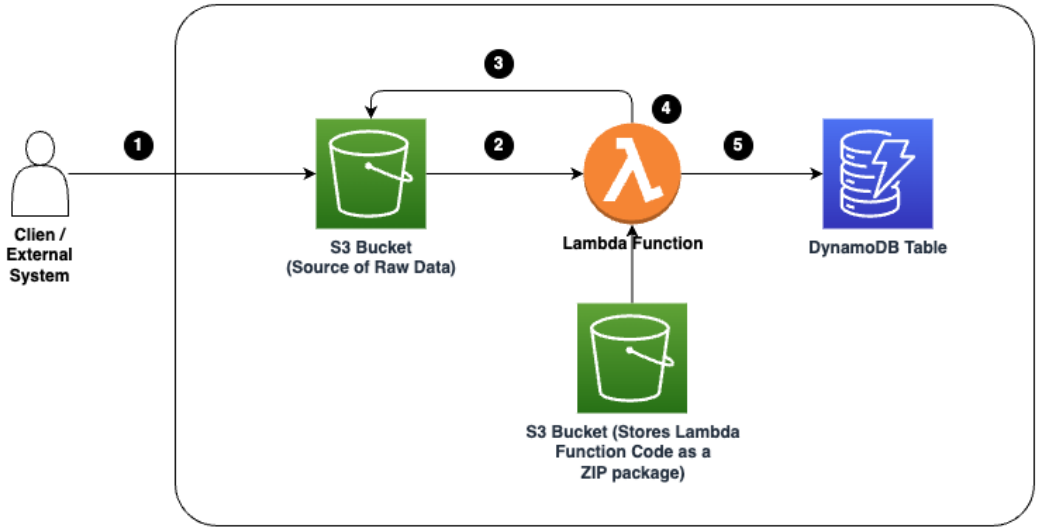
A Serverless data processing pipeline, using AWS S3, DynamoDB and Lambda services

Introduction

In this architecture, a serverless data pipeline is described. It is capable of,

- Fetch raw data from an S3 bucket,
- Process them accordingly
- Export processed data into a DynamoDB table

Architecture Diagram and High Level Overview



1. Client / External system uploads the CSV file into Source S3 bucket
2. S3 bucket triggers an event, and sends details of the newly created file into the Lambda function
3. Lambda function fetches the CSV file
4. Lambda function processes the CSV file accordingly
5. Stores the processed data in DynamoDB table

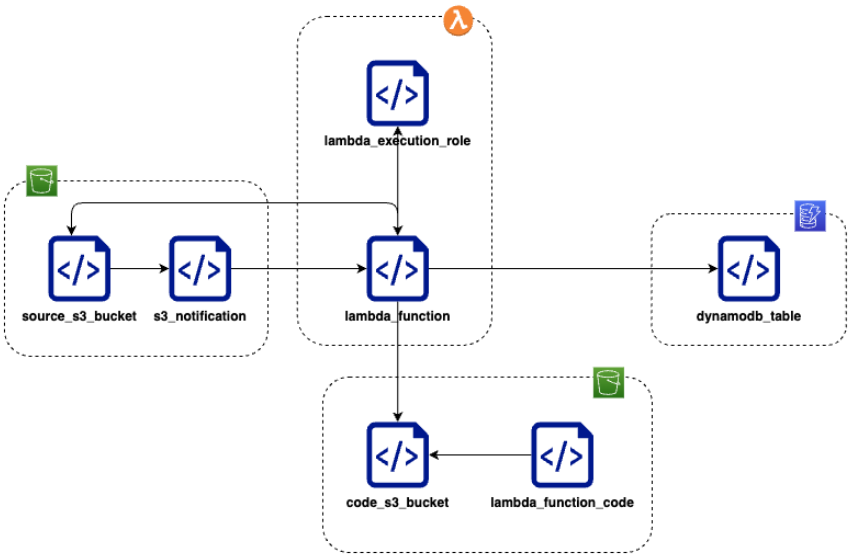
Terraform can be used to deploy the whole infrastructure, while the lambda function is written in Python

Terraform Script

The Terraform script is written in a modular structure, which allows the user to manage and scale the infrastructure easily.

All the variable values are stored in a separate file so that it might help if the Terraform code is migrated to Terraform cloud later.

Modular structure of the Terraform Script :



Structure of the Code base :

Module Explanation

S3 Module

This module is to deploy the S3 buckets which act as the data source, and hosts the Lambda function code.

This Module contains following resources/configurations :

- S3 Bucket
- S3 Bucket versioning
- S3 Bucket public access block
- Ownership controls
- S3 Bucket policy

S3Notification Module

Purpose of this module is to create the S3 bucket notification, which required to trigger the lambda function, when a new CSV file was uploaded. Apart from the S3 bucket notification, there is another resource in this module called `aws_lambda_permission` which gives S3 bucket permission to invoke the Lambda function

S3Object Module

The ZIP package that contains the Lambda function, is considerably large (approximately 60MB). AWS Lambda function can hold a ZIP files no larger than 50MB. Therefore, any ZIP package larger than 50MB must be stored in a S3 bucket. The

purpose of this module is to upload the ZIP package to the S3 bucket that contains the code

(Please refer to the **Deployment package (.zip file archive) size** of this official documentation : https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html)

LambdaFunction Module

As the name implies, this module contains the lambda function resource.

Specifications of the Lambda function :

Runtime	Python 3.8
Architecture	arm64

The parameters such as source bucket name and the destination DynamoDB table name are fetched in to the lambda function as environment variables.

In the lambda function, the `batch_writer()` function is used, in order to minimise the number of API calls made to the DynamoDB table

ExecutionRole Module

This holds the necessary permissions for the Lambda function to,

- Pull source CSV file from the S3 bucket
- Upload processed data into DynamoDB table
- Create CloudWatch log groups and log streams

For each action above mentioned, least permission has been granted as an AWS best practice

DynamoDBTable Module

Responsible for creating DynamoDB table.

Specs of the DynamoDB table :

Billing Mode	Provisioned
Read Capacity	5
Write Capacity	5

Commands to Deploy and Destroy the infrastructure

Initialise the Terraform Project

```
terraform init
```

View the resources to be created

```
terraform plan
```

Create Resources

```
terraform apply
```

Destroy the infrastructure

```
terraform destroy
```

Limitations

- The size of the ZIP package must be less than 50MB if it is uploaded directly to the Lambda function. Therefore, the ZIP package has to be uploaded to a separate S3 bucket