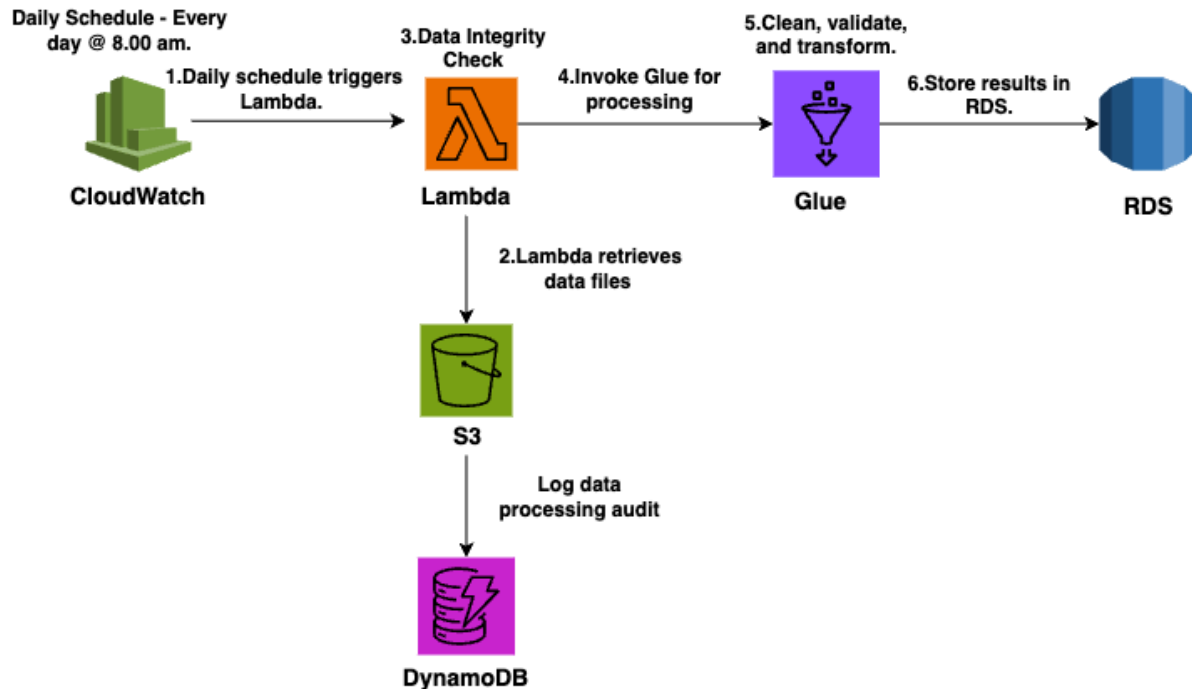


Batch Based ETL Pipeline Documentation

Architecture Diagram



Flow of our Architecture Diagram

1. CloudWatch:

- **Scheduled Event:** Triggers the Lambda function daily at 8:00 am.

2. Lambda Function:

- **Retrieve Data from S3:** Fetches the data files from the specified S3 bucket.
- **Data Integrity Check:** Performs initial checks to ensure data quality (e.g., checking for missing CustomerID, validating TransactionDate).
- **Log Data Processing Audit:** Writes an entry to the DynamoDB table for tracking and auditing the data processing status.

3. S3 Bucket:

- **Storage:** Holds the raw data files in the 'source' folder and processed data files in the 'processed' folder.

4. DynamoDB:

- **Audit Log:** Records the processing details such as file name, upload timestamp, processing start and end times, status, and Glue job ID.

5. Invoke Glue Job:

- **Trigger:** The Lambda function invokes the Glue job for further processing after the initial data integrity checks.

6. Glue:

- **Data Cleaning:** Ensures fields like CustomerID are not empty; replaces missing values with 'N/A'.
- **Data Validation:** Validates date fields like TransactionDate and replaces invalid dates with the current date.
- **Transformation:** Applies necessary data transformations to prepare the data for loading into RDS.
- **Load Data to RDS:** Stores the cleaned and transformed data into the RDS instance.

7. RDS:

- **Data Storage:** Holds the final processed data ready for querying and analysis.

1. Setting Up IAM Roles and Policies

1.1 Lambda Role

1.1.1 Create Role:

- AWS Console >> IAM >> Create role >> Entity Type (AWS Service) >> Service (Lambda)

1.1.2 Add Permissions:

- Attach AmazonS3FullAccess
- Attach CloudWatchLogsFullAccess
- Attach AmazonDynamoDBFullAccess

1.1.3 Review and Create Role:

- Name: **LambdaS3ExecutionRole**

1.2 Glue Service Role

1.2.1 Create Role:

- AWS Console >> IAM >> Create role >> Entity Type (AWS Service) >> Service (Glue)

1.2.2 Add Permissions:

- Attach AmazonS3FullAccess
- Attach AWSGlueServiceRole
- Attach AWSGlueServiceNotebookRole

1.2.3 Review and Create Role:

- Name: **GlueServiceRole**

1.3 EventBridge Role

1.3.1 Create Role:

- AWS Console >> IAM >> Create role >> Entity Type (AWS Service) >> CloudWatch Events >> Create Custom Policy.

1.3.2 Custom Policy:

- Name: EventBridgeInvokeLambdaPolicy
- JSON Policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction",
        "lambda:InvokeAsync"
      ],
      "Resource": "*"
    }
  ]
}
```

1.3.3 Attach to Existing Role:

- Name: **EventBridgeInvokeLambdaRole**

1.4 Additional Policy for DynamoDB Access

1.4.1 Attach AmazonDynamoDBFullAccess to Lambda Role:

- Go to IAM >> Roles >> LambdaS3ExecutionRole >> Attach policy >> AmazonDynamoDBFullAccess

2. Setting Up Lambda Function

2.1 Create Lambda Function

2.1.1 Create Function:

- AWS Console >> Lambda >> Create function >> Author from scratch
- Function name: **'ETL_Lambda_Function'**
- Runtime: Python 3.x
- Role: **LambdaS3ExecutionRole**

2.1.2 Upload Script:

- Place the script in the S3 bucket under scripts/ folder.
- Update the script path in the Lambda function configuration if needed.

2.2 Description of the Role Used

The **LambdaS3ExecutionRole** has permissions to read from S3, write logs to CloudWatch, and access DynamoDB for auditing purposes.

3. Steps to Create an RDS Instance

1. Sign in to the AWS Management Console.
2. Navigate to the RDS Dashboard.
3. Click on "Create database".
4. Select "Standard Create" and choose "PostgreSQL".
5. Configure the instance:
 - Enter DB identifier, master username, and password.
 - Select instance class (e.g., db.t3.micro).
 - Set storage type and size.
6. Configure connectivity:
 - Select VPC, subnet group, and security group.
 - Set to publicly accessible if needed.
7. Review and create the database.
8. After creation, note down the endpoint and port for database connections.

3.1 RDS Configuration

1. DB Identifier: Choose a unique identifier.
2. Instance Class: Select an instance class (e.g., db.t3.micro).
3. Engine: Choose PostgreSQL.
4. Region & Availability Zone: Select your preferred region (e.g., us-east-1a).

3.2 Connectivity & Security

1. Endpoint: Generated after instance creation.
2. Port: Use port 5432 for PostgreSQL.
3. VPC: Select the VPC.
4. Subnets: Ensure multiple subnets are listed.
5. Security Group:
 - Inbound Rule: Allow traffic from 0.0.0.0/0 on port 5432.
 - Outbound Rule: Allow all outbound traffic (0.0.0.0/0).
6. Publicly Accessible: Set to Yes if needed.
7. Certificate Authority: Choose appropriate SSL certificate.

4. Creating the Audit Table for S3

4.1 Importance of Auditing

Auditing is crucial to track the processing status of files, ensuring data integrity and successful pipeline execution. The audit table logs the start and end times of processing each file, along with their status.

4.2 Creating Audit Table in DynamoDB

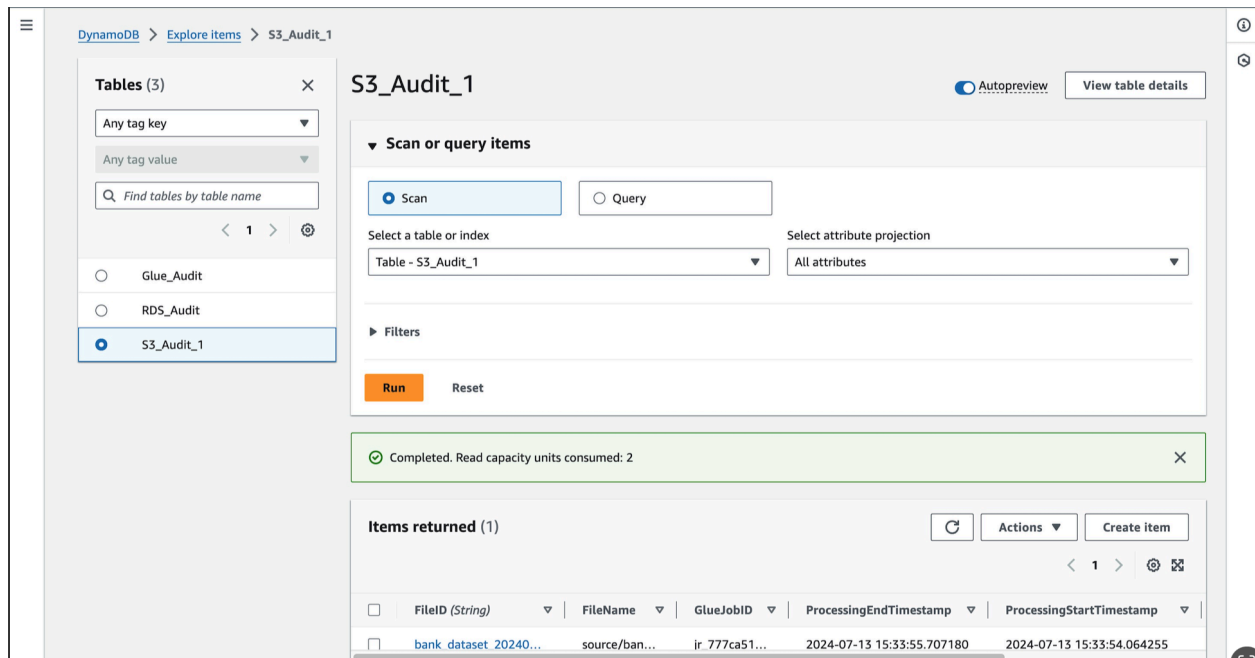
1.Create Table:

- Table Name: S3_Audit_1
- Partition Key: FileID (String)
- On-Demand Capacity Mode

4.3 Example Table Structure

- FileID: Unique identifier for each file
- FileName: Name of the file in S3
- UploadTimestamp: Timestamp when the file was uploaded to S3
- ProcessingStartTimestamp: Timestamp when processing started
- ProcessingEndTimestamp: Timestamp when processing ended

- Status: Processing status (e.g., processing, completed)
- GlueJobID: ID of the Glue job processing the file



5. Setting Up Glue Job

5.1 Glue Job Configuration

- Name: Select_Any_name
- IAM Role: GlueServiceRole
- Type: Spark
- Glue Version: Glue 4.0
- Language: Python 3
- Worker Type: G 1X (4vCPU and 16GB RAM)
- Requested Number of Workers: 2
- Job Bookmark: Enable
- Script Filename: ETL_Job_to_RDS.py
- Script Path: your_scripts_location
- Temporary Path: your_temp_location
- Maximum Concurrency: 1
- Connections: None

5.2 Script for Glue Job : Will be provided as a separate python file.

5.3 Steps to Upload PostgreSQL JDBC Driver

5.3.1 Upload JDBC Driver:

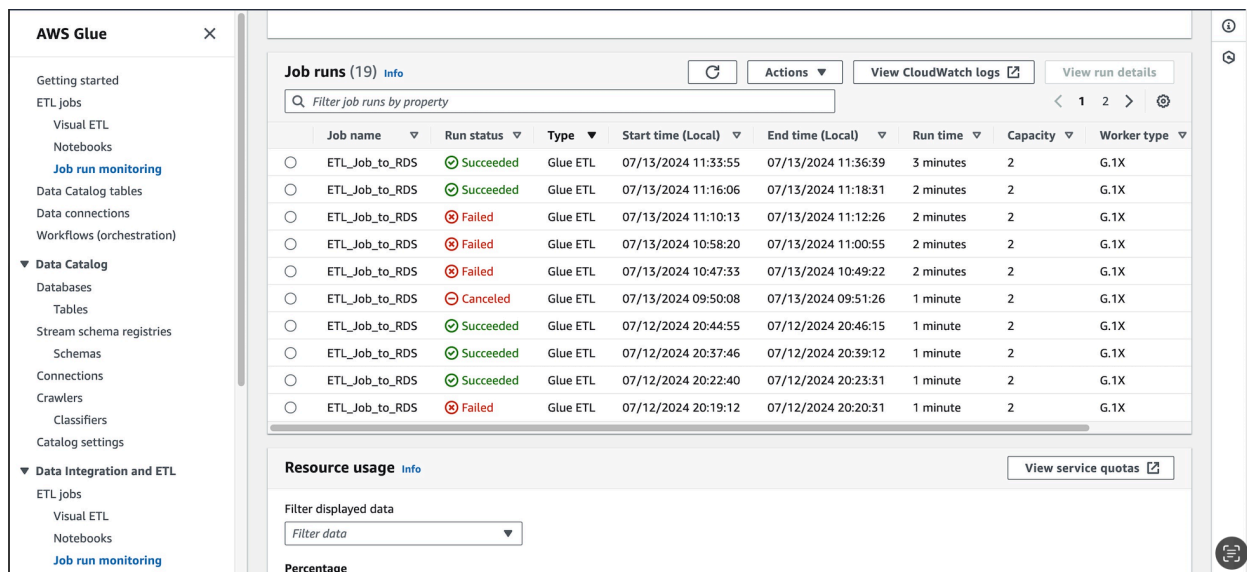
- Upload the PostgreSQL JDBC driver (e.g., postgresql-42.2.19.jar) to an S3 bucket.
- Ensure the Glue job has access to this S3 bucket.

5.3.2 Create and Configure Glue Job:

- Go to the AWS Glue Console.
- Create a new Glue job.
- Provide the necessary configurations and script path.
- Ensure the IAM role used by the Glue job has the necessary permissions to access the RDS instance and the S3 bucket where the JDBC driver is stored.

5.3.3 Run and Monitor Glue Job:

- Run the Glue job and monitor the execution through CloudWatch logs.



The screenshot displays the AWS Glue console interface. On the left, a navigation pane shows the 'Job run monitoring' section selected under 'Data Integration and ETL'. The main panel, titled 'Job runs (19) info', contains a table of job runs. The table has columns for Job name, Run status, Type, Start time (Local), End time (Local), Run time, Capacity, and Worker type. The job name for all entries is 'ETL_Job_to_RDS'. The run statuses are a mix of 'Succeeded' (green checkmark) and 'Failed' (red X). The types are all 'Glue ETL'. The start and end times are listed in local time. The run times are mostly 1 or 2 minutes. The capacity is consistently 2, and the worker type is 'G.1X'. Below the table, there is a 'Resource usage' section with a 'Filter displayed data' dropdown and a 'Percentage' label.

| Job name | Run status | Type | Start time (Local) | End time (Local) | Run time | Capacity | Worker type |
|----------------|------------|----------|---------------------|---------------------|-----------|----------|-------------|
| ETL_Job_to_RDS | Succeeded | Glue ETL | 07/13/2024 11:33:55 | 07/13/2024 11:36:39 | 3 minutes | 2 | G.1X |
| ETL_Job_to_RDS | Succeeded | Glue ETL | 07/13/2024 11:16:06 | 07/13/2024 11:18:31 | 2 minutes | 2 | G.1X |
| ETL_Job_to_RDS | Failed | Glue ETL | 07/13/2024 11:10:13 | 07/13/2024 11:12:26 | 2 minutes | 2 | G.1X |
| ETL_Job_to_RDS | Failed | Glue ETL | 07/13/2024 10:58:20 | 07/13/2024 11:00:55 | 2 minutes | 2 | G.1X |
| ETL_Job_to_RDS | Failed | Glue ETL | 07/13/2024 10:47:33 | 07/13/2024 10:49:22 | 2 minutes | 2 | G.1X |
| ETL_Job_to_RDS | Canceled | Glue ETL | 07/13/2024 09:50:08 | 07/13/2024 09:51:26 | 1 minute | 2 | G.1X |
| ETL_Job_to_RDS | Succeeded | Glue ETL | 07/12/2024 20:44:55 | 07/12/2024 20:46:15 | 1 minute | 2 | G.1X |
| ETL_Job_to_RDS | Succeeded | Glue ETL | 07/12/2024 20:37:46 | 07/12/2024 20:39:12 | 1 minute | 2 | G.1X |
| ETL_Job_to_RDS | Succeeded | Glue ETL | 07/12/2024 20:22:40 | 07/12/2024 20:23:31 | 1 minute | 2 | G.1X |
| ETL_Job_to_RDS | Failed | Glue ETL | 07/12/2024 20:19:12 | 07/12/2024 20:20:31 | 1 minute | 2 | G.1X |

6. Accessing RDS Instance with DBeaver

6.1 Install DBeaver:

- Download and install DBeaver from the official website.

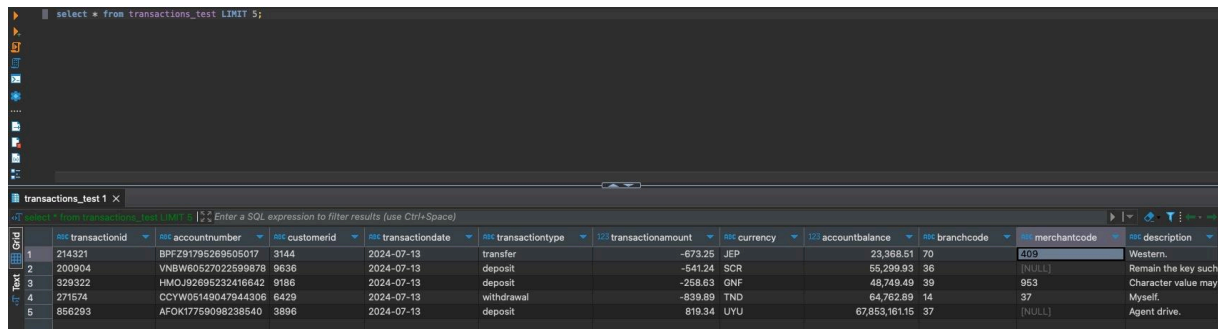
6.2 Create Connection:

- Open DBeaver and create a new connection.
- Select PostgreSQL as the database type.
- Enter the RDS endpoint, port, database name, username, and password.

6.3 Test Connection:

- Test the connection to ensure it is successful.
- Once connected, you can execute queries and interact with the RDS instance.
- Example Query:

SELECT * FROM transactions_test LIMIT 5;



The screenshot shows a database client interface. The top pane displays the SQL query: `select * from transactions_test LIMIT 5;`. The bottom pane shows the results of the query as a table with 10 columns and 5 rows of data.

| | transactionid | accountnumber | customerid | transactiondate | transactiontype | transactionamount | currency | accountbalance | branchcode | merchantcode | description |
|---|---------------|--------------------|------------|-----------------|-----------------|-------------------|----------|----------------|------------|--------------|---------------------|
| 1 | 214321 | BPF291795269505017 | 3144 | 2024-07-12 | transfer | -673.25 | JEP | 23,988.51 | 70 | 400 | Western. |
| 2 | 200904 | VNBW60527022698878 | 9636 | 2024-07-13 | deposit | -541.24 | SCR | 55,299.93 | 36 | [NULL] | Remain the key such |
| 3 | 329322 | HMCJ92695232416642 | 9186 | 2024-07-13 | deposit | -258.63 | GNF | 48,749.49 | 39 | 953 | Character value may |
| 4 | 271574 | CCYW05149047944306 | 6429 | 2024-07-13 | withdrawal | -839.89 | TND | 64,762.89 | 14 | 37 | Myself. |
| 5 | 856293 | AFOK17759098238540 | 3896 | 2024-07-13 | deposit | 819.34 | UYU | 67,853,161.15 | 37 | [NULL] | Agent drive. |

7. CloudWatch Event Configuration

Required Roles and Permissions

7.1 EventBridge Role:

- Name: EventBridgeInvokeLambdaRole
- Policy: EventBridgeInvokeLambdaPolicy (as described above)

7.2 Lambda Role:

- Name: LambdaS3ExecutionRole
- Policy: AmazonS3ReadOnlyAccess, CloudWatchLogsFullAccess, AmazonDynamoDBFullAccess

Creating CloudWatch Event Rule

7.3 Create Rule:

- AWS Console >> CloudWatch >> Events >> Create rule

7.4 Configure Rule:

- Event Source: S3
- Event Type: Object Created
- Target: Lambda Function (ETL_Lambda_Function)

7.5 Set Permissions:

- Ensure the EventBridgeInvokeLambdaRole has permissions to invoke the Lambda function.

Conclusion

This documentation provides a step-by-step guide to setting up an ETL pipeline using AWS services. Follow the steps carefully, and refer to the attached images and scripts for additional help. This setup ensures a smooth and