

## Advance Devops Practical Examination: AWS Case Study Assignment

### Topic : Serverless Image Processing Workflow

#### 1. Introduction

**Concepts Used:** AWS Lambda, S3, 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.

### Case Study Overview:

The case study involves building a **serverless image processing workflow** using AWS services such as Lambda, S3, and CodePipeline. The goal is to create a serverless workflow that automatically triggers a Lambda function when a new image is uploaded to an S3 bucket. Additionally, the project sets up CodePipeline to automate the deployment of Lambda function updates.

### Key Feature and Application:

The case study demonstrates how AWS Lambda can process events triggered by S3 uploads, facilitating **real-time image processing** without manual intervention. The project also highlights how CodePipeline automates Lambda function deployment, ensuring **continuous integration and continuous delivery (CI/CD)**.

## 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**
  - **AmazonCodeBuildAdminAcess**
  - **AmazonCodePipeline\_FullAccess**
  - **AWSLambda\_FullAccess**
  - **CloudWatchLogsFullAccess**
- Give the role a name: **AWS\_Case\_Study** and create it.
- 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**

**Roles (2) Info**

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AWSServiceRoleForSupport	AWS Service: support (Service-Linker)	27 days ago
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linker)	-

**Roles Anywhere Info**

Authenticate your non AWS workloads and securely provide access to AWS services.

Access AWS from your non AWS workloads

Operate your non AWS workloads using the same authentication and authorization strategy that you use within AWS.

X.509 Standard

Use your own existing PKI infrastructure or use AWS Certificate Manager Private Certificate Authority to authenticate identities.

Temporary credentials

Use temporary credentials with ease and benefit from the enhanced security they provide.

**IAM > Roles > Create role**

**Select trusted entity**

**Step 1 Select trusted entity**

**Step 2 Add permissions**

**Step 3 Name, review, and create**

**Trusted entity type**

AWS service

Allow AWS services like EC2, Lambda, or others to perform actions in this account.

AWS account

Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

SAML 2.0 federation

Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

Custom trust policy

Create a custom trust policy to enable others to perform actions in this account.

**Use case**

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

**Service or use case**

Lambda

**Use case**

Choose a use case for the specified service.

Lambda

Allows Lambda functions to call AWS services on your behalf.

**Cancel** **Next**

**IAM > Roles > Create role**

**Name, review, and create**

**Step 1 Select trusted entity**

**Step 2 Add permissions**

**Step 3 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**

**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:     ]
16: }

```

**Edit**

```

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 }

```

+ Add new statement

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

Role name	Trusted entities	Last activity
AWS_Case_Study	AWS Service: lambda	-
AWSServiceRoleForSupport	AWS Service: support (Service-Linked)	27 days ago
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service)	-

View role Delete Create role

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

## 2. Create an S3 Bucket:

- Navigate to the **S3** service.
- Click on **Create Bucket**.
  - Bucket Type: **General Purpose**.
  - Provide a unique bucket name. (**bhushan-aws-bucket**)
  - **Uncheck the Block all public access option**.
  - Keep Rest of the things to default.

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

**Amazon S3**

Store and retrieve any amount of data from anywhere

Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance.

**Create a bucket**

Every object in S3 is stored in a bucket. To upload files and folders to S3, you'll need to create a bucket where the objects will be stored.

**Pricing**

With S3, there are no minimum fees. You only pay for what you use. Prices are based on the location of your S3 bucket.

Estimate your monthly bill using the [AWS Simple Monthly Calculator](#).

[View pricing details](#)

**Resources**

[User guide](#)

[API reference](#)

[FAQs](#)

**How it works**

[Introduction to Amazon S3](#)

[aws.amazon.com/S3](#)

**Create bucket**

Buckets are containers for data stored in S3.

**General configuration**

AWS Region: US East (N. Virginia) us-east-1

Bucket type: **General purpose**

Bucket name: **bhushan-aws-bucket**

Bucket settings from existing bucket - optional

**Object Ownership**

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

**Block Public Access settings for this bucket**

Block all public access

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

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

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

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

**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**

**Bucket Versioning**

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

**Bucket Versioning**

Disable  
 Enable

**Tags - optional (0)**

You can use bucket tags to track storage costs and organize buckets. [Learn more](#)

No tags associated with this bucket.

[Add tag](#)

**Default encryption** [Info](#)

Server-side encryption is automatically applied to new objects stored in this bucket.

**Encryption type** [Info](#)

- Server-side encryption with Amazon S3 managed keys (SSE-S3)
- Server-side encryption with AWS Key Management Service keys (SSE-KMS)
- Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)  
Secure your objects with two separate layers of encryption. For details on pricing, see DSSE-KMS pricing on the Storage tab of the [Amazon S3 pricing page](#).

**Bucket Key**

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

Disable  
 Enable

[CloudShell](#) [Feedback](#)

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

**Default encryption** [Info](#)

Server-side encryption is automatically applied to new objects stored in this bucket.

**Encryption type** [Info](#)

- Server-side encryption with Amazon S3 managed keys (SSE-S3)
- Server-side encryption with AWS Key Management Service keys (SSE-KMS)
- Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)  
Secure your objects with two separate layers of encryption. For details on pricing, see DSSE-KMS pricing on the Storage tab of the [Amazon S3 pricing page](#).

**Bucket Key**

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

Disable  
 Enable

**Advanced settings**

After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

[Cancel](#) [Create bucket](#)

[CloudShell](#) [Feedback](#)

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

**Successfully created bucket "bhushan-aws-bucket"**

To upload files and folders, or to configure additional bucket settings, choose [View details](#).

[View details](#) [X](#)

[Amazon S3](#) > [Buckets](#)

**Account snapshot - updated every 24 hours** [All AWS Regions](#)

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

[View Storage Lens dashboard](#)

**General purpose buckets** [Directory buckets](#)

**General purpose buckets (1)** [Info](#) [All AWS Regions](#)

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
bhushan-aws-bucket	US East (N. Virginia) us-east-1	<a href="#">View analyzer for us-east-1</a>	October 20, 2024, 11:34:46 (UTC+05:30)

[Copy ARN](#) [Empty](#) [Delete](#) [Create bucket](#)

[Find buckets by name](#)

[CloudShell](#) [Feedback](#)

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

### 3. Create a Lambda Function:

- Navigate to the **Lambda Console**.
- Click **Create Function**.
  - Choose **Author from scratch**.
  - Provide a function name **ImageDetectorLambda**.
  - Choose a free-tier eligible runtime **Python 3.12**.
  - Assign the **IAM role** created earlier **AWS\_Case\_Study**.
  - Keep Rest of things to default.
- 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)

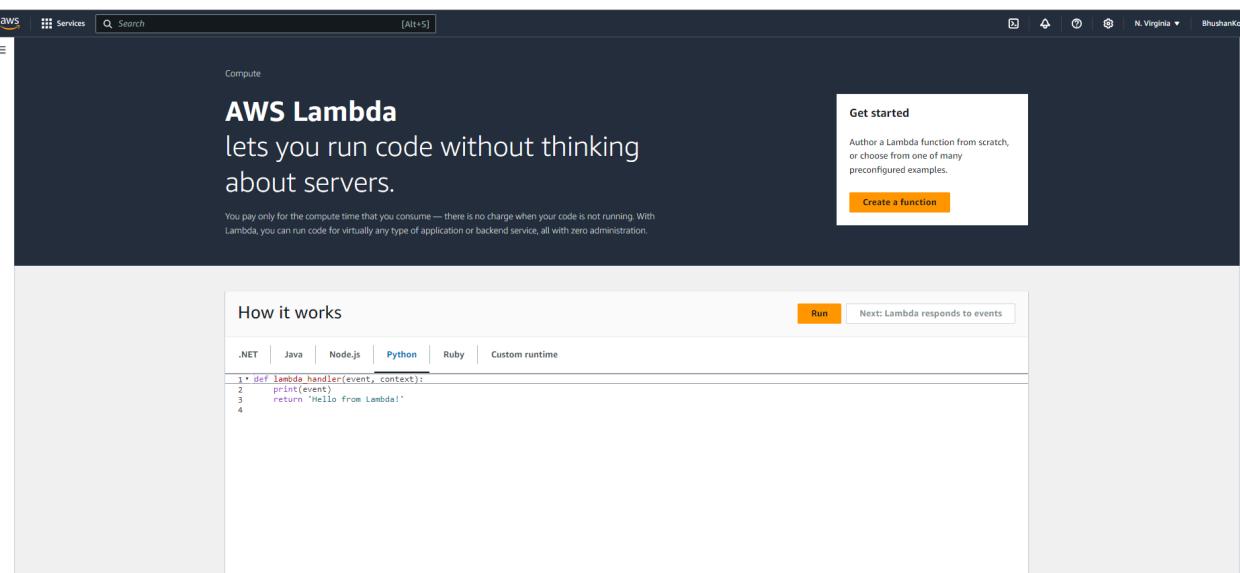
def lambda_handler(event, context):
    # Log the event details
    logger.info(f'Received event: {json.dumps(event)}')

    # 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.')
    }
```

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



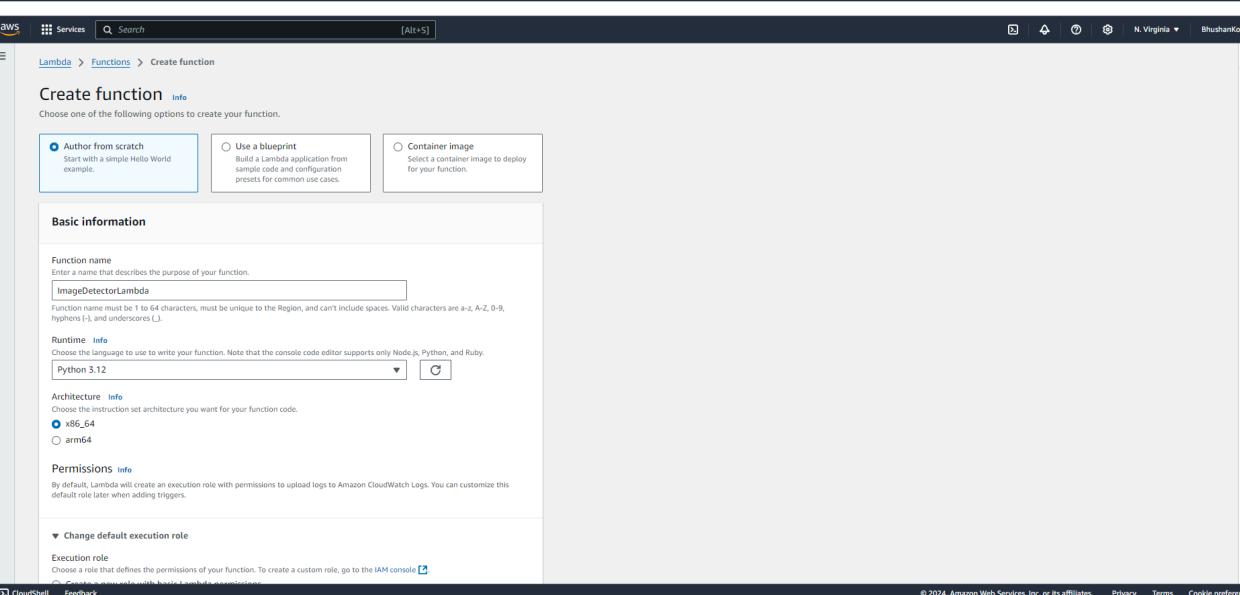
The screenshot shows the AWS Lambda 'How it works' page. It features a large central code editor window with Python code examples. The code is as follows:

```

1 def lambda_handler(event, context):
2     print(event)
3     return 'Hello from Lambda!'
4

```

Below the code editor, there are tabs for .NET, Java, Node.js, Python, Ruby, and Custom runtime. A 'Run' button is located at the top right of the code editor. To the right of the code editor, there is a 'Get started' box with a 'Create a function' button.

The screenshot shows the 'Create function' wizard. The first step, 'Choose one of the following options to create your function.', has 'Author from scratch' selected. Other options include 'Use a blueprint' and 'Container image'. The 'Basic information' section requires entering a function name ('ImageDetectorLambda'), choosing a runtime ('Python 3.12'), and selecting an architecture ('x86\_64'). The 'Permissions' section indicates that Lambda will create an execution role with CloudWatch Logs permissions. At the bottom, there is a 'Change default execution role' link and an 'Execution role' section.

**ImageDetectorLambda**

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 for 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 [\[IAM\]](#).

- 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.

AWS\_Case\_Study

[View the AWS\\_Case\\_Study 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.

[Create function](#)

**Successfully created the function ImageDetectorLambda. You can now change its code and configuration. To invoke your function with a test event, choose "Test".**

[Lambda > Functions > ImageDetectorLambda](#)

### ImageDetectorLambda

**Function overview** [Info](#)

[Diagram](#) [Template](#)

**ImageDetectorLambda**

**Layers** (0)

[+ Add trigger](#) [+ Add destination](#)

[Throttle](#) [Copy ARN](#) [Actions ▾](#)

[Export to Application Composer](#) [Download ▾](#)

**Description**  
-

**Last modified**  
3 seconds ago

**Function ARN**  
arn:aws:lambda:us-east-1:010928205712:function:ImageDetectorLambda

**Function URL** [Info](#)

[Code](#) [Test](#) [Monitor](#) [Configuration](#) [Aliases](#) [Versions](#)

**Code source** [Info](#)

[Upload from ▾](#)

[File](#) [Edit](#) [Find](#) [View](#) [Go](#) [Tools](#) [Window](#) [Test](#) [Deploy](#)

Environment Var X Execution results X lambda\_function X

```
1 import json
2 import logging
3
4 logger = logging.getLogger()
5 logger.setLevel(logging.INFO)
6
7 def lambda_handler(event, context):
8     logger.info("Received event: %s", json.dumps(event))
9
10    # Get the object from the event and show its info
11    bucket_name = event['Records'][0]['s3']['bucket']['name']
12    object_key = event['Records'][0]['s3']['object']['key']
13
14    logger.info("New image added: [object_key] in bucket: [bucket_name]")
15
16    return {
17        'statusCode': 200,
18        'body': json.dumps('Image processed successfully')
19    }
20
21
```

[CloudShell](#) [Feedback](#)

**Successfully updated the function ImageDetectorLambda.**

**Code source** [Info](#)

[Upload from ▾](#)

[File](#) [Edit](#) [Find](#) [View](#) [Go](#) [Tools](#) [Window](#) [Test](#) [Deploy](#)

Environment Var X Execution results X lambda\_function X

```
1 import json
2 import logging
3
4 logger = logging.getLogger()
5 logger.setLevel(logging.INFO)
6
7 def lambda_handler(event, context):
8     logger.info("Received event: %s", json.dumps(event))
9
10    # Get the object from the event and show its info
11    bucket_name = event['Records'][0]['s3']['bucket']['name']
12    object_key = event['Records'][0]['s3']['object']['key']
13
14    logger.info("New image added: [object_key] in bucket: [bucket_name]")
15
16    return {
17        'statusCode': 200,
18        'body': json.dumps('Image processed successfully')
19    }
20
21
```

20:8 Python Spaces: 4

[Code properties](#) [Info](#)

[CloudShell](#) [Feedback](#)

**Add trigger**

**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.  
Q s3/bhushan-aws-bucket [X](#) [C](#)  
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 [X](#) PUT [X](#) POST [X](#) COPY [X](#)  
Multipart upload completed [X](#)

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.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

[Cancel](#) [Add](#)

**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.  
Q s3/bhushan-aws-bucket [X](#) [C](#)  
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 [X](#) PUT [X](#) POST [X](#) COPY [X](#)  
Multipart upload completed [X](#)

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.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

[Cancel](#) [Add](#)

**ImageDetectorLambda**

The trigger bhushan-aws-bucket was successfully added to function ImageDetectorLambda. The function is now receiving events from the trigger.

**Function overview** [Info](#)

[Diagram](#) [Template](#)

**Triggers** [1](#) [Info](#)

imageDetectorLambda

**Configuration**

**Code** **Test** **Monitor** **Configuration** **Aliases** **Versions**

**General configuration**

**Triggers** [1](#) [Info](#)

Find triggers [C](#) Fix errors [Edit](#) [Delete](#) [Add trigger](#)

Trigger [s3: bhushan-aws-bucket](#)

[CloudShell](#) [Feedback](#)

Name:Bhushan Mukund Kor

Academic Year:2024-2025

Division: D15C

Roll No: 28

- Now Let's test that Lambda is properly working by uploading image and checking logs.

The screenshot shows the AWS S3 'Upload' interface. At the top, the navigation bar includes 'Services', a search bar, and a 'Upload' button. Below the navigation, the path 'Amazon S3 > Buckets > bhushan-aws-bucket > Upload' is displayed. A large central area is titled 'Upload' with a sub-section 'Info'. It contains a 'Files and folders' table with one item: 'Day.jpg' (1 Total, 674.6 KB). Below this is a 'Destination' section with the target bucket 's3://bhushan-aws-bucket'. There are sections for 'Permissions' and 'Properties'. At the bottom right is an orange 'Upload' button.

The screenshot shows the AWS S3 'Upload' status page. At the top, a green banner indicates 'Upload succeeded' with a link to 'View details below.' Below this, the heading 'Upload: status' is shown. A summary table provides a high-level overview of the upload results. The 'Files and folders' section lists the uploaded file 'Day.jpg' with its details: Name (Day.jpg), Folder (-), Type (image/jpeg), Size (674.6 KB), Status (Succeeded), and Error (-). The status column shows 'Succeeded' for the single file and 'Failed' for zero errors.

**CloudWatch Log group details for /aws/lambda/ImageDetectorLambda**

**Log group details**

Log class: Info Standard	Stored bytes: -	KMS key ID: -
ARN: arn:aws:logs:us-east-1:010928205712:log-group:/aws/lambda/ImageDetectorLambda*	Metric filters: 0	Anomaly detection: Configure
Creation time: 1 minute ago	Subscription filters: 0	Data protection: -
Retention: Never expire	Contributor insights rules: -	Sensitive data count: -

**Log streams**

- Filter log streams or try prefix search: 2024/10/20/[SLATEST]e39ca3c4e0dd447a97297756715f9767
- Show expired:
- Last event time: 2024-10-20 11:47:50 (UTC+05:30)

**CloudWatch Log events for /aws/lambda/ImageDetectorLambda**

**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](#)

Timestamp	Message	Actions
No older events at this moment. <a href="#">Retry</a>		
2024-10-20T11:47:59.020+05:30	INIT_START Runtime Version: python3.12.v16 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:188d9cxa2714ff5637bd2b0e81ec81ec1bc488bf277db184c1dc081d0881	<a href="#">Edit</a>
	INIT_START Runtime Version: python3.12.v36 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:188d9cxa2714ff5637bd2b0e81ec81ec1bc488bf277db184c1dc081d0881	<a href="#">Edit</a>
2024-10-20T11:47:59.108+05:30	START RequestId: 7f65b755-71eb-4a10-ad48-9609810b0a41 Version: \$LATEST	<a href="#">Edit</a>
	START RequestId: 7f65b755-71eb-4a10-ad48-9609810b0a41 Version: \$LATEST	<a href="#">Edit</a>
2024-10-20T11:47:59.109+05:30	[INFO] 2024-10-20T00:17:58.108Z 7f65b755-71eb-4a10-ad48-9609810b0a41 Received event: {"Records": [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "us-east-1", "eventTime": "2024-10-20T00:17:48.625Z", "eventName": "ObjectCreate.."}]}	<a href="#">Edit</a>
2024-10-20T11:47:59.109+05:30	[INFO] 2024-10-20T00:17:58.109Z 7f65b755-71eb-4a10-ad48-9609810b0a41 New image added: Day.jpg in bucket: bhushan-aws-bucket	<a href="#">Edit</a>
	[INFO] 2024-10-20T00:17:58.109Z 7f65b755-71eb-4a10-ad48-9609810b0a41 New image added: Day.jpg in bucket: bhushan-aws-bucket	<a href="#">Edit</a>
2024-10-20T11:47:59.110+05:30	END RequestId: 7f65b755-71eb-4a10-ad48-9609810b0a41	<a href="#">Edit</a>
	END RequestId: 7f65b755-71eb-4a10-ad48-9609810b0a41	<a href="#">Edit</a>
2024-10-20T11:47:59.110+05:30	REPORT RequestId: 7f65b755-71eb-4a10-ad48-9609810b0a41 Duration: 2.19 ms Billed Duration: 3 ms Memory Size: 128 MB Max Memory Used: 32 MB Init Duration: 85.11 ms	<a href="#">Edit</a>
	REPORT RequestId: 7f65b755-71eb-4a10-ad48-9609810b0a41 Duration: 2.19 ms Billed Duration: 3 ms Memory Size: 128 MB Max Memory Used: 32 MB Init Duration: 85.11 ms	<a href="#">Edit</a>
No newer events at this moment. <a href="#">Auto retry</a> . <a href="#">Resume</a>		

#### 4. Create a Github Repository:

- Create Repository with name **AWS-CodePipeline**.
- 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:

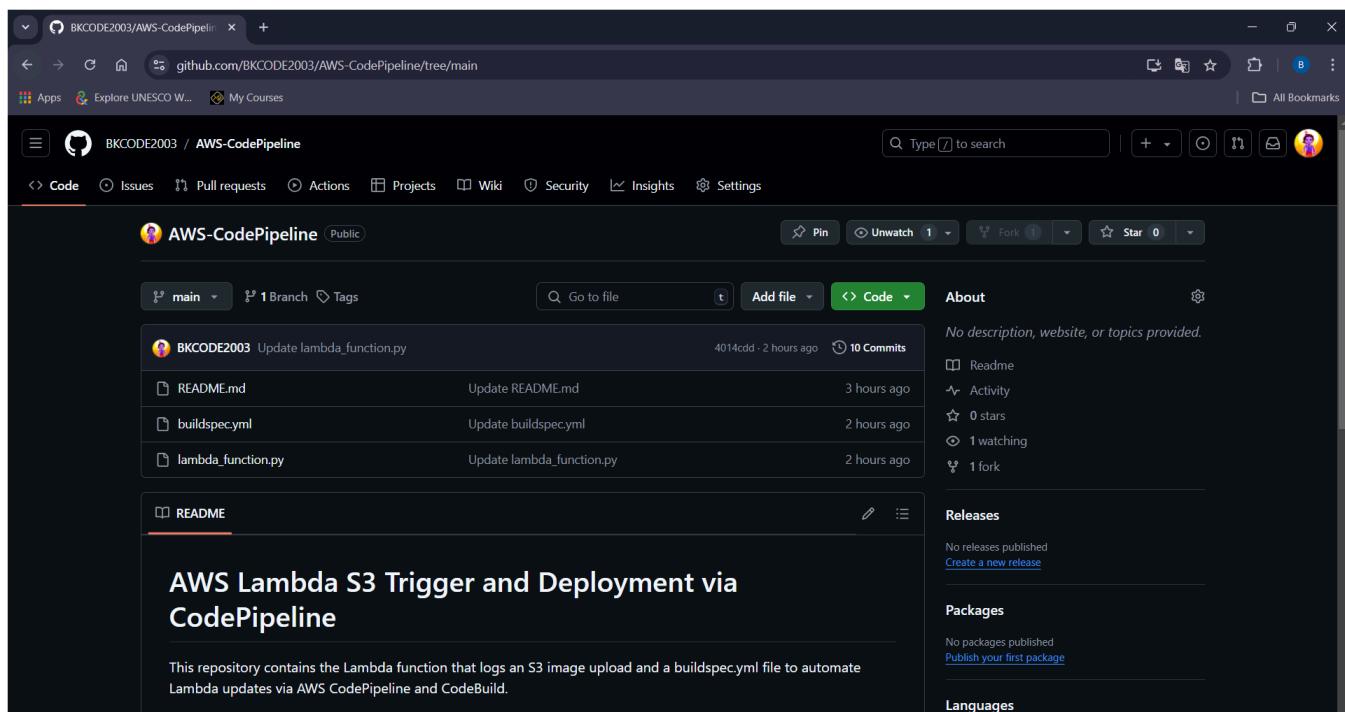
- echo "Packaging Lambda function..."

- zip -r lambda\_function.zip .

- echo "Updating Lambda function in AWS..."

- aws lambda update-function-code --function-name ImageDetectorLambda --zip-file fileb://lambda\_function.zip

#### **lambda\_function.py Same as in step 3**



A screenshot of a GitHub repository page for 'AWS-CodePipeline/lambda\_function.py'. The file contains Python code for a Lambda function. The code imports json and logging, sets up a logger, and defines a lambda\_handler function that logs the received event, extracts bucket and object details, and returns a success response.

```
1 import json
2 import logging
3
4 logger = logging.getLogger()
5 logger.setLevel(logging.INFO)
6
7 def lambda_handler(event, context):
8     # Log the event details
9     logger.info(f"Received event: {json.dumps(event)}")
10
11     # Extract bucket name and object key (file name)
12     bucket_name = event['Records'][0]['s3']['bucket']['name']
13     object_key = event['Records'][0]['s3']['object']['key']
14
15     logger.info(f"New image added: {object_key} in bucket: {bucket_name}")
16
17     return {
18         'statusCode': 200,
19         'body': json.dumps('Image processed successfully. Also Lambda Function Updated.')
20     }
```

A screenshot of a GitHub repository page for 'AWS-CodePipeline/buildspec.yml'. The file is a YAML build specification for AWS CodeBuild. It defines two phases: 'install' and 'build'. The 'install' phase runs pip install awscli. The 'build' phase packages the Lambda function code into a zip file and updates the Lambda function with the new code using the AWS CLI.

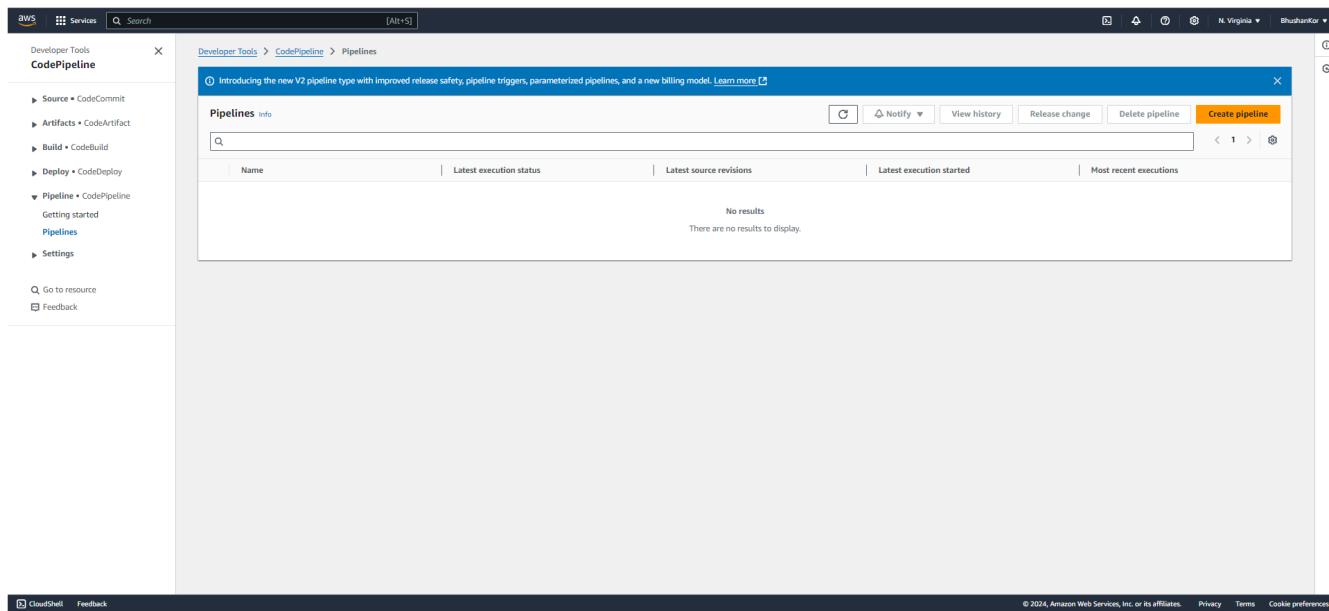
```
1 version: 0.2
2
3 phases:
4   install:
5     commands:
6       - pip install awscli # Ensure AWS CLI is available
7   build:
8     commands:
9       - echo "Packaging Lambda function..."
10      - zip -r lambda_function.zip .
11      - echo "Updating Lambda function in AWS..."
12      - aws lambda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

## 5. Create a CodePipeline:

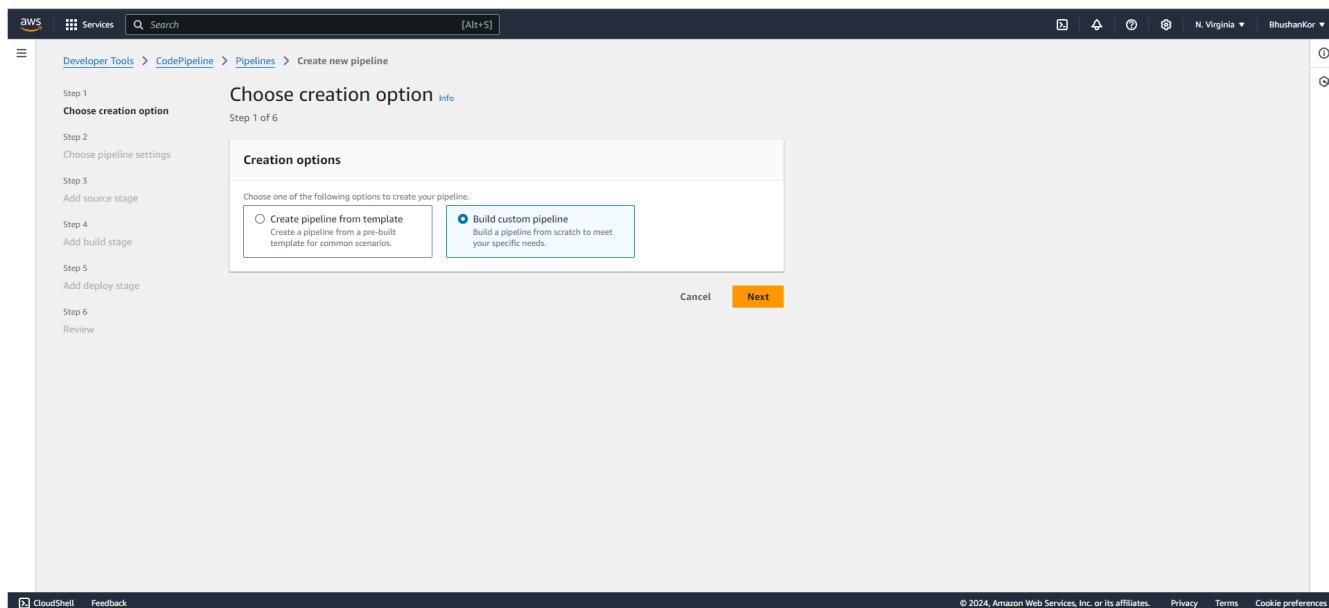
- Go to **AWS CodePipeline** in the AWS Management Console and create a new pipeline.

### Step 1:Choose Creation option.

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



The screenshot shows the AWS CodePipeline Pipelines page. On the left, there's a navigation sidebar with options like Source, Artifacts, Build, Deploy, Pipeline, Pipelines, and Settings. The main area is titled 'Pipelines' and shows a table with columns for Name, Latest execution status, Latest source revisions, Latest execution started, and Most recent executions. A message at the top says 'Introducing the new V2 pipeline type with improved release safety, pipeline triggers, parameterized pipelines, and a new billing model. Learn more'. At the bottom right of the main area, there's a prominent orange 'Create pipeline' button.



The screenshot shows the 'Choose creation option' step of the AWS CodePipeline wizard. The left sidebar lists steps from 1 to 6: 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), and Step 6 (Review). The main content area is titled 'Choose creation option' with a sub-note 'Step 1 of 6'. It contains a section titled 'Creation options' with two choices: 'Create pipeline from template' (radio button not selected) and 'Build custom pipeline' (radio button selected). Below the radio buttons, it says 'Create a pipeline from scratch to meet your specific needs.' At the bottom right are 'Cancel' and 'Next' buttons.

## Step 2:Choose Pipeline Settings.

- Pipeline Name: **ImageDetectorPipeline** .
- Select Existing Role : **AWS\_Case\_Study** .
- Keep Rest all to default.

Developer Tools > CodePipeline > Pipelines > Create new pipeline

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

**Pipeline settings**

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

**Pipeline type**  
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**

**Variables**  
You can add variables at the pipeline level. You can choose to assign the value when you start the pipeline. Choosing this option requires pipeline type V2. [Learn more](#)

No variables defined at the pipeline level in this pipeline.

**Add variable**  
You can add up to 50 variables.

**Notes**  
The first pipeline execution will fail if variables have no default values.

**Advanced settings**

Cancel Previous Next

### Step 3: Add Source Storage.

- Source provider: **GitHub (Version 2)** .
- Connect Your account where you have created the repository. (**BKCODE2003**)
- Select Repository. (**BKCODE2003/AWS-CodePipeline**)
- Select Branch to **main**.
- In Trigger Just add **main** in include.
- Keep Rest all to default.

Choose creation option

Step 1 Choose pipeline settings

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.

Q awsmodeconnectionsus-east-1:010928205712:connection/6f08433b-a3 X or [Connect to GitHub](#)

Repository name  
Choose a repository in your GitHub account.

Q BKCODE2003/AWS-CodePipeline 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.

Q main X

Output artifact format  
Choose the output artifact format.

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

Output artifact format  
Choose the output artifact format.

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

**Full clone**  
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.

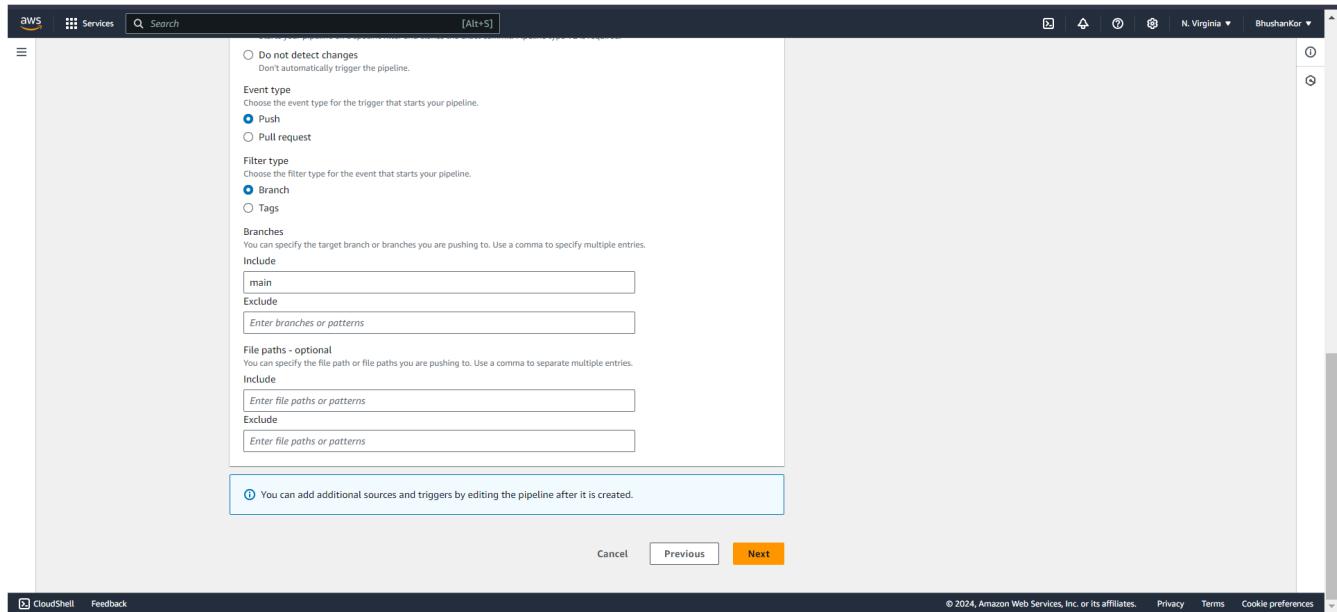
**Branch**

Tags

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

Include  
main

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



#### 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 file** .
  - Buildspec name: **buildspec.yml** .
  - Keep Rest all to Default.
- Select Created Project.
- Keep Rest all to default.

Create build project | CodeBuild | us-east-1 - Google Chrome  
us-east-1.console.aws.amazon.com/codesuite/codebuild/project/new?preset=codepipeline&region=us-east-1

Developer Tools > CodeBuild > Build projects > Create build project

**Continue to CodePipeline**  
Create a new CodeBuild build project and return to CodePipeline to finish configuring your pipeline.

### Create build project

#### Project configuration

Project name: **Image\_Detector\_Build**  
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: **arn:aws:iam::010928205712:role/AWS\_Case\_Study**

Allow AWS CodeBuild to modify this service role so it can be used with this build project  
arn:aws:iam::010928205712:role/AWS\_Case\_Study

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

Create build project | CodeBuild | us-east-1 - Google Chrome  
us-east-1.console.aws.amazon.com/codesuite/codebuild/project/new?preset=codepipeline&region=us-east-1

Developer Tools > CodeBuild > Build projects > Create build project

Allow AWS CodeBuild to modify this service role so it can be used with this build project  
arn:aws:iam::010928205712:role/AWS\_Case\_Study

**Additional configuration**  
Description, Build badge, Concurrent build limit, tags

#### 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**

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

Create build project | CodeBuild | us-east-1 - Google Chrome  
us-east-1.console.aws.amazon.com/codesuite/codebuild/project/new?preset=codepipeline&region=us-east-1

**Image**  
aws/codebuild/amazonlinux2-x86\_64-standard:5.0

**Image version**  
Always use the latest image for this runtime version

Use GPU-enhanced compute

**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:awsiam:01092805712:role/AWS\_Case\_Study

Allow AWS CodeBuild to modify this service role so it can be used with this build project

**Additional configuration**  
Timeout, privileged, certificate, VPC, compute type, environment variables, file systems

**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

**Batch configuration**  
You can run a group of builds as a single execution. Batch configuration is also available in advanced option when starting build.

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

Create build project | CodeBuild | us-east-1 - Google Chrome  
us-east-1.console.aws.amazon.com/codesuite/codebuild/project/new?preset=codepipeline&region=us-east-1

**buildspec.yml**

**Batch configuration**  
You can run a group of builds as a single execution. Batch configuration is also available in advanced option when starting build.

Define batch configuration - optional  
You can also define or override batch configuration when starting a build batch.

**Logs**

**CloudWatch**

CloudWatch logs - optional  
Checking this option will upload build output logs to CloudWatch.

**Group name - optional**  
aws/codebuild/Image\_Detector\_Build  
The group name of the logs in CloudWatch Logs. The log group name will be /aws/codebuild/<project-name> by default.

**Stream name prefix - optional**  
The prefix of the stream name of the CloudWatch Logs.  
S3

S3 logs - optional  
Checking this option will upload build output logs to S3.

**Cancel** **Continue to CodePipeline**

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

The screenshot shows the AWS CodePipeline interface. The user is on Step 4 of 6, titled 'Add build stage'. In the 'Build - optional' section, 'Other build providers' is selected. A success message at the bottom states: 'Successfully created Image\_Detector\_Build in CodeBuild.'.

The screenshot shows the AWS CodePipeline interface. The user is on Step 6 of 6, titled 'Review'. The pipeline configuration includes a build stage with 'Image\_Detector\_Build' selected. The 'Single build' option is chosen under 'Build type'. The 'Next' button is highlighted at the bottom.

**Step 5: Skip The Deploy Stage.**

**Step 6: Review the Pipeline and Click on Create.**

**Step 1: Choose pipeline settings**

Pipeline name	ImageDetectorPipeline
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::010928205712:role/AWS_Case_Study

**Variables**

Name	Default value	Description
No variables		

No variables defined at the pipeline level in this pipeline.

**Step 2: Add source stage**

**Source action provider**

Source action provider	GitHub (Version 2)
OutputArtifactFormat	CODE_ZIP
DetectChanges	false
ConnectionArn	arn:aws:codeconnection:us-east-1:010928205712:connection/6f08433b-a361-4f2e-b871-12ad7eed502
FullRepositoryId	BKCODE2003/AWS-CodePipeline
Default branch	main
Enable automatic retry on stage failure	Enabled

**Trigger configuration**

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

**Trigger type**

- Specify filter
- Event type
- Push

**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

Name: Bhushan Mukund Kor

Academic Year: 2024-2025

Division: D15C

Roll No: 28

The screenshot shows the 'Step 3: Add build stage' configuration screen. It includes sections for 'Build action provider' (set to AWS CodeBuild) and 'ProjectName' (set to Image\_Detector\_Build). Under 'Commands', there is a note about enabling automatic retry on stage failure, which is set to 'Enabled'. At the bottom are 'Cancel', 'Previous', and 'Create pipeline' buttons.

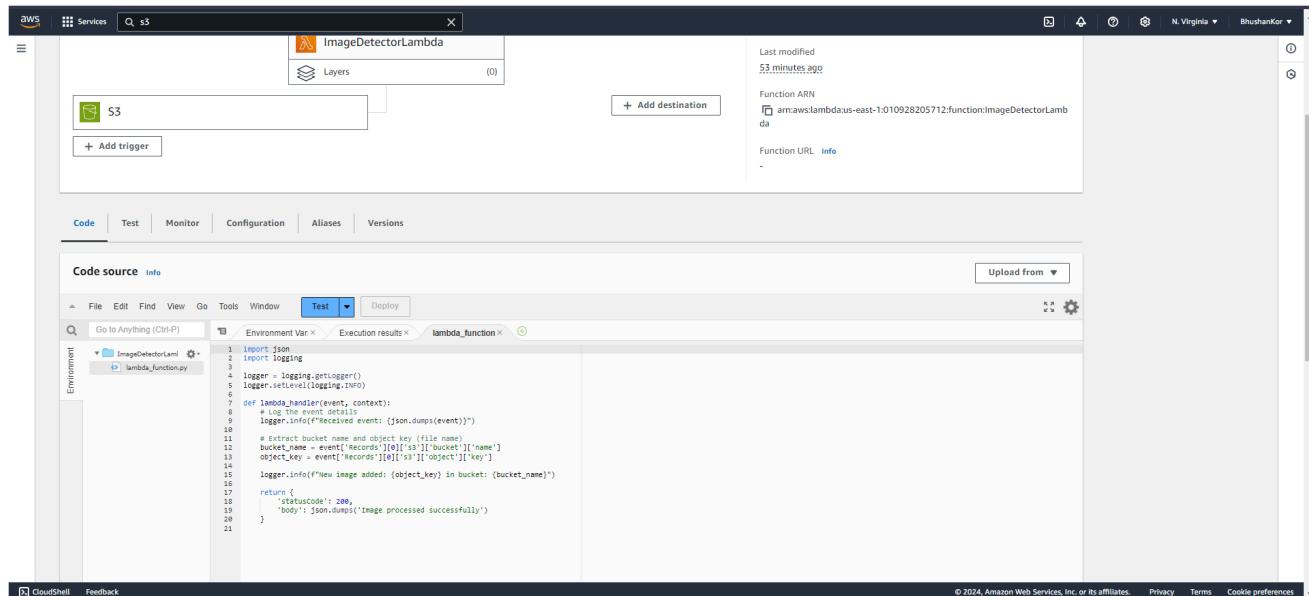
The screenshot shows the success message 'Congratulations! The pipeline ImageDetectorPipeline has been created.' Below it, the pipeline details are shown: Pipeline type: V2, Execution mode: QUEUED. The pipeline consists of two stages: Source and Build. The Source stage is succeeded, showing GitHub (Version 2) and a successful build step. The Build stage is also succeeded, showing AWS CodeBuild and a successful build step. There are buttons for Notify, Edit, Stop execution, Clone pipeline, and Release change.

The screenshot shows the execution details for the ImageDetectorPipeline. It lists the pipeline type (V2), execution mode (QUEUED), and the current execution ID. The execution details show the Source and Build stages, both of which have succeeded. The Source stage details include GitHub (Version 2) and a successful build step. The Build stage details include AWS CodeBuild and a successful build step. There are buttons for Notify, Edit, Stop execution, Clone pipeline, and Release change.

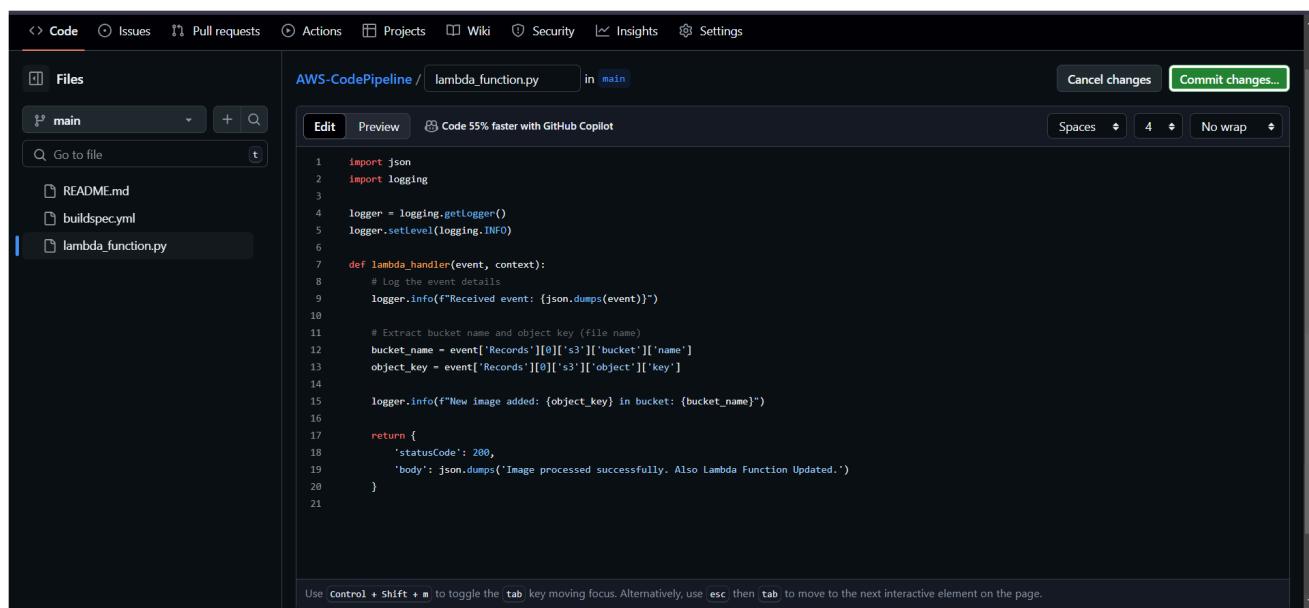
## 6.Test the Pipeline:

- 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 Before push of new code in repository



### Code Change in Repository



Name:Bhushan Mukund Kor

Academic Year:2024-2025

Division: D15C

Roll No: 28

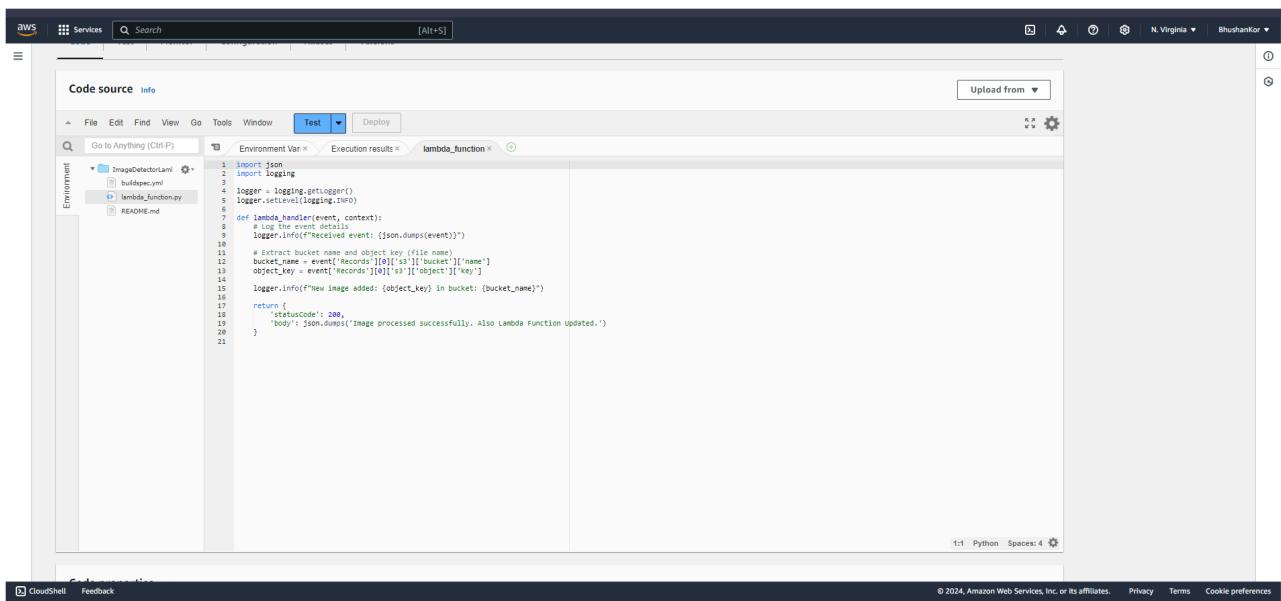
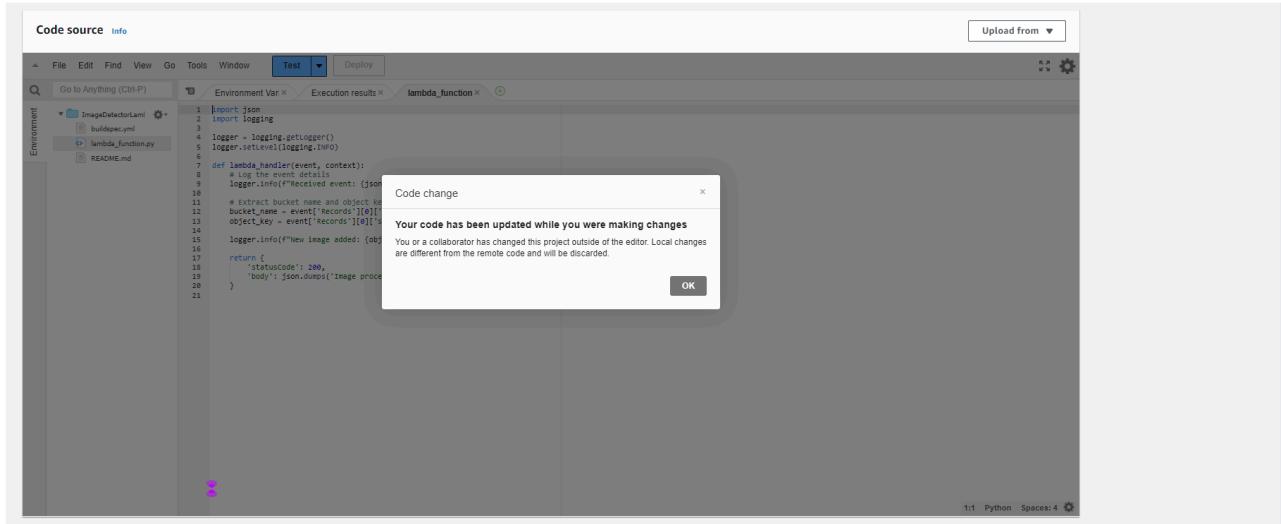
## Auto Triggered Pipeline

The screenshot shows the AWS CodePipeline console with the pipeline named "ImageDetectorPipeline". The pipeline type is V2 and the execution mode is QUEUED. The pipeline consists of two stages: Source and Build. The Source stage is successful, and the Build stage is in progress. The pipeline execution ID is #f460171-cb85-4eed-8e87-66111072f8e1. The interface includes standard AWS navigation elements like CloudShell and Feedback, and footer links for Privacy, Terms, and Cookie preferences.

**Successful Build.**

This screenshot is identical to the one above, showing the "ImageDetectorPipeline" in the AWS CodePipeline console. The Build stage has now completed successfully, indicated by a green checkmark icon and the text "Succeeded". A "Start rollback" button is visible next to the Build stage. The pipeline execution ID remains #f460171-cb85-4eed-8e87-66111072f8e1.

## Lambda After push of new code in repository .



### 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.

**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.