COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI ICADE CIHS

Sprint 2

Grupo 11

Tecnologías de Procesamiento Big Data
3º Grado en Ingeniería Matemática e Inteligencia Artificial

Índice

Introducción

Esta práctica, al igual que la anterior, se va a desarrollar en AWS; esta vez, utilizando el servicio AWS Glue, que es un servicio cuya especialidad es la integración de datos.
Dentro de AWS Glue, nos centraremos en usar AWS Glue Data Catalog para almacenar los datos en tablas y AWS Glue Crawler que es la herramienta encargada de, basada en fuentes de datos, organizar los datos dentro del Data Catalog.

Durante este sprint los principales objetivos van a ser organizar los datos ya almacenados en el S3 en un catálogo de datos y familiarizarnos con los conceptos de AWS Data Catalog y AWS Crawler.

Este tipo de problemas es interesante abordarlos ya que optimiza de manera rápida y eficiente la organizacion de nuestros datos pensando en posteriores consultas sobre estos.

Metodología

Para llevar a cabo este sprint, hemos utilizado Python junto la librería boto3 para interactuar con los servicios de AWS. El entorno de ejecución se compone de AWS Glue para la catalogación de datos, AWS S3 para el almacenamiento, AWS Glue Crawler para la exploración de los datos e IAM para la gestión de permisos y acceso.
En la solución diseñada, en primer lugar, se configura la conexión con AWS Glue mediante la inicialización de un cliente glue_client, especificando la región de AWS (Figura 1) y las credenciales de acceso. Posteriormente, se crea la base de datos en AWS Glue Data Catalog, (Figura 2), donde se almacenarán los datos de S3 en tablas. Por último, se configura un Crawler en AWS Glue (Figura 3) y se ejecuta para poder descubrir de forma automática la estructura de los datos almacenados en S3, y así poder crear las tablas dentro de Glue Data Catalog.

```
glue_client = boto3.client('glue', region_name=AWS_REGION,
                           aws_access_key_id = aws_access_key_id,
                           aws_secret_access_key = aws_secret_access_key,
                           aws_session_token = aws_session_token
                           )
```

Figura 1 Creación glue client

```python
def create_database():
    try:
        glue_client.create_database(
            DatabaseInput={
                'Name': DATABASE_NAME,
                'Description': 'Base de datos para almacenar metadatos de datos históricos en S3'
            }
        )
        print(f"Base de datos '{DATABASE_NAME}' creada.")
    except glue_client.exceptions.AlreadyExistsException:
        print(f"La base de datos '{DATABASE_NAME}' ya existe.")
```

Figura 2. Creación Base de datos

```python
def create_crawler():
    try:
        glue_client.create_crawler(
            Name=CRAWLER_NAME,
            Role=IAM_ROLE,
            DatabaseName=DATABASE_NAME,
            Targets={'S3Targets': [{'Path': S3_TARGET_PATH}]},
            TablePrefix="trade_data_"
        )
        print(f"Crawler '{CRAWLER_NAME}' creado.")
    except glue_client.exceptions.AlreadyExistsException:
        print(f"El crawler '{CRAWLER_NAME}' ya existe.")
```

Figura 3. Creación crawler

```python
def start_crawler():
    glue_client.start_crawler(Name=CRAWLER_NAME)
    print(f"Crawler '{CRAWLER_NAME}' iniciado.")
```

Figura 4. Inicialización crawler

Resultados

El script que hemos implementado ha dado este resultado. Al estar ya subido nos ha devuelto esto, pero si se hiciese desde un inicio devolvería "Base de datos creada", "Crawler creado", "Crawler iniciado".

```
JoseJuan@MacBook-Pro HU-3 % /usr/local/bin/python3 /Users/JoseJuan/Documents/IMAT_3/BigData2/FinalProject/HU-3/hu3.py
La base de datos 'trade_data_imat3a11' ya existe.
El crawler 'trade_data_crawler' ya existe.
Crawler 'trade_data_crawler' iniciado.
```

Este es el nombre de nuestra base de datos "trade_data_imat3a11" y tiene la siguiente estructura:

trade_data_imat3a11 — Base de datos para almacenar metadatos d - — February 5, 2025 at 18:40:21

Schema (10)

View and manage the table schema.

Edit schema as JSON | Edit schema

Filter schemas

#	Column name	Data type	Partition key	Comment
1	datetime	string	-	-
2	symbol	string	-	-
3	open	double	-	-
4	high	double	-	-
5	low	double	-	-
6	close	double	-	-
7	volume	double	-	-
8	partition_0	string	Partition (0)	-
9	partition_1	string	Partition (1)	-
10	partition_2	string	Partition (2)	-

Las primeras 7 columnas corresponden a la estructura del dataset, que es datetime, symbol, open, high, low, close y volume. Lo que nos ha sorprendido ha sido los partitions, sin embargo, luego nos hemos dado cuenta que son las carpetas de bitcoins, año y mes. Al ser carpetas jerarquicas, Glue interpreta las carpetas como niveles jerárquicos de segmentación, lo que ayuda a filtrar datos más rápido. Por lo que también tendría sentido este resultado.

El sistema se ha comportado de la manera prevista subiendo los datos correspondientes de manera correcta. El único comportamiento un poco extraño fue lo de las particiones que rápidamente nos dimos cuenta el porqué.

Conclusión

Para completar este sprint lo primero que hemos hecho ha sido entender con claridad qué eran y cuáles eran las funciones de AWS Glue, AWS Glue Data Catalog y Aws Glue Crawler.

A partir de ahí, teníamos claro que el procedimiento iba a ser similar al del sprint 1: crear un script en python que, ejecutado desde el entorno local, cumpliese las funcionalidades pedidas.

Después, seguimos el tutorial que AWS tiene publicado para completar este proceso usando Boto3 y fuimos implementando una a una las cuatro funcionalidades: crear el cliente AWS Glue, crear el AWS Glue Data Catalog, crear el AWS Glue Crawler e iniciar el propio Crawler.

Los logros más importantes consideramos que han sido comprender los conceptos de Data Catalog y Crawler así como ser capaces de ver su enfoque práctico e implementarlos en este sprint. Además, hemos conseguido una buena organización de nuestros datos para realizar consultas en futuros sprints.