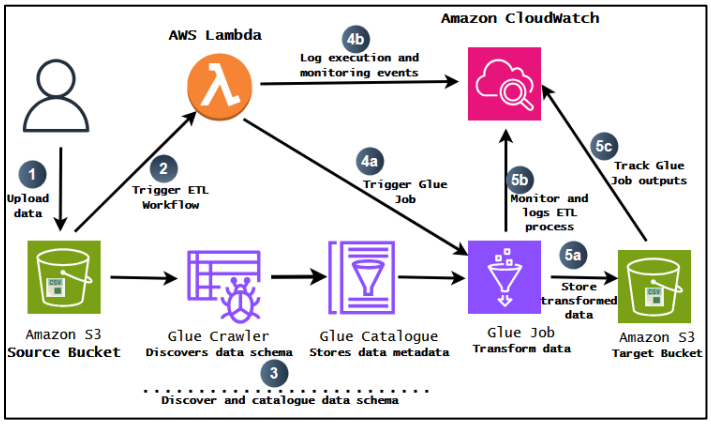
# Automating ETL Workflows with AWS Glue, Lambda, and S3 - A Guide with Python Integration

## Table of Contents

# Introduction

In today's data-driven business landscape, the ability to efficiently process, transform, and analyze large volumes of data is crucial for organizations to gain valuable insights and maintain a competitive edge. A common challenge many companies face is the need to automate Extract, Transform, Load (ETL) workflows for data stored in cloud environments, such as Amazon S3.

This article presents a serverless solution that leverages the following AWS services to automate the process of transforming and analyzing data files stored in Amazon S3 and persistently storing the results:




By using this approach, you can seamlessly integrate data transformation and analysis into your processing pipeline, without the need for overseeing the underlying infrastructure.

The core of the solution involves several AWS services working in concert:

1. A cloud storage service that stores the input data files and receives the transformed output.
2. An ETL service that:
    a. Automatically discovers and catalogs the schema of the input data.
    b. Reads data from the source storage, performs transformations using Python-based distributed processing code, and writes processed data to the target storage.
3. A serverless compute service that triggers the ETL job automatically when new data is uploaded to the source storage.
4. A monitoring service that tracks the ETL process and stores logs for troubleshooting purposes.

This serverless architecture, with a managed ETL service as the processing engine, a serverless compute service as the trigger mechanism, and cloud storage as the data lake, allows for a scalable, efficient, and automated data processing workflow, without the need for managing and maintaining complex infrastructure.

# Tasks

In this article, you will learn how to set up and automate an ETL workflow using AWS Glue, AWS Lambda, and Amazon S3. The architecture described enables seamless data transformation and integration into your data pipeline, leveraging serverless technologies for scalability and efficiency.

To complete the steps in this article, you will:

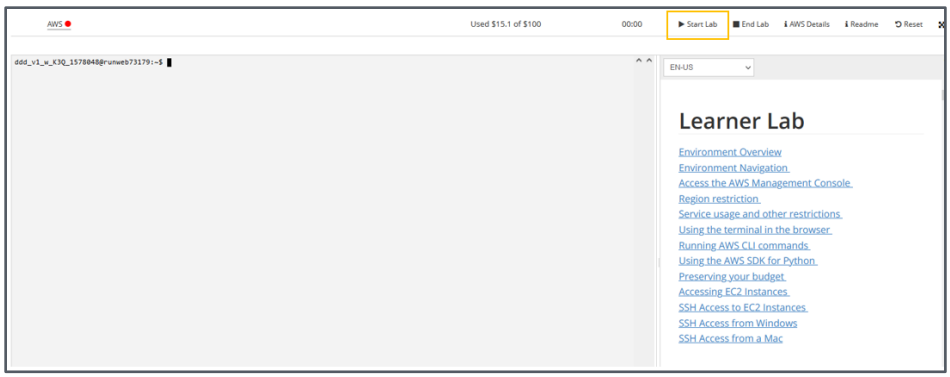
1. **Set Up Amazon S3 Buckets**: Create an S3 bucket to act as the source for raw data files and another one as the target for transformed data.
2. **Configure AWS Glue Data Catalog**: Define a Glue Data Catalog to automatically discover and catalog the schema of the input data files stored in S3.
3. **Create an AWS Glue ETL Job**: Set up an AWS Glue ETL job to:
    - Read data from the S3 bucket.
    - Perform Python-based distributed data transformations.
    - Write the transformed data back to the target S3 location.
4. **Deploy an AWS Lambda Function**: Create a Lambda function that will:
    - Trigger the AWS Glue ETL job whenever new data is uploaded to the source S3 bucket.
    - Monitor the Glue job status and log the results in Amazon CloudWatch.
5. **Test Glue Job ETL Workflow**: Upload sample data files to the S3 bucket to verify that:
    - The Lambda function triggers the Glue ETL job as expected.
    - Transformed data is written to the target S3 bucket.
    - Logs and metrics are correctly recorded in CloudWatch.

By the end of this article, you will have an automated, serverless data processing pipeline that is scalable, resilient, and easy to maintain, ensuring your data is transformed and analyzed efficiently.

# Prerequisites

Before continuing with the tasks listed above, you will need:

a. **Access to an AWS Academy Learner Lab (LL) course**: From the Student View of the LL, click the Start Lab button as shown in the screenshot below:




**Note:** For information on how to view a Learner Lab course in student view, please review **How to view a Learner Lab in Student View** section from [AWS Academy Learner Lab Educator Guide.]

b. After starting the lab, wait until the AWS icon turns green as shown below:


Then click on the AWS icon to display the AWS Management console.

# Implementation

## Set Up Amazon S3 Buckets

The purpose of creating two separate S3 buckets is to establish a clear data flow and organization in your data processing pipeline. The source bucket will store the raw, unprocessed data files, while the target bucket will contain the transformed, processed data. This separation allows for better data management, easier tracking of data transformations, and improved organization of your data assets.

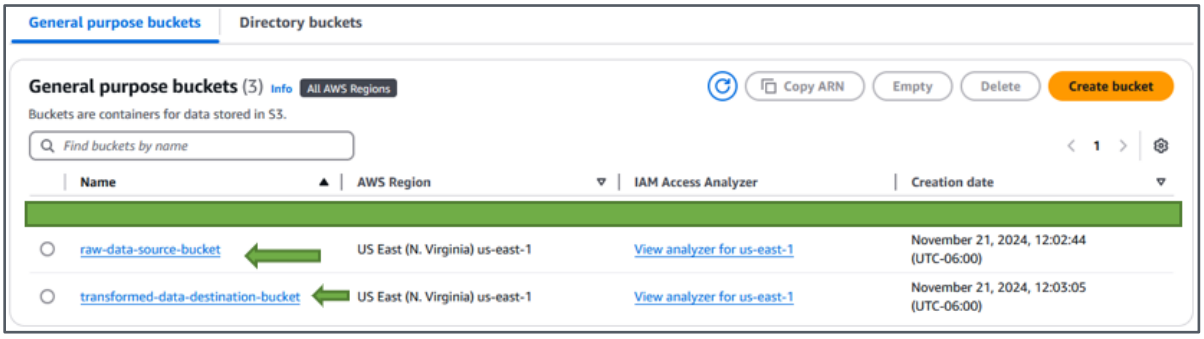
Steps for creating the S3 buckets:

a. From the Management console, in the search box at the top of the console, type **S3**.
b. Click on **S3** in the search results to navigate to the Amazon S3 console.
c. On the S3 console, click the **Create bucket** button in the upper right corner of the page.
d. For the **source** bucket:

- Enter a unique name for your source bucket (e.g., ***raw-data-source-bucket-123***). Replace '123' with a random number to ensure your bucket name is unique.
- Leave all other options set to their default values and choose **Create bucket**.

e. Repeat the process for the **target** bucket:
- Click **Create bucket** again.
- Enter a unique name for your destination bucket (e.g., ***transformed-data-destination-bucket-123***). Replace '123' with a random number to ensure your bucket name is unique.
- Leave all other options set to their default values and choose **Create bucket**.

After completing these steps, you will have created two Amazon S3 buckets - one for raw data and one for transformed data - both ready to use in your analysis workflow.




## Upload a data file to source bucket

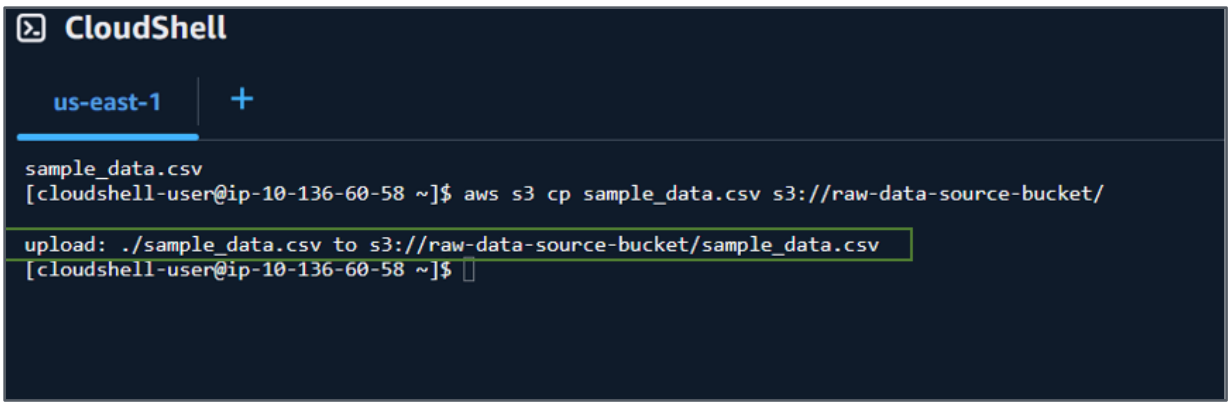
To upload the sample data file to your source bucket, you will use AWS CloudShell as your terminal to execute an AWS CLI command. You can either create your own sample .csv data file or use the provided sample_data.csv file included in the accompanying folder for this demo. Follow these steps:

- Download the sample_data.csv file from the folder accompanying this article to your local machine.
- Open AWS CloudShell in the AWS Management Console.
- Use the AWS CloudShell file upload feature to upload the sample_data.csv file from your local machine to CloudShell.
- Once the file is in CloudShell, use the following AWS CLI command to copy the sample data file to your source bucket:

***aws s3 cp sample_data.csv s3://raw-data-source-bucket/***

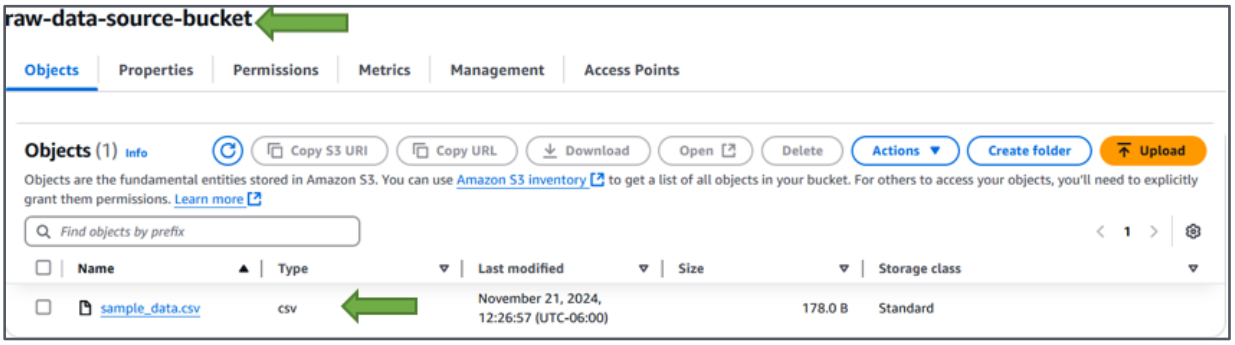
This process will transfer the sample data file from AWS CloudShell to your designated S3 source bucket.

After successfully executing the above CLI command, you should see confirmation that the data file has been copied to the source bucket. The output will look similar to the following:




**NB:** Don't forget to replace ***raw-data-source-bucket*** with your actual source bucket name and ***sample_data.csv*** with your specific sample data filename in the CLI command.

To verify the upload, navigate to the Amazon S3 console in your AWS Management Console, locate and click on your source bucket from the list of S3 buckets, and check the bucket's contents. You should now see your sample data file listed, which should appear similar to the following:




## Configure AWS Glue Data Catalog

The AWS Glue Data Catalog is a centralized metadata repository that automatically discovers and catalogs the schema of your input data files stored in S3, facilitating easier data querying and analysis. Follow these steps to configure the AWS Glue Data Catalog:

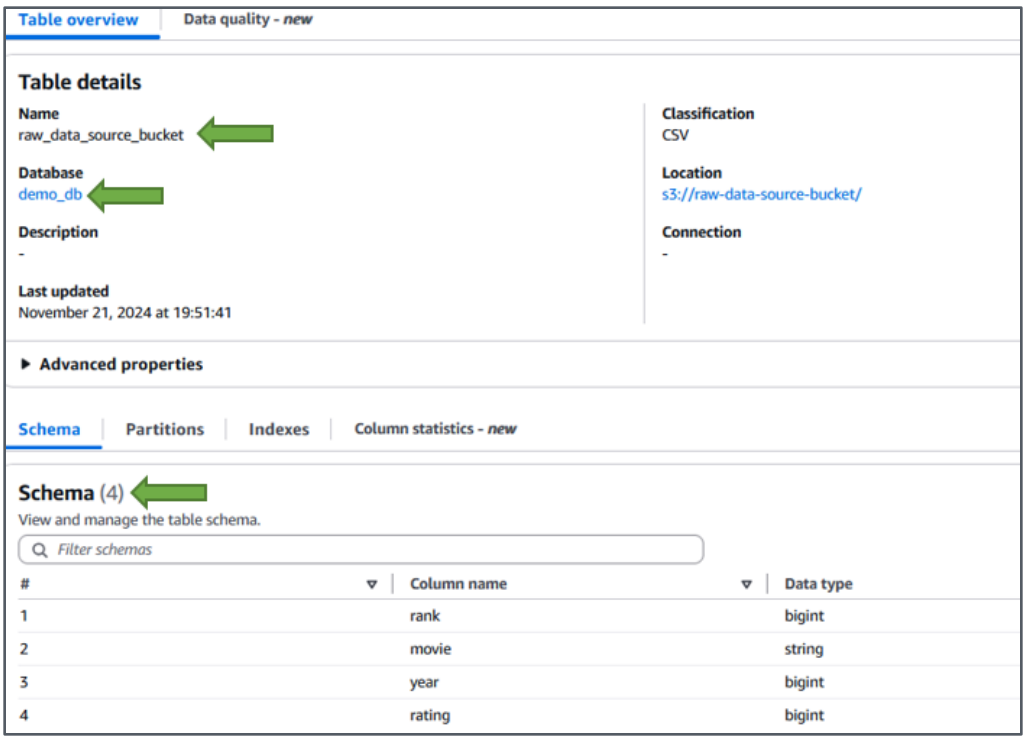
1. From the Management console, in the search box at the top of the console, type **Glue**.
2. Click on **AWS Glue** in the search results to navigate to the AWS Glue console.
3. On the Glue Console, locate the **Data Catalog** section from the navigation menu and click on **Databases**.
4. Click **Add database** and provide a name (e.g., demo_db), then click **Create database**.
5. In the **Data Catalog** section, click on **Tables**.
6. Select **Add tables using crawler**.

7. Name your crawler (e.g., S3SourceDataCrawler) and click **Next**.
8. Under **Data sources**, click **Add a data source**.
9. Select **S3** as the data source and specify the S3 path for your CSV files (e.g., s3://raw-data-source-bucket/). Click **Add an S3 data source** to continue.
10. Click **Next**.
11. From the **Existing IAM role** dropdown menu, choose **LabRole**.
12. Click **Next**.
13. Select the database you created earlier as your target database (e.g., demo_db).
14. Set the crawler frequency to **On demand** for this activity.
15. Click **Next**.
16. Review your settings and click **Create crawler**.
17. Once the crawler is created, select it and click the **Run** button to start the cataloging process.

    **Note**: It may take a few seconds for the crawler to be created.
18. After the crawler completes, navigate to **Databases** and then **Tables** in the AWS Glue console to view the discovered schema and table information.

This process will create a metadata catalog of your S3 data, enabling efficient querying and analysis in subsequent steps of your data processing workflow.

Your table and its details should appear similar to the following:



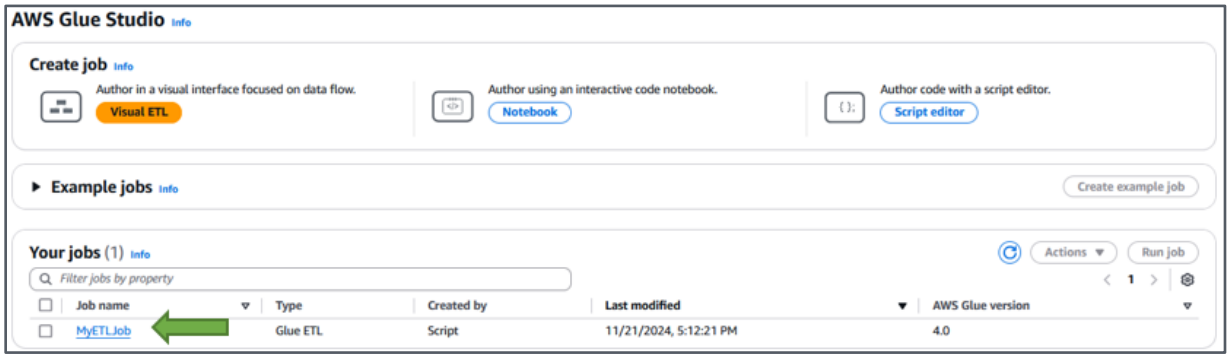

## Create an AWS Glue ETL Job

The following steps guide you through creating an AWS Glue ETL job, which will transform your data as specified in the Job script.

1. Navigate to the **ETL Jobs** section in the AWS Glue Console.
2. Under the **Create job** section, click on the **Script editor** option.
3. Choose **Upload script** option and select the ***MyETLJob.py*** file from the folder accompanying this article. Leave **Spark** as the Engine.
4. Click **Create script.**
5. In the **Job details** tab, provide a name for your job (e.g., MyETLJob).
6. Select **LabRole** from the IAM role dropdown.
7. Leave the Glue version as default, ensure the job type is set to **Spark**, and the Language is set to **Python**.
8. In the **Advanced properties** section, add the following job parameter (this is required):
    - **Key**: --JOB_NAME
    - **Value**: MyETLJob
9. Click the **Script** tab. To adapt the script to your environment, modify the following parameters as necessary:

- **glue_db** - Your Glue database name (from AWS Glue Data Catalog)
- **glue_tbl** - Your Glue table name (from AWS Glue Data Catalog)
- **s3_write_path** - Target S3 bucket path for storing transformed data

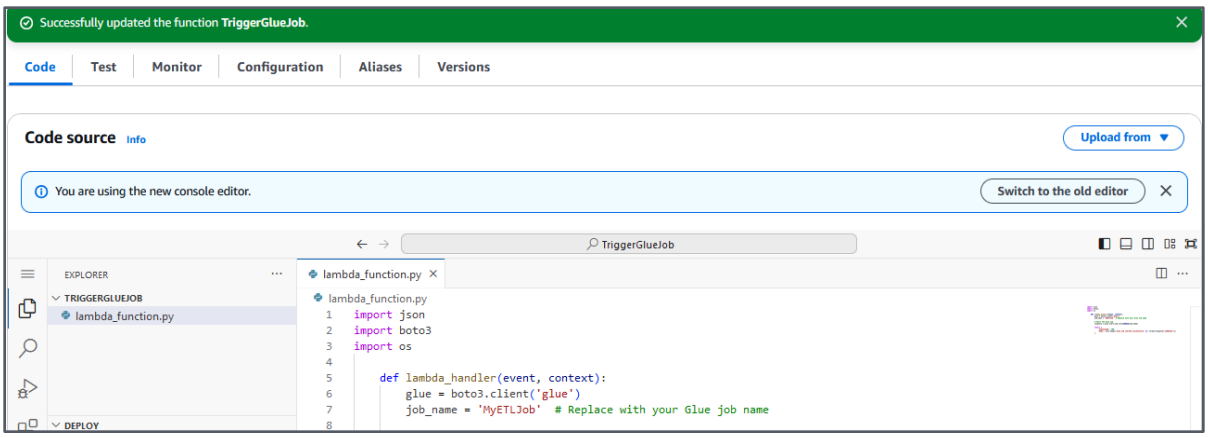
10. Leave all other settings at their default values.
11. Review your Job details and click **Save** to create the job.




## Deploy an AWS Lambda Function

The following steps guide you through creating an AWS Lambda function that automatically triggers the Glue ETL job when new data is uploaded to your S3 bucket.

1. Navigate to the AWS Lambda console.
2. Click on **Create function**.
3. Choose **Author from scratch** and provide a name for your function (e.g., TriggerGlueJob).
4. Select Python 3.8 from the Runtime dropdown.
5. Choose **Use an existing role** under the **Execution role** in the Change default execution role section.
6. Select **LabRole** from the Existing role dropdown.
7. Click **Create function**.
8. In the Lambda function code editor, replace the existing code with the contents of the ***lambda_function.py*** file in the folder accompanying this article.
9. On line 7 of the code, replace the placeholder with your actual **Glue job name**.
10. Click the **Deploy** button to save your changes and update the Python code for your Lambda function.

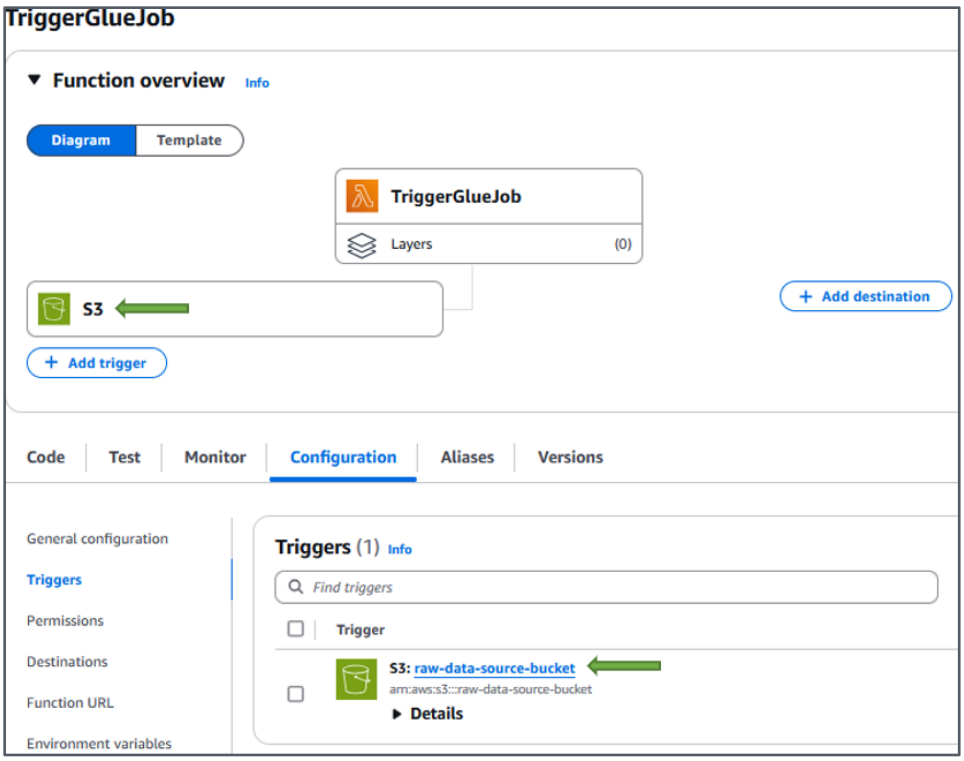



## Adding a Trigger to a Lambda Function

A trigger allows your Lambda function to automatically execute in response to specific events in AWS. In this case, you will configure the trigger to respond to new object uploads in your source S3 bucket. This enables automation of your data processing workflow by triggering your AWS Glue ETL job whenever a new file is uploaded to the bucket.

1. Navigate to **Functions** in the AWS Lambda Console.
2. Select the Lambda function you created in the previous section
3. Scroll to the **Function overview** section.
4. Click **Add trigger** to open the trigger configuration panel.
5. From the **Trigger configuration** menu, choose **S3** as the trigger type.
6. Select or specify the name of your source S3 bucket where your input files (e.g., CSV files) was uploaded.
7. From the **Event type** dropdown, choose **All object create events** to listen for any new object uploads.
8. Under **Recursive invocation**, select the check box to acknowledge that using the same Amazon S3 bucket for input and output is not recommended.
9. Choose **Add**.

After completing these steps, the Lambda function will automatically execute whenever a new object is uploaded to the source S3 bucket. This setup triggers your AWS Glue ETL job, enabling seamless and automated data processing from ingestion to transformation.




## Test your AWS Glue ETL Job Workflow

Testing your AWS Glue ETL job is crucial to ensure that the data transformation and processing workflow operates as expected. The test involves running the job, monitoring its progress, and verifying the output through logs. Follow these steps to validate your ETL job:

1. Navigate to the **AWS Glue service**.
2. In the left navigation pane, click on **ETL Jobs**.
3. Locate your ETL job in the list and select it.
4. Click the **Run** button at the top of the page to start the job execution.
5. Once the job starts running, click on the **Runs** tab on the job page.
6. Monitor the status of the job. It should initially display as **Running**.
7. Keep track of the job's progress in this tab.

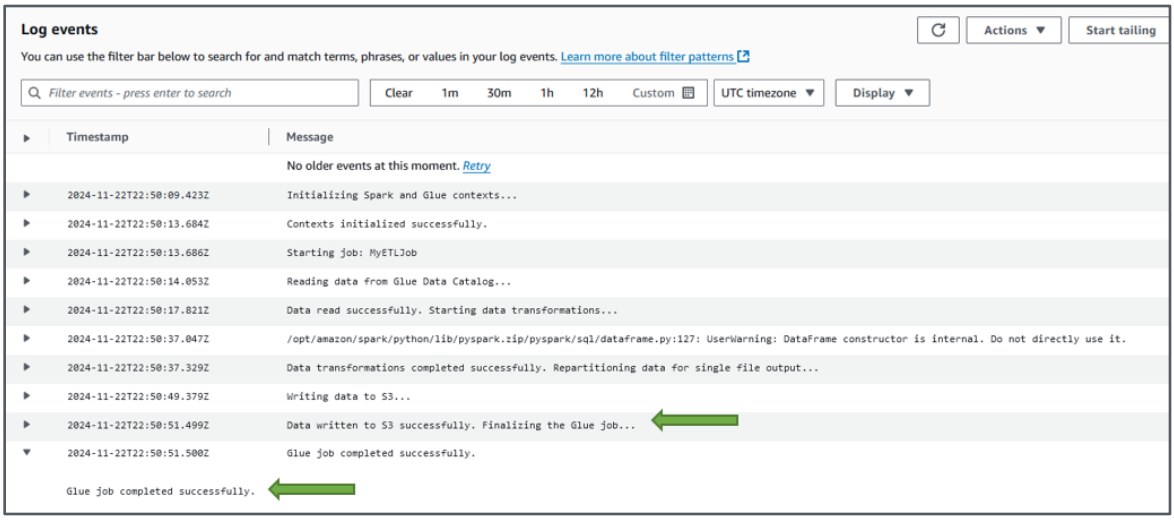
If the job is correctly configured, the status should change to **Succeeded** within approximately two minutes. If the job fails or takes longer than expected, review the configuration and troubleshoot as needed using the error logs.

To view detailed logs:

1. Go to the **Run details** tab of your job.
2. Click on **Output logs** to access the CloudWatch Logs.
3. In the log stream, you should see various outputs as printed by the print statements in your script.

These logs are generated due to the print statements in your Glue job Python code. They can provide valuable information about the job's execution and help in debugging if issues arise.

Remember, thorough testing and monitoring of your ETL job are essential for maintaining a reliable data processing pipeline.




After the job completes successfully, you can retrieve the output file from the S3 bucket you specified as the output location in your Glue job script. The output file will be named in the following format:

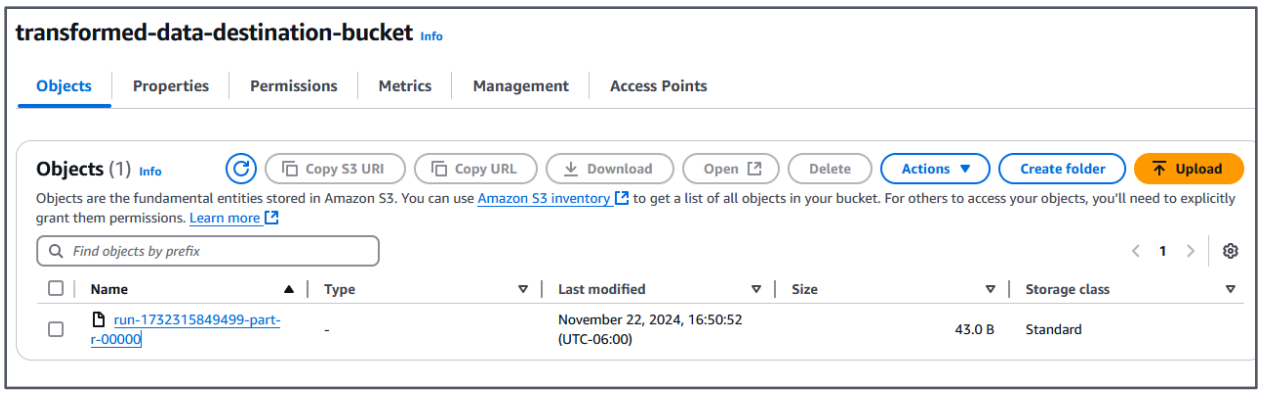
***run-[JobRunId]-part-r-[PartitionNumber]***. For example: run-1732315849499-part-r-00000

To access this file:

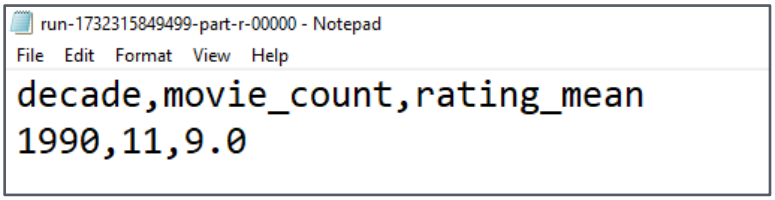
1. Open the Amazon S3 console.
2. Navigate to the bucket you specified as the output location in your Glue job script.
3. Look for the file with the naming pattern described above.
4. You can download this file to verify the results of your ETL job.

**Note that:**

- The [JobRunId] is a unique identifier for each job run.
- The [PartitionNumber] is typically 00000 for single-partition output, but may vary if your job produces multiple output partitions.
- Depending on your job configuration, you might see multiple output files, each representing a different partition of your data.




Reviewing this output file is an important step in validating that your ETL job has processed and transformed the data as expected.




## Clean Up Your Resources

After testing your AWS Glue ETL workflow and verifying its functionality, it's important to clean up the resources to maintain an organized AWS environment and avoid incurring unnecessary costs.

Below are the steps to delete the resources provisioned for this project:

### 1. Delete the AWS Glue ETL Job

a. Navigate to the AWS Glue Console.
b. Select the ETL Jobs tab and locate the ETL job you created.
c. Select the job, click Actions, and choose Delete.
d. Confirm the deletion in the dialog box.

### 2. Delete the S3 Buckets

a. Go to the Amazon S3 Console.
b. Locate the source and target buckets used for the ETL workflow.
c. For each bucket:
   - Click on the bucket and select Empty.
   - In the confirmation dialog, type permanently delete to confirm the deletion of all objects in the bucket.
   - Once the bucket is empty, click Delete.
   - Type the full name of the bucket in the confirmation dialog and click Delete bucket.

### 3. Delete the AWS Lambda Function

a. Navigate to the AWS Lambda Console.
b. Locate the Lambda function you created to trigger the Glue ETL job.
c. Select the function, click Actions, and choose Delete.
d. In the confirmation dialog, type delete to confirm and click Delete.

### 4. Delete Logs in Amazon CloudWatch

a. Open the Amazon CloudWatch Console.
b. Go to the Log groups section.
c. Locate the log groups associated with your Glue job and Lambda function (e.g., /aws-glue/jobs/output and /aws/lambda/YourFunctionName).
d. Select the log groups, click Actions, and choose Delete log group.
e. Confirm the deletion in the dialog box.

By following these steps, you will completely remove the resources created for this AWS Glue ETL workflow, including the Glue job, S3 buckets, Lambda function, and CloudWatch logs. This ensures that your AWS environment is clean and free from unused resources, preventing unexpected charges.

Perform these cleanup steps only after completing your testing and ensuring you no longer need the resources for this activity.