Getting Started with AWS Glue: A Step-by-Step Guide

Learn how to set up AWS Glue, create a crawler, catalog your data, and run jobs to convert CSV files into Parquet format, optimizing your ETL processes.

Sep 9, 2024 · 25 min read

Zoumana Keita
A data scientist who likes to write and share knowledge with the data and IA community

TOPICS

AWS

Data Engineering

Cloud-native solutions are increasingly popular across industries due to their security, flexibility, and scalability. One such solution is AWS Glue, which simplifies data processing—a core component in making informed decisions and optimizing business operations.

In this tutorial, we will use AWS Glue to demonstrate a common ETL (Extract, Transform, Load) task: converting CSV files stored in an S3 bucket into Parquet format. Using Parquet enhances both data processing efficiency and query performance.

We'll introduce AWS Glue, its core features, and the benefits of automating data preparation. Next, we'll walk through setting up your AWS account, creating IAM roles, and configuring access to your S3 bucket. Finally, we'll guide you in creating a Glue crawler to scan your datasets and generate Parquet files.

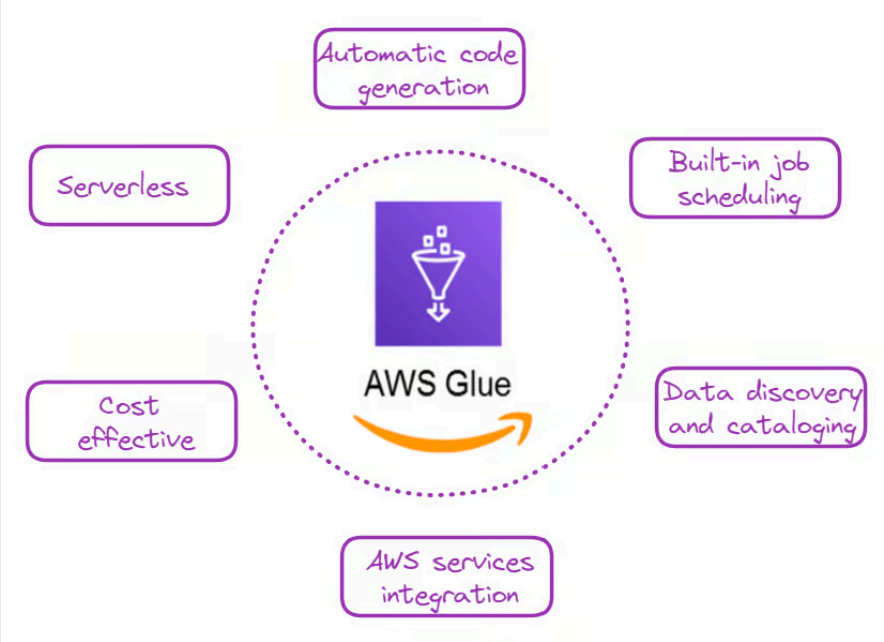
What Is AWS Glue?

Before proceeding with the practical steps, let's first understand what AWS Glue is and why it's useful for data processing tasks.

AWS Glue is a fully managed ETL service that makes preparing and loading your data for analytics easy. It provides a serverless environment to create, run, and monitor ETL jobs. AWS Glue automatically discovers and profiles your data via the AWS Glue Data Catalog, recommends and generates ETL code, and provides a flexible scheduler to handle dependency resolution, job monitoring, and retries.

Why Use AWS Glue?

Understanding the benefits of AWS Glue will help you appreciate its value in the data processing workflow. There are several compelling reasons to use AWS Glue and some of them are illustrated below:



Why choose AWS Glue? Image by author

- **Serverless:** There is no need to provision or manage servers, which reduces operational overhead.
- **Automatic code generation**: AWS Glue can automatically generate Python or Scala code for ETL jobs.
- **Built-in job scheduling and monitoring**: This capability makes scheduling and monitoring ETL jobs easy.
- **Data discovery and cataloging**: AWS Glue crawlers can automatically discover, catalog, and prepare data for analysis.
- **Integration with other AWS services**: Seamlessly works with services like Amazon S3, Amazon RDS, and Amazon Redshift.
- **Cost-effective**: You only pay for the resources consumed while ETL jobs run.

AWS Glue: A Step-by-Step Guide

Here comes the exciting part: let's learn how to use Glue for a very common data processing task: converting CSV files to Parquet.

1. AWS account setup

For those new to AWS, the first step would be to create an account and then log in to the management console.

- Go to AWS and sign up.
- Once you have an account, log in to the AWS Management Console.

2. IAM role creation

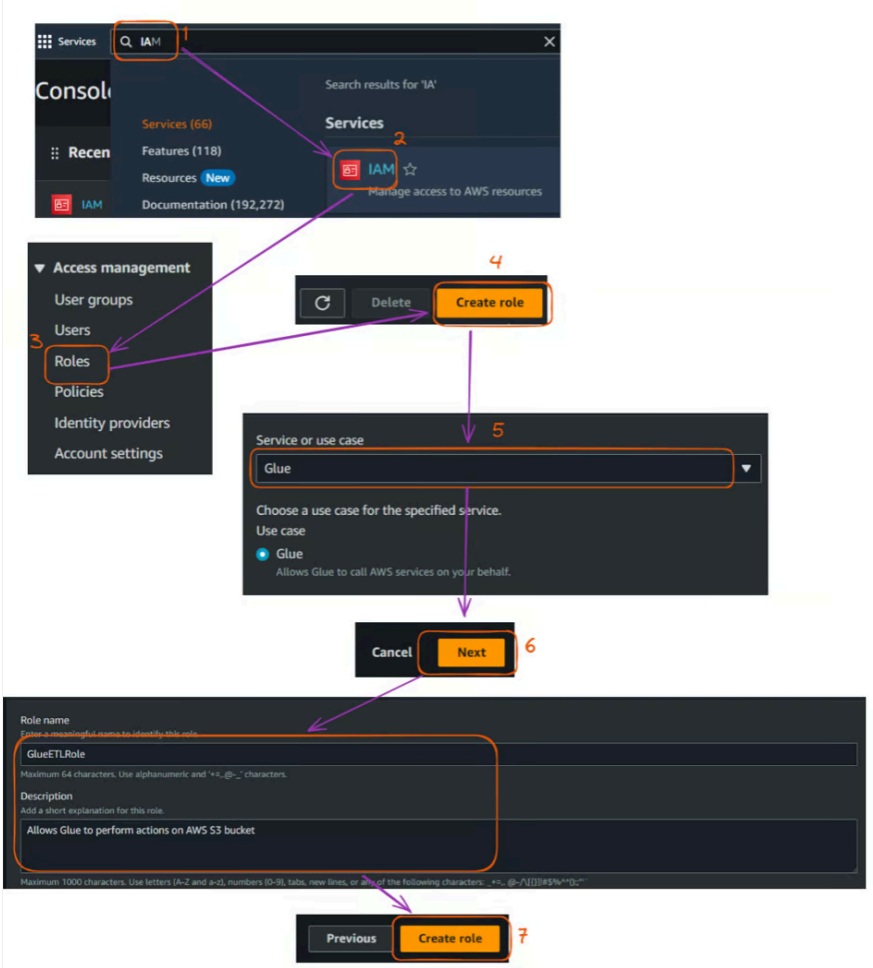
AWS Identity and Access Management (IAM) allows users to securely manage access to AWS services and resources.

We need to create an IAM role to allow Glue to access relevant services. In our case, the main service we are accessing is S3, which hosts our CSV files.

The **AWS Storage Tutorial** is a great way for new AWS users to learn more about systems like S3.

The main steps to creating that IAM role are described below, starting from the Management Console:

- Navigate to the IAM service.
- Click "Roles" in the left sidebar, then click "Create role".
- Under "Choose a use case," select "Glue" and click "Next: Permissions."
- Search for and select the following policies:
    - `AWSGlueServiceRole`
    - `AmazonS3FullAccess` (Note: In a production environment, you should create a more restrictive policy).
- Name your role (e.g., "GlueETLRole"), and eventually give a description, but that is not mandatory. Finally, click "Create role".



Steps to create an IAM role in the AWS console. Image by author

The **Complete Guide to AWS IAM** walks through the steps to use IAM to secure AWS environments, manage access with users, groups, and roles, and outline best practices for robust security.

3. S3 bucket setup

We'll store both input CSV files and output Parquet files on Amazon S3.

The input data for this example was created for illustration purposes and can be found in the author's GitHub repository:

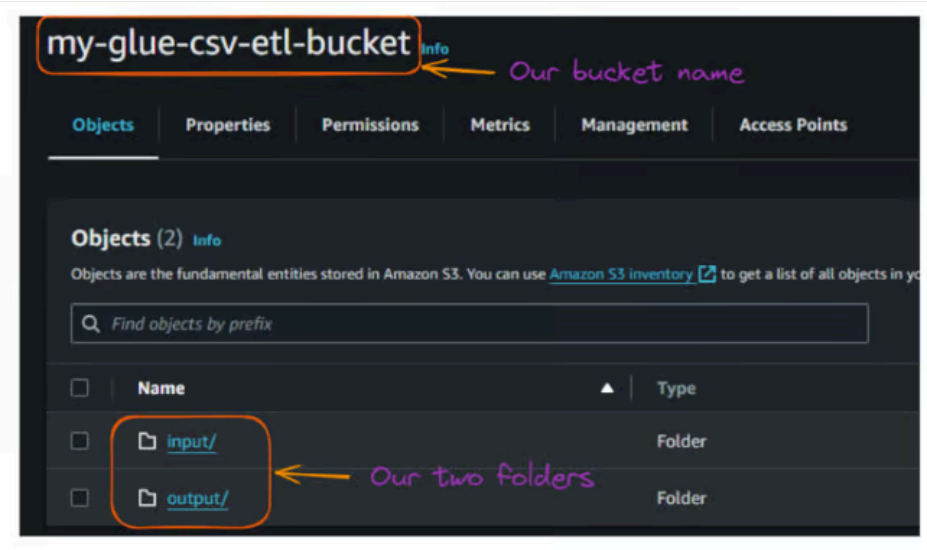
- `athletes.csv` : Contains information about athletes, including their ID, name, country, sport, and age.
- `events.csv` : Lists various events, including event ID, sport, event name, date, and venue.
- `medals.csv` : Records medal information, linking events and athletes to the medals they won.

Now, we can set up our S3 bucket as follows:

- Go to the S3 service in the AWS Management Console.
- Click "Create bucket" and give it a unique name (e.g., "my-glue-csv-etl-bucket").
- Leave the bucket default settings for simplicity's sake and create the bucket.
- Create two folders inside your bucket: "input" and "output."

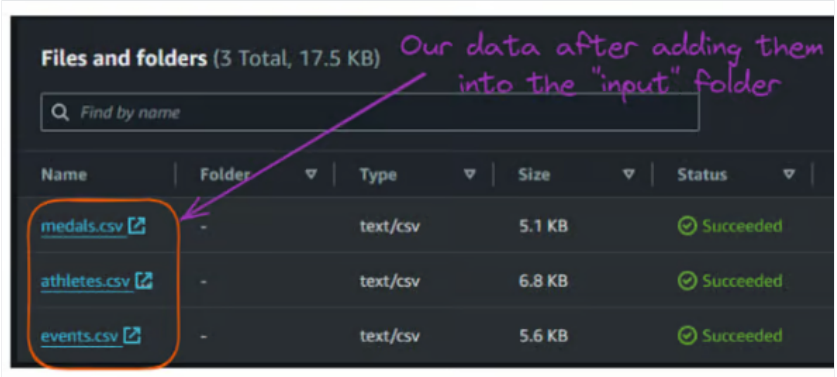
S3 bucket and folder creations flow. Image by author

After creating the two folders, the content of your bucket should look like this:



View of the folders inside the S3 bucket. Image by author

Now, we can upload **these input CSV files** to the "input" folder.



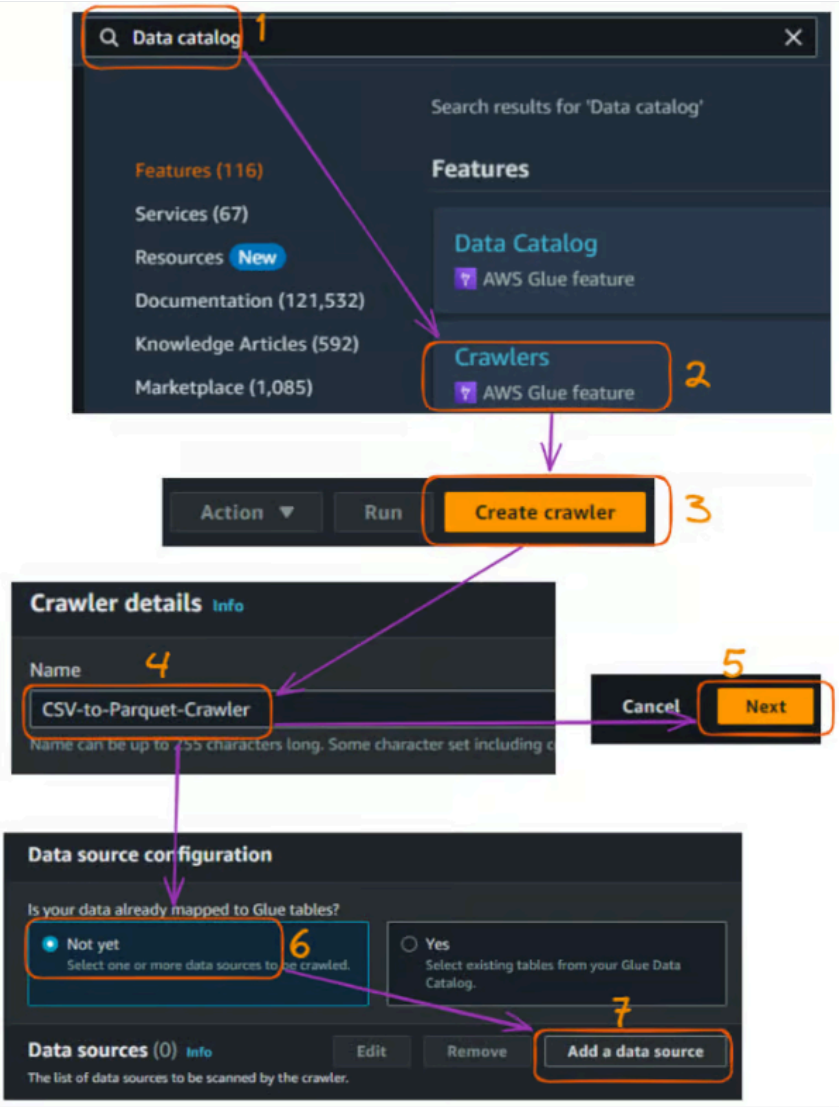
Contents of the S3 bucket "input" folder. Image by author

4. Creating a Glue crawler

A Glue crawler is used to discover and catalog our data automatically. This section explains how to create a crawler that scans all the input CSV files.

The main steps are explained below:

- Go to the AWS Glue service in the AWS Management Console.
- In the left sidebar, under "Data catalog," click on "Crawlers."
- Click "Create crawler".
- Name your crawler (e.g., "CSV-to-Parquet-Crawler") and leave the description field empty, then click "Next."
- We need to specify "S3" as the data source, which is done by choosing "Add data source " as the crawler source type and clicking "Next."



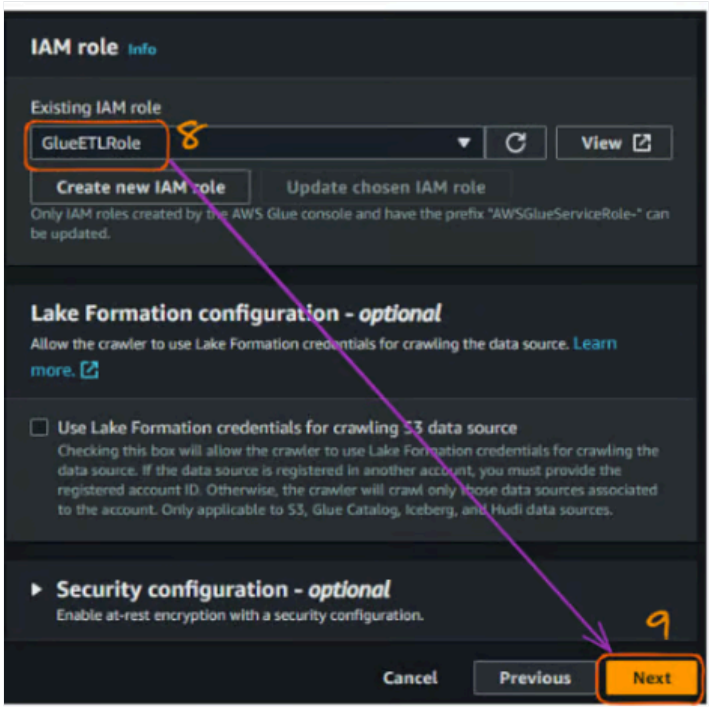
AWS Glue crawler creation flow in the console - part 1. Image by author

- Choose "S3" as the data store and specify your S3 bucket path, which you can do by selecting “Browse S3.” Then click "Next."
- Select “Crawl all sub-folders”
- Select “Add an S3 data source”, then click "Next."
- Before moving further, check the “S3” data type, then “Next.”

AWS Glue crawler creation flow in the console - part 2. Image by author

Once we have connected the S3 bucket, we need to connect the IAM role that was previously created to allow Glue to access S3 buckets. The name of our role is "GlueETLRole."

- Choose the role name from the "Existing IAM role" field.
- Click "Next."



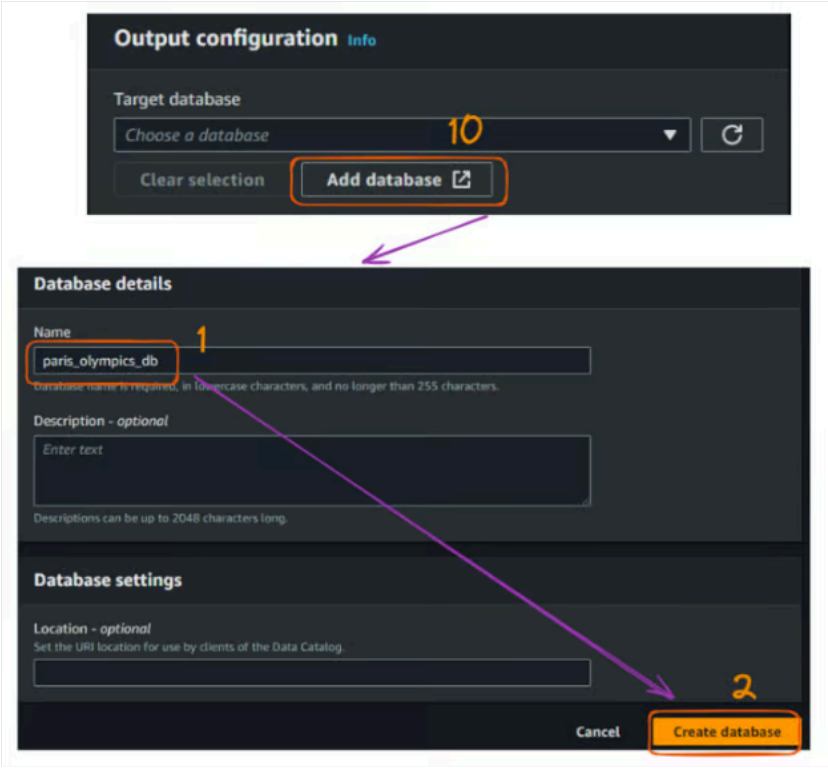
AWS Glue crawler creation flow in the console - part 3. Image by author

The next step is to create a database. The “paris_olympics_db” database we will create serves as a central repository in the AWS Glue Data Catalog to store and organize metadata about our CSV files.

The catalog will enable efficient data discovery and simplifying our ETL process for combining and converting the athletes, events, and medals data into Parquet format.

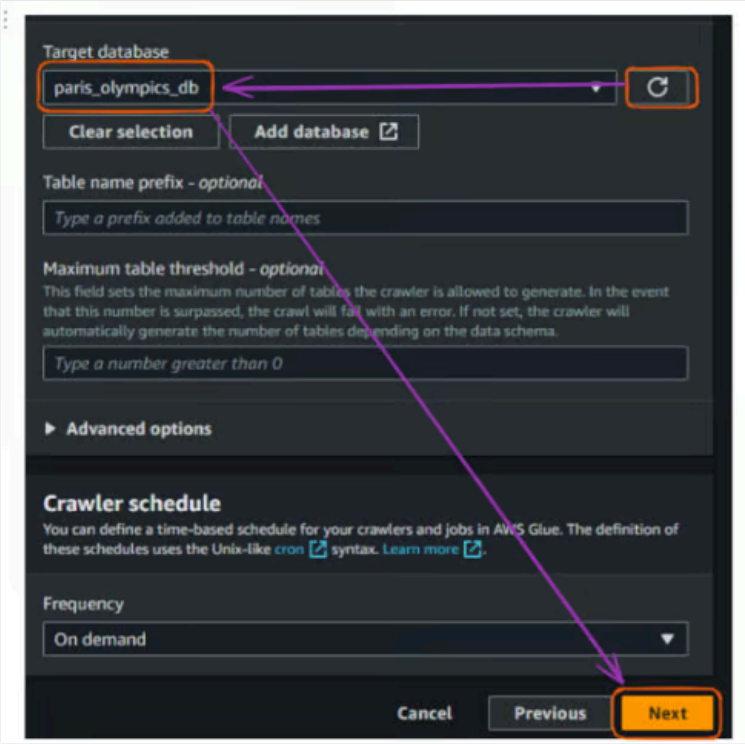
At this moment, we do not have any database yet, and the creation steps are illustrated below:

- Select “Add database,” which will open a new window for creating our database. Give it the name “paris_olympics_db” while leaving everything else as default, and click “Create database.”



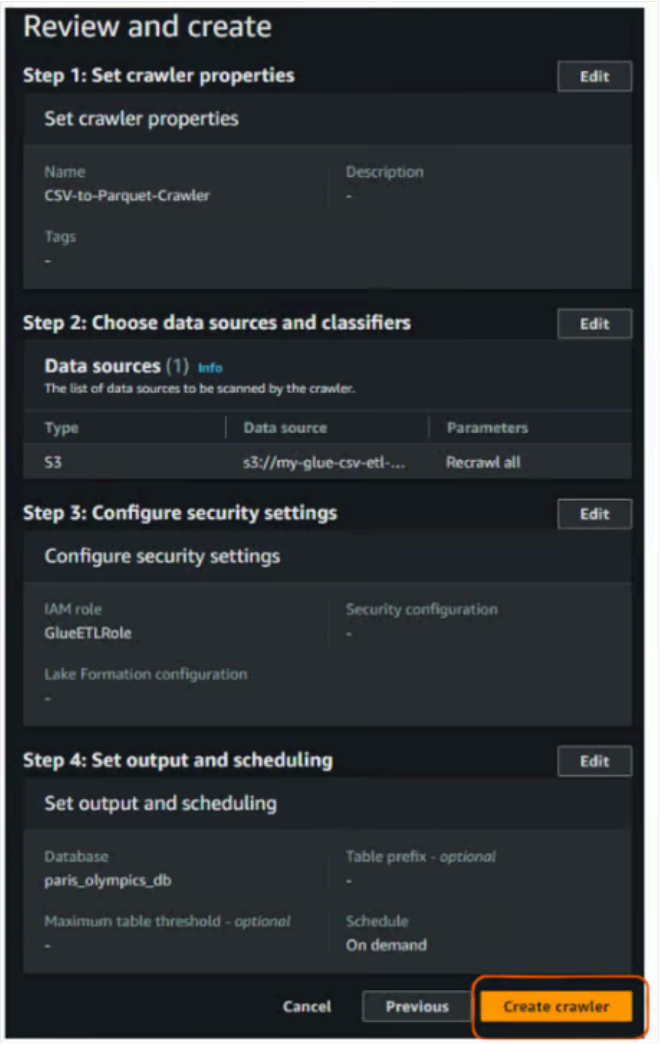
AWS Glue crawler creation flow in the console - part 4. Image by author

From the original tab where we connect a database, refresh the “Target database” section and choose the newly created database. Then click “Next.”



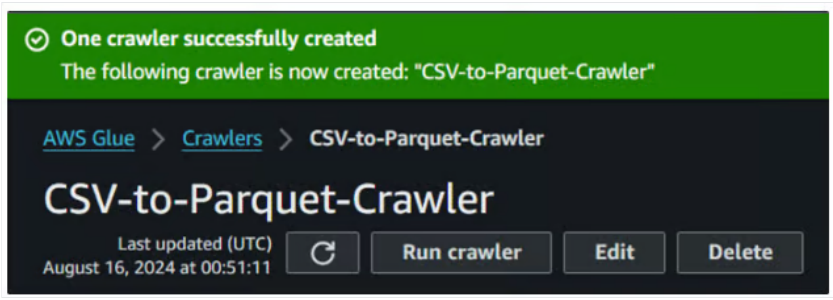
AWS Glue crawler creation flow in the console - part 5. Image by author

Finally, review everything and click “Create crawler.”



AWS Glue crawler "Review and create" view. Image by author

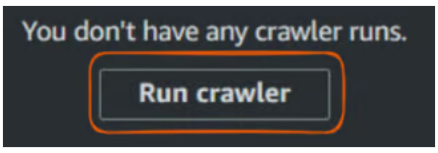
After the previous action of creating a crawler, we should see the following result confirming its creation:



Glue crawler successful creation. Image by author

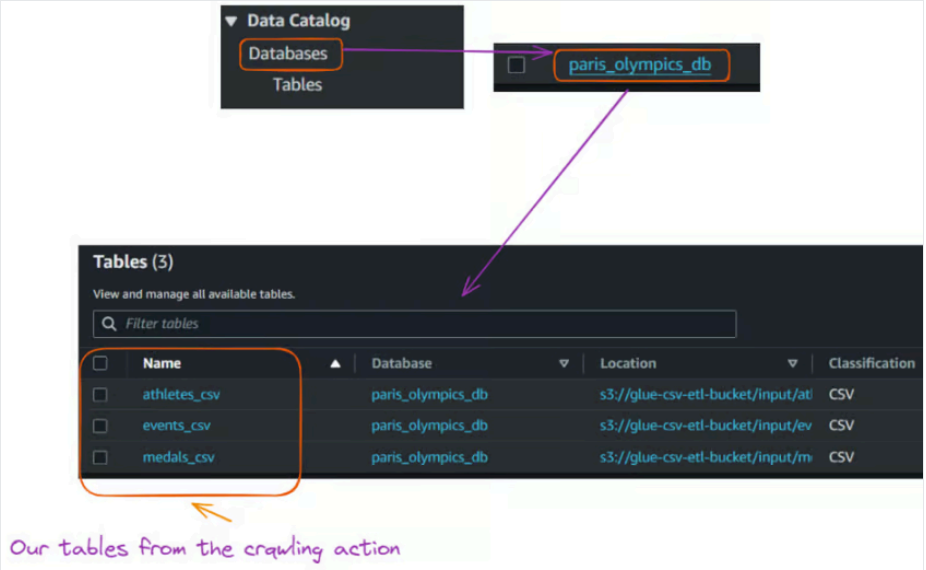
5. Running a Glue crawler

Now that we've created our crawler, it's time to run it and see how it catalogs our data.



Glue "Run crawler" button. Image by author

- Select your crawler from the list.
- Click "Run crawler."
- Wait for the crawler to finish. This may take a few minutes.
- Once complete, go to "Databases" under "Data catalog" in the left sidebar.
- Click on the database that was created.
- You should see a table that represents your CSV data structure.

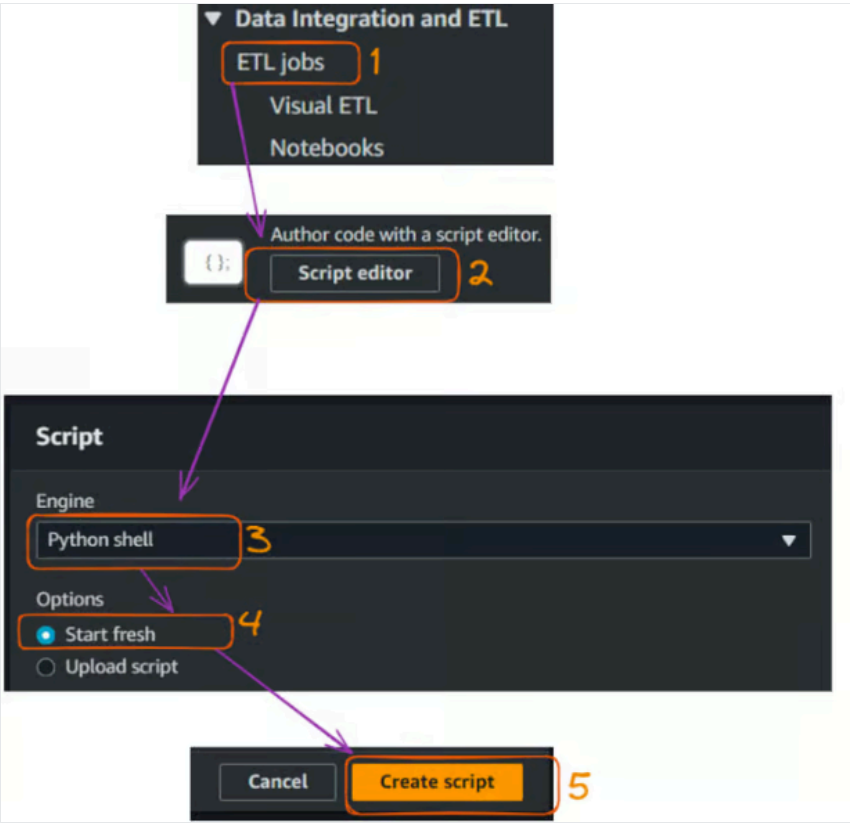


View of tables created in the database from Glue crawler execution. Image by author

6. Creating and running a Glue job

With our data catalog created, we can now create a Glue job to combine our CSV files and convert them to Parquet format. This section walks you through creating the Glue job and provides the necessary Python code.

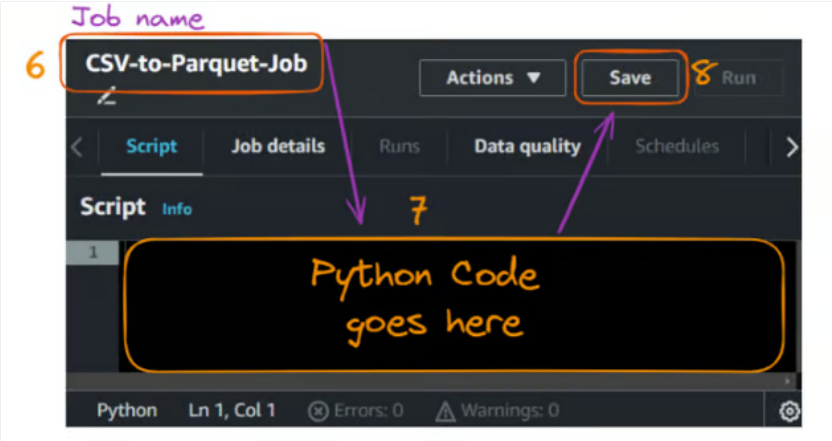
- Go to "Jobs" under "ETL" in the AWS Glue console in the left sidebar.
- Select the Script editor to open an editor to write a Python code, and click “Create script.”



Creating and running a Glue job - part 1. Image by author

The action of creating the script opens the following script tab where we can give the job a name, which is “CS” in our case.

- Within the code section, paste the following code, then click “Save.”



Creating and running a Glue job - part 2. Image by author

The corresponding Python code is given below:

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from pyspark.sql.functions import col, to_date

# Initialize the Spark and Glue contexts
sc = SparkContext()
glueContext = GlueContext(sc)
```

```
spark = glueContext.spark_session
job = Job(glueContext)

# Set Spark configurations for optimization
spark.conf.set("spark.sql.adaptive.enabled", "true")
spark.conf.set("spark.sql.adaptive.coalescePartitions.enabled", "true")

# Get job parameters
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

# Set the input and output paths
input_path = "s3://glue-csv-etl-bucket/input/"
output_path = "s3://glue-csv-etl-bucket/output/"

# Function to read CSV and write Parquet
def csv_to_parquet(input_file, output_file):
    try:
        # Read CSV
        df = spark.read.option("header", "true") \
                       .option("inferSchema", "true") \
                       .option("mode", "PERMISSIVE") \
                       .option("columnNameOfCorruptRecord", "_corrupt_record") \
                       .csv(input_file)

        # Print schema for debugging
        print(f"Schema for {input_file}:")
        df.printSchema()

        # Convert date column if it exists (assuming it's in the format M/d/yyyy)
        if "date" in df.columns:
            df = df.withColumn("date", to_date(col("date"), "M/d/yyyy"))

        # Write Parquet
        df.write.mode("overwrite").parquet(output_file)

        print(f"Successfully converted {input_file} to Parquet at {output_file}")
        print(f"Number of rows processed: {df.count()}")
    except Exception as e:
        print(f"Error processing {input_file}: {str(e)}")

# List of files to process
files = ["athletes", "events", "medals"]

# Process each file
for file in files:
    input_file = f"{input_path}{file}.csv"
    output_file = f"{output_path}{file}_parquet"
    csv_to_parquet(input_file, output_file)

job.commit()
print("Job completed.")
```

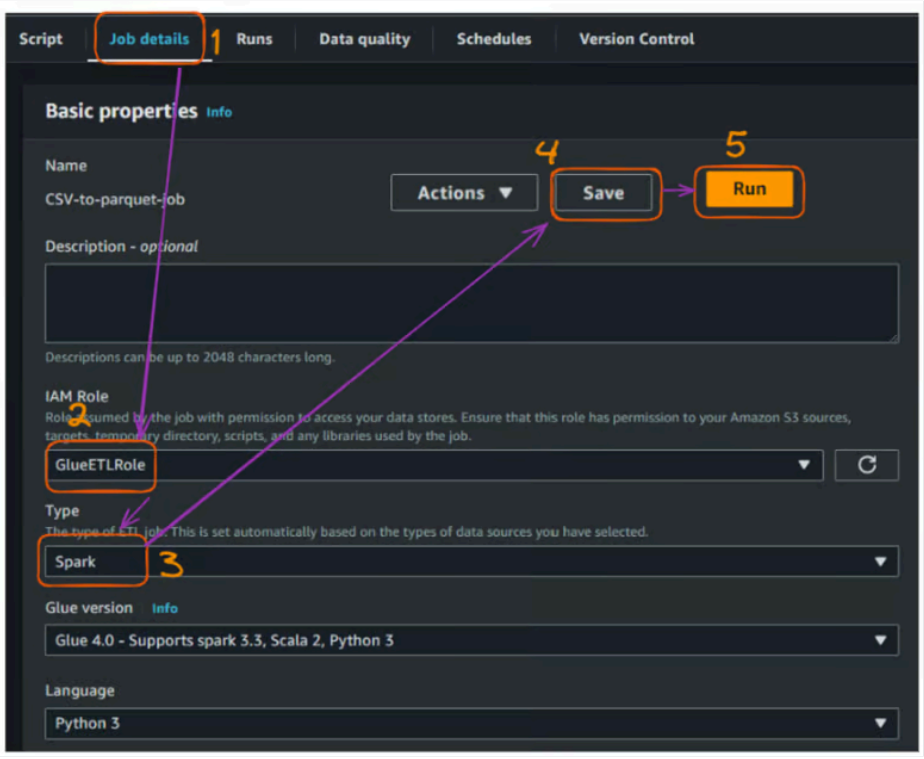
Explain code

Let's briefly understand what is happening in the code:

1. **Imports**: Imports necessary libraries from AWS Glue, PySpark, and Python's standard library.
2. **Initialization**: Initializes the Spark and Glue contexts and sets Spark configurations for optimization.
3. **Job parameters**: Retrieves job parameters, specifically the job name, from command line arguments.
4. **Paths**: Defines input and output paths for the CSV and Parquet files in the S3 bucket.
5. **Function definition**: Defines the `csv_to_parquet()` function to read a CSV file, print its schema, convert a date column if it exists, and write the data as a Parquet file. Handles exceptions during processing.
6. **File processing**: Processes a list of files (`athletes`, `events`, `medals`) by calling the `csv_to_parquet()` function on each file.
7. **Job commit**: Commits the job and prints a completion message.

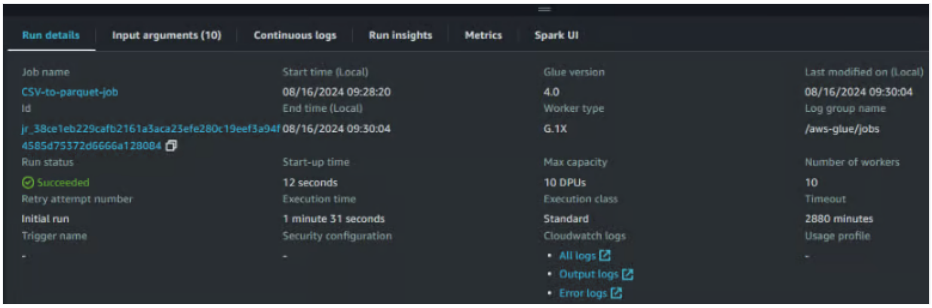
Before running the job, we need to fill in the following additional configurations in the "Job details" section:

- Provide the IAM role we previously created to allow the job execution.
- Set the Job "Type" to "Spark," leave the rest of the fields by default, and "Save".
- Then, run the job!



Creating and running a Glue job - part 3. Image by author

After successfully running the job, we can see more details about the status of the execution:



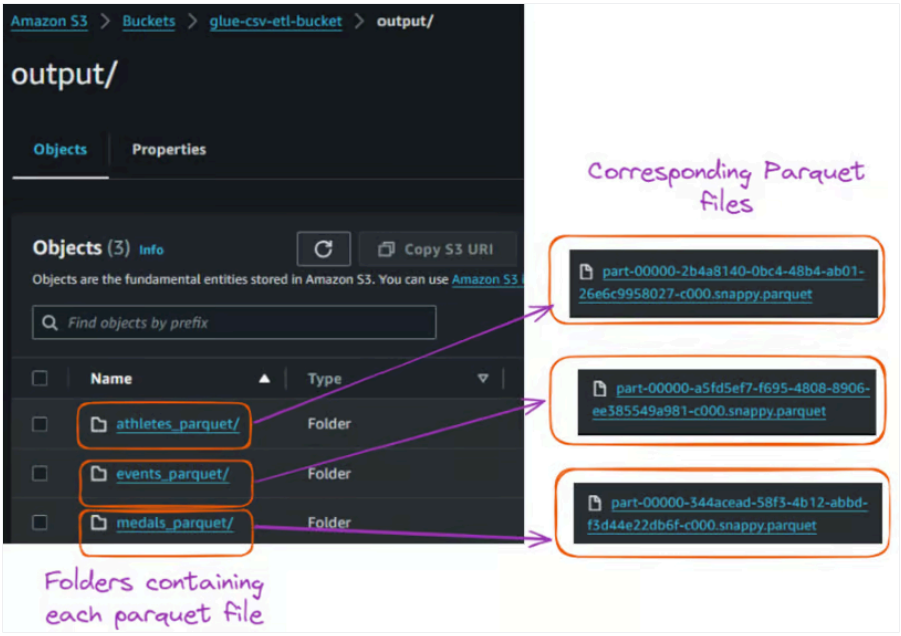
AWS Glue job run details view. Image by author

Access to and monitoring the run details is important for maintaining efficient and reliable data pipelines. These metrics provide valuable insights and help us with the following:

- **Performance optimization:** Track execution times to improve job efficiency and resource allocation.
- **Cost management:** Monitor resource usage (DPUs, workers) to align with the budget and optimize expenses.
- **Capacity planning:** Use execution metrics to forecast resource needs and plan for scalability.

The successful execution of this job converted each CSV file into its corresponding Parquet file and stored them in the “output” folder as shown below:

- On the left, we have the folders with the same name as the original CSV file.
- The ride hand-side corresponds to the Parquet files for each CSV file.



Contents of the output folder inside the S3 bucket. Image by author

Conclusion and Next Steps

This tutorial covered AWS Glue and its features. It guided you through setting up an AWS environment and exploring the AWS Glue interface. It also showed you how to build and run a Glue crawler to catalog data, create a Glue job to transform it, and successfully convert CSV files to Parquet format.

You can expand on what you've learned by setting up triggers to automate workflows, implementing error handling and logging, and optimizing Parquet files for even better

performance. You can also explore querying your data with AWS Athena or Amazon Redshift Spectrum.

Additionally, monitor your AWS usage and costs and clean up resources when they're no longer needed to avoid unnecessary charges.

The courses Introduction to AWS and AWS Cloud Technology and Services could be excellent next steps for further learning!

The first offers a solid AWS and cloud computing foundation, ideal for beginners or those brushing up on key concepts. The second focuses on hands-on learning, helping you deepen your practical knowledge of AWS services.

FAQs

What are some best practices for optimizing Parquet file performance?

To optimize Parquet file performance, consider adjusting file sizes (128 MB to 1 GB is ideal), partitioning your data for efficient querying, and setting the right compression codec (like Snappy or Gzip). These strategies reduce I/O and improve query speeds when working with large datasets in tools like AWS Athena.

How can I implement error handling and logging in AWS Glue jobs?

What are some common use cases for AWS Glue beyond CSV-to-Parquet conversion?

How does AWS Glue compare to other ETL tools like Apache Airflow or Databricks?

How can I automate AWS Glue jobs with triggers?

AUTHOR
Zoumana Keita

A multi-talented data scientist who enjoys sharing his knowledge and giving back to others, Zoumana is a YouTube content creator and a top tech writer on Medium. He finds joy in speaking, coding, and teaching . Zoumana holds two master's degrees. The first one in computer science with a focus in Machine Learning from Paris, France, and the second one in Data Science from Texas Tech University in the US. His career path started as a Software Developer at Groupe OPEN in France, before moving on to IBM as a Machine Learning Consultant, where he developed end-to-end AI solutions for insurance companies. Zoumana

joined Axionable, the first Sustainable AI startup based in Paris and Montreal. There, he served as a Data Scientist and implemented AI products, mostly NLP use cases, for clients from France, Montreal, Singapore, and Switzerland. Additionally, 5% of his time was dedicated to Research and Development. As of now, he is working as a Senior Data Scientist at IFC-the world Bank Group.

TOPICS

AWS Data Engineering

Learn more about AWS and data engineering with these courses!

COURSE

AWS Security and Cost Management

3 hr 1.4K

Master AWS security, governance, and cost optimization to prepare for the Cloud Practitioner certification.

See Details → Start Course

See More →

Related

TUTORIAL

How to Set Up and Configure AWS: A Comprehensive Tutorial

TUTORIAL

Getting Started with AWS Athena: A Hands-On Guide for...

TUTORIAL

AWS EC2 Tutorial For Beginners

See More →