name"/>											
Name	Folder	Type	Size	Status	Error						
about.jpg	-	image/jpeg	30.4 KB	✔ Succeeded	-						

The screenshot shows the AWS CloudWatch console. The top navigation bar includes 'Log streams', 'Tags', 'Anomaly detection', 'Metric filters', 'Subscription filters', 'Contributor Insights', and 'Data protection'. The 'Log streams' section is active, showing a list of log streams. Below the list, there's a search bar and buttons for 'Filter log streams or try prefix search', 'Exact match', 'Show expired', and 'Info'. The log stream '2024/10/21/[\$LATEST]ca7e5238db114b7d982fcbfb49b7f37d' is selected. The 'Log events' section is also visible, showing a list of log events with columns for 'Timestamp' and 'Message'. The events include 'INIT_START', 'START', '[INFO]', and 'END' messages.

Log streams (1)

Filter log streams or try prefix search ☐ Exact match ☐ Show expired [Info](#)

☐ Log stream ☐ Last event time

☐ 2024/10/21/[\$LATEST]ca7e5238db114b7d982fcbfb49b7f37d 2024-10-21 04:32:58 (UTC)

Log events

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](#)

Filter events - press enter to search Clear 1m 30m 1h 12h Custom UTC timezone Display

Timestamp	Message
	No older events at this moment. Retry
2024-10-21T04:32:58.695Z	INIT_START Runtime Version: python:3.12-v36 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:188d9ca2e2714ff5637bd22be0e0cebb81ec3bc408a0f277da0104c14cd814b081
2024-10-21T04:32:58.792Z	START RequestId: 81b3cf41-9f8f-4af1-812a-454bc204458e Version: \$LATEST
2024-10-21T04:32:58.793Z	[INFO] 2024-10-21T04:32:58.793Z 81b3cf41-9f8f-4af1-812a-454bc204458e Received event: {"Records": [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "us-east-1", "eventTime": "2024-10-21T04:32:58.793Z", "eventName": "ObjectCreated", "eventSize": 1024, "eventSourceARN": "arn:aws:s3:::mycasestudybucket40", "eventSourceAccount": "123456789012"}]}
2024-10-21T04:32:58.793Z	[INFO] 2024-10-21T04:32:58.793Z 81b3cf41-9f8f-4af1-812a-454bc204458e New image added: about.jpg in bucket: mycasestudybucket40
2024-10-21T04:32:58.795Z	END RequestId: 81b3cf41-9f8f-4af1-812a-454bc204458e
2024-10-21T04:32:58.795Z	REPORT RequestId: 81b3cf41-9f8f-4af1-812a-454bc204458e Duration: 2.63 ms Billed Duration: 3 ms Memory Size: 128 MB Max Memory Used: 32 MB Init Duration: 93.71 ms
	No newer events at this moment. Auto retry paused. Resume

4. Create a Github Repository:

- Create Repository with name **CodePipeline-48**.
- Add the file **buildspec.yml** and **lambda_function.py**.

buildspec.yml

version: 0.2

phases:

install:

commands:

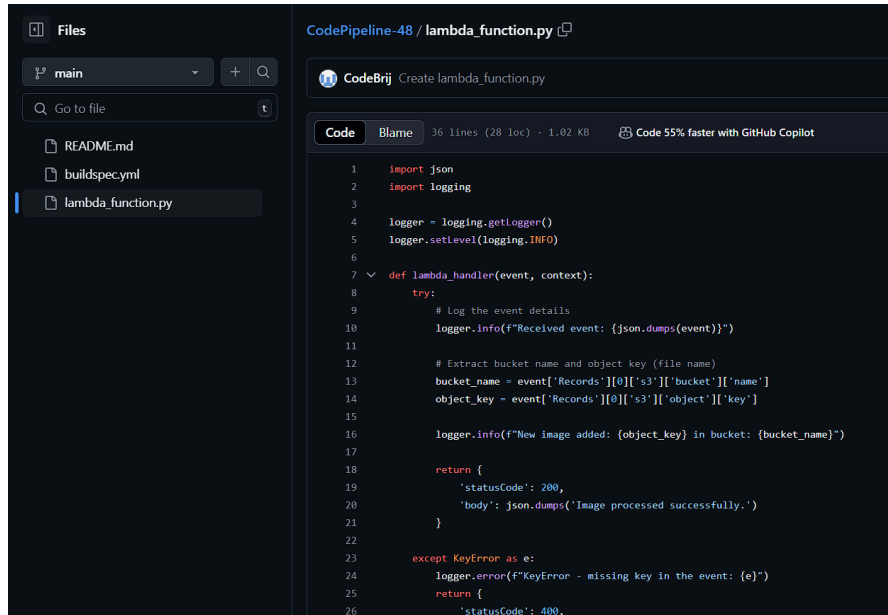
- pip install awscli # Ensure AWS CLI is available

build:

commands:

- echo "Packaging Lambda function..."
- zip -r lambda_function.zip .
- echo "Updating Lambda function in AWS..."
- aws lambda update-function-code --function-name ImageProcessorLambda --zip-file fileb://lambda_function.zip

lambda_function.py Same as in step 3



The screenshot shows the AWS CodePipeline console interface. On the left, a 'Files' sidebar lists 'main', 'README.md', 'buildspec.yml', and 'lambda_function.py'. The main area displays the 'lambda_function.py' file 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     try:
9         # Log the event details
10        logger.info(f"Received event: {json.dumps(event)}")
11
12        # Extract bucket name and object key (file name)
13        bucket_name = event['Records'][0]['s3']['bucket']['name']
14        object_key = event['Records'][0]['s3']['object']['key']
15
16        logger.info(f"New image added: {object_key} in bucket: {bucket_name}")
17
18        return {
19            'statusCode': 200,
20            'body': json.dumps('Image processed successfully.')
21        }
22
23    except KeyError as e:
24        logger.error(f"KeyError - missing key in the event: (e)")
25        return {
26            'statusCode': 400,
```



The screenshot shows the AWS CodePipeline console interface for the 'buildspec.yml' file. The 'Files' sidebar lists 'main', 'README.md', 'buildspec.yml', and 'lambda_function.py'. The main area displays the 'buildspec.yml' file with the following content:

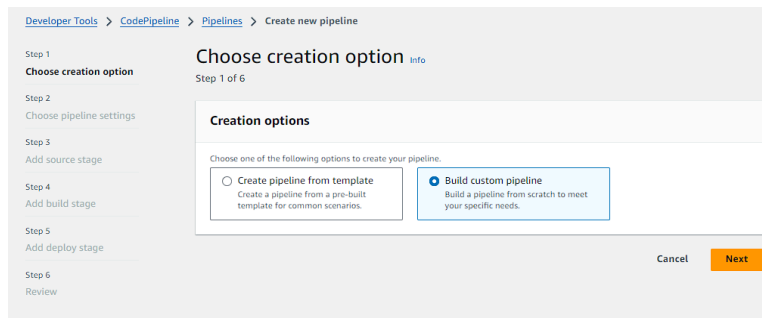
```
1 version: 0.2
2 phases:
3   install:
4     commands:
5       - pip install awscli # Ensure AWS CLI is available
6   build:
7     commands:
8       - echo "Packaging Lambda function..."
9       - zip -r lambda_function.zip .
10      - echo "Updating Lambda function in AWS..."
11      - aws lambda update-function-code --function-name ImageProcessorLambda --zip-file fileb://lambda_function.zip
```

5. Create a CodePipeline:

- Go to **AWS CodePipeline** in the AWS Management Console and create a new pipeline.

Step 1: Choose Creation option.

- In creation option Select **Build Custom pipeline**.



Step 2: Choose Pipeline Settings.

- Pipeline Name: **ImageDetectorPipeline** .
- Select Existing Role : **MyCaseStudyRole48** .
- Keep Rest all to default.

Choose pipeline settings [Info](#)

Step 2 of 6

Pipeline settings

Pipeline name
Enter the pipeline name. You cannot edit the pipeline name after it is created.

MyCaseStudyPipeline

No more than 100 characters

Pipeline type

☒ You can no longer create V1 pipelines through the console. We recommend you use the V2 pipeline type with improved release safety, pipeline triggers, parameterized pipelines, and a new billing model.

Execution mode
Choose the execution mode for your pipeline. This determines how the pipeline is run.

☐ Superseded
A more recent execution can overtake an older one. This is the default.

☒ Queued (Pipeline type V2 required)
Executions are processed one by one in the order that they are queued.

☐ Parallel (Pipeline type V2 required)
Executions don't wait for other runs to complete before starting or finishing.

Service role

☐ New service role
Create a service role in your account

☒ Existing service role
Choose an existing service role from your account

Role ARN

arn:aws:iam::017820672175:role/MyCaseStudyRole48

Step 3: Add Source Storage.

- Source provider: **GitHub (Version 2)** .
- Connect Your account where you have created the repository. (**CodeBrij**)
- Select Repository. (**CodeBrij/CodePipeline-48**)
- Select Branch to **main**.
- In Trigger Just add **main** in include.
- Keep Rest all to default.

Step 1

Choose creation option

Step 2

Choose pipeline settings

Step 3

Add source stage

Step 4

Add build stage

Step 5

Add deploy stage

Step 6

Review

Add source stage info

Step 3 of 6

Source

Source provider

This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

GitHub (Version 2) ▼

New GitHub version 2 (app-based) action

To add a GitHub version 2 action in CodePipeline, you create a connection, which uses GitHub Apps to access your repository. Use the options below to choose an existing connection or create a new one. [Learn more](#)

Connection

Choose an existing connection that you have already configured, or create a new one and then return to this task.

arn:aws:codeconnections:us-east-1:017820672175:connection/686901d3-74 X

 or

Connect to GitHub

Repository name

Choose a repository in your GitHub account.

CodeBrij/CodePipeline-48 X

You can type or paste the group path to any project that the provided credentials can access. Use the format 'group/subgroup/project'.

Default branch

Default branch will be used only when pipeline execution starts from a different source or manually started.

main X

Output artifact format

Choose the output artifact format.

☒ **CodePipeline default**

☐ **Full clone**

AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.

AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions.

☒ Enable automatic retry on stage failure

Trigger

Trigger type

Choose the trigger type that starts your pipeline.

☐ No filter
Starts your pipeline on any push and clones the HEAD.

☒ Specify filter
Starts your pipeline on a specific filter and clones the exact commit. Pipeline type V2 is required.

☐ Do not detect changes
Don't automatically trigger the pipeline.

Event type

Choose the event type for the trigger that starts your pipeline.

☒ Push

☐ Pull request

Filter type

Choose the filter type for the event that starts your pipeline.

☒ Branch

☐ Tags

Branches

You can specify the target branch or branches you are pushing to. Use a comma to specify multiple entries.

Include

main

Exclude

Enter branches or patterns

File paths - optional

You can specify the file path or file paths you are pushing to. Use a comma to separate multiple entries.

Include

Enter file paths or patterns

Exclude

Enter file paths or patterns

Step 4: Add Build Stage.

- Select **Other build Provider**.
- Select **AWS CodeBuild**.
 - Now Click on **Create Project**.
 - Project name: **Image_Detector_Build**
 - Enable public access.
 - Select Existing Role : **AWS_Case_Study** On 2 Places.
 - Buildspec: **Use a buildspec.yml file** .
 - Buildspec name: **buildspec.yml** .
 - Keep Rest all to Default.
- Select Created Project.
- Keep Rest all to default.

Project configuration

Project name

A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and _.

Public build access - *optional*

Public build access allows you to make the build results, including logs and artifacts, for this project available for the general public.

☒ Enable public build access**Public build access enabled**

Your build results, including logs and artifacts, are accessible to the general public. Downloading logs and/or artifacts will increase your AWS costs. [Learn more](#)

Public build service role

The public build service role is used to provide read access to your logs and artifacts for public builds. You can let CodeBuild create a new role, or you can choose an existing role.



New service role

Create a service role in your account



Existing service role

Choose an existing service role from your account

Service role

☒ Allow AWS CodeBuild to modify this service role so it can be used

Environment

Provisioning model [Info](#)

- ☒ **On-demand**
Automatically provision build infrastructure in response to new builds.

- ☐ **Reserved capacity**
Use a dedicated fleet of instances for builds. A fleet's compute and environment type will be used for the project.

Environment image

- ☒ **Managed image**
Use an image managed by AWS CodeBuild

- ☐ **Custom image**
Specify a Docker image

Compute

- ☒ **EC2**
Optimized for flexibility during action runs

- ☐ **Lambda**
Optimized for speed and minimizes the start up time of workflow actions

Operating system

Amazon Linux

Runtime(s)

Standard

Image

aws/codebuild/amazonlinux2-x86_64-standard:5.0

Image version

Always use the latest image for this runtime version

Buildspec

Build specifications

- ☐ **Insert build commands**
Store build commands as build project configuration

- ☒ **Use a buildspec file**
Store build commands in a YAML-formatted buildspec file

Buildspec name - optional

By default, CodeBuild looks for a file named buildspec.yml in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root here (for example, buildspec-two.yml or configuration/buildspec.yml).

buildspec.yml

Build - optional

Build provider
Choose the tool you want to use to run build commands and specify artifacts for your build action.

☐ Commands

☒ Other build providers

AWS CodeBuild ▼

Project name
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

X or

✔ Successfully created Image_Detector_Build in CodeBuild. X

Environment variables - optional
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

Build type

☒ **Single build**
Triggers a single build.

☐ **Batch build**
Triggers multiple builds as a single execution.

Region

US East (N. Virginia) ▼

Input artifacts
Choose an input artifact for this action. [Learn more](#)

SourceArtifact X
Defined by: Source

No more than 100 characters

☒ Enable automatic retry on stage failure

Step 5: Skip The Deploy Stage.

Step 6: Review the Pipeline and Click on Create.

Step 1: Choose pipeline settings**Pipeline settings**

Pipeline name
MyCaseStudyPipeline

Pipeline type
V2

Execution mode
QUEUED

Artifact location
A new Amazon S3 bucket will be created as the default artifact store for your pipeline

Service role name
arn:aws:iam::017820672175:role/MyCaseStudyRole48

Variables

Name	Default value	Description
------	---------------	-------------

No variables

No variables defined at the pipeline level in this pipeline.

Trigger configuration

You can add additional pipeline triggers after the pipeline is created.

Trigger type
Specify filter

Event type
Push

Filter type
Branch

Include branches
main

Exclude branches
-

Include file paths
-

Exclude file paths
-

Step 3: Add build stage

Build action provider

Build action provider

AWS CodeBuild

ProjectName

Image_Detector_Build

Commands

-

Enable automatic retry on stage failure

Enabled

Step 4: Add deploy stage

Deploy action provider

Deployment stage

No deploy

Cancel

Previous

Create pipeline

Success

Stage Build successfully retried

Developer Tools > CodePipeline > Pipelines > MyCaseStudyPipeline

MyCaseStudyPipeline

Pipeline type: V2 Execution mode: QUEUED

Notify

Edit

Stop execution

Clone pipeline

Release change

Source

Succeeded

Pipeline execution ID: 2e063a13-4d0a-42a3-b018-52ef09387d25

Source

[Github \(Version 2\)](#)

Succeeded - 9 minutes ago

18624f4e

View details

[18624f4e](#) Source: Update buildspec.yml

Disable transition

Build

Succeeded

Manual retry attempt

View retry metadata

Pipeline execution ID: 2e063a13-4d0a-42a3-b018-52ef09387d25

Build

[AWS CodeBuild](#)

Succeeded - Just now

View details

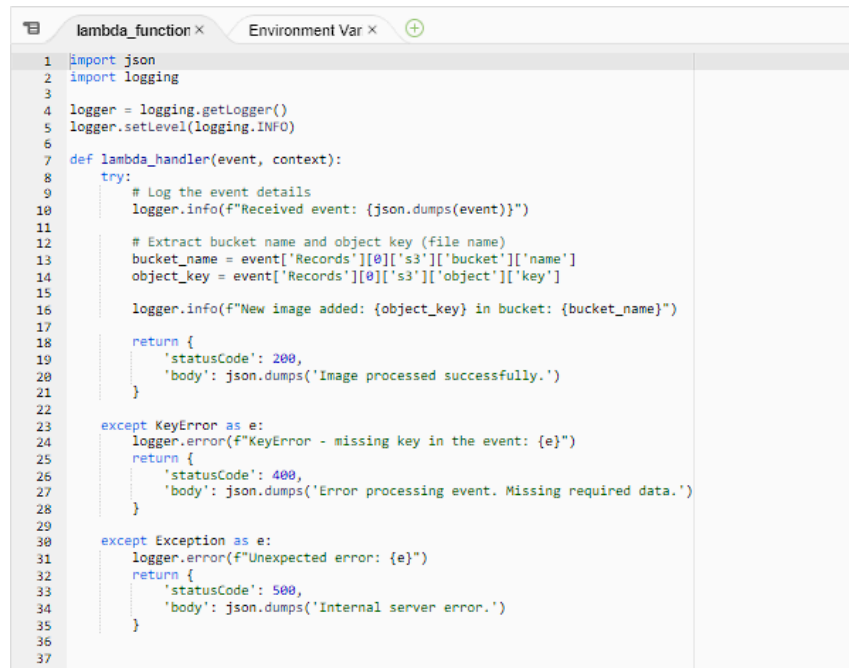
[18624f4e](#) Source: Update buildspec.yml

Start rollback

6.Test the Pipeline:

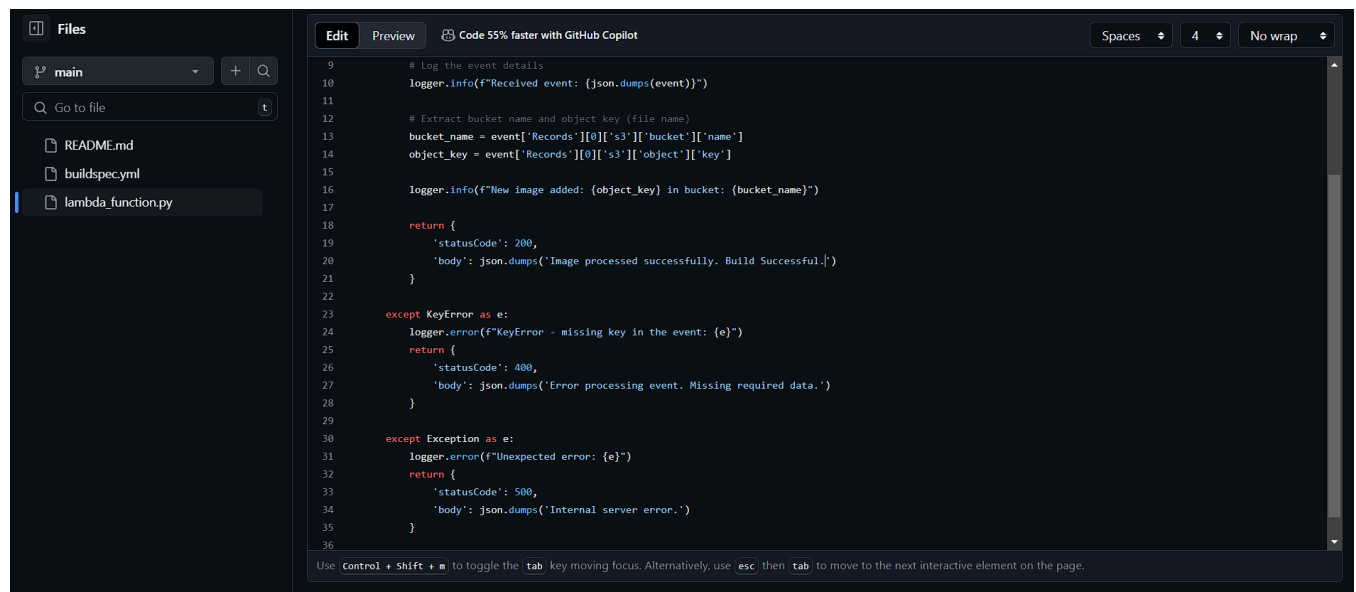
- Push/Update code to your GitHub repository.
- CodePipeline will trigger the CodeBuild project, which will package the code and update the Lambda function using the AWS CLI command `aws lambda update-function-code`.

Lambda function before changes



```
1 import json
2 import logging
3
4 logger = logging.getLogger()
5 logger.setLevel(logging.INFO)
6
7 def lambda_handler(event, context):
8     try:
9         # Log the event details
10        logger.info(f"Received event: {json.dumps(event)}")
11
12        # Extract bucket name and object key (file name)
13        bucket_name = event['Records'][0]['s3']['bucket']['name']
14        object_key = event['Records'][0]['s3']['object']['key']
15
16        logger.info(f"New image added: {object_key} in bucket: {bucket_name}")
17
18        return {
19            'statusCode': 200,
20            'body': json.dumps('Image processed successfully.')
21        }
22
23    except KeyError as e:
24        logger.error(f"KeyError - missing key in the event: {e}")
25        return {
26            'statusCode': 400,
27            'body': json.dumps('Error processing event. Missing required data.')
28        }
29
30    except Exception as e:
31        logger.error(f"Unexpected error: {e}")
32        return {
33            'statusCode': 500,
34            'body': json.dumps('Internal server error.')
35        }
36
37
```

Code Change in Repository



```
9 # Log the event details
10 logger.info(f"Received event: {json.dumps(event)}")
11
12 # Extract bucket name and object key (file name)
13 bucket_name = event['Records'][0]['s3']['bucket']['name']
14 object_key = event['Records'][0]['s3']['object']['key']
15
16 logger.info(f"New image added: {object_key} in bucket: {bucket_name}")
17
18 return {
19     'statusCode': 200,
20     'body': json.dumps('Image processed successfully. Build Successful.!')
21 }
22
23 except KeyError as e:
24     logger.error(f"KeyError - missing key in the event: {e}")
25     return {
26         'statusCode': 400,
27         'body': json.dumps('Error processing event. Missing required data.')
28     }
29
30 except Exception as e:
31     logger.error(f"Unexpected error: {e}")
32     return {
33         'statusCode': 500,
34         'body': json.dumps('Internal server error.')
35     }
36
```


Auto Triggered Pipeline

Developer Tools > CodePipeline > Pipelines > MyCaseStudyPipeline

MyCaseStudyPipeline

Pipeline type: V2 Execution mode: QUEUED

Source

Succeeded

Pipeline execution ID: c6305e98-2226-423e-b1bf-2530067c9146

Source

GitHub (Version 2)

Succeeded - 1 minute ago

9561db61

View details

9561db61 Source: Update lambda_function.py

Disable transition

Build

Succeeded

Pipeline execution ID: c6305e98-2226-423e-b1bf-2530067c9146

Build

AWS CodeBuild

Succeeded - Just now

View details

9561db61 Source: Update lambda_function.py

Start rollback

Successful Build.

BWS Services Search [Alt+S]

Developer Tools CodePipeline

Source • CodeCommit

Artifacts • CodeArtifact

Build • CodeBuild

Deploy • CodeDeploy

Pipeline • CodePipeline

Getting started

Pipelines

Pipeline

History

Settings

Settings

Go to resource

Feedback

ImageDetectorPipeline

Pipeline type: V2 Execution mode: QUEUED

Source

Succeeded

Pipeline execution ID: #660171-c885-4e0d-8e87-661110278a1

Source

GitHub (Version 2)

Succeeded - 2 minutes ago

4014c0d4

View details

4014c0d4 Source: Update lambda_function.py

Disable transition

Build

Succeeded

Pipeline execution ID: #660171-c885-4e0d-8e87-661110278a1

Build

AWS CodeBuild

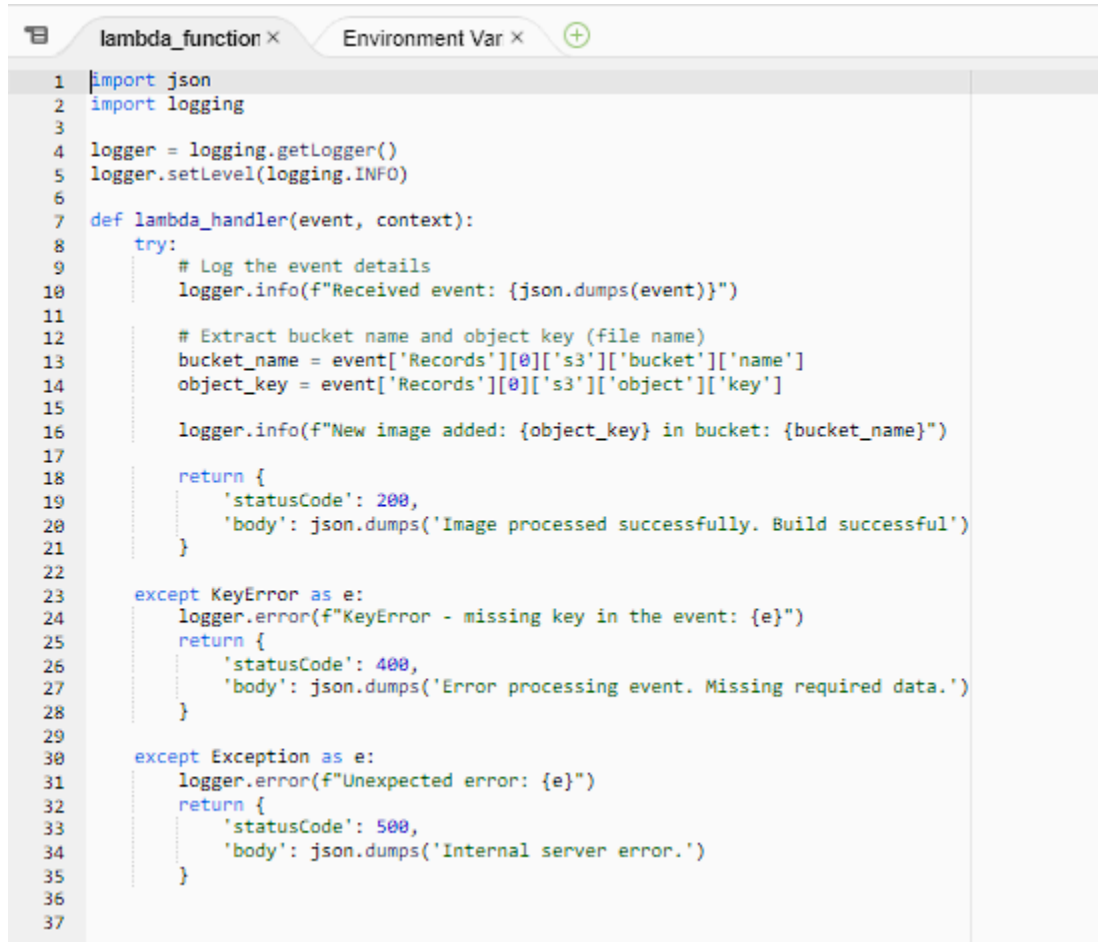
Succeeded - Just now

View details

4014c0d4 Source: Update lambda_function.py

Start rollback

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Lambda After push of new code in repository .

```
1 import json
2 import logging
3
4 logger = logging.getLogger()
5 logger.setLevel(logging.INFO)
6
7 def lambda_handler(event, context):
8     try:
9         # Log the event details
10        logger.info(f"Received event: {json.dumps(event)}")
11
12        # Extract bucket name and object key (file name)
13        bucket_name = event['Records'][0]['s3']['bucket']['name']
14        object_key = event['Records'][0]['s3']['object']['key']
15
16        logger.info(f"New image added: {object_key} in bucket: {bucket_name}")
17
18        return {
19            'statusCode': 200,
20            'body': json.dumps('Image processed successfully. Build successful')
21        }
22
23    except KeyError as e:
24        logger.error(f"KeyError - missing key in the event: {e}")
25        return {
26            'statusCode': 400,
27            'body': json.dumps('Error processing event. Missing required data.')
28        }
29
30    except Exception as e:
31        logger.error(f"Unexpected error: {e}")
32        return {
33            'statusCode': 500,
34            'body': json.dumps('Internal server error.')
35        }
36
37
```

Conclusion:

In this case study, we successfully implemented a serverless image processing workflow using AWS Lambda, S3, and CodePipeline. The solution demonstrated how Lambda functions can be triggered by S3 events, allowing real-time image processing without manual intervention. By leveraging CodePipeline, we automated the deployment of updates to the Lambda function, ensuring a streamlined CI/CD process. Testing confirmed that the workflow functions as expected, with images uploaded to S3 triggering the Lambda function, which logs the event and processes the image.

Guidelines:

- **Use AWS Personal.**
- **IAM Role Creation:**
 - **Why It's Important:** The IAM role is critical as it grants the necessary permissions for Lambda, CodeBuild, and CodePipeline to interact with each other and with other AWS services like S3. Without proper permissions, the workflow will not function, and components will fail to trigger or deploy.
 - **Guidelines:**
 - Attach policies such as **AWSLambda_FullAccess**, **AmazonS3FullAccess**, **AmazonCodeBuildAdminAccess**, **AmazonCodePipeline_FullAccess**, and **CloudWatchLogsFullAccess** to allow access to Lambda, S3, CodeBuild, and CodePipeline.
 - Ensure that the trust relationship includes both **codepipeline.amazonaws.com** and **codebuild.amazonaws.com** services to allow these services to assume the role.