AI-Powered Data Pipeline for E-commerce Warehouse Demand Forecasting

# Implementation Steps

Data Ingestion Using Terraform and Python

**Data Processing Using AWS Glue & Lambda**

Machine Learning with Amazon SageMaker (for demand forecasting)

Model Integration & Optimisation

Deployment & Scaling



Table of Contents

[Implementation Steps 1](#_Toc192929696)

[Step 2: Data Processing Using AWS Glue & Lambda 2](#_Toc192929697)

[Objective: 2](#_Toc192929698)

[Verify you have all the required infrastructure 3](#_Toc192929699)

[Set Up AWS Glue for Data Processing 3](#_Toc192929700)

[Create an AWS Glue ETL Job 5](#_Toc192929701)

[Automate Processing with AWS Lambda 6](#_Toc192929702)

[Test the Data Processing Pipeline 7](#_Toc192929703)

[Issues Encountered 9](#_Toc192929704)

[Summary of Key Deliverables 10](#_Toc192929705)

# Step 2: Data Processing Using AWS Glue & Lambda

## Objective:

* Use AWS Glue to clean and transform raw data stored in S3.
* Automate data pipelines using AWS Lambda for event-driven processing.
* Structure data into a format suitable for machine learning models.

## Verify you have all the required infrastructure

1. Check if Kinesis Firehose is active:



1. Check if data exists in S3



1. Check the latest data in S3



## Set Up AWS Glue for Data Processing

* 1. Create an AWS Glue database

Go to the AWS Glue Console → Databases → Click Add database → Enter Database name → Click Create database.



* 1. Create an AWS Glue Crawler

Go to AWS Glue Console → Crawlers → Click Create Crawler → Enter Crawler name → Choose Data Source → Select S3 → Enter S3 path.

Set IAM Role: Choose "Create new IAM Role" and name it or choose an existing role.

Set Output Database: Choose your database → Click Create crawler.



Run the crawler



* 1. Verify AWS Glue table

Go to AWS Glue Console → Databases → Click your database → Verify your database table is created → Check the schema (should match JSON structure from Kinesis Firehose).



## Create an AWS Glue ETL Job

1. Go to AWS Glue Console → Jobs → Click Create job → enter a job name → choose an IAM role → choose script type → set the data source → set the data target → replace the script



1. Save the job and run it



## Automate Processing with AWS Lambda

1. Create an AWS Lambda Function

Go to AWS Lambda Console → click create function → author from scratch → enter function name → choose runtime → set IAM role → click create function.



1. Add Lambda Code to Trigger Glue Job and deploy it



1. Create an S3 event trigger

Go to S3 console → select data ingestion bucket → properties → event notifications → click create event notification → enter event name and type → destination select lambda function and choose your lambda function.



## Test the Data Processing Pipeline

1. Re-run the python script to simulate data ingestion

 python3 simulate\_data.py



1. Check S3

aws s3 ls s3://ecommerce-warehouse-ingestion/processed-data/



1. Verify glue job execution

aws glue get-job-runs --job-name warehouse\_data\_processing



# Issues Encountered

**Problem 1:** Glue Job Failing Due to Incorrect DataFrame Conversion**.** The Glue job initially failed with a *TypeError: frame\_or\_dfc must be DynamicFrame or DynamicFrameCollection because .toDF()* was used without converting it back to a *DynamicFrame.*

**Resolution 1:** Fixed the error by properly converting back to a **DynamicFrame** before writing to S3:

**Python script**

*df = datasource0.toDF()*

*df = df.withColumnRenamed("quantity\_sold", "sales")*

*df = df.withColumnRenamed("current\_inventory", "inventory")*

*dynamic\_frame = DynamicFrame.fromDF(df, glueContext)*

*glueContext.write\_dynamic\_frame.from\_options(*

 *frame=dynamic\_frame,*

 *connection\_type="s3",*

 *connection\_options={"path": "s3://ecommerce-warehouse-ingestion/processed-data/"},*

 *format="json"*

*)*

**Problem 2:** Lambda Execution Role Not Showing in the Dropdown.When creating the AWS Lambda function, the execution role did not appear in the dropdown.

**Resolution 2:** Manually verified IAM roles using:

*aws iam list-roles | grep "AWSGlueServiceRole-warehouse"*

Manually attached the role to Lambda after creation via the AWS Console.

**Problem 3:** IAM Permission Denied When Updating Lambda Role via CLI. Attempting to update the Lambda function’s IAM role via CLI resulted in *AccessDeniedException.*

**Resolution 3:** Updated the IAM permissions for the Kinesis Firehose user to allow role modification.

**Added this permission to the user's policy:**

**json**

*{*

 *"Effect": "Allow",*

 *"Action": [*

 *"lambda:UpdateFunctionConfiguration",*

 *"iam:PassRole"*

 *],*

 *"Resource": "\*"*

*}*

**Problem 4:** AWS Glue Job Did Not Trigger Automatically**.** After setting up an S3 event notification to trigger the Lambda function, the Glue job did not start automatically.

**Resolution 4:** Verified S3 event configuration and Lambda permissions and manually invoked the Lambda function using:

**bash**

*aws lambda invoke --function-name trigger\_glue\_processing output.txt*

# Summary of Key Deliverables

1. **AWS Glue Job Debugging:** Successfully identified and corrected database and table name mismatches, ensuring proper DataFrame conversions.
2. **IAM Permission Handling:** Modified IAM roles and policies to grant appropriate permissions for AWS Lambda, AWS Glue, and Amazon Kinesis Firehose, ensuring smooth execution.
3. **Automated Event-Driven Processing:** Integrated Amazon S3, AWS Lambda, and AWS Glue to create a fully serverless data pipeline, improving efficiency and reducing manual intervention.