MODULO 09 - EJERCICIO 07-A

ALEXIS YURI M.

Diseña una solución basada en AWS Lambda que se dispare automáticamente cuando se suba un archivo a un bucket S3. La función debe procesar los datos (por ejemplo extraer campos relevantes o convertir el formato) y dejar un archivo transformado en otro bucket.

Diseño del Flujo Serverless: Ingesta y Validación de Datos para un Data Lake.

Se necesita un sistema que, cada vez que un archivo (por ejemplo, un CSV de ventas) sea subido al bucket data\_lake\_inicial, lo valide, lo comprima en formato Parquet, lo mueva al bucket data\_lake\_procesado y notifique el resultado.

Paso 1: Arquitectura Serverless y Disparador (Trigger).

Se utiliza la arquitectura orientada a eventos (event-driven) donde la función Lambda sólo se ejecuta en respuesta a una acción específica.

Componente Principal: Función AWS Lambda (en Python).

Disparador (Trigger): Se configura el bucket de origen (data\_lake\_inicial) para que emita un evento de creación de objeto en S3 cada vez que se carga un nuevo archivo.

Flujo: El evento S3 llama a la función Lambda, pasándole como payload la ubicación exacta del archivo recién cargado (bucket y clave).

Paso 2: Lógica de la Función Lambda (Procesamiento).

La función debe encapsular la lógica de Extracción y Transformación para la ingesta de datos:

- Extracción: La función utiliza el módulo boto3 para descargar el archivo de datos desde el bucket de origen S3 al entorno de ejecución temporal (/tmp) de Lambda.

- Validación: Se valida el archivo (ej. se comprueba el número de columnas o la presencia de encabezados). Si la validación falla, se mueve el archivo a un bucket de errores (data\_lake\_errores) y se notifica al equipo.

- Transformación: Si la validación es exitosa, se usa una biblioteca de procesamiento de datos (como Pandas dentro de Lambda) para:

- Aplicar limpieza básica (ej. eliminar valores nulos).

- Convertir los datos al formato Parquet, optimizado para el Data Lake.

- Carga: El archivo Parquet transformado se sube al bucket de destino (data\_lake\_procesado).

Estructura básica del código y Permisos.

La función Lambda debe estar escrita en Python (usando la librería boto3 para interactuar con AWS) y, preferiblemente, usar librerías de procesamiento de datos como Pandas.

import json

import os

import boto3

s3 = boto3.client('s3')

# Variables de entorno definidas en la configuración de Lambda

BUCKET\_DESTINO = os.environ.get('BUCKET\_PROCESADO')

BUCKET\_ERRORES = os.environ.get('BUCKET\_FALLIDO')

def validar\_y\_transformar(df):

# 1. Se valida el esquema.

if 'id\_cliente' not in df.columns:

raise ValueError("Error de esquema: Falta la columna 'id\_cliente'")

# 2. Se eliminan filas con los valores nulos.

df = df.dropna(subset=['monto\_venta'])

# 3. Se transforma a formato Parquet.

ruta\_tmp\_parquet = '/tmp/datos\_procesados.parquet'

df.to\_parquet(ruta\_tmp\_parquet, index=False)

return ruta\_tmp\_parquet

# Se crea Función que se dispara con el evento de carga de S3.

def lambda\_handler(event, context):

# 1. Se extraen detalles del evento S3.

try:

# Se extrae el bucket y la clave del archivo cargado.

bucket\_origen = event['Records'][0]['s3']['bucket']['name']

clave\_origen = event['Records'][0]['s3']['object']['key']

nombre\_archivo\_parquet = clave\_origen.replace('.csv', '.parquet')

ruta\_tmp\_csv = f'/tmp/{clave\_origen}'

# 2. Se descarga el archivo a /tmp

s3.download\_file(bucket\_origen, clave\_origen, ruta\_tmp\_csv)

# 3. Se lee el archivo usando Pandas.

# df = pd.read\_csv(ruta\_tmp\_csv)

# print(f"Archivo leído. Filas: {len(df)}")

# 4. Se valida y transforman los datos.

# ruta\_final = validar\_y\_transformar(df)

return {

'statusCode': 200,

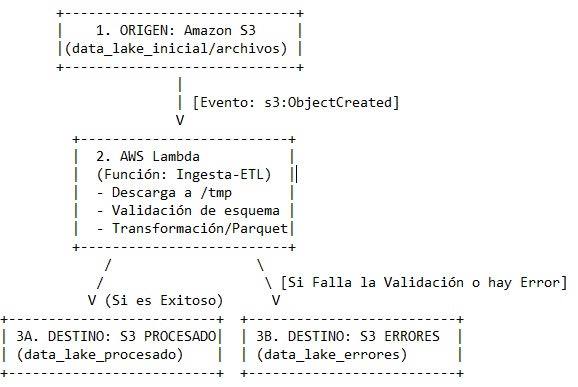
'body': json.dumps('Proceso ETL Serverless completado con éxito.')

}

La función AWS Lambda debe asumir un Rol de Ejecución (Execution Role) que le otorgue los permisos mínimos necesarios para interactuar con los servicios utilizados.



Esquema del Flujo Serverless.



La justificación del diseño serverless se resume en su eficiencia operativa y económica, ya que el proceso está construido para ser orientado a eventos, altamente escalable y de bajo costo sin requerir administración de infraestructura.

- Costo y Eficiencia Serverless: Sólo se paga por el tiempo de cómputo exacto que utiliza AWS Lambda (a nivel de milisegundos), eliminando el costo fijo y el tiempo de inactividad de servidores tradicionales (EC2 o EMR).

- Diseño Orientado a Eventos: El proceso se activa automáticamente (s3:ObjectCreated) solo cuando llega un archivo, sin necesidad de un chequeo constante, garantizando una respuesta inmediata a la carga de datos.

- Escalabilidad Instantánea: Lambda escala la capacidad de procesamiento en paralelo de forma automática, pudiendo manejar alzas de carga de miles de archivos sin que el equipo de ingeniería deba aprovisionar recursos.

- Calidad de Datos: La función centraliza la validación y transformación a Parquet, asegurando que los datos que llegan al Data Lake procesado ya estén limpios y en un formato optimizado para el análisis posterior.