



COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI

ICADE

CIHS

Sprint 2

**Grupo 2: María Carreño Nin de Cardona, Raquel
Fernández Esquinas, Teresa X. Garvía Gallego e
Isabel V. Morell Maudes**

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

Índice

INTRODUCCIÓN	3
METODOLOGÍA	4
RESULTADOS	5
CONCLUSIÓN	7

Introducción

Este sprint supone la segunda práctica del proyecto final de la asignatura Tecnologías de Procesamiento Big Data sobre las criptomonedas. En este sprint del proyecto, daremos un paso crucial para guardar y leer los metadatos relacionados con los datos históricos almacenados en S3 mediante la creación de un crawler utilizando el servicio de AWS Glue Data Catalog y AWS Glue Crawler.

El objetivo principal de este sprint es desarrollar un script en Python que automatice tanto la creación del catálogo de datos, como la configuración del crawler.

Dado que la seguridad es un pilar fundamental para el cliente, el código subido no debe contener ningún tipo de credencial o información sensible.

Este proceso permitirá mejorar la gobernanza de los datos almacenados en S3, asegurando una gestión eficiente y segura de los metadatos necesarios para el análisis en futuras fases del proyecto.

Adjuntamos el enlace al repositorio de Git donde se encuentra toda la información, ficheros, datos... sobre este Sprint 2. Toda la información se encuentra en la rama de develop:

https://github.com/tgarviagallego/Proyecto_BigData.git

Metodología

Para completar la segunda historia de usuario hemos utilizado Glue de AWS que es un servicio de integración de datos ‘serverless’ que facilita la detección, preparación y la combinación de los datos para analítica, machine learning y desarrollo de aplicaciones.

Comenzamos utilizando Glue Data Catalog para almacenar y organizar los metadatos de las tablas que hicimos en el sprint anterior desde la página web. Para ello, hemos creado un rol de glue que permita acceder a todos sus servicios gracias a la política de `AWSGlueServiceRole` y que nos permitirá leer los datos almacenados en el bucket que creamos en el Sprint 1 gracias a la política de `AmazonS3ReadOnlyAccess`. A continuación, creamos la base de datos con nombre “trade_data_imat3a02” y añadimos una tabla utilizando un crawler que hemos llamado “crawler_imat02”. Este crawler ha tomado los datos del bucket de s3 “s3://trading-view-data” y los ha almacenado en la tabla de nombre “trade_data_trading_view_data”.

Después, hemos hecho un script de Python llamado `create_bbdd.py` que utiliza la librería `boto3` para conseguir la funcionalidad pedida. Comenzamos creando un cliente de glue con `boto3.client` que recibe el servicio de “glue” y las diferentes credenciales necesarias. Éstas se encuentran en un fichero llamado `credenciales.py` que tenemos almacenado en local con el fin de no hacerlas públicas. El cliente comienza creando una base de datos con nombre “trade_data_imat3a02” y, si salta un error porque está ya creada se maneja el error. Luego, creamos un crawler al que pasamos el enlace donde se encuentra el bucket. Así conseguimos crear la base de datos con las diferentes tablas con los metadatos de los diferentes ficheros del buckets de s3.

Resultados

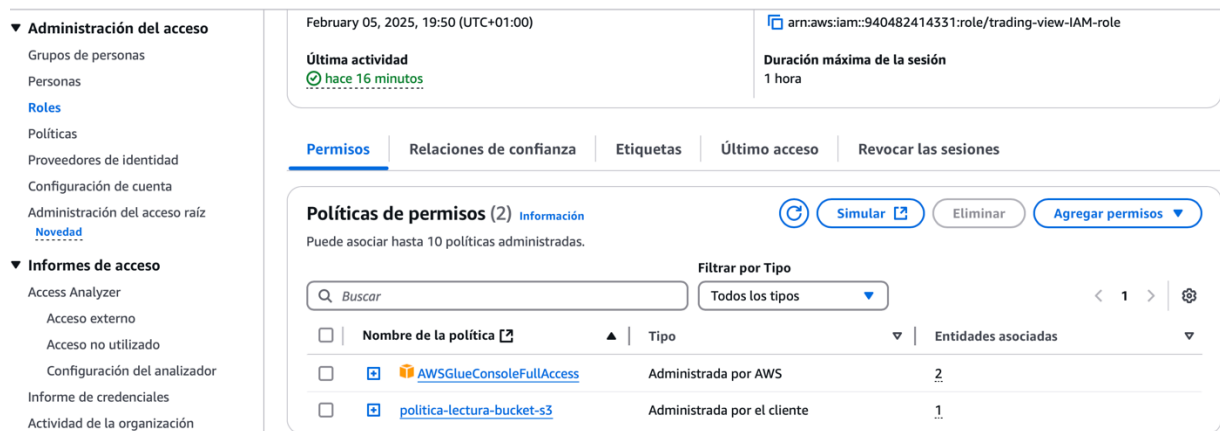
En primer lugar, decidimos hacer a mano la creación de la database, el datalog y el crawler en Glue de AWS, siguiendo el tutorial con el fin de afianzarnos más con el tema.

Una vez que ya interiorizamos qué teníamos que hacer, hicimos un programa de Python que nos permitía crear todo lo anterior con solo ejecutarlo, como ya hemos explicado anteriormente.

A la hora de crear el crawler se nos pedía la creación de una política, donde tuvimos que crear un rol y asociárselo a esa política, donde se le permitía al usuario poder hacer cualquier operación de lectura en el bucket. Para la creación de la política también añadimos las de Glue.

Como resultados, podemos observar las siguientes creaciones que hemos hecho.

En primer lugar, vemos como el rol que hemos creado para la creación del crawler está asociado a las 2 políticas mencionadas anteriormente, la de Glue y la de lectura de los buckets de s3.



February 05, 2025, 19:50 (UTC+01:00)

Última actividad
 hace 16 minutos

arn:aws:iam::940482414331:role/trading-view-IAM-role

Duración máxima de la sesión
 1 hora

Permisos Relaciones de confianza Etiquetas Último acceso Revocar las sesiones

Políticas de permisos (2) Información

Puede asociar hasta 10 políticas administradas.

Filtrar por Tipo
 Todos los tipos

<input type="checkbox"/>	Nombre de la política	Tipo	Entidades asociadas
<input type="checkbox"/>	AWSGlueConsoleFullAccess	Administrada por AWS	2
<input type="checkbox"/>	politica-lectura-bucket-s3	Administrada por el cliente	1

La política final sería la que observamos en la imagen de debajo, la de trading-view que usaremos a la hora de crear el crawler.



Identity and Access Management (IAM)

Panel

Administración del acceso

Grupos de personas

Personas

Roles

Políticas

Proveedores de identidad

Configuración de cuenta

Administración del acceso raíz

Informes de acceso

Access Analyzer

Acceso externo

Acceso no utilizado

trading-view-data-catalog-policy Información

Detalles de la política

Tipo Administrada por el cliente	Hora de creación February 05, 2025, 19:48 (UTC+01:00)	Hora de modificación February 11, 2025, 18:36 (UTC+01:00)	ARN arn:aws:iam::940482414331:policy/trading-view-data-catalog-policy
-------------------------------------	--	--	--

Permisos Entidades asociadas Etiquetas Versiones de la política (2) Último acceso

Esta política define algunas acciones, recursos o condiciones que no proporcionan permisos. Para otorgar acceso, las políticas deben tener una acción que tenga un recurso o condición aplicable. Para obtener más información, elija **Mostrar lo restante**. [Más información](#)

Permisos definidos en esta política Información

Los permisos definidos en este documento de política especifican qué acciones se permiten o deniegan. Para definir permisos para una identidad de IAM (persona, grupo de personas o rol), asíciela a una política

Mostrar los 436 cambios restantes

Una vez que ya tenemos las políticas y el rol definido podemos crear el crawler. Pero antes, creamos el datalog donde se va a almacenar el formato de la estructura de los datos del bucket. (trade_data_imat3a02).

Databases (3)					Last updated (UTC) February 11, 2025 at 18:52:48	Edit Delete Add database
A database is a set of associated table definitions, organized into a logical group.						
<input type="text" value="Filter databases"/>					< 1 >	Settings
<input type="checkbox"/>	Name	Description	Location URI	Created on (UTC)		
<input type="checkbox"/>	console_glueworkshop	-	-	February 5, 2025 at 18:22:10		
<input type="checkbox"/>	glue-database	-	-	February 5, 2025 at 18:21:52		
<input type="checkbox"/>	trade_data_imat3a02	Base de datos con los datos de tradi	-	February 5, 2025 at 18:54:21		

Tras la implementación del código, donde realizamos una conexión al bucket, creamos el datalog y crawler que contienen los metadatos de nuestros datos, la forma con la que se ve en el AWS es:

Databases (1)					Last updated (UTC) February 12, 2025 at 18:21:56	Edit Delete Add database
A database is a set of associated table definitions, organized into a logical group.						
<input type="text" value="Filter databases"/>					< 1 >	Settings
<input type="checkbox"/>	Name	Description	Location URI	Created on (UTC)		
<input type="checkbox"/>	trade_data_imat3a02	Base de datos con los datos de tradi	-	February 5, 2025 at 18:54:21		

Dentro de dicha base de datos encontramos el Datalog, donde se almacenan los metadatos de los datos del bucket, donde podemos ver su esquema en la imagen de debajo:

AWS Glue <ul style="list-style-type: none"> Getting started ETL jobs <ul style="list-style-type: none"> Visual ETL Notebooks Job run monitoring Data Catalog tables Data connections Workflows (orchestration) Data Catalog <ul style="list-style-type: none"> Databases Tables Connections Crawlers <ul style="list-style-type: none"> Classifiers Catalog settings Data Integration and ETL 		trade_data_trading_view_data	
Description -		Connection -	
Last updated February 12, 2025 at 17:58:36		Advanced properties	
Schema		Partitions	
Indexes		Column statistics - new	
Schema (8) View and manage the table schema.		Edit schema as JSON Edit schema	
<input type="text" value="Filter schemas"/>		< 1 >	
#	Column name	Data type	Partition key
1	datetime	string	-
2	symbol	string	-
3	open	double	-
4	high	double	-
5	low	double	-
6	close	double	-
7	partition_0	string	Partition (0)
8	partition_1	string	Partition (1)

Como podemos observar en las imágenes anteriores, conseguimos obtener todos los requisitos necesarios, con muchas dificultades, pero conseguido. Hemos conseguido que el sistema funcione de una forma correcta.

Conclusión

Este sprint ha sido un paso clave para la gestión y gobernanza de los datos relacionados con el proyecto de criptomonedas, permitiéndonos aprovechar las capacidades de AWS Glue para organizar y estructurar los metadatos de manera eficiente. A lo largo del proceso, hemos aprendido a utilizar AWS Glue Data Catalog y AWS Glue Crawler para integrar, analizar y gestionar datos almacenados en S3.

La automatización lograda mediante el script en Python desarrollado con la librería boto3 no solo cumple con los requisitos establecidos en la historia de usuario, sino que también establece una base sólida para futuras implementaciones. Este enfoque nos permite optimizar el flujo de trabajo y garantizar que los metadatos estén correctamente organizados y accesibles para análisis futuros.

Además, se han tomado medidas rigurosas para mantener la seguridad de las credenciales, asegurando que el código sea seguro y cumpla con las mejores prácticas. Gracias a estos avances, hemos sentado las bases para un gobierno efectivo de los datos y mejorado la capacidad del equipo para abordar problemas más complejos en las siguientes fases del proyecto.