

AWS Glue II

[AWS Glue](#) es la herramienta ETL *serverless* completamente administrada que ofrece Amazon Web Services, con la cual podemos crear soluciones basadas en código tanto para la ingesta como para la transformación de los datos.



ETL Jobs

AWS Glue incluye un amplio conjunto de conectores tanto para los servicios AWS (RDS, S3, Redshift, etc.) como para servicios de terceros (MongoDB, MongoDB Atlas, conectores JDBC, etc.)

Como vimos en el tema anterior, cuando el esquema de los datos es desconocido, AWS Glue permite inferirlo. Para ello, hemos de construir un rastreador (*crawler*) para descubrir su estructura.

Con esa información, Glue crea un catálogo que contiene metadatos sobre las diferentes fuentes de datos (compatible con el Hive Catalog).

Además de los rastreadores Glue incorpora otra serie de componentes que nos permiten la automatización de las tareas de ETL.

Finalmente, Glue simplifica la orquestación de las ETL, permitiendo ejecutar miles de *jobs* en *workflows* y desarrollar e incluso poder administrarlos de forma visual.

Componentes

AWS Glue tiene varios componentes que podrían haberse dividido en varios servicios independientes, pero suelen trabajar todos juntos, por lo que AWS los ha agrupado en la familia AWS Glue.

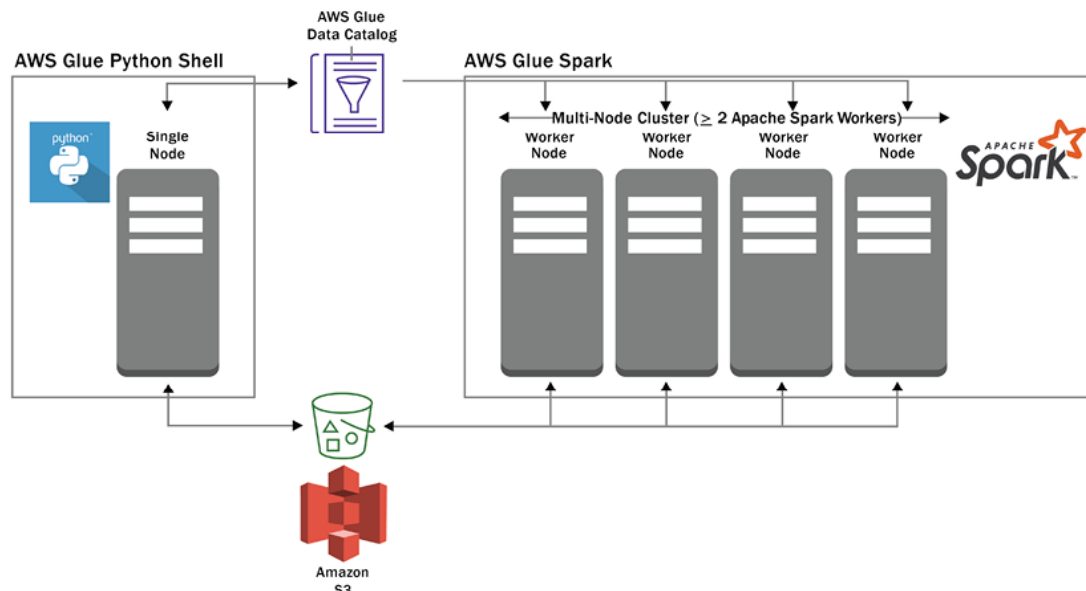
A continuación, vamos a examinar los componentes principales de Glue relacionados con el procesamiento de datos.

Glue ETL jobs

Glue permite trabajar con diferentes motores de integración mediante Python, Spark y Ray para el procesamiento de datos almacenados en diferentes fuentes como S3 o RDS y registrados en el AWS Glue Data Catalog.

Las diferentes tareas ETL se facturan por DPUs (Data Processing Units) utilizadas y la cantidad de tiempo empleado en la ejecución (de forma similar a AWS Lambda). Aproximadamente, y dependiendo de la tecnología, cada DPU cuesta 0,44\$ por hora. Más información en la [página oficial de precios de AWS Glue](#).

La siguiente imagen muestra una posible configuración de DPUs, con dos de los motores Glue (un único nodo Python mediante *shell* a la izquierda, y un clúster de nodos Spark a la derecha):



Un shell de Python en Glue Python y un clúster de Spark

Glue Studio

Por ejemplo, mediante Glue Studio vamos a crear una ETL utilizando herramientas visuales preprogramadas o programas en Python o Spark para realizar distintas operaciones de Extracción, Transformación y Carga de Datos.

▼ Data Integration and ETL

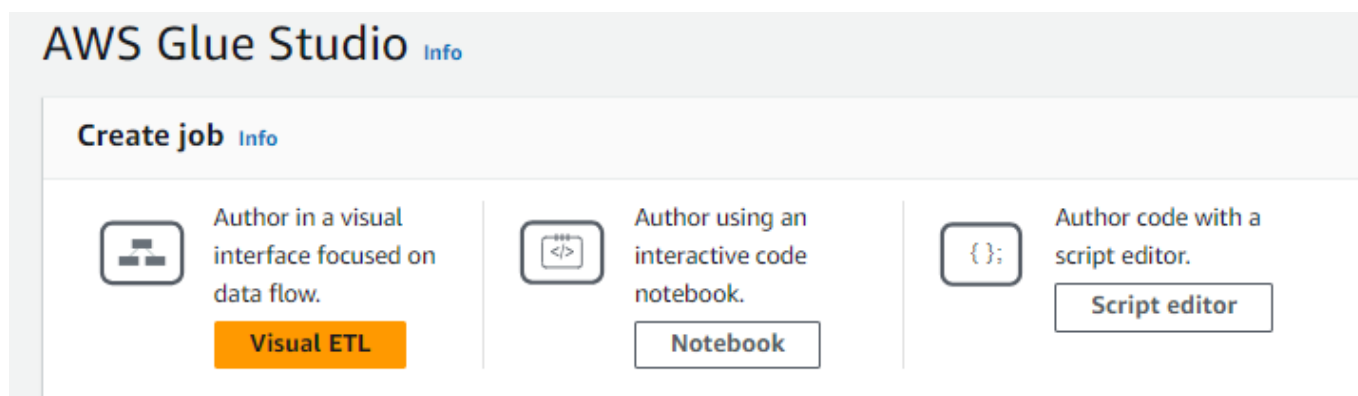
ETL jobs

Visual ETL

Notebooks

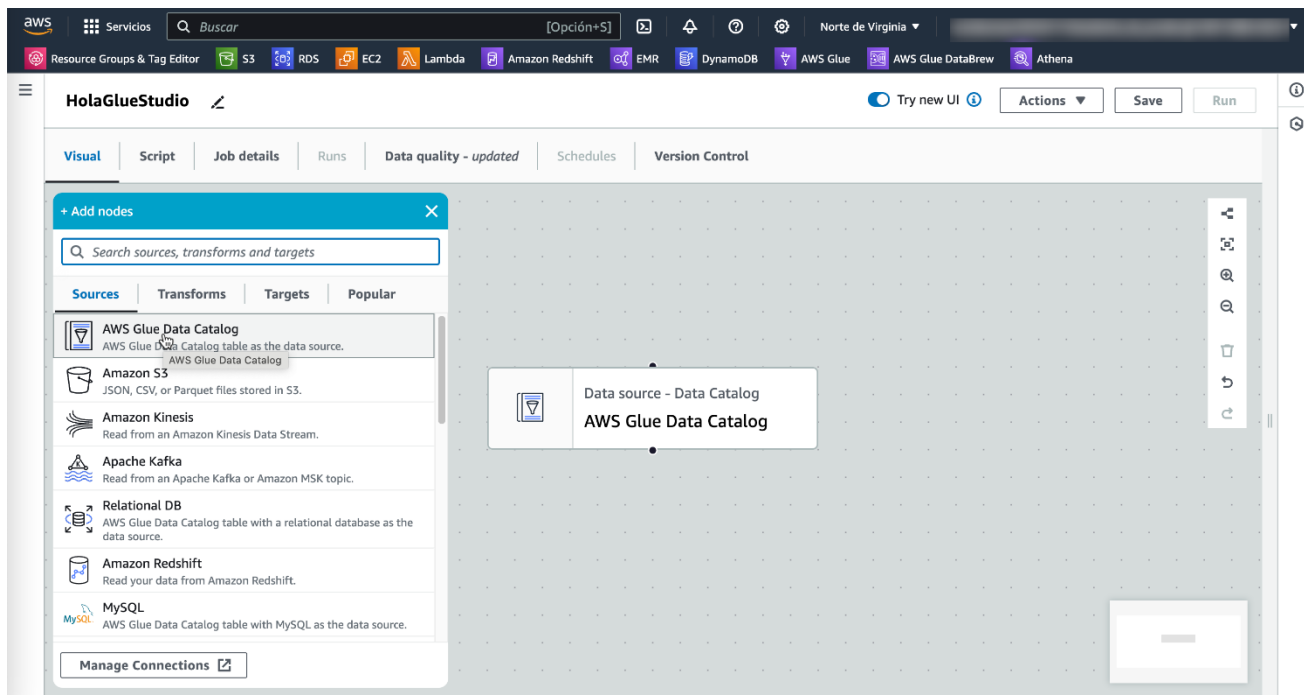
Job run monitoring

Interactive Sessions



Realizando la extracción

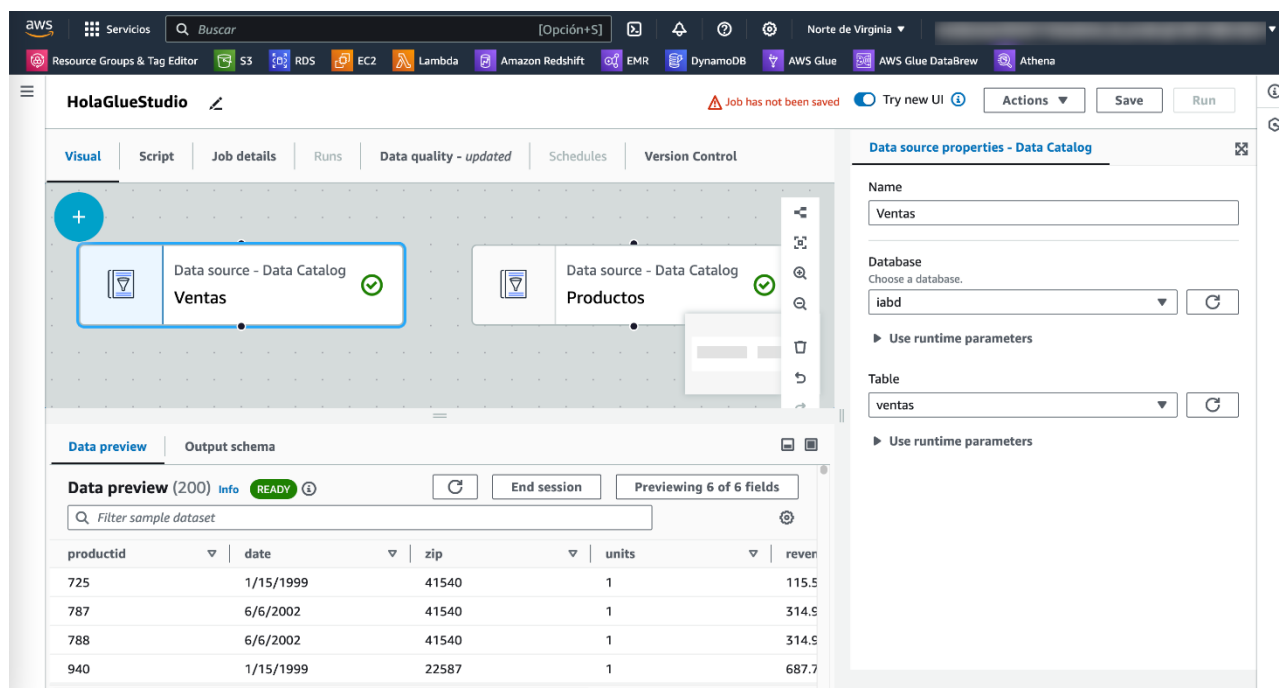
Veremos un ejemplo en Glue Studio desde el interfaz de Visual ETL. En él realizaremos una operación JOIN entre dos tablas (ventas y productos) previamente cargadas con un *crawler* en el Data Catalog y crear nuestra ETL arrastrando como fuente una tabla del catálogo (como se ve en la imagen hay otras fuentes de datos disponibles):



Hola Glue Studio - Tabla del catálogo como fuente

Una vez seleccionado el elemento, seleccionamos la base de datos, en el ejemplo **iabd**, la tabla **ventas** y utilizamos el LabRole que nos ofrece AWS Academy.

Tras seleccionarlo, Glue Studio nos mostrará una previsualización de los datos:

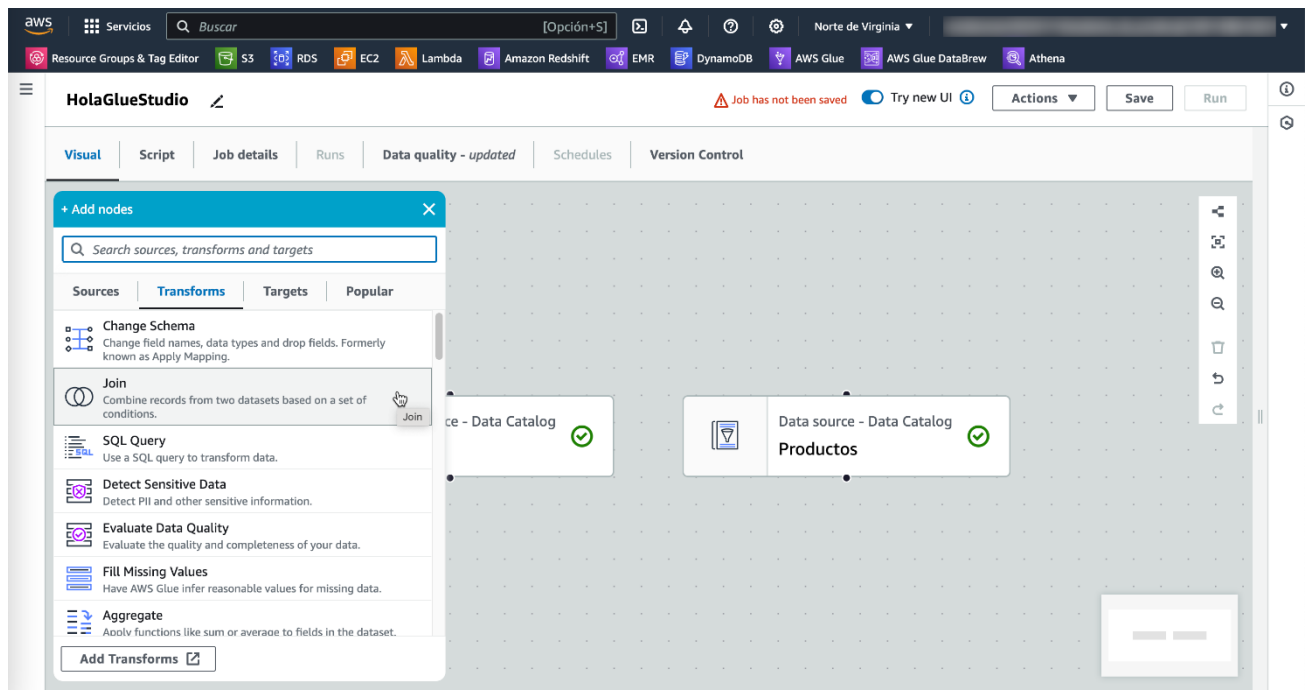


Glue Studio - Tabla de ventas

A continuación, repetimos el proceso, pero ahora con la tabla **productos**.

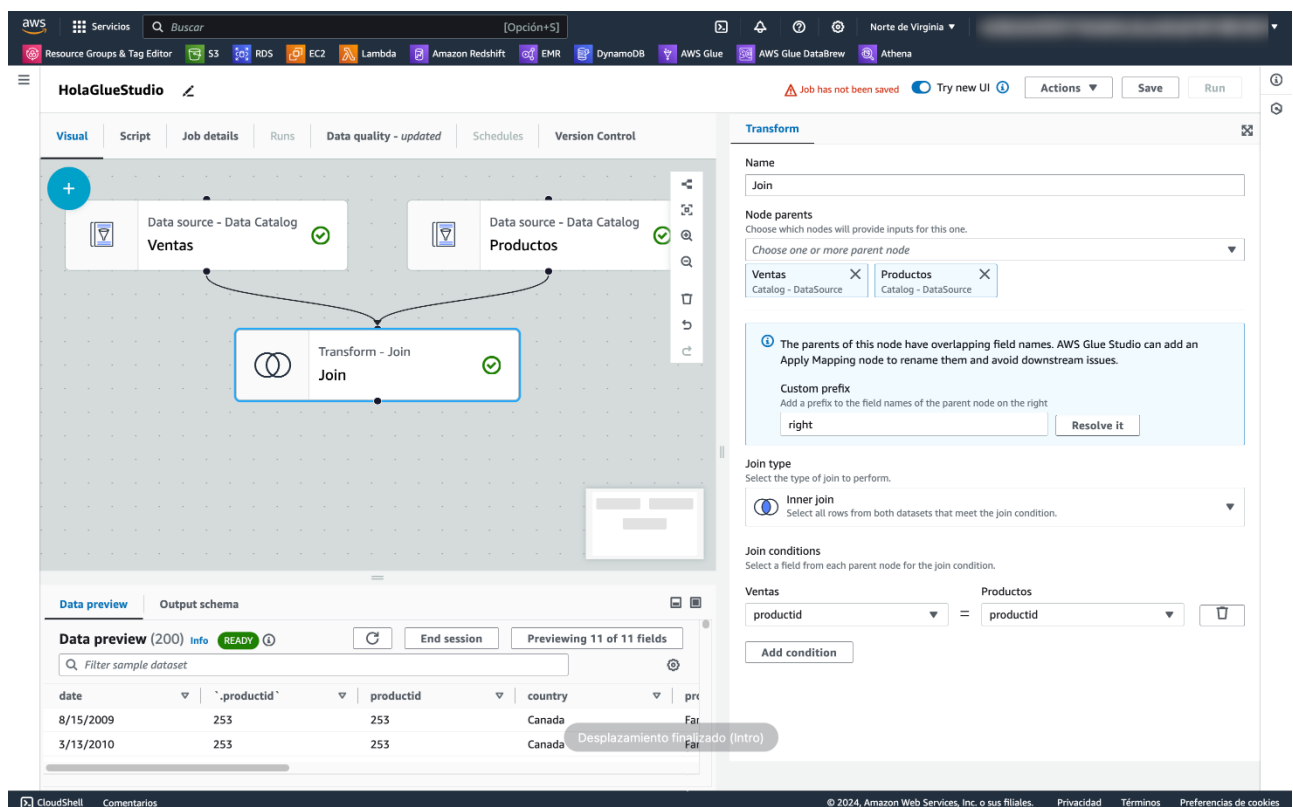
Transformando los datos

Una vez cargados los datos, desde la pestaña de Transformaciones, seleccionamos el componente de **Join**:



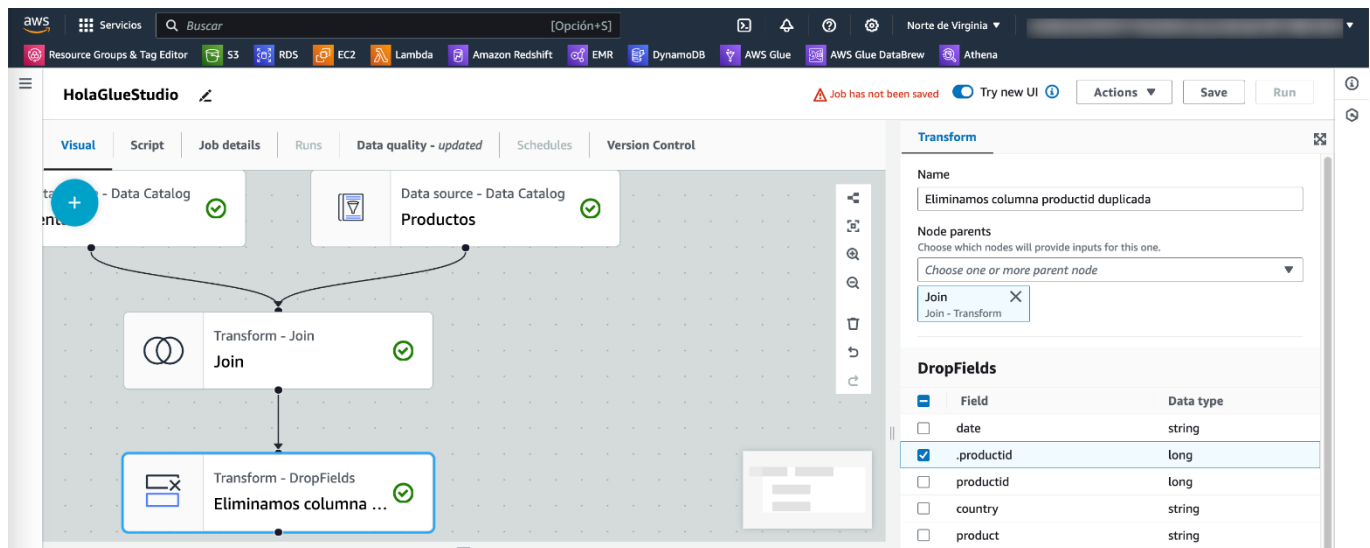
Glue Studio - Componente Join

Y configuramos el **Join** seleccionando ambas tablas e indicando la condición de clave ajena con clave primaria:



Glue Studio - Componente Join

Como ahora tenemos dos campos *productid*, añadimos otra transformación de **Drop Fields** para eliminar la columna duplicada:

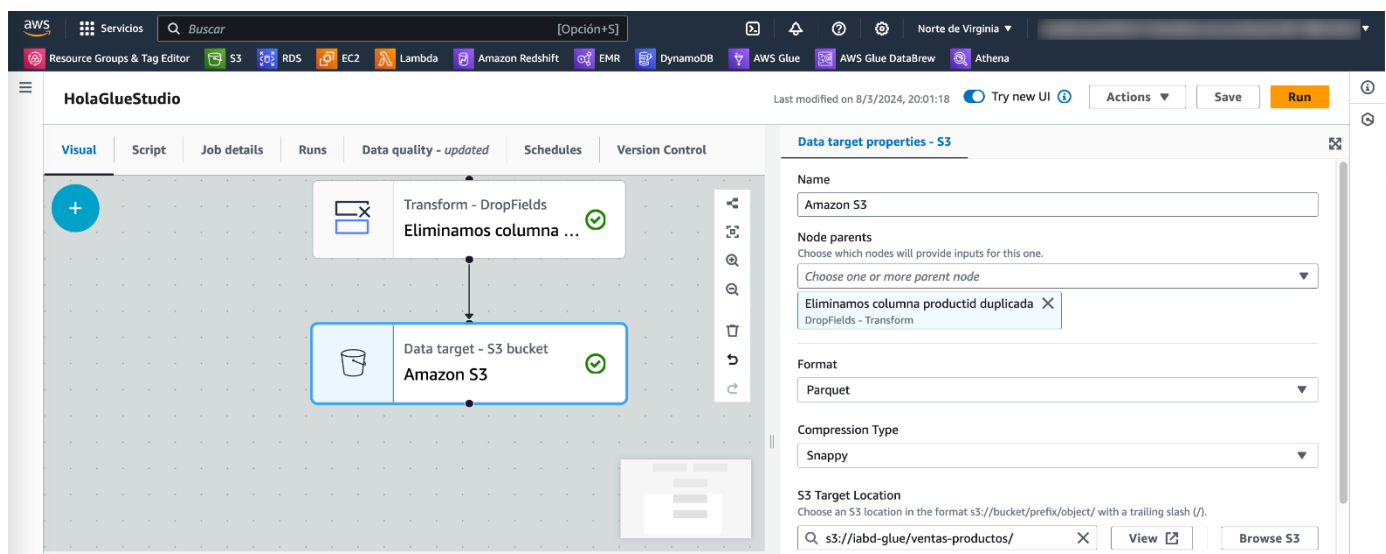


Eliminando columna duplicada

Seleccionamos la carga

Y, para terminar, vamos a almacenar el resultado en S3 en formato Parquet particionando los datos por país (country) y crear una tabla en el catálogo que apunte a dichos datos.

Para ello, primero elegimos para el destino el servicio S3 y configuramos el nodo padre, el formato Parquet y la ruta de destino (en nuestro caso, le ponemos una nueva carpeta ventas-productos):



Hola Glue Studio - Eliminando columna duplicada

Y a continuación le pedimos que guarde una tabla en el catálogo en la base de datos **iabd** en una nueva tabla que llamamos **ventas_productos**:

Data Catalog update options [Info](#)

Choose how you want to update the Data Catalog table's schema and partitions. These options will only apply if the Data Catalog table is an S3 backed source.

☐ Do not update the Data Catalog
 ☒ Create a table in the Data Catalog and on subsequent runs, update the schema and add new partitions
 ☐ Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions

Database

Choose the database from the AWS Glue Data Catalog.

iabd ↻

► **Use runtime parameters**

Table name

Enter a table name for the AWS Glue Data Catalog.

ventas_productos

Partition keys - optional

Add partition keys.

Partition (0)

country 🗑

[Add a partition key](#)

Glue Studio – Actualizando el catálogo de datos con la nueva tabla

Probamos la ETL

Tras grabar, ya tenemos lista nuestra ETL y la ejecutamos mediante el botón *Run* naranja situado en la esquina superior derecha.

Si vamos a la pestaña *Runs* de nuestra ETL, podremos ver el estado y tiempo de ejecución (2m50s lo cual es mucho tiempo para tan poca cantidad de datos, pero el procesamiento distribuido tiene una sobrecarga que, en volúmenes pequeños de datos, penaliza).

HolaGlueStudio Last modified on 8/3/2024, 20:01:18 Try new UI Actions Save Run

Visual | Script | Job details | **Runs** | Data quality - updated | Schedules | Version Control

Job runs (1/1) [Info](#) Last updated (UTC) March 9, 2024 at 07:21:45 View details Stop job run Table View Card View

🔍 Filter job runs by property < 1 > ⚙

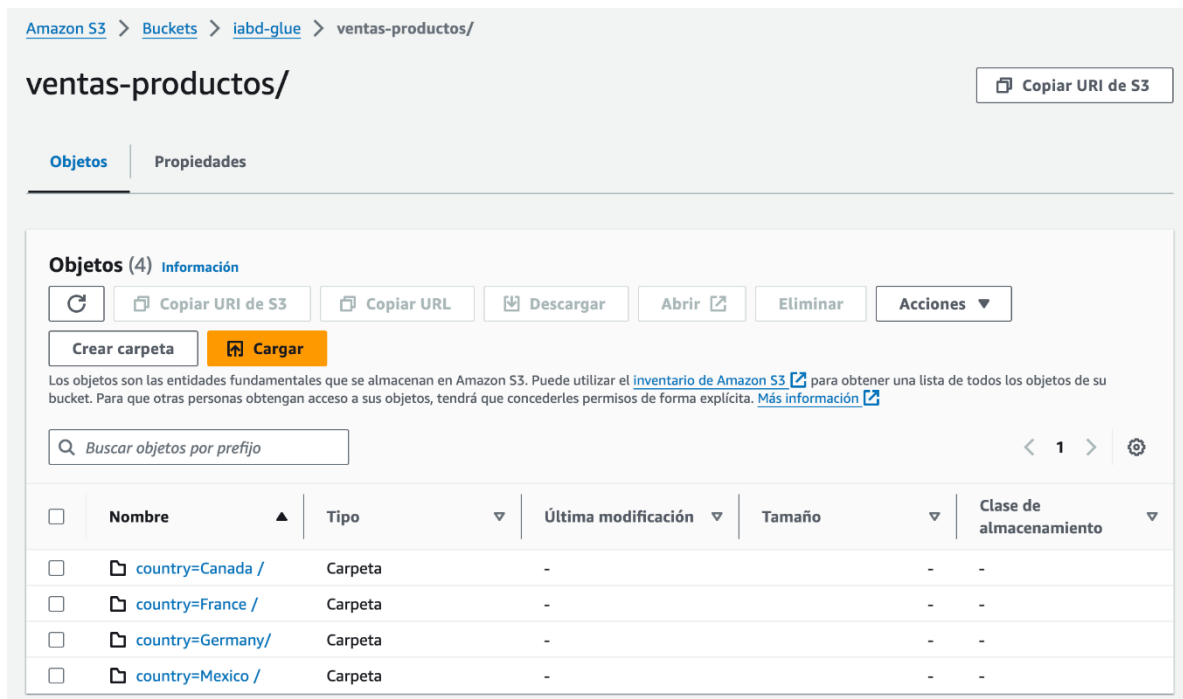
Run status	Retries	Start time (UTC)	End time (UTC)	Duration	Capacity (DP...	Worker type	Glue ver...
🟢 Succeeded	0	2024/03/08 19:06:30	2024/03/08 19:09:31	2 m 50 s	10 DPU's	G.1X	4.0

[Run details](#) | [Input arguments \(10\)](#) | [Continuous logs](#) | [Run insights](#) | [Metrics](#) | [Spark UI](#) 📄 📊

Job name	Start time (UTC)	Glue version	Last modified on (UTC)
HolaGlueStudio	2024/03/08 19:06:30	4.0	2024/03/08 19:09:31
Id	End time (UTC)	Worker type	Log group name
jr_bb5bd93c746c68817d83987508be53eab42572786590dc0537d95e469f8a8fa 📄	2024/03/08 19:09:31	G.1X	/aws-glue/jobs
Run status	Start-up time	Max capacity	Number of workers
🟢 Succeeded	11 seconds	10 DPU's	10
Retry attempt number	Execution time	Execution class	Timeout
Initial run	2 minutes 50 seconds	Standard	2880 minutes
Trigger name	Security configuration	Cloudwatch logs	
-	-	<ul style="list-style-type: none"> All logs Output logs 	

Glue Studio - Estadísticas de la ejecución

Y el último paso es comprobar cómo en S3 se han creado los datos particionados por país:



Glue Studio - Datos particionados por país

Glue Triggers (Disparadores)

[AWS Glue triggers - AWS Glue \(amazon.com\)](#)

Los disparadores nos permiten concatenar trabajos y rastreadores en función de determinados eventos.

▼ Data Integration and ETL

ETL jobs

Visual ETL

Notebooks

Job run monitoring

Interactive Sessions

Data classification tools

Sensitive data detection

Record Matching

Triggers

Pueden ser de tres tipos:

Trigger type

☒ On demand

Fire the trigger immediately when started.

☐ Schedule

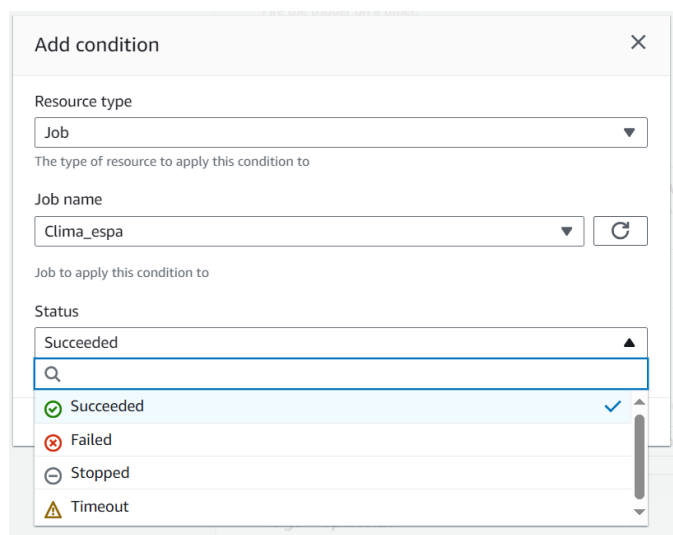
Fire the trigger on a timer.

☐ Job or crawler event

Fire the trigger when job or crawler events match your watched list.

Lanzados manualmente, programados por fechas o que se lancen en función de un determinado evento procedente del resultado de la ejecución de un trabajo o disparador. En este último caso debemos seleccionar el trabajo o rastreador previamente programado y el evento al que ha de responder:

Por ejemplo, seleccionamos un trabajo que carga los datos de las estaciones meteorológicas de España a partir de los datos mundiales que tenemos en un *bucket*:



Add condition

Resource type
Job

The type of resource to apply this condition to

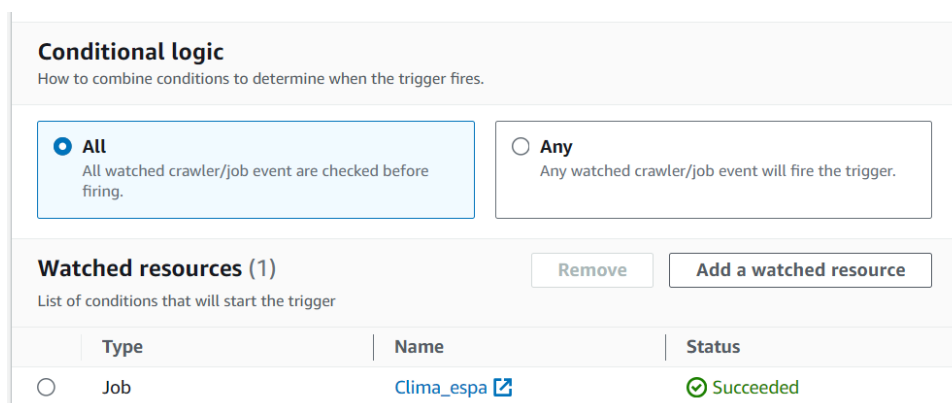
Job name
Clima_espa

Job to apply this condition to

Status
Succeeded

Succeeded
Failed
Stopped
Timeout

Si añadimos más de un trabajo o rastreador habría que configurar si queremos que se lance el trabajo posterior una vez finalizadas todas las tareas previas o solamente alguna de ellas:



Conditional logic
How to combine conditions to determine when the trigger fires.

☒ **All**
All watched crawler/job event are checked before firing.

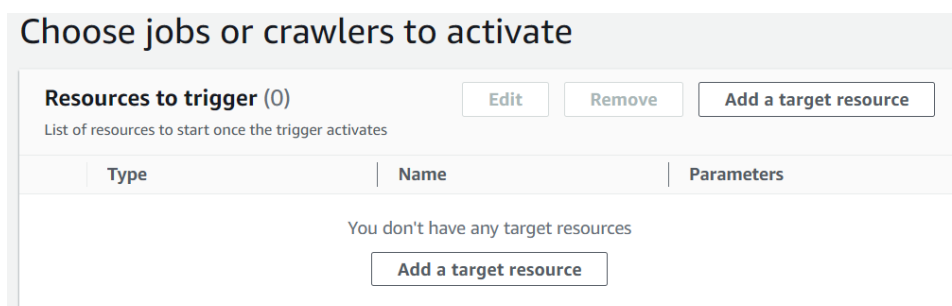
☐ **Any**
Any watched crawler/job event will fire the trigger.

Watched resources (1)
List of conditions that will start the trigger

Remove Add a watched resource

Type	Name	Status
Job	Clima_espa	Succeeded

En el siguiente paso seleccionaremos el trabajo o rastreador que queremos que se active después de finalizado el trabajo con éxito (en el ejemplo podría ser un rastreador que nos crea el catálogo de datos para una nueva tabla con solo las estaciones meteorológicas españolas que ya están en nuestro *bucket* como resultado del trabajo anterior):



Choose jobs or crawlers to activate

Resources to trigger (0)
List of resources to start once the trigger activates

Edit Remove Add a target resource

Type	Name	Parameters
------	------	------------

You don't have any target resources

Add a target resource

Add target

Resource type

Crawler

The type of resource for this trigger target

Crawler name

estaciones

Crawler to start when this trigger fires

Cancel

Add

Como resultado de todo ello, después de lanzado nuestro trabajo que copia los ficheros de las estaciones meteorológicas españolas a nuestro S3 debería lanzarse automáticamente el rastreador que lo añade al catálogo de datos. Es decir, deberíamos tener los ficheros de las estaciones en nuestro *bucket* y la tabla con sus campos añadida al *Data Catalog* disponible para poder hacer consultas sobre ella.

Finalmente, una vez creado el disparador, debemos activarlo desde el panel general:

Triggers

A trigger starts a job when it fires.

Triggers (1/3)

Last updated (UTC)
January 18, 2026 at 17:30:39

Action

Add trigger

View and manage all available triggers.

Filter triggers

	Name	Status	Type	Parameter
<input checked="" type="checkbox"/>	csvtabla	Deactivated	Conditional	1 condition
<input type="checkbox"/>	inicio	Created	On demand	-

Edit trigger

Deactivate trigger

Activate trigger

Start trigger

Delete trigger