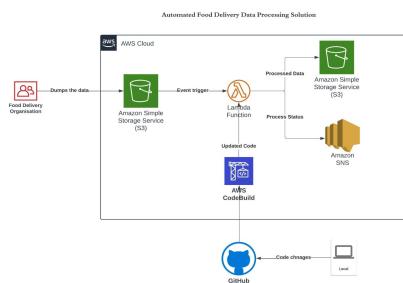


# Streamlining Delivery Data Processing with AWS

## Problem statement:

Create an automated AWS-based solution for processing daily delivery data from DoorDash. JSON files containing delivery records will be uploaded to an Amazon S3 bucket. An AWS Lambda function, triggered by the file upload, will filter the records based on delivery status and save the filtered data to another S3 bucket. Notifications regarding the processing outcome will be sent via Amazon SNS. Lambda should have CI/CD integration using CodeBuild so that any new requirements in processing can be quickly addressed.

## Architecture diagram:



## Step1:

Created JSON file, '2024-04-28-raw\_data.json' with json data.

```
{"id": 1, "status": "delivered", "amount": 20.5, "date": "2024-03-09"}  
{"id": 2, "status": "cancelled", "amount": 15.0, "date": "2024-03-09"}  
{"id": 3, "status": "order placed", "amount": 22.5, "date": "2024-03-09"}  
{"id": 4, "status": "delivered", "amount": 19.5, "date": "2024-03-09"}  
{"id": 5, "status": "cancelled", "amount": 18.0, "date": "2024-03-09"}  
{"id": 6, "status": "delivered", "amount": 23.5, "date": "2024-03-09"}  
{"id": 7, "status": "order placed", "amount": 20.0, "date": "2024-03-09"}  
{"id": 8, "status": "delivered", "amount": 25.5, "date": "2024-03-09"}  
{"id": 9, "status": "delivered", "amount": 21.5, "date": "2024-03-09"}  
{"id": 10, "status": "cancelled", "amount": 17.5, "date": "2024-03-09"}
```

## Step2:

Create two general-purpose Amazon S3 buckets. One designated for raw data landing and the other for processed data. Implement lifecycle rules to manage object storage, automatically transitioning objects between storage classes or deleting them after a specified number of days, aimed at cost optimization and audit trail maintenance.

Name	AWS Region	IAM Access Analyzer	Creation date
doordash-target-bucket	Asia Pacific (Mumbai) ap-south-1	<a href="#">View analyzer for ap-south-1</a>	April 27, 2024, 15:33:20 (UTC+05:30)
doordash-landing-bucket	Asia Pacific (Mumbai) ap-south-1	<a href="#">View analyzer for ap-south-1</a>	April 27, 2024, 15:32:20 (UTC+05:30)

## Step3:

Create an AWS Simple Notification Service (SNS) topic with a subscription using Email as the protocol for the endpoint.

Configure the Lambda function to send a notification to this SNS topic once the processing is completed, updating the status of the process to stakeholders via email.

The screenshot shows the 'DoorDashUpdate' SNS topic configuration page. The top navigation bar includes links for 'Amazon SNS > Topics > DoorDashUpdate'. The main section is titled 'DoorDashUpdate' with three buttons: 'Edit', 'Delete', and 'Publish message'. Below this is a 'Details' card containing fields: 'Name' (DoorDashUpdate), 'Display name' (empty), 'ARN' (arn:aws:sns:ap-south-1:...DoorDashUpdate), and 'Topic owner' (a placeholder icon). At the bottom of the card are tabs for 'Subscriptions', 'Access policy', 'Data protection policy', 'Delivery policy (HTTP/S)', 'Delivery status logging', 'Encryption', 'Tags', and 'Integrations'. The 'Subscriptions' tab is selected, showing one entry: 'Subscriptions (1)'. A search bar is present above the table. The table columns are 'ID', 'Endpoint', 'Status', and 'Protocol'. The single row shows an ID 'e2998f16-7222-478f-a113-148db11b891e', a blurred Endpoint, a 'Confirmed' Status, and an EMAIL Protocol. There are buttons for 'Edit', 'Delete', 'Request confirmation', 'Confirm subscription', and 'Create subscription'.

## Step4:

Create a Lambda function and configure it with an S3 trigger to be invoked when an object with the suffix "Raw\_Data" and prefix ".json" is uploaded to the dorrdash-landing S3 bucket.

The screenshot shows the AWS Lambda console. In the top navigation bar, the 'Diagram' tab is selected. Below the tabs, there's a search bar and a 'Layers' section. On the left, there's a sidebar with various configuration options like General configuration, Triggers, Permissions, Destinations, Function URL, Environment variables, Tags, VPC, RDS databases, Monitoring and operations tools, and Concurrency. The 'Configuration' tab is currently active. Under 'Triggers', there is one entry: 'S3: doordash-landing-bucket'. This entry includes details such as Bucket arn: arn:aws:s3:::doordash-landing-bucket, Event types: s3:ObjectCreated:, isComplexStatement: No, Notification name: 58b16df1-9c86-4e2d-853b-1c70b1586cc3, Prefix: Raw\_data/, Service principal: s3.amazonaws.com, Source account: [REDACTED], Statement ID: lambda-ee3b60de-22b9-47bb-9d85-d4c706da219c, and Suffix: .json. To the right of the trigger list are buttons for 'Edit', 'Delete', and 'Add trigger'.

## Step5:

Update the Lambda function's execution role permissions to include the necessary permissions to read data from the S3 bucket, publish messages through SNS, and write logs to CloudWatch.

The screenshot shows the AWS IAM Roles page. A role named 'DoorDashDelivery-role-5c5plh6i' is selected. The 'Summary' tab is active, displaying information like Creation date (April 27, 2024, 15:30 (UTC+05:30)), Last activity (4 days ago), ARN (arn:aws:iam: [REDACTED] role/service-role/DoorDashDelivery-role-5c5plh6i), and Maximum session duration (1 hour). Below the summary, the 'Permissions' tab is selected, showing four attached policies: AmazonS3FullAccess, AmazonSNSFullAccess, AWSLambdaBasicExecutionRole-d97ec2d6-fe32-4def-bd30-f..., and CloudWatchLogsFullAccess. There are buttons for 'Edit', 'Delete', 'Simulate', 'Remove', and 'Add permissions'.

## Step6:

Store the final bucket name where the processed data has to be dumped and the SNS ARN in the Lambda's environmental variables to avoid hardcoding.

## DoorDashDelivery

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

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

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

Description  
-

Last modified  
2 hours ago

Function ARN  
[arn:aws:lambda:ap-south-1:123456789012:function:DoorDashDelivery](#)

Function URL [Info](#)  
-

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

[+ Add destination](#)

[+ Add trigger](#)

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

General configuration	Environment variables (2)	Edit						
Triggers	The environment variables below are encrypted at rest with the default Lambda service key.							
Permissions								
Destinations								
Function URL								
<b>Environment variables</b>	<table border="1"> <thead> <tr> <th>Key</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>TopicArn</td> <td>arn:aws:sns:ap-south-1:123456789012:DoorDashUpdate</td> </tr> <tr> <td>output_bucket</td> <td>doordash-target-bucket</td> </tr> </tbody> </table>		Key	Value	TopicArn	arn:aws:sns:ap-south-1:123456789012:DoorDashUpdate	output_bucket	doordash-target-bucket
Key	Value							
TopicArn	arn:aws:sns:ap-south-1:123456789012:DoorDashUpdate							
output_bucket	doordash-target-bucket							

**Step 7:**

To implement continuous integration and continuous delivery (CI/CD) for the Lambda function:

- You would set up a pipeline that automatically builds, tests, and deploys changes to your Lambda function's code. This pipeline typically involves using AWS CodeBuild to compile and package your Lambda function code, along with any necessary dependencies. Once the code is built, it can be automatically deployed to your Lambda function. This CI/CD pipeline ensures that any updates or changes to your Lambda function can be quickly and reliably deployed to production, allowing you to iterate on your code efficiently and respond to new requirements or updates as they arise.

To achieve this:

- Create a GitHub repository to host your Lambda function code and the build file required by CodeBuild.
- Create a project on CodeBuild with the following parameters:
  - Use the GitHub repository created above as the source.
  - Configure CodeBuild to trigger builds automatically whenever a pull request is merged into the main branch of the repository.
  - Ensure that the environment parameters in CodeBuild match those used for the Lambda function to prevent build failures.
  - Select the option to use a buildspecfile.yml containing sequential steps for the entire build process, defining how CodeBuild should build, test, and deploy the Lambda function code.
- Create another general-purpose Amazon S3 bucket to store the code after it is built by CodeBuild. This bucket will serve as the source for updating the Lambda function.
- Update the CodeBuild's execution role by adding the necessary permissions, including the ARNs of the services.

**Project configuration**

**Source**

**Source 1 - Primary**

Source provider: GitHub

Repository: doordashcodebuild

Build events:

- Source code changes
- Refetch every time a code change is pushed to this repository

Build type: Single build

Webhook event filter groups

Filter group 1: Event type: optional

Start a build under these conditions - optional

Don't start a build under these conditions - optional

**Environment**

Provisioning model info:

- On-demand: Create new build infrastructure in response to each build request
- Reserved capacity: Reserve a specific amount of resources for builds. A fixed number of builds will run simultaneously.

Execution image:

- Image managed by AWS CodeBuild
- Custom image

Compute:

- AZ: Preferred for flexibility during action runs
- Lambda: Preferred for speed and minimizes the start up time of Lambda actions

Operating system: Amazon Linux

Rooted build:

- Standard
- Amazon Linux 2 (2023.09.0)
- Amazon Linux 2 (2023.09.0)
- Always use the latest image for this runtime version

Additional configuration: Cache dependencies, VPC, environment variables, file systems

**Buildspec**

Build specifications:

- Insert build commands
- Use a buildspec file: Provide a buildspec file in a local, mounted buildspec file

Buildspec file - optional:

Batch configuration

Define batch configuration - optional

Artifacts

Artifact 1 - Primary

Type: Docker artifacts

Additional configuration: Cache dependencies

Logs

CloudWatch logs - optional:

Group name - optional

Stream name prefix - optional

S3 logs - optional:

Service role permissions

Service role: doordashcodebuild

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

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:logs:ap-south-1:log-group:/aws/codebuild/DoorDashLambdaFunction",
        "arn:aws:logs:ap-south-1:log-group:/aws/codebuild/DoorDashLambdaFunction-*"
      ],
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ]
    },
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::codepipeline-ap-south-1-*",
        "arn:aws:s3:::doordashcodebuildbucket",
        "arn:aws:s3:::doordashcodebuildbucket/*"
      ],
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:CreateReportGroup",
        "codebuild:CreateReport",
        "codebuild:UpdateReport",
        "codebuild:BatchPutTestCases",
        "codebuild:BatchPutCodeCovverages"
      ],
      "Resource": [
        "arn:aws:codebuild:ap-south-1:report-group/DoorDashLambdaFunction-*"
      ]
    },
    {
      "Sid": "Statement1",
      "Effect": "Allow",
      "Action": [
        "lambda:AddPermission",
        "lambda:RemovePermission",
        "lambda>CreateAlias",
        "lambda:UpdateAlias",
        "lambda>DeleteAlias",
        "lambda:UpdateFunctionCode",
        "lambda:UpdateFunctionConfiguration",
        "lambda:PutFunctionConcurrency",
        "lambda:DeleteFunctionConcurrency",
        "lambda:PublishVersion"
      ],
      "Resource": [
        "arn:aws:lambda:ap-south-1:function:DoorDashDelivery"
      ]
    }
  ]
}
```

Name	AWS Region	IAM Access Analyzer	Creation date
doordashcodebuildbucket	Asia Pacific (Mumbai) ap-south-1	<a href="#">View analyzer for ap-south-1</a>	April 27, 2024, 20:09:08 (UTC+05:30)

## Let's go through the lambda\_function.py and buildspec.yml files on the DoorDash repository:

### Lambda function:

- It extracts the bucket name and object key from the event parameter when triggered.
- Utilizing boto3, it retrieves the data of the JSON object, filters records with a "delivered" status, and converts them into a CSV format.
- It uploads the CSV file to the target bucket using boto3.
- In case of any errors during the upload process, it publishes a message to an SNS topic to notify the stakeholders.
- Upon successful upload, it publishes a message to an SNS topic to inform stakeholders about the completion of the process.
- Logs are utilized extensively throughout the function for monitoring and debugging purposes, providing insights into the execution flow and any encountered errors.

### Buildspec:

- Version and Phases Declaration:
  - Version is set to 0.2.
  - Phases are defined: install, build, and post\_build.
- Install Phase:
  - Set the Python runtime version to 3.12.
  - Install dependencies listed in the requirements.txt file into the lib directory.
- Build Phase:
  - Zip the contents of the lib directory to create a deployment package.
  - Add the lambda\_function.py file to the deployment package.
- Post\_build Phase:
  - Upload the deployment package to the S3 bucket doordashcodebuildbucket.
  - Update the Lambda function DoorDashDelivery using the deployment package stored in the S3 bucket.
  - Print a message indicating the completion of the deployment process.

requirements.txt file contains all the dependencies used in the lambda\_function.

After setting up everything, After merging changes to the main branch on GitHub, the configured CI/CD pipeline automatically initiates a code build process. The DoorDashDelivery lambda function gets updated.

Now, upload the 2024-04-28-raw\_data.json file to the landing bucket. DoorDashDelivery lambda would be automatically triggered to process the data, and this is how the final notification

