

Title: Serverless Image Processing Workflow

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.

Introduction

Overview:

The case study focuses on developing a serverless image processing workflow using AWS services. When an image is uploaded to an S3 bucket, an AWS Lambda function is triggered to log and process the image. AWS CodePipeline automates the deployment of the Lambda function, ensuring that any updates to the function are seamlessly integrated and deployed. This setup leverages the event-driven architecture and scalability of AWS, providing an efficient and cost-effective solution for image processing.

Key Features and Applications:

Key Features:

1. **Serverless Architecture:** Utilizes AWS Lambda to execute code in response to image uploads without provisioning or managing servers.
2. **Event-Driven Processing:** Automatically triggers the Lambda function upon image upload to the S3 bucket.

3. Automated Deployment: Employs AWS CodePipeline to automate the build, test, and deployment stages of the Lambda function, ensuring continuous integration and delivery.
4. Scalability: Leverages AWS's inherent scalability to handle varying loads without manual intervention.
5. Cost-Effectiveness: Pay-as-you-go model for AWS Lambda and S3, reducing costs by eliminating the need for dedicated server resources.

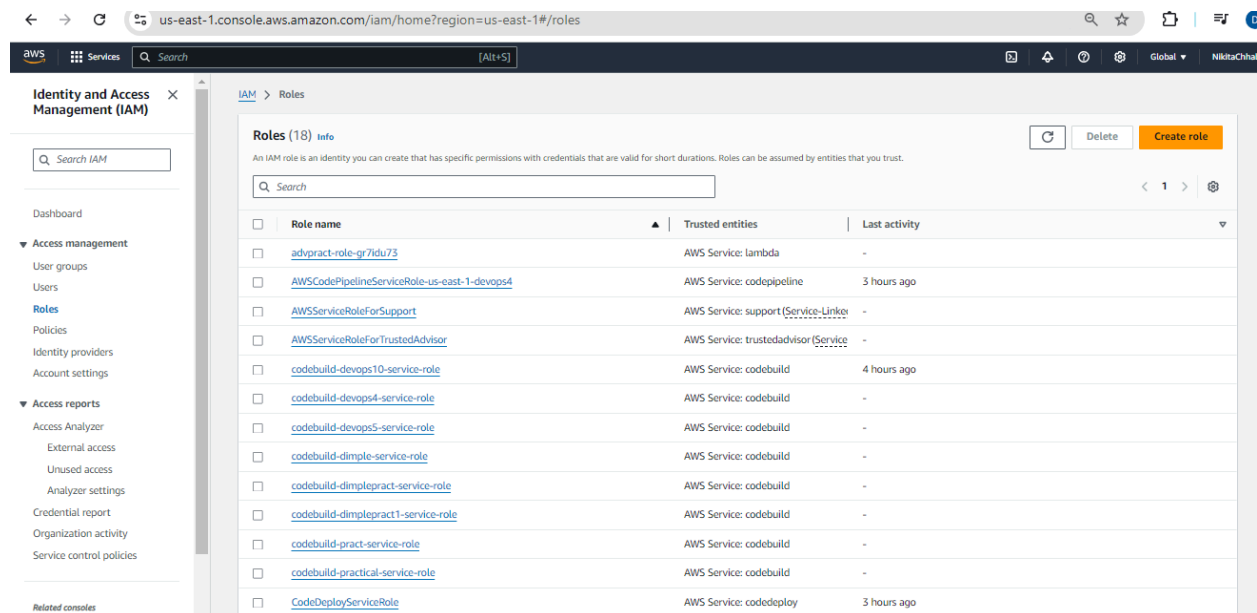
Applications:

1. Real-Time Image Processing: Automatically resize, convert, or analyze images upon upload, useful for applications like photo-sharing platforms, e-commerce sites, and social media.
2. Metadata Extraction: Extract and log metadata from images for cataloging and indexing purposes.
3. Image Moderation: Automatically analyze images for inappropriate content, facilitating content moderation.
4. Thumbnail Generation: Create thumbnails for uploaded images, which can be used in web and mobile applications to improve loading times and user experience.

Step-by-Step Explanation

STEP I: Create IAM Role:

STEP 1:Login to to your AWS console account because AWS academy account doesn't give you permission ,then go on roles and click on create role.



STEP 2:In trusted entity type select “ AWS service” in use case select “lambda”and then click on next.

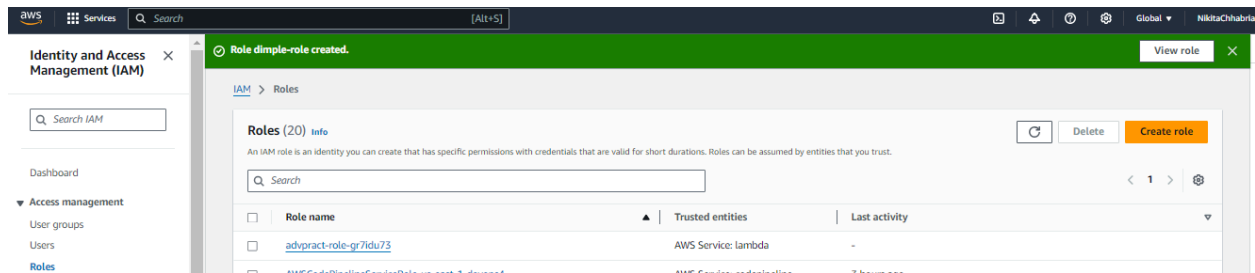
The screenshot shows the AWS IAM console interface for creating a role. The left sidebar indicates the current step is 'Step 1: Select trusted entity'. The main content area is titled 'Select trusted entity' and includes an 'Info' link. Under the 'Trusted entity type' section, five options are listed: 'AWS service' (selected), 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy'. Each option has a brief description. Below this, the 'Use case' section shows a dropdown menu with 'Lambda' selected. At the bottom right, there are 'Cancel' and 'Next' buttons.

STEP 3: Attach policies such as:

- AmazonS3FullAccess
- AmazonCodeBuildAdminAccess
- AmazonCodePipeline_FullAccess
- AWSLambda_FullAccess
- CloudWatchLogsFullAccess

Then in Role name add the name of the role and then click on create role.

The screenshot shows the 'Name, review, and create' step of the AWS IAM console. The left sidebar indicates the current step is 'Step 3: Name, review, and create'. The main content area is titled 'Name, review, and create'. Under the 'Role details' section, there are two input fields: 'Role name' and 'Description'. The 'Role name' field contains the text 'dimple-role'. The 'Description' field contains the text 'Allows Lambda functions to call AWS services on your behalf.' At the bottom right, there are 'Cancel' and 'Next' buttons.



STEP 4:After that created go ob IAM >Roles >role that you have created >edit trust policies in that write this code and then click on update the policy .

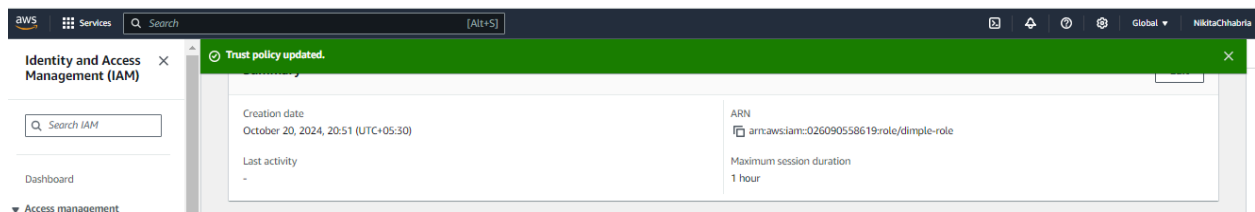
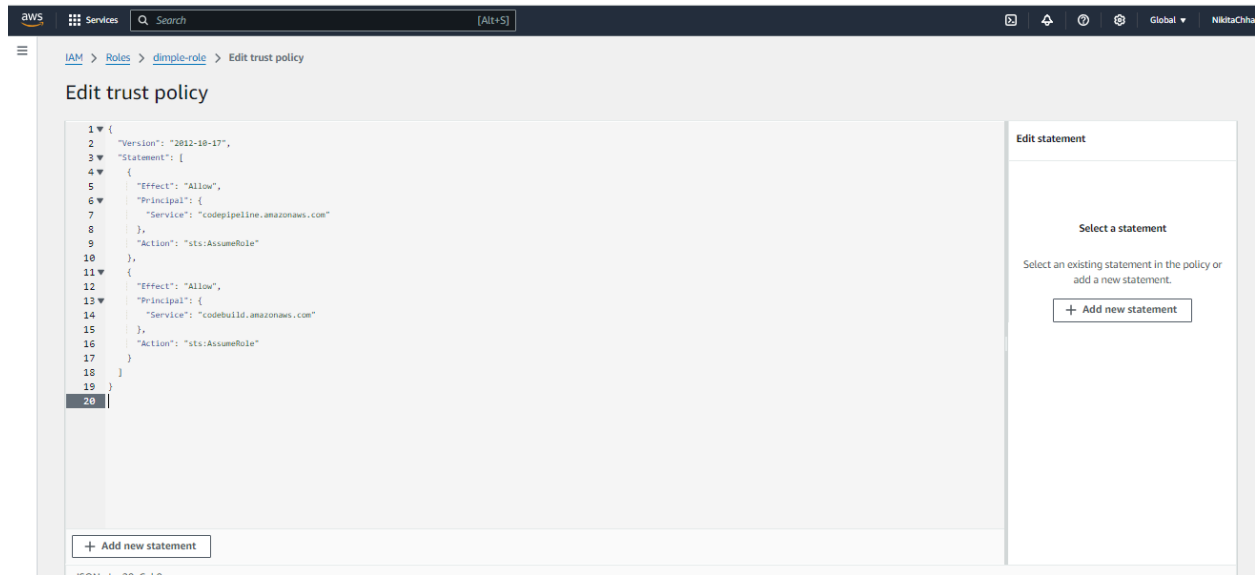
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "codepipeline.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Name:DIMPLE DALWANI

Year:2024-2025

Division: D15C

Roll No: 8



STEP II:Create an S3 bucket :

STEP 1:

- Login to AWS Management Console.
- Go to S3 Services and forSearch for "S3" in the search bar and select it

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](#)

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.

Format: s3://bucket/prefix

STEP 2:Enable all block public access settings for this bucket.

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.

Warning: 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.

STEP 3:Let all the default settings remain same and then Click on create 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

► Advanced settings

1

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

Cancel

Create bucket

STEP 4: The bucket gets created successfully.

Services

Search

(Alt+S)

N. Virginia

NikitaChhabra

Successfully created bucket "dimplecasestudy"

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](#)

General purpose buckets

Directory buckets

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

Refresh

Copy ARN

Empty

Delete

Create bucket

Find buckets by name

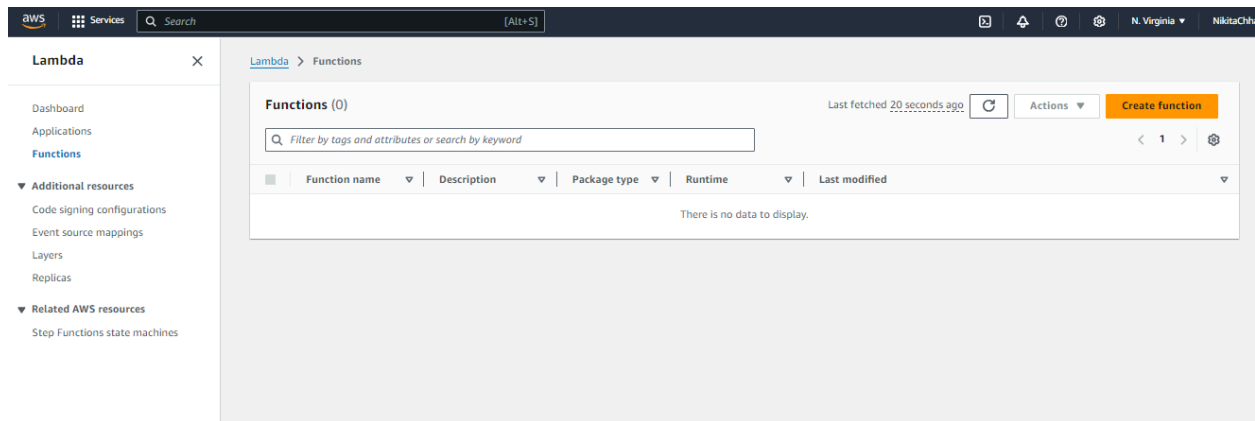
< 1 > ⚙

Name	AWS Region	IAM Access Analyzer	Creation date
dimplecasestudy	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 20, 2024, 21:09:01 (UTC+05:30)

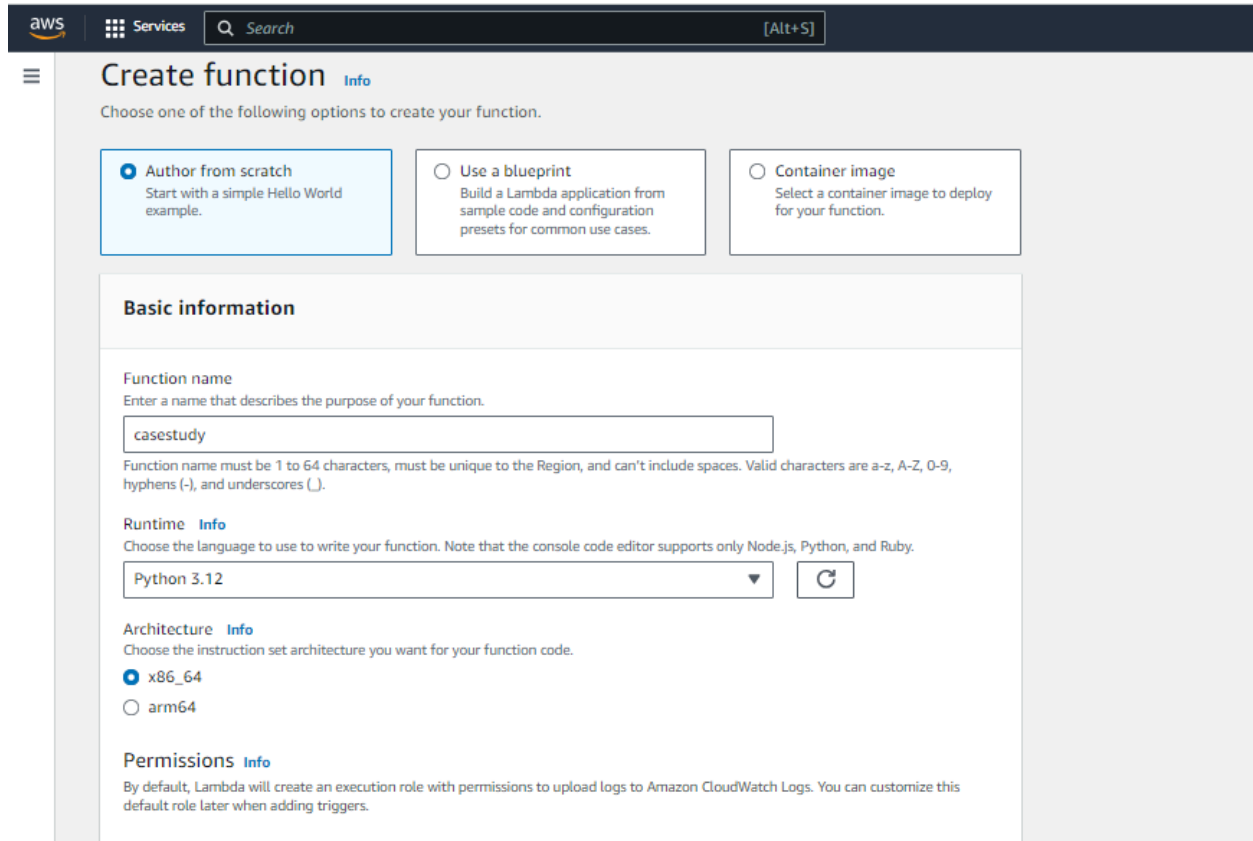
STEP III:Create a Lambda Function

STEP1:Go to Lambda Service and search for "Lambda" in the search bar and select it.

STEP 2: Create a New Lambda Function:Click on create new



STEP 3:Select "Author from scratch" and then add then add the name of the function name and then select the latest python version and then click on create function.



The screenshot shows the AWS Lambda 'Create function' page. At the top, there's a navigation bar with the AWS logo, 'Services', a search bar, and a keyboard shortcut '[Alt+S]'. Below the navigation bar, the page title is 'Create function' with an 'Info' link. A sub-header says 'Choose one of the following options to create your function.' There are three radio button options: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. The 'Author from scratch' option is highlighted with a blue border. Below these options is the 'Basic information' section. It contains three fields: 'Function name' with the value 'casestudy', 'Runtime' with a dropdown menu set to 'Python 3.12', and 'Architecture' with a radio button set to 'x86_64'. There are also 'Info' links for each section. At the bottom, there's a 'Permissions' section with a brief description of the default execution role.

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 [Info](#)
Enter a name that describes the purpose of your function.

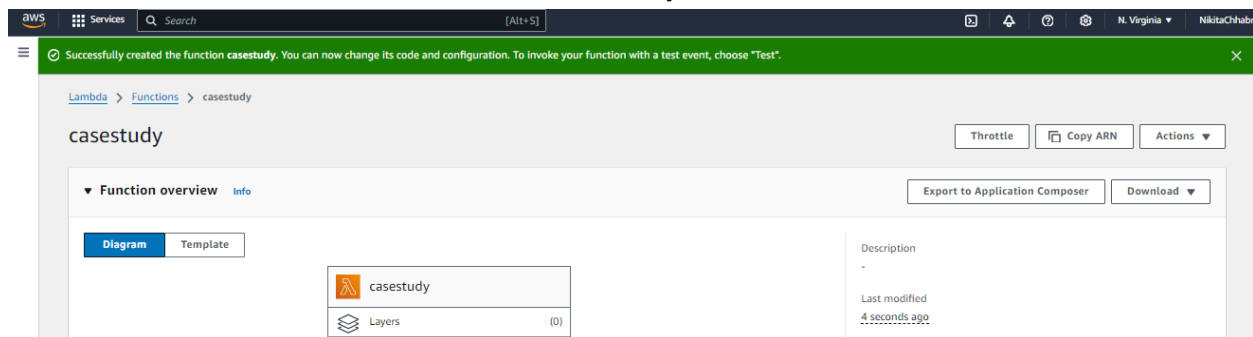
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

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

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.

STEP 4:The lambda function is created successfully.



STEP 5:

Write the Lambda Function Code:

In the "Function code" section, replace the default code with the following Python 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']

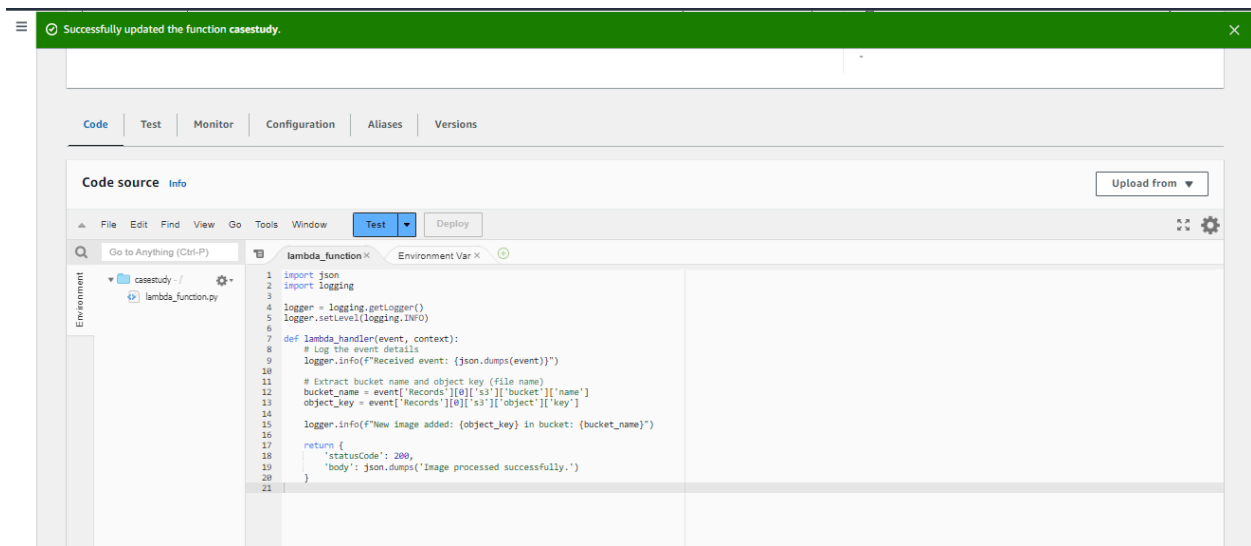
    logger.info(f"New image added: {object_key} in bucket:

    {bucket_name}") return {

        'statusCode': 200,

        'body': json.dumps('Image processed successfully.')

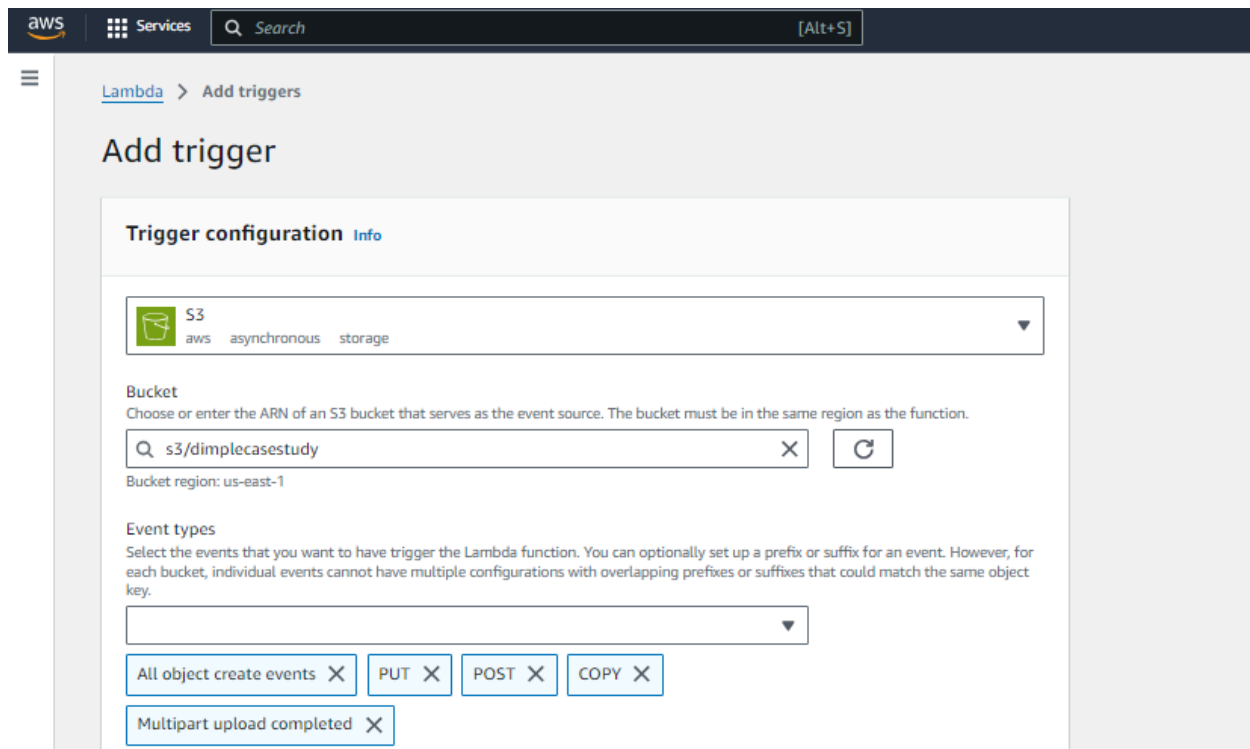
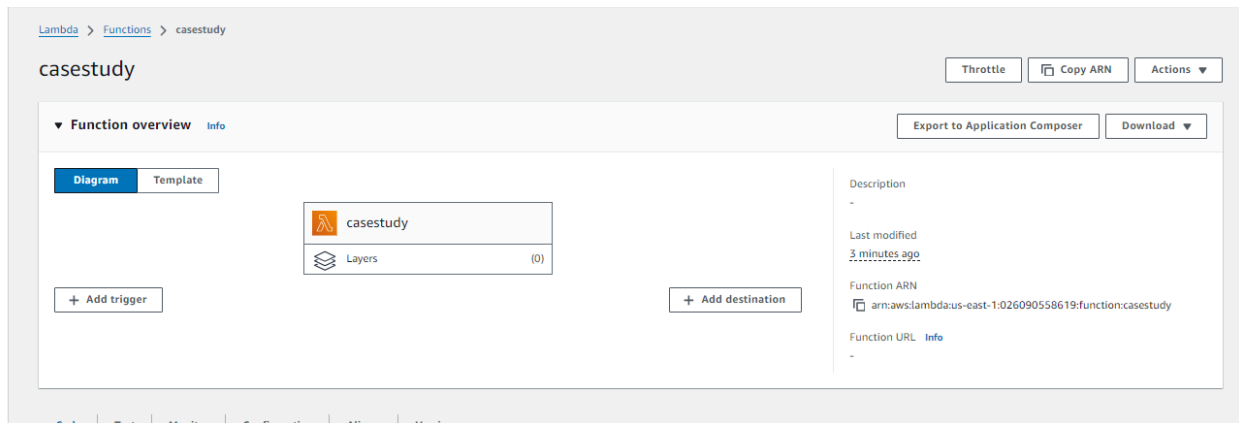
    }
```



STEP 6:

Add S3 Trigger to Lambda Function:

- Go back to your Lambda function.
- Click on "Add trigger" under the "Designer" section.
- Select "S3" as the trigger.
- Choose the bucket you created earlier.
- Select "All object create events".
- Click "Add".



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.

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.

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

Services

Search

[Alt+S]

N. Virginia

NikitaChhabra

Lambda > Functions > casestudy

casestudy

Throttle Copy ARN Actions

The trigger dimplecasestudy was successfully added to function casestudy. The function is now receiving events from the trigger.

Function overview

Diagram Template

casestudy

Layers (0)

S3

+ Add trigger

+ Add destination

Description

-

Last modified

5 minutes ago

Function ARN

arn:aws:lambda:us-east-1:026090558619:function:casestudy

Function URL

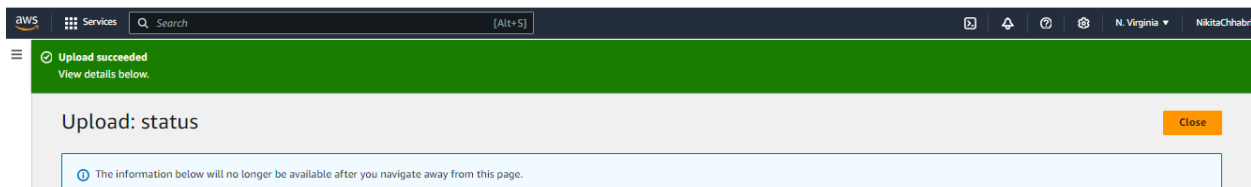
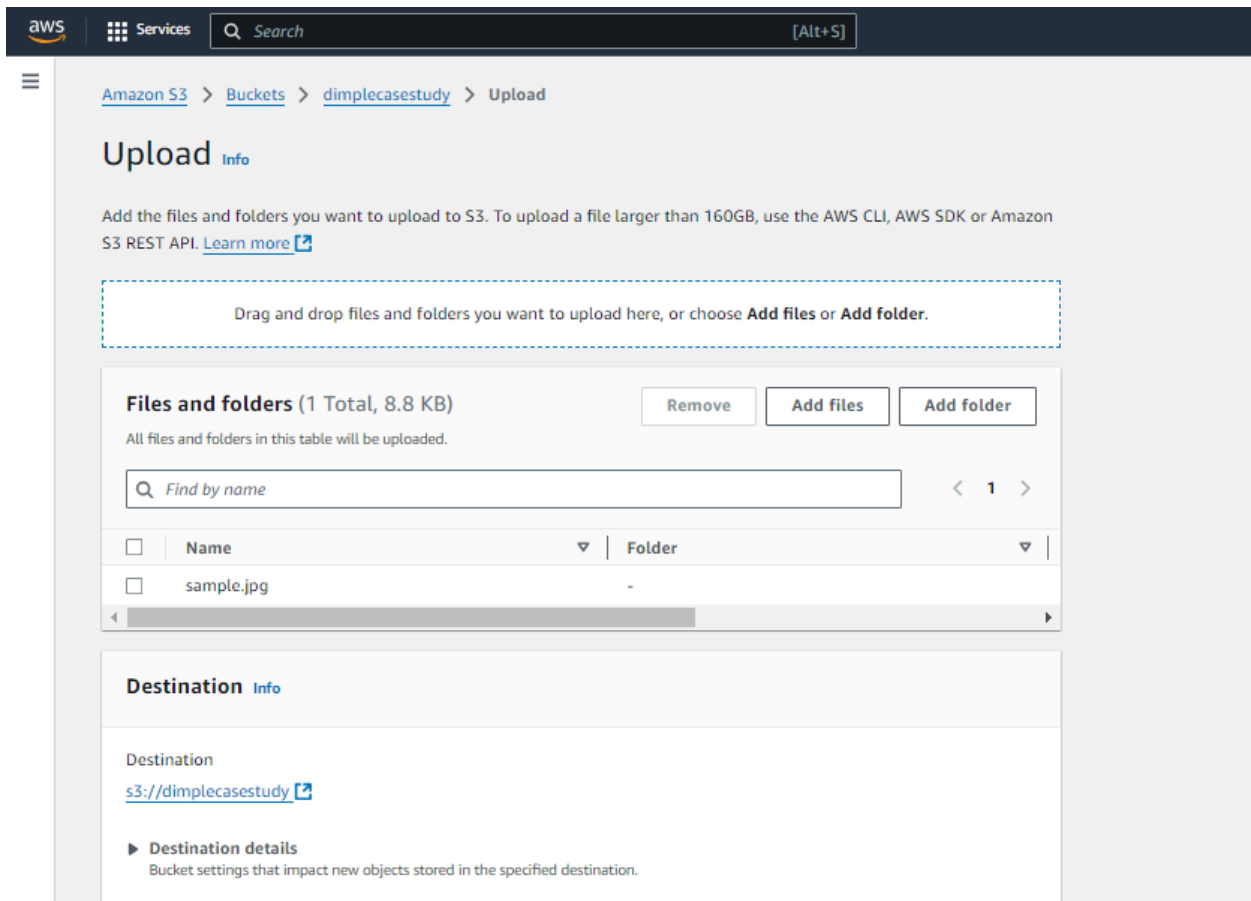
Info

STEP IV :Upload a Sample Image to S3:

STEP 1:Go to the S3 bucket created earlier.

STEP 2:Click on "Upload" and select a sample image.

STEP 3: Click "Upload".



STEP V:Verify Lambda Execution:

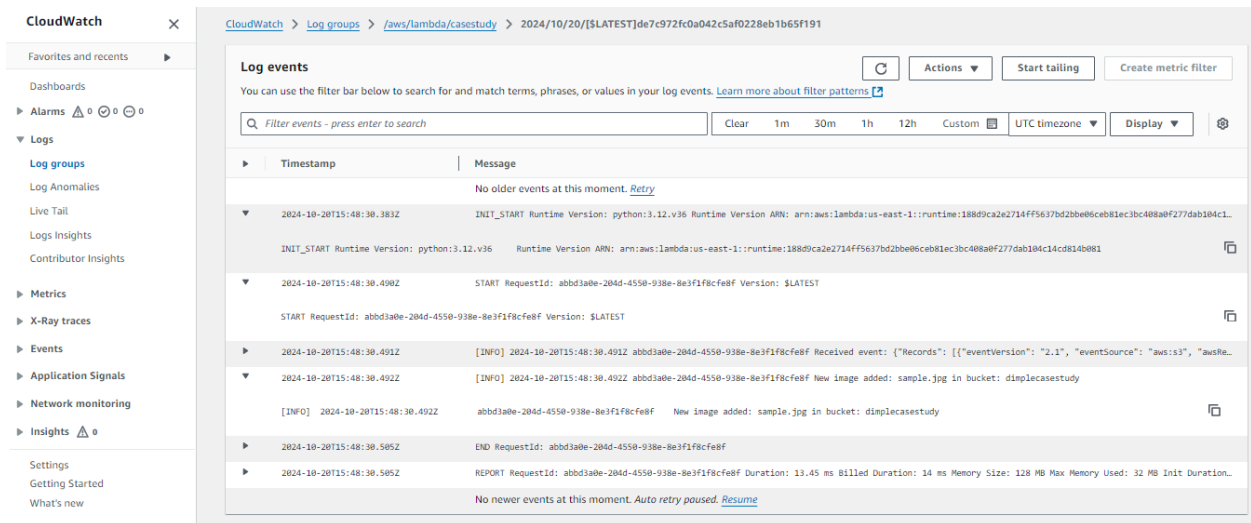
STEP 1:Go to CloudWatch Logs by searching for "CloudWatch" in the search bar and selecting it.

STEP 2:Click on "Logs" and find the log group for your Lambda function (e.g., /aws/lambda/ImageProcessor).

STEP 3:Verify that the logs show the details of the uploaded image.

The screenshot displays the AWS CloudWatch console interface. The top navigation bar shows the AWS logo, Services menu, a search bar, and the current region (N. Virginia) and user (NikitaCh). The left sidebar contains the CloudWatch logo and a navigation menu with options like Dashboards, Alarms, Logs, Metrics, X-Ray traces, Events, Application Signals, Network monitoring, and Insights. The main content area is titled '/aws/lambda/casestudy' and features a 'Log group details' section. This section provides information about the log group, including its class (Standard), ARN, creation time (1 minute ago), retention (Never expire), stored bytes, metric filters, subscription filters, contributor insights rules, KMS key ID, anomaly detection, data protection, and sensitive data count. Below the details, there are tabs for Log streams, Tags, Anomaly detection, Metric filters, Subscription filters, Contributor Insights, and Data protection. The 'Log streams (1)' tab is active, showing a table with one log stream. The table has columns for 'Log stream' and 'Last event time'. The log stream is named '2024/10/20/[LATEST]de7c972fc0a042c5af0228eb1b65f191' and its last event time is '2024-10-20 15:48:30 (UTC)'.

Log stream	Last event time
2024/10/20/[LATEST]de7c972fc0a042c5af0228eb1b65f191	2024-10-20 15:48:30 (UTC)



The screenshot displays the AWS CloudWatch console interface. On the left, a sidebar menu shows navigation options like 'Dashboards', 'Alarms', 'Logs', 'Metrics', 'Events', 'Application Signals', 'Network monitoring', and 'Insights'. The 'Logs' section is selected, and the 'Log groups' link is highlighted. The main panel shows the breadcrumb path: 'CloudWatch > Log groups > /aws/lambda/casestudy > 2024/10/20/[\${LATEST}]de7c972fca042c5af0228eb1b65f191'. Below this, there are buttons for 'Actions', 'Start tailing', and 'Create metric filter'. A search bar is present with the placeholder text 'Filter events - press enter to search'. The log events are displayed in a table with columns for 'Timestamp' and 'Message'. The events show the initialization of a Lambda function, including the runtime version (python:3.12.v36) and the function's ARN. The messages indicate the start of the function execution, the receipt of an event, the addition of a new image (sample.jpg) to the bucket 'dimplecasestudy', and the end of the function execution.

STEP VI:Create a Github Repository:

- Create Repository with name AWS-CodePipeline.
- Add the file buildspec.yml ,requirements.txt and lambda_function.py.

buildspec.yml file contains the code:

version: 0.2

phases:

install:

runtime-versions:

python: 3.12

build:

commands:

- echo "builld package"
- zip -r lambda_function.zip . # Packages the code

artifacts:

files:

- lambda_function.zip

requirements.txt file contains the code:

Example dependency\nrequests

lambda_function.py file contains the 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.')
```

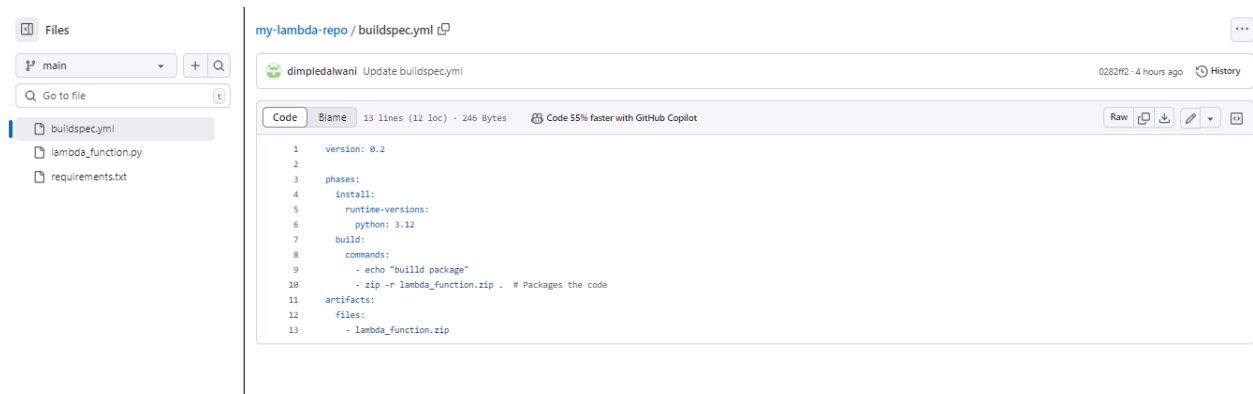
```
}
```

Name:DIMPLE DALWANI

Year:2024-2025

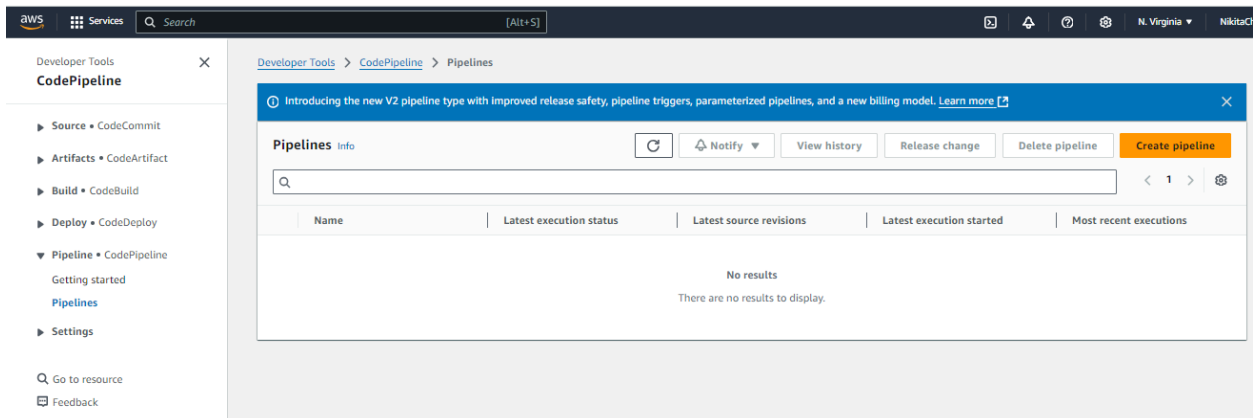
Division: D15C

Roll No: 8



STEP VII:Set Up AWS CodePipeline:

STEP 1:Go to CodePipeline service and click on create pipeline.



STEP 2:Choose creation option stage:

Choose “build custom pipeline” and then click on next.

The screenshot shows the AWS CodePipeline console interface for creating a new pipeline. The breadcrumb navigation at the top reads: [Developer Tools](#) > [CodePipeline](#) > [Pipelines](#) > Create new pipeline. On the left, a vertical sidebar lists the steps of the wizard: 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 an 'Info' link and indicates 'Step 1 of 6'. Below this, a box titled 'Creation options' contains the instruction 'Choose one of the following options to create your pipeline.' There are two options: 'Create pipeline from template' (with an unselected radio button) and 'Build custom pipeline' (with a selected radio button). The 'Build custom pipeline' option is highlighted with a blue border and includes the subtext 'Build a pipeline from scratch to meet your specific needs.' At the bottom right of the main area are 'Cancel' and 'Next' buttons, with 'Next' being orange and highlighted.

STEP 3:Choose pipeline setting stage:

Add the name of your pipeline click on existing role and then add your role arn and then click on next.

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.

casestudy

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::026090558619:role/dimple-role

Role name

Type your service role name

☒ Allow AWS CodePipeline to create a service role so it can be used with this new pipeline

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.

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

► Advanced settings

Cancel

Previous

Next

STEP 4:Add source stage:

Select “Github (Version2)” this will redirect you to another page and enter your connection name and it will connect you to your github account and then add your repository name and the default branch name and then click on next.

The screenshot shows the AWS CodePipeline console interface. The top navigation bar includes the AWS logo, 'Services', a search bar, and a '[Alt+S]' shortcut. The left sidebar displays a list of steps: 'Step 2: Choose pipeline settings', 'Step 3: Add source stage' (which is highlighted), 'Step 4: Add build stage', 'Step 5: Add deploy stage', 'Step 6: Review', and 'Review'. The main content area is titled 'Step 3 of 6' and 'Source'. It contains a 'Source provider' dropdown menu set to 'GitHub (Version 2)'. Below this is a blue information box titled 'New GitHub version 2 (app-based) action' with a message about creating a connection and a 'Learn more' link. The 'Connection' section shows a search bar with the text 'arn:aws:codeconnections:us-east-1:026090558619:connection/a1541214-89' and a 'Connect to GitHub' button. The 'Repository name' section has a search bar with the text 'dimpledalwani/my-lambda-repo'. The 'Default branch' section has a search bar with the text 'main'.

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-1:026090558619:connection/a1541214-89 or [Connect to GitHub](#)

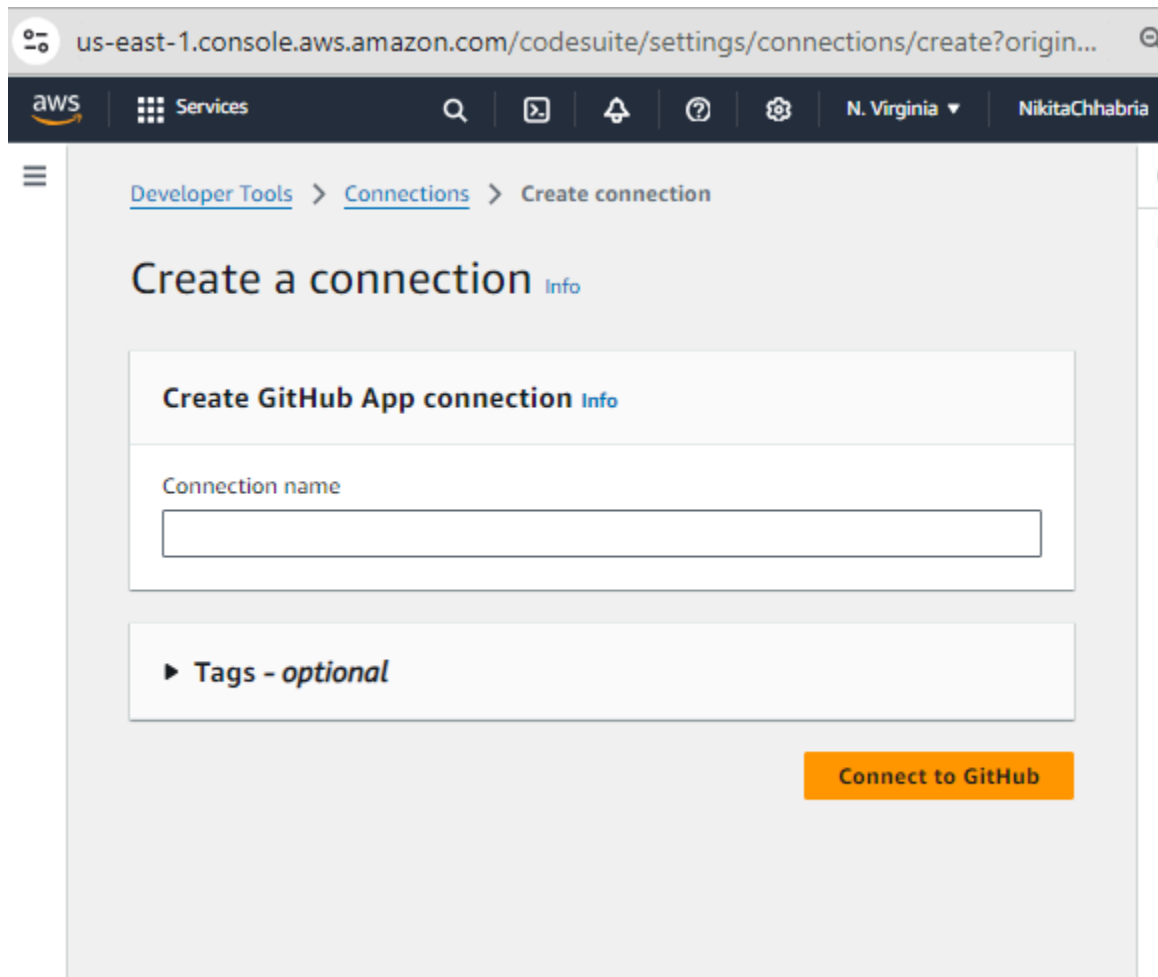
Repository name
Choose a repository in your GitHub account.

dimpledalwani/my-lambda-repo

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



The screenshot shows the AWS CodeSuite console interface for creating a new connection. The browser address bar displays the URL: `us-east-1.console.aws.amazon.com/codesuite/settings/connections/create?origin...`. The top navigation bar includes the AWS logo, a 'Services' menu, search and navigation icons, the active region 'N. Virginia', and the user name 'NikitaChhabria'. The breadcrumb trail indicates the path: `Developer Tools > Connections > Create connection`. The main heading is 'Create a connection' with an 'Info' link. Below this, there is a section titled 'Create GitHub App connection' with its own 'Info' link. A form field labeled 'Connection name' is present, followed by an optional section for 'Tags'. An orange button labeled 'Connect to GitHub' is located at the bottom right of the form area.

us-east-1.console.aws.amazon.com/codesuite/settings/connections/create?origin...

aws Services 🔍 📄 🔔 ⓘ ⚙️ N. Virginia NikitaChhabria

☰ Developer Tools > Connections > Create connection

Create a connection [Info](#)

Create GitHub App connection [Info](#)

Connection name

▶ Tags - *optional*

[Connect to GitHub](#)

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.

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


Exclude

File paths - optional

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

Include

Exclude

 You can add additional sources and triggers by editing the pipeline after it is created.

Cancel Previous Next

Step 5: 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.

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

casestudy

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

► Additional configuration

Description, Build badge, Concurrent build limit, tags

Environment

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 with this build project

arn:aws:iam::026090558619:role/dimple-role

► Additional configuration

Description: Build badges, Concurrent build limit, etc.

aws

Services

Search

[Alt+S]

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

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

☒ 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

aws Services Search [Alt+S]

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.

☐ 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*

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](#)

aws

Services

Search

[Alt+S]

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

Add build stage Info

Step 4 of 6

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.

or [Create project](#)

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](#)

Add environment variable

Build type

☒ Single build
Triggers a single build.

☐ Batch build
Triggers multiple builds as a single execution.

and then return to this task.

✕ or Create project [↗](#)

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](#) [↗](#)

Add environment variable

Build type

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

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

Region

▼

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

SourceArtifact ✕
Defined by: Source

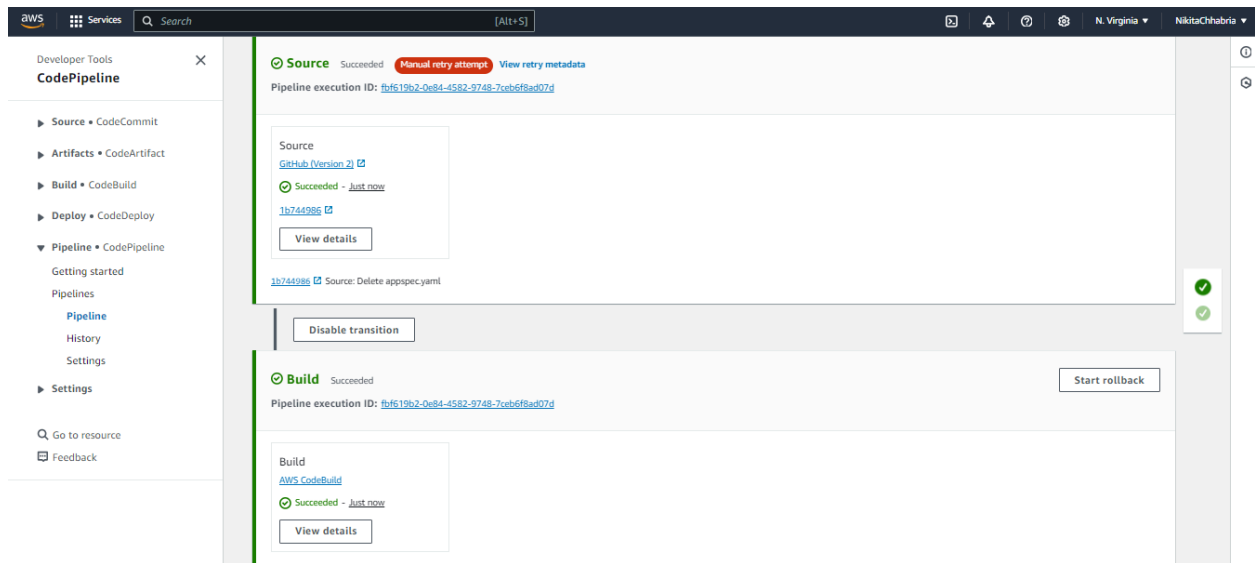
No more than 100 characters

☒ Enable automatic retry on stage failure

Cancel Previous Skip build stage Next

STEP 6:Add deploy stage:
Skip the deploy stage .

STEP 7:The CodePipeline is created successfully.



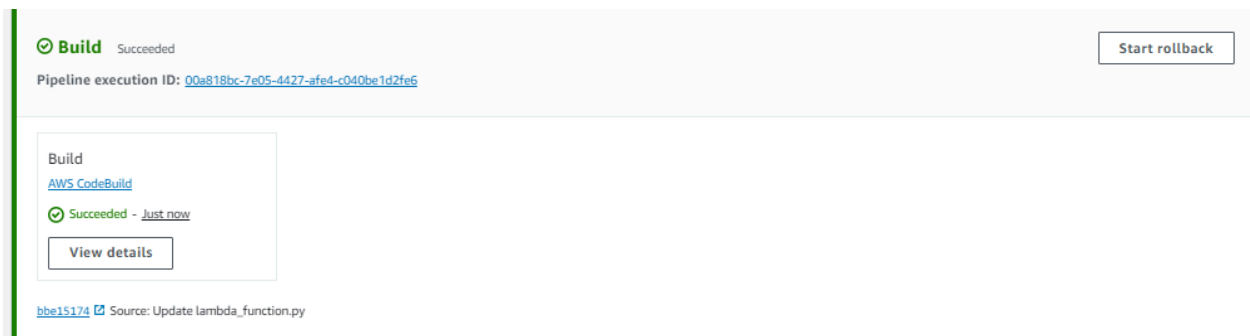
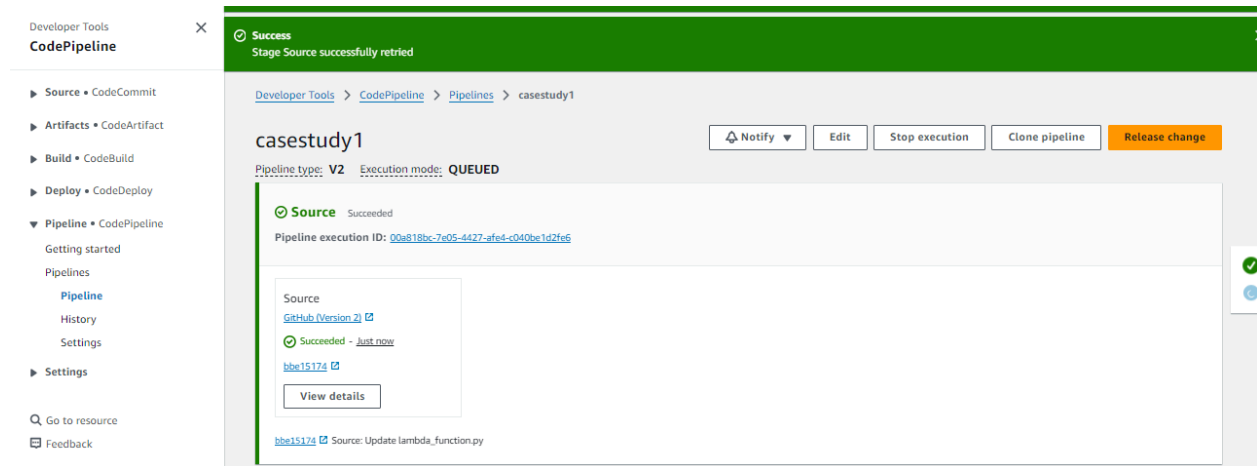
STEP VIII:Update Lambda Code and Test CodePipeline:

- Make a change to your Lambda function code in your repository.
- Commit and push the changes to the repository.
- Check CodePipeline to see the pipeline execution and verify that the Lambda function is updated.

```

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.')
20     }

```



Conclusion:Implementing a serverless image processing workflow with AWS Lambda, S3, and CodePipeline provides an efficient, scalable, and cost-effective solution for real-time image processing tasks. By leveraging the event-driven architecture of AWS Lambda, images uploaded to S3 buckets can be processed automatically without the need for manual intervention or server management. The integration with AWS CodePipeline ensures that any updates to the Lambda function are seamlessly deployed, enabling continuous integration and delivery. This setup is highly versatile and can be adapted for various applications such as image resizing, metadata extraction, content moderation, and thumbnail generation, making it an ideal choice for modern, cloud-based image processing needs.

GUIDELINES:Using AWS Personal for setting up your serverless image processing workflow involves creating and configuring IAM roles to ensure seamless integration and operation of various AWS services like Lambda, S3, CodeBuild, and CodePipeline. The IAM role is essential as it provides the necessary permissions for these services to interact with each other. Without the correct permissions, the workflow components will fail to trigger or deploy, causing the entire system to malfunction.

To ensure proper access, attach specific policies to the IAM role. These policies should include `AWSLambda_FullAccess`, `AmazonS3FullAccess`, `AmazonCodeBuildAdminAccess`, `AmazonCodePipeline_FullAccess`, and `CloudWatchLogsFullAccess`. These permissions allow Lambda to execute code in response to events, S3 to store and retrieve images, CodeBuild to build and package the Lambda function, and CodePipeline to automate the deployment process. Additionally, the `CloudWatchLogsFullAccess` policy enables logging and monitoring of the functions and processes involved.

It is also crucial to configure the trust relationship for the IAM role to include the services `codepipeline.amazonaws.com` and `codebuild.amazonaws.com`. This configuration allows CodePipeline and CodeBuild to assume the role and perform the necessary actions within your AWS environment. By setting up these permissions and trust relationships correctly, you ensure that your serverless workflow operates smoothly and efficiently, with all components having the access they need to function properly.