Serverless Data pipelines in AWS to process CSV files

Team	
Status	In progress

AWS PROJECT - 01

This project involves building serverless data pipelines on AWS to process CSV files. The pipeline automates ingestion, transformation and visualization of data. CSV files are uploaded to an raw data S3 bucket(csv-raw-data),triggering an AWS Lambda function to preprocess the data and store it in the processed data bucket(csv-processed-data).

AWS Glue is then used for further ETL operations(Extract, Transform, Load), and final data is stored in final data bucket(csv-final-data).Finally, Amazon QuickSight is used to create interactive dashboards and reports for visualizing the data.

1.**DATA INGESTION**: CSV files are uploaded to s3 bucket which serves as an central storage for raw and processed data.

2.**TRIGGER AND TRANSFORMATION**: An AWS lambda function is automatically triggered to preprocess the data, such as filtering and formatting, and passing it to AWS Glue for detailed ETL operations.

3.**DATA STORAGE**: Processed data is stored in Amazon S3 for scalable storage.

4.**VISUALIZATION**: Amazon QuickSight connects to the processed data and provides an interactive visualization dashboard.

SERVICES USED

1.**Amazon S3**:Used for scalable storage of raw and processed data, providing event driven architecture capabilities upon upload to s3.

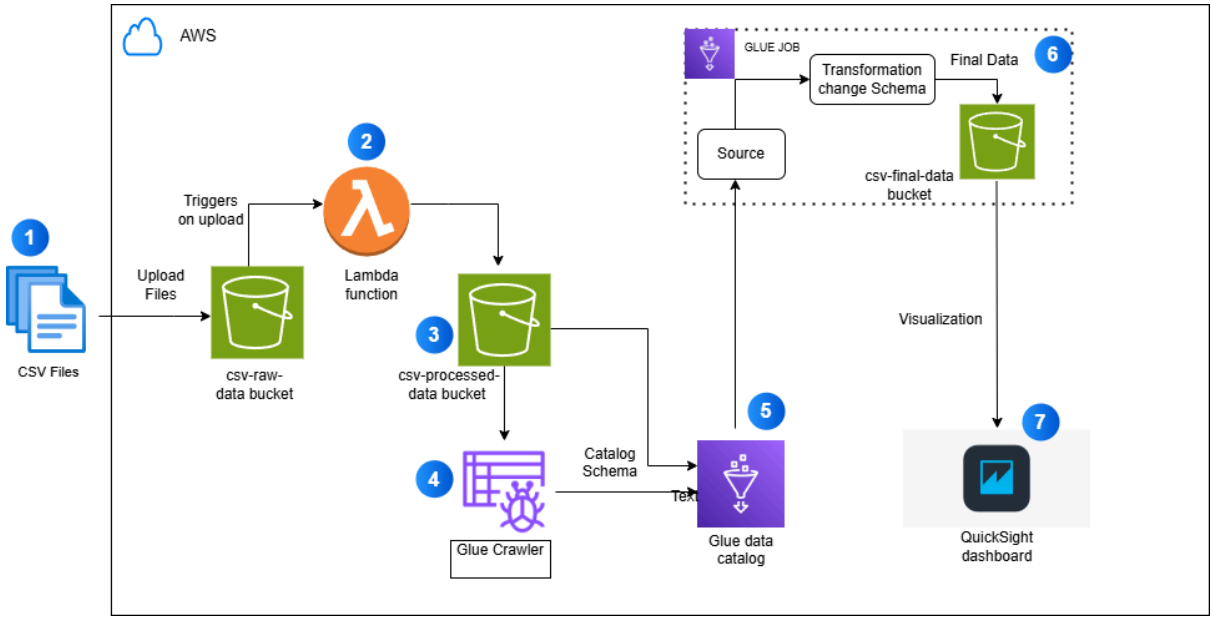
2.**AWS Lambda**: Acts as an serverless compute layer, automatically triggered to preprocess and clean csv files upon upload to s3.

3.**AWS Glue**: Provides ETL(Extract, Transform, Load) capabilities to turn data into usable format.

4.**Amazon QuickSight:** Offers interactive dashboards and reports for real time data visualization.

5.**IAM Roles and Policies**: Ensures secure access to s3,Lambda,Glue,and QuickSight.[Permissions]

ARCHITECTURAL DIAGRAM



Action items

1.Raw CSV files are uploaded to the csv-raw-data S3 bucket, initiating the pipeline

2.An **AWS Lambda function** is automatically triggered to read and preprocess the CSV files

3.The Lambda function filters/formats data and stores the cleaned files in the csv-processed data s3-bucket.

4.An **AWS Glue Crawler** scans the processed data and identifies the schema for further processing

5.The crawler updates the **AWS Glue Data Catalog**, creating a structured table for ETL processes.

6.Source → Transform → Store — An AWS Glue job extracts data from the table created in Glue Data Catalog, transforms it, and loads the final output into the **csv-final-data S3 bucket.**

7.**Amazon Quicksight** connects to final dataset, enabling an interactive dashboard of analytics

Steps to be performed

1.Create s3 buckets

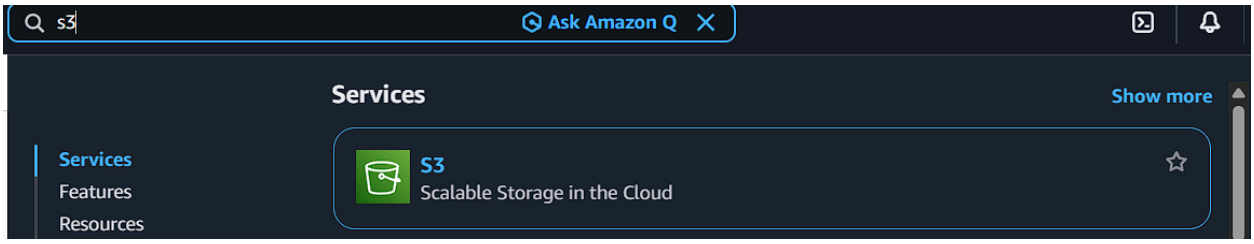
2.Configure IAM Roles and Policies

3.Setup Amazon Quicksight

#1.CREATE S3 BUCKETS

Amazon S3 will be the backbone of our pipeline, acting as storage for raw, processed and final data. Here we will be creating three s3 buckets for storing-Raw,processed and final data respectively.

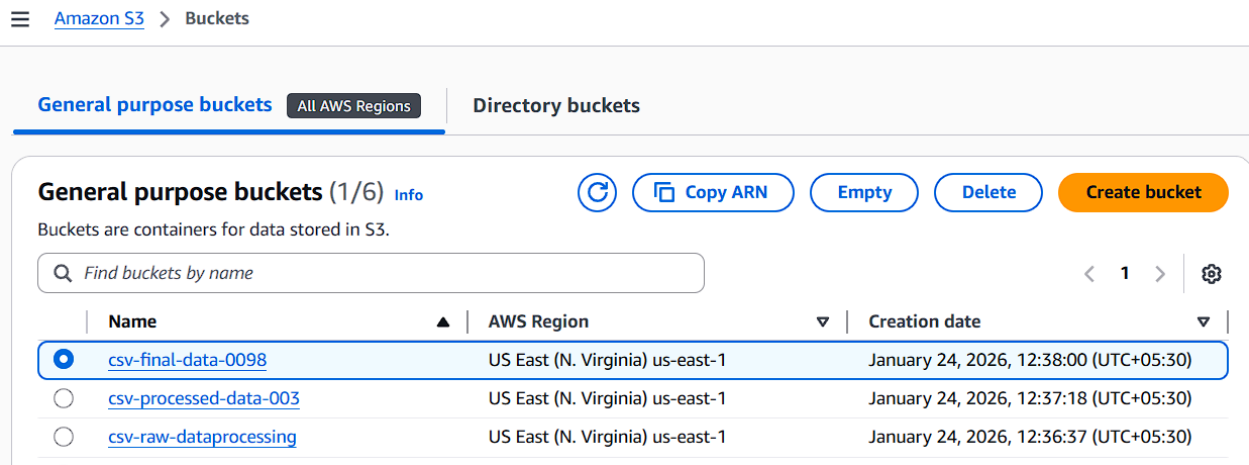
*Log in to AWS console and navigate to s3 service.



*Click on “Create bucket”.

*Fill out details like 1.Bucket name (csv-raw-data for raw data)(csv-processed-data for processed data)(csv-final-data for final data)

2.Region (Availability Zone): us-east-1 for all buckets



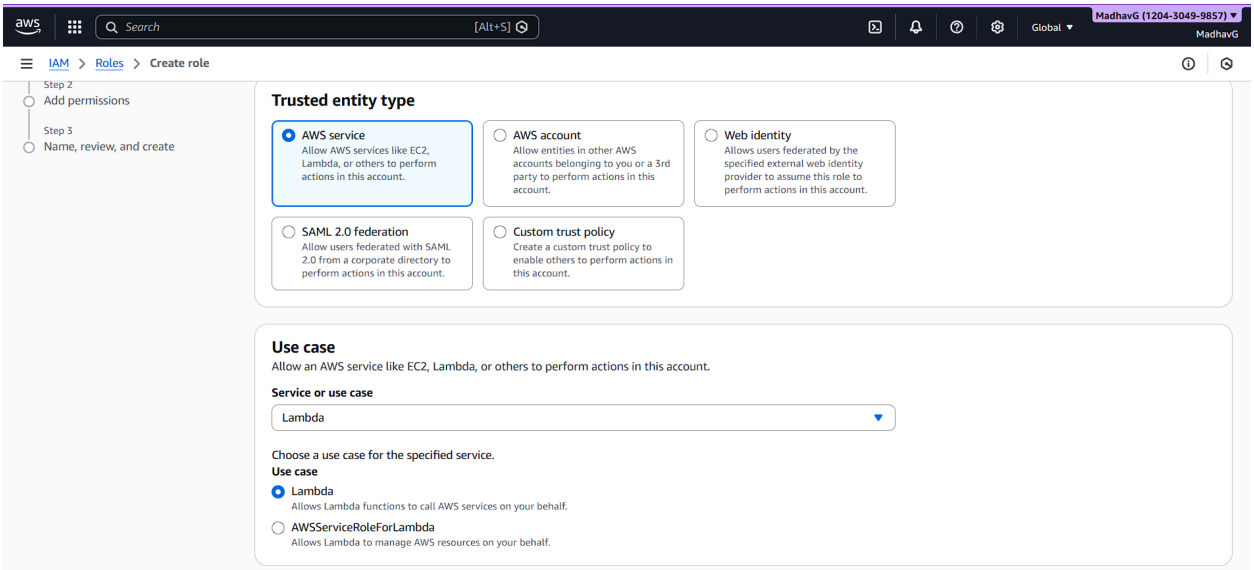
#2.CONFIGURE IAM ROLES AND POLICIES

IAM (Identity and Access Management) ensures our services can securely communicate and operate.Proper permissions to be given to Lambda and Glue

Create an IAM Role for Lambda:

*Go to the IAM Console → Roles → Create Role

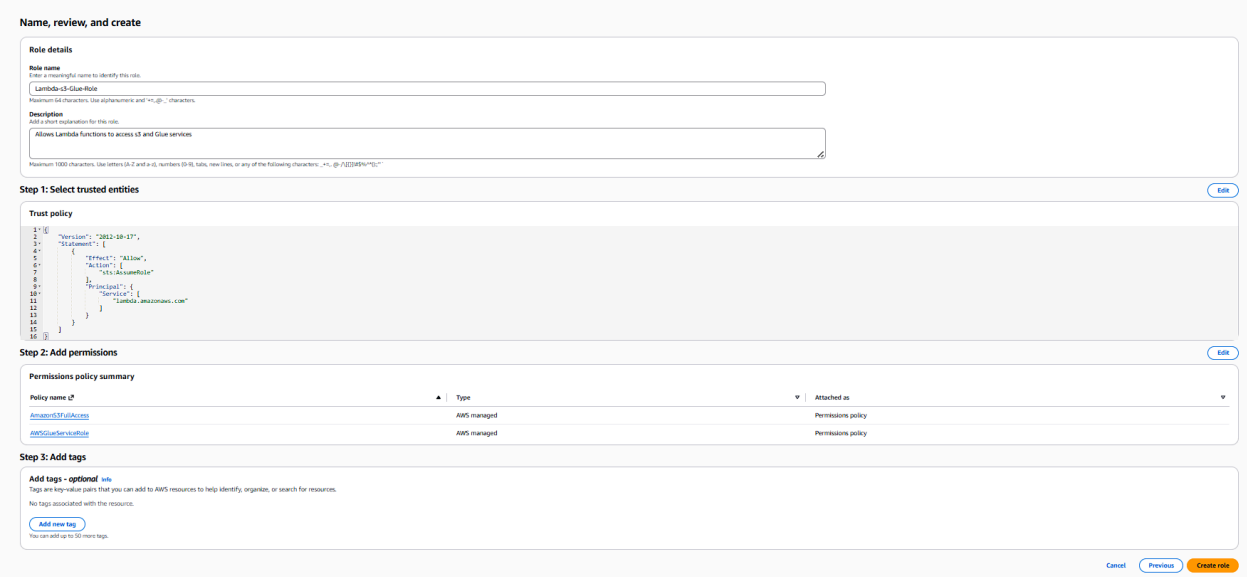
*Select AWS Service as the trusted entity and choose Lambda



*Attach following policies

1.AmazonS3FullAccess (to read/write s3 buckets)

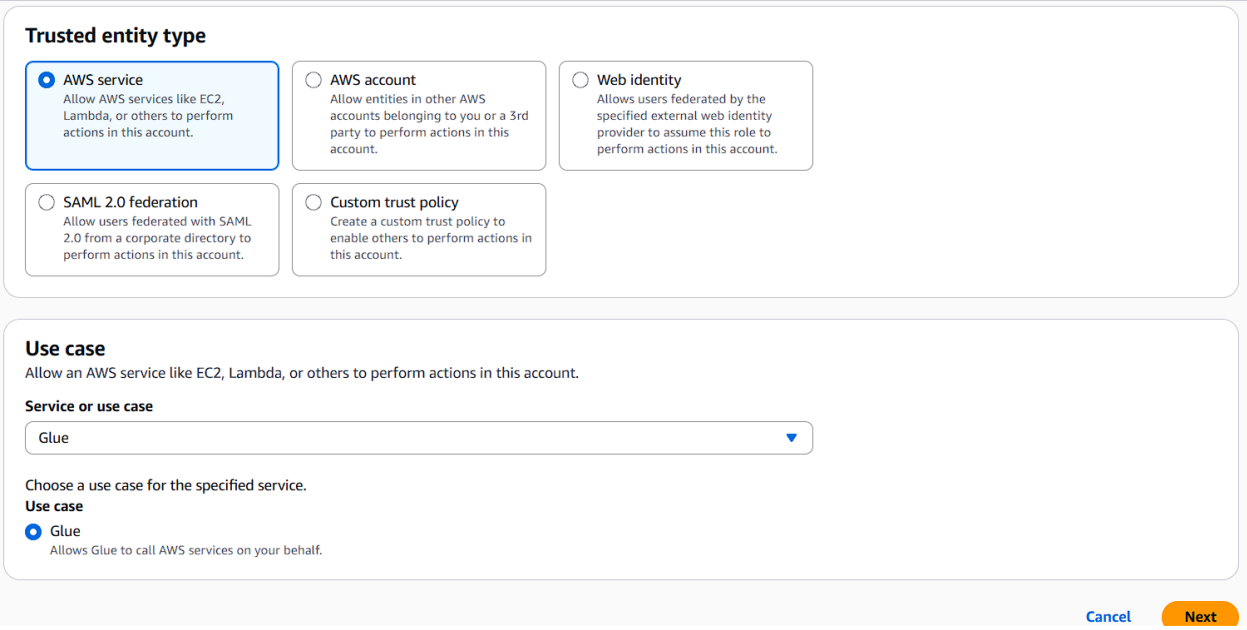
2.AWSGlueServiceRole(for ETL operations)



Create an IAM Role for Glue:

Go to roles → Create Role

Select AWS Service and choose Glue



Attach these policies

***AWSGlueServiceRole**

***AmazonS3FullAccess**

Name the role Glue-Service-Role and click Create Role

Note:Attach only permissions necessary for the pipeline to reduce security risks

#3.SET UP AMAZON Quicksight

Amazon Quicksight allows us to visualize our processed data.

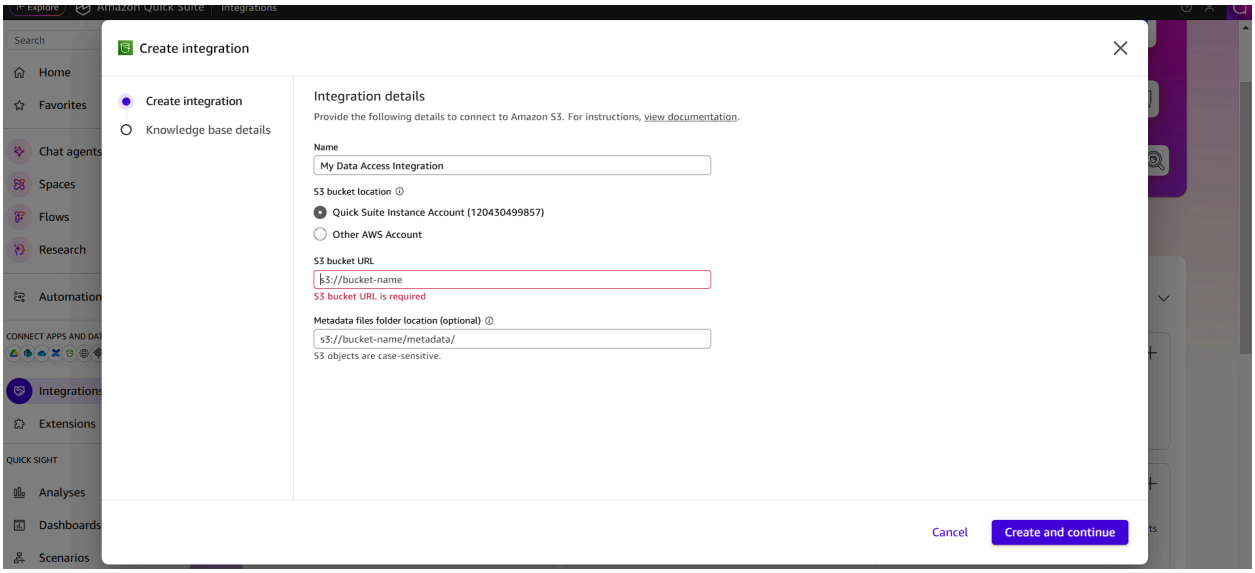
Create an Quicksight account.Choose same region as your s3 bucket's.Add account name.

Configure Quicksight Settings

*In the Allow Access, click on select s3 Buckets.

*Select previously created three S3 buckets. Click on finish

*Now Quicksight is ready to access and visualize data from s3



DATA INGESTION AND PREPROCESSING

STEPS:

1.Create an Lambda function

2.Write the lambda function code

3.Set up s3 event trigger for lambda

4.Test data ingestion and preprocessing

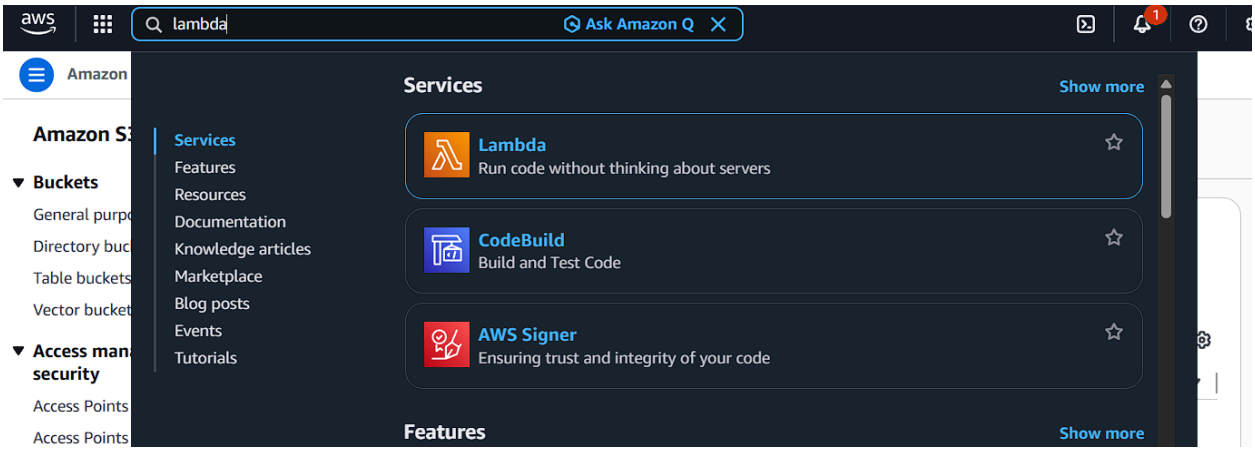
#1 Create an Lambda Function

Now that we have the foundational setup, let's automate the ingestion and preprocessing of CSV files. In this step, we will deploy a Lambda function that triggers automatically when the file is uploaded to the raw data s3 bucket.

The function will clean and transform the data before saving it back to processed data bucket.

AWS lambda allows us to execute the code without an server.

*Navigate to AWS console and create an lambda function



*Choose author from scratch and add these

1.Function Name: CSVPreprocessorFunction

2.Runtime: Select Python 3.13

3.Role: Choose an existing role and choose the role (Lambda-s3-Glue-Role) created earlier.

#2.WRITE LAMBDA FUNCTION CODE

Let’s write a basic preprocessing Lambda function to clean and filter CSV files.

- Scroll down to the **Code** section and replace the default code with the following:

```python
import boto3
import csv
import io
s=boto3.client('s3')
def lambda_handler(event,context):
    bucket_name=event["Records"][0]["s3"]["bucket"]["name"]
    file_key=event["Records"][0]["s3"]["object"]["key"]
    try:
        response=s3.get_object(Bucket=bucket_name,Key=file_key)
        csv_content=response["Body"].read().decode("utf-8")
        processed_rows=[]
        reader = csv.reader(io.StringIO(csv_content))
        header = next(reader)  # Extract the header row
        for row in reader:
            if all(row):
                processed_rows.append(row)
        output_csv=io.StringIO()
        writer=csv.writer(output_csv)
        writer.writerow(header)
        writer.writerows(processed_rows)
        processed_file_key=file.key.replace("raw/","processed/")
        s3.put_object(
            Bucket="<csv-processed-data-003>",
            Key=processed_file_key,
            Body=output_csv.getvalue()
        )
        print(f"Processed file uploaded to: {processed_file_key}")

    except Exception as e:
        print(f"Error processing file: {str(e)}")
        raise
```

This Lambda Function is designed to automatically preprocess csv files uploaded to the s3 bucket(csv-raw-data)

1.Triggered by s3 upload

Whenever an csv file is uploaded to the s3 bucket (csv-raw-data) the Lambda function gets triggered upon the event (event-driven).

2.Reads the CSV file

It fetches the file from the s3 bucket and reads it.

3.Cleans the data

The function read the data and removes missing values and keeps only complete rows

4.Creates a new CSV

Writes the cleaned data into a new CSV file in memory

5.Uploads the processed file

The function uploads the processed file into an new s3 bucket.(csv-processed-data-003)

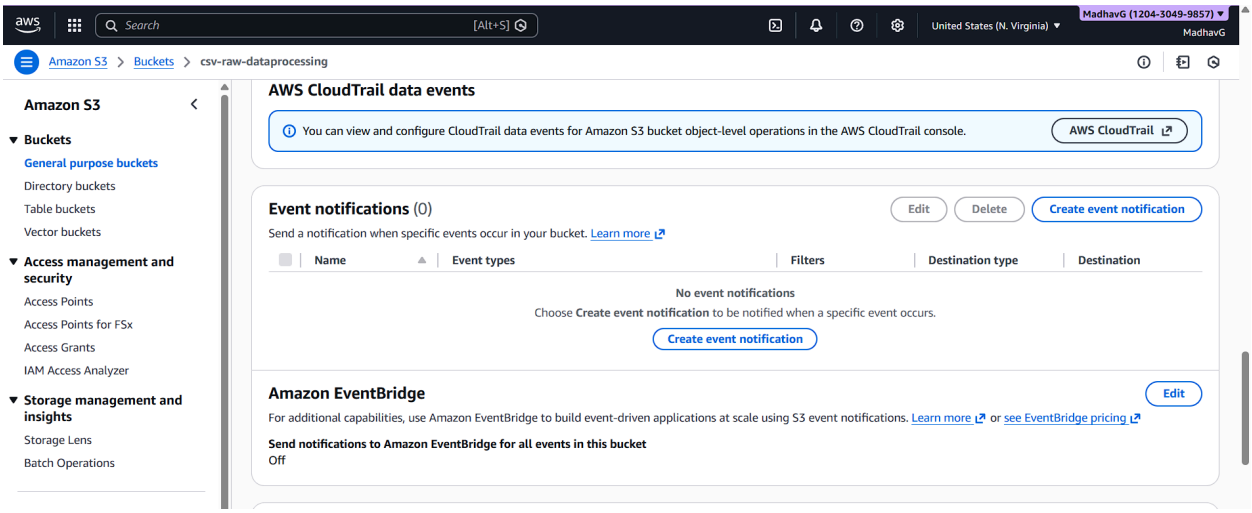
6.Click deploy to save code

#3.SETUP S3 EVENT TRIGGER FOR LAMBDA

Now that our Lambda function is setup, we need to configure the raw data S3 bucket to automatically trigger the Lambda function whenever a new file is uploaded in the bucket.

*Go to s3 console select your raw data bucket

*Navigate to properties and scroll to Event Notifications



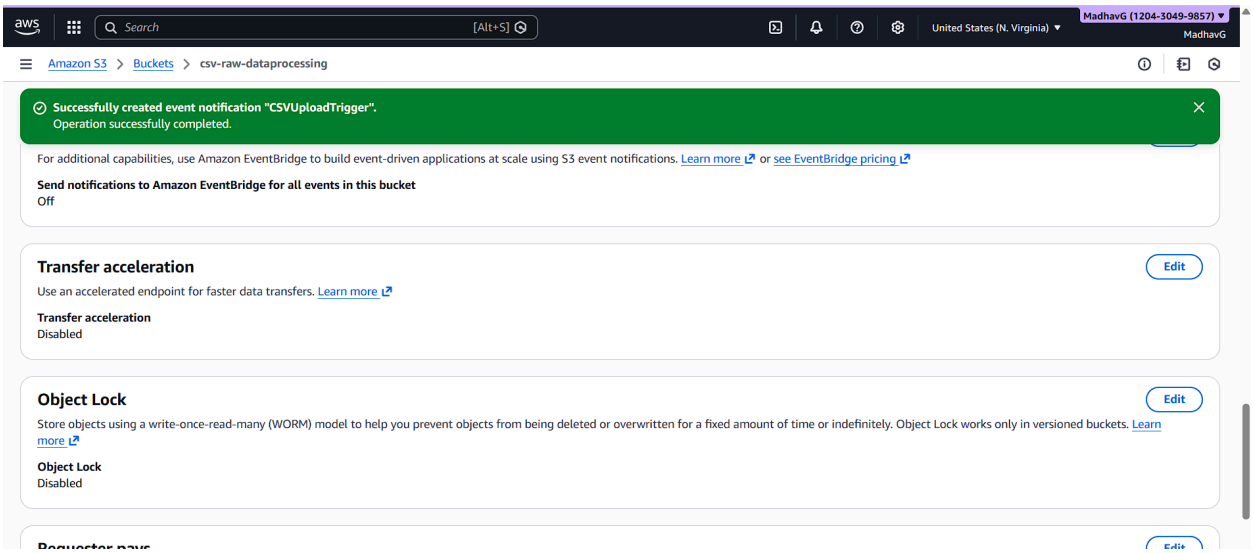
*Click create Event notification and configure the following

Name:CSVUploadTrigger

Events:Select PUT

Prefix:raw/

Destination:Choose Lambda function and select CSVPreprocessorFunction

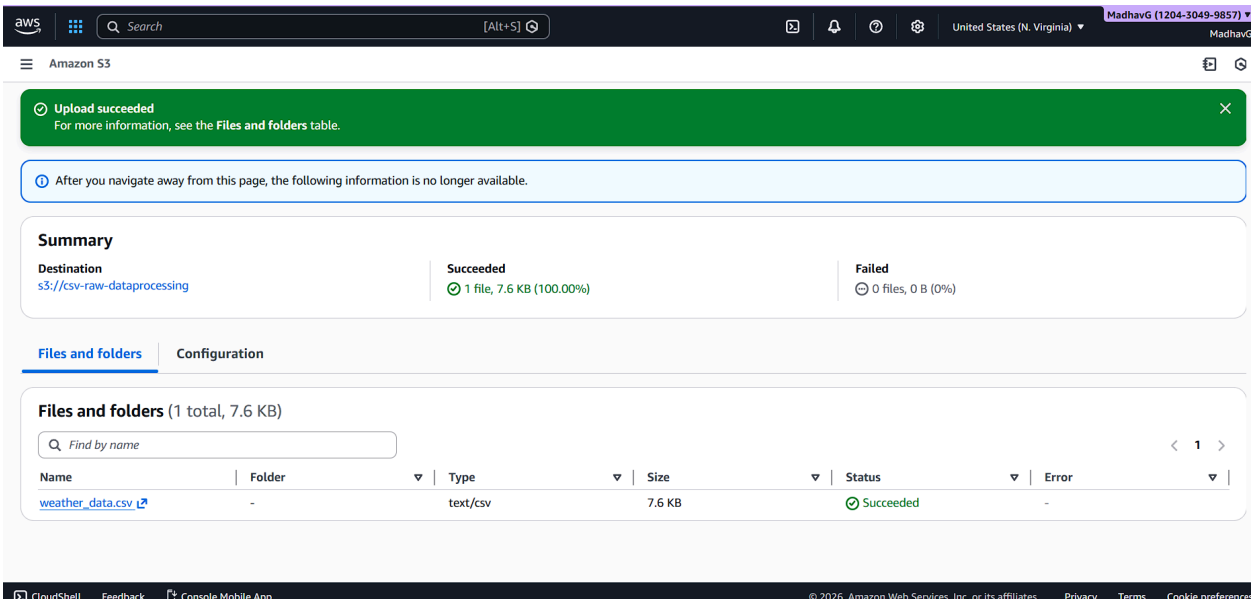


#4.TEST DATA INGESTION AND PREPROCESSING

Lets test if the Lambda function is getting triggered when we upload a file in the `raw` data bucket and if it is getting preprocessed and stored in the `processed` bucket. To do this—

Upload a Sample CSV File:

- Go to the **S3 Console** and navigate to your `csv-raw-data` bucket.
- Create a folder named `raw` and upload a sample CSV file (`weather-data.csv`) to this folder.
- After uploading the file to the folder try navigating to the folder again to check if the file is in the same hierarchy



DATA TRANSFORMATION WITH AWS GLUE

1.Setup AWS glue data catalog

2.Create a crawler to discover data schema

3.Create and configure AWs Glue Job using visual ETL

4.Verify and prepare transformed data for visualization

#1 .SETUP AWS GLUE DATA CATALOG

AWS Glue is a fully managed ETL service (Extract, Transform, Load) that helps to transform data and move it between different storage layers. Glue Catalog is a centralized metadata repository that stores data about datasets.(makes it easy for searching)

Lambda is ideal for lightweight, real-time preprocessing of small files, while Glue ETL is better suited for large-scale, complex transformations on big data with built-in schema discovery and job orchestration

Navigate to AWS Glue Console

2.Click **Data catalogs → Databases → Add database**

Set database name as: **csv_data_pipeline_catalog**

#2.CREATE CRAWLER TO DISCOVER DATA SCHEMA

A crawler automatically scans the data and creates meta data tables

*Go to crawler and click **Add crawler**

Fill the details:

Name:ProcessedCSVDataCrawler

Data Source:choose s3 and provide location of csv-processed-data bucket

Create and Configure an AWS Glue Job Using Visual ETL

Now lets proceed with creating the AWS Glue Job ETL pipeline using Visual ETL. To do this—

Access AWS Glue Studio:

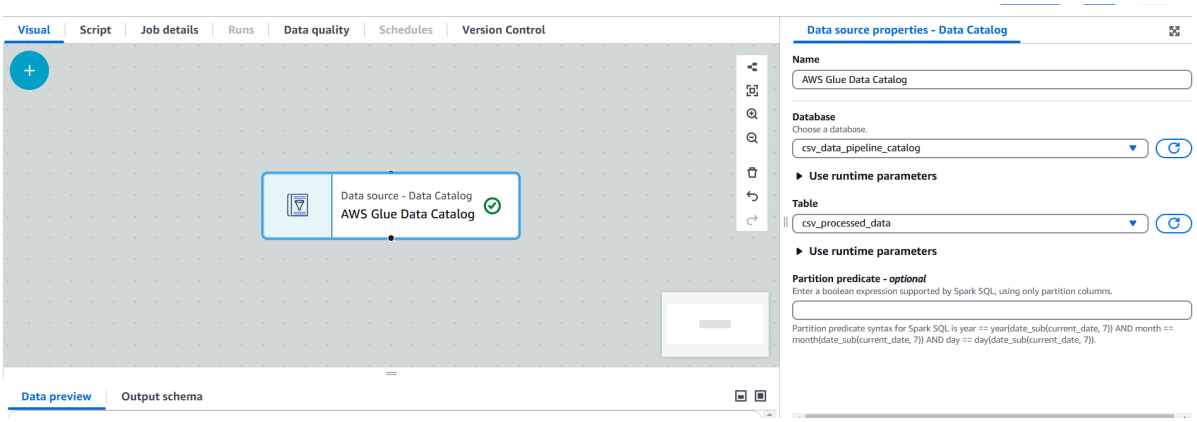
- Navigate to the AWS Glue Console, then click **AWS Glue Studio** from the left menu.

Create a New Job:

- Click **Jobs** → **Create ETLJob**.
- Select **Visual ETL**.

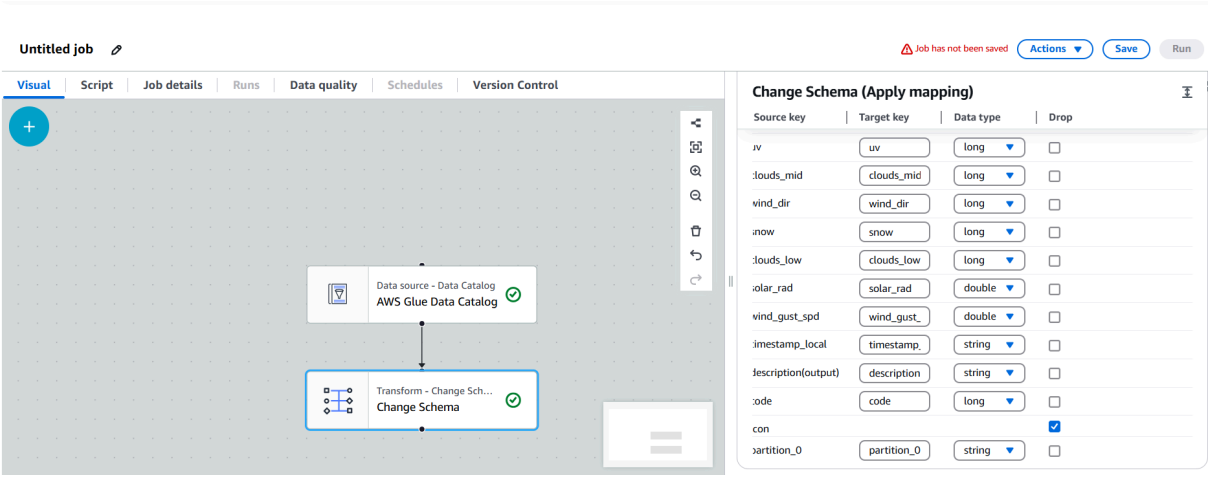
Define the Source:

- In the visual canvas, click on the add button and go to **Data Source**.
- Choose the AWS Glue Data Catalog. Under the database, choose the created `csv_data_pipeline_catalog` database.
- Under the table, choose the `csv_processed_data` table created by the Crawler job.



Add Transformations:

- Click the **+** button after the source block and choose **Change Schema** for basic transformations.
- Here I am going to drop the icon column so I will choose the checkbox for icon column.



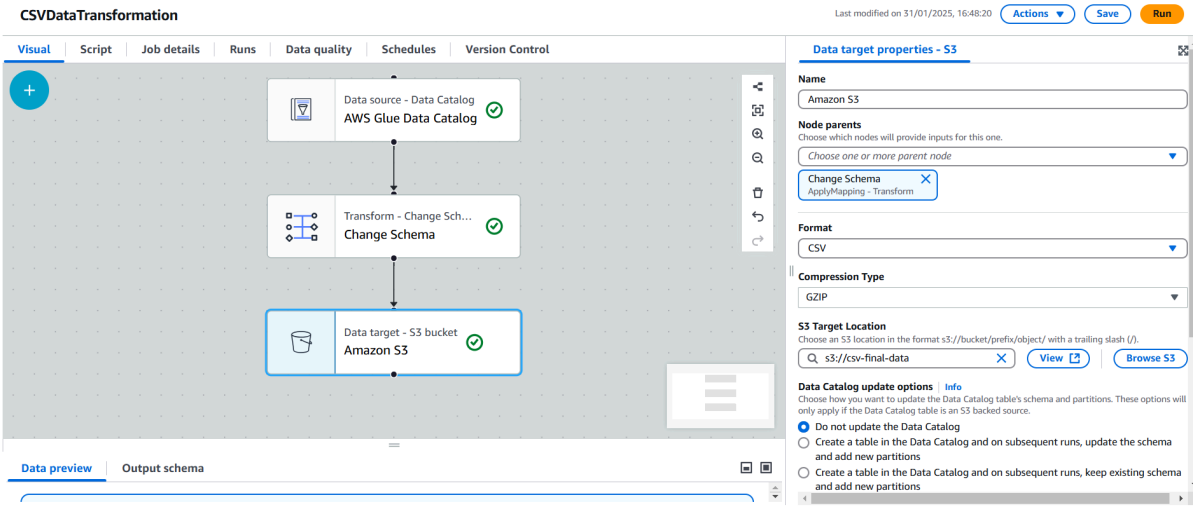
Define the Target (Destination):

- Click the **+** button after the transformation block and select **Data Target**.
- Choose **Amazon S3** as the target and specify your `csv-final-data` bucket location where the transformed data will be stored.

- Under **IAM Role**, select the `Glue-Service-Role` we created earlier.
- Set the **Name** to `CSVTransformationJob`.
- Click on the **Job details** tab at the top of the page.

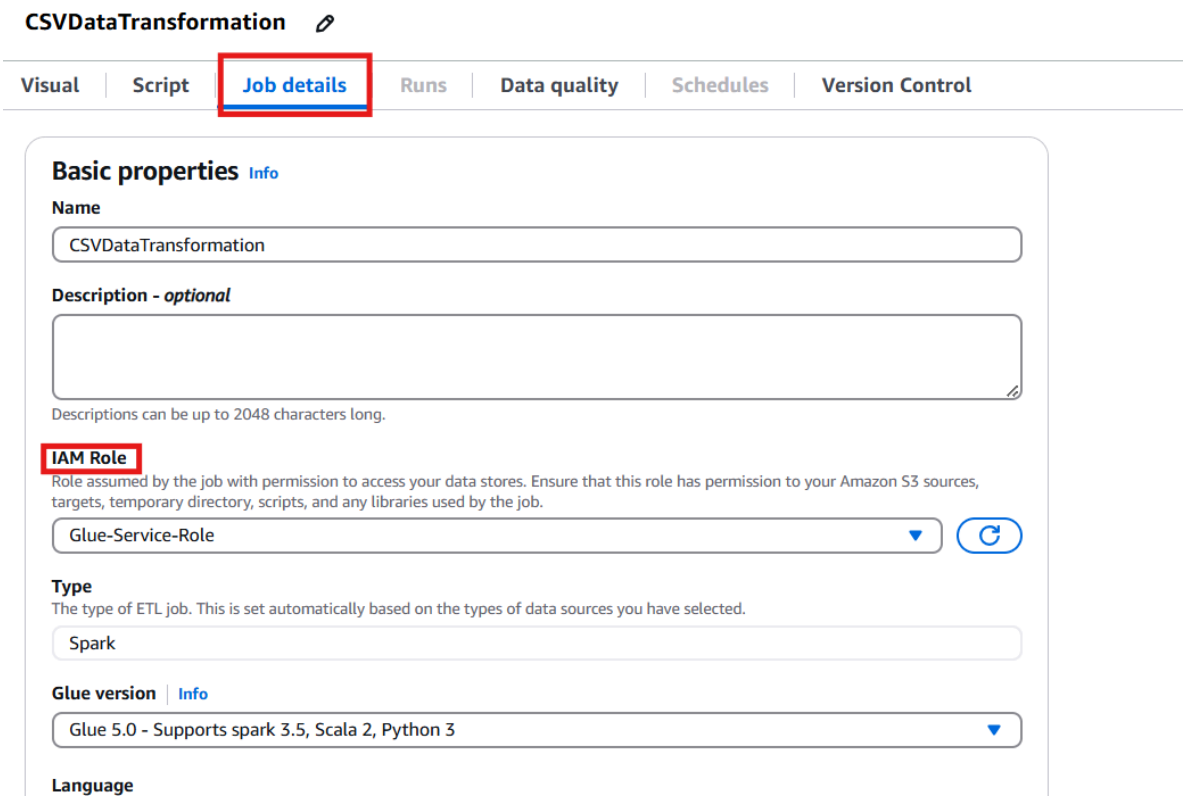
Configure Job Settings:

- **Compression:** Choose GZIP as the compression type.



Configure Job Properties:

- Click the **Job Details** tab on the right panel and provide the following details:
- **Name:** `CSVDataTransformation`
- **IAM Role:** Select an existing Glue role with access to S3 or create a new one.
- Leave other advanced settings as default.



Save and Run the Job:

- Click **Save** and then **Run**.
- Monitor the job status in the **Runs** tab. It may take a few minutes to complete.

#4 Verify and Prepare Transformed Data for Visualization

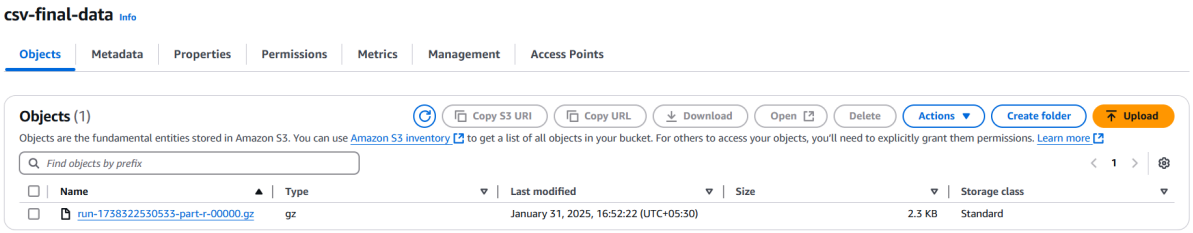
After the Glue pipeline execution, the transformed data will be stored as a **ZIP** file in the target S3 bucket. Since the file inside may not have an extension, follow these steps to convert it to a proper `.csv` format.

Navigate to the S3 Bucket:

- Open the **AWS S3 Console**.
- Locate and select the bucket where the transformed data is stored (e.g., `csv-final-data`).

Download the ZIP File

- Click on the folder where the output is stored.



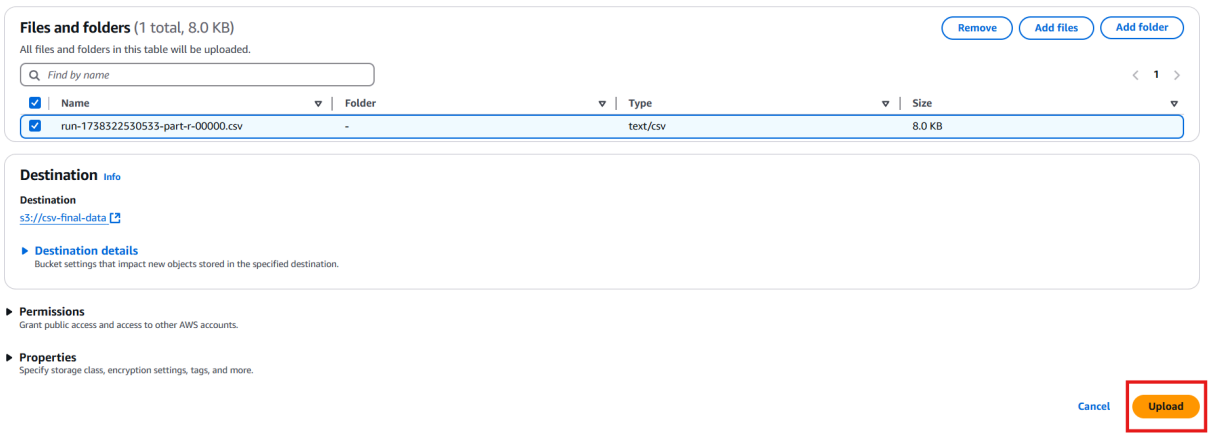
- Select the compressed **ZIP** file and click **Download**.

Extract and Rename the File

- On your computer, extract the contents of the ZIP file.
- Locate the extracted file (it may not have an extension).
- **Rename the file** to include a `.csv` extension at the end.

Re-upload the CSV to S3

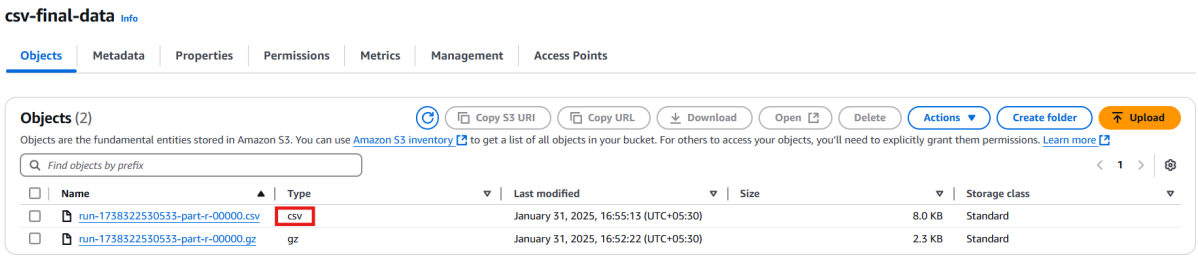
- Go back to the **AWS S3 Console**.
- Click on **Upload** and select the renamed `.csv` file.



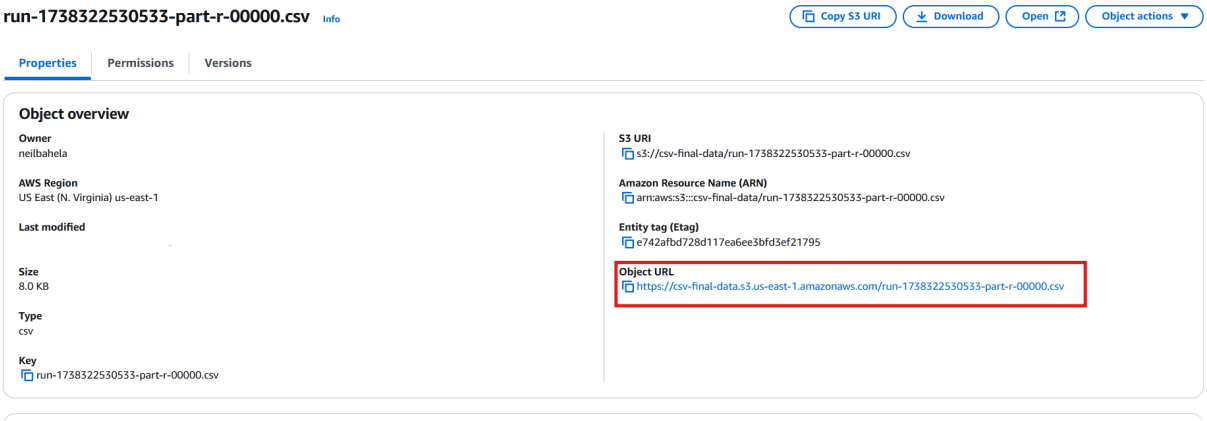
- Click **Upload**.

Verify the Upload

- Confirm that the file now appears in the `csv-final-data` bucket with a `.csv` extension



- It is now ready for visualization and analytics tasks. You can click on the csv file and copy the Object URL to use it for the further steps.



Final Check

- Data Catalog created and configured.
- Crawler set up and successfully discovered schema.
- Glue job created and executed with data transformations.
- Transformed data verified in S3.

Steps to be Performed

1. Connect to the Data Source

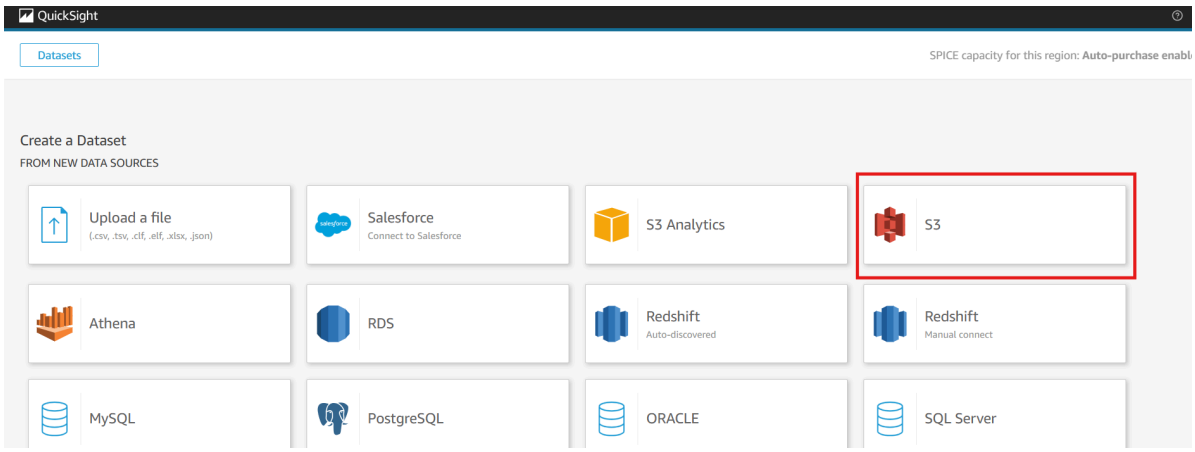
2. Build a QuickSight Dashboard
3. Share and Publish the Dashboard

#1 Connect to the Data Source

We've successfully automated the ingestion and transformation of CSV data using S3, Lambda, and AWS Glue. The final step is to visualize this processed data using Amazon QuickSight, which will help turn processed data into actionable insights.

We need to connect QuickSight to the processed data stored in S3.

- In the QuickSight console, go to the **Datasets** section and click **New Dataset**.
- Select **S3** as the data source.



- Provide the following details:
  - **Data Source Name:** `ProcessedCSV`
  - **Manifest File:** Create a manifest file in your local machine with the following content and upload it from your local machine. **Add the Object URL you copied earlier and replace it in the placeholder.**

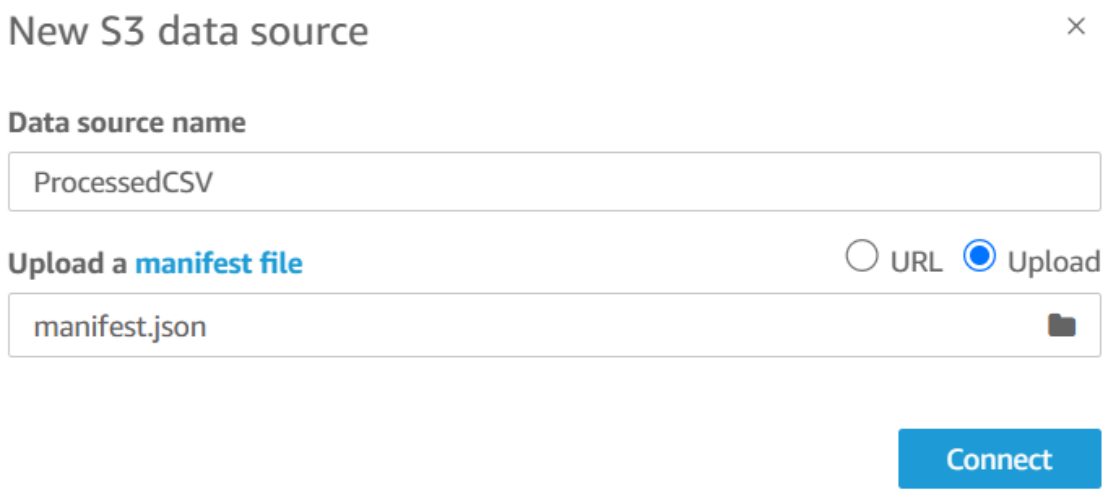
json

```
{
  "fileLocations": [
```

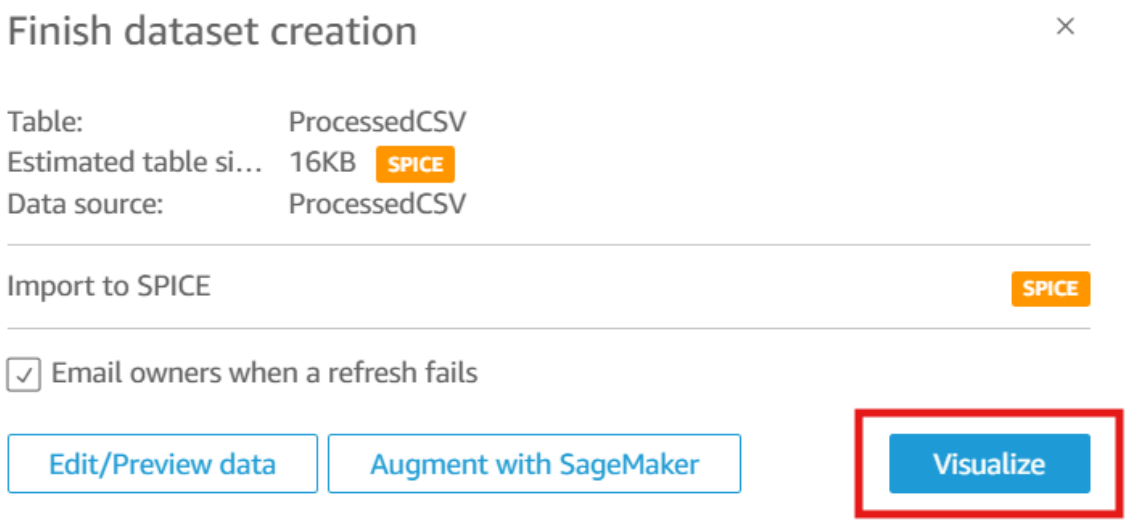
```
    {
      "URIs": [
        "ENTER YOUR OBJECT URL HERE"
      ]
    }
  ],
  "globalUploadSettings": {
    "textqualifier": "\""
  }
}
```

```
{
  "fileLocations": [
    {
      "URIs": [
        "https://csv-final-data.s3.us-east-1.amazonaws.com/run-1738302894596-part-r-00000.csv"
      ]
    }
  ],
  "globalUploadSettings": {
    "textqualifier": "\""
  }
}
```

- Upload the manifest file and click **Connect**.



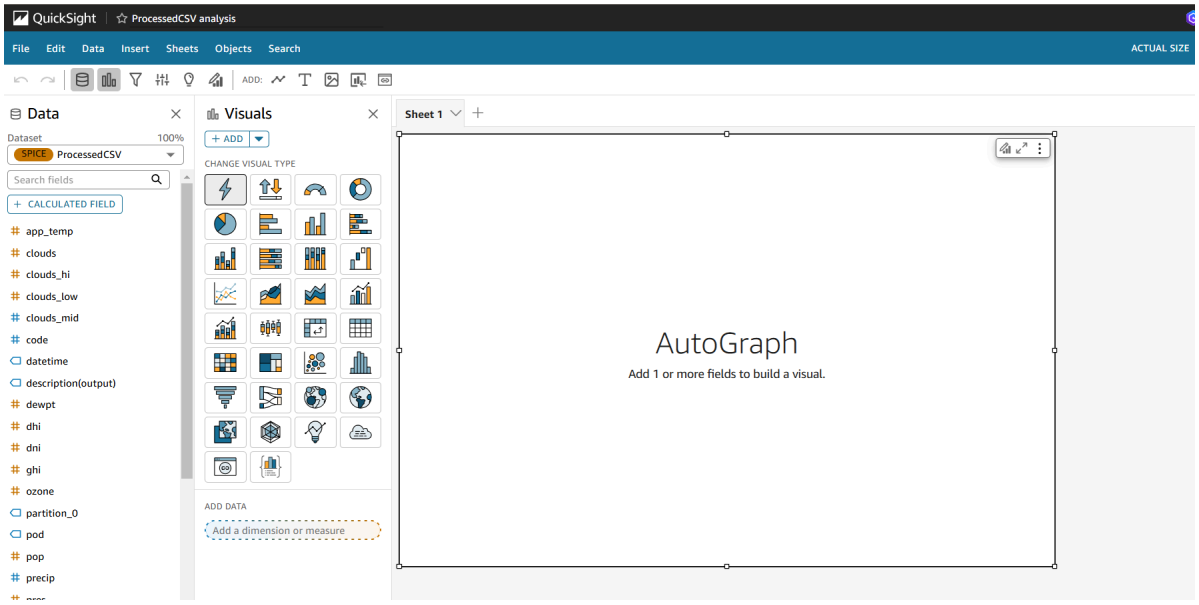
- Click **Visualize** to load the data into QuickSight.



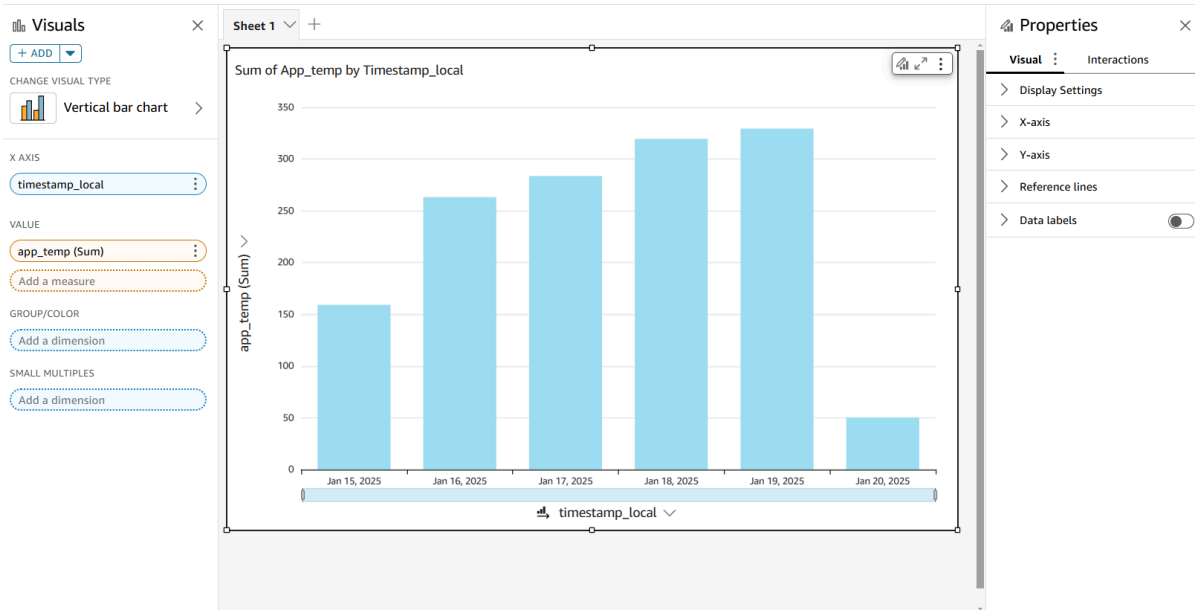
#2 Build a QuickSight Dashboard

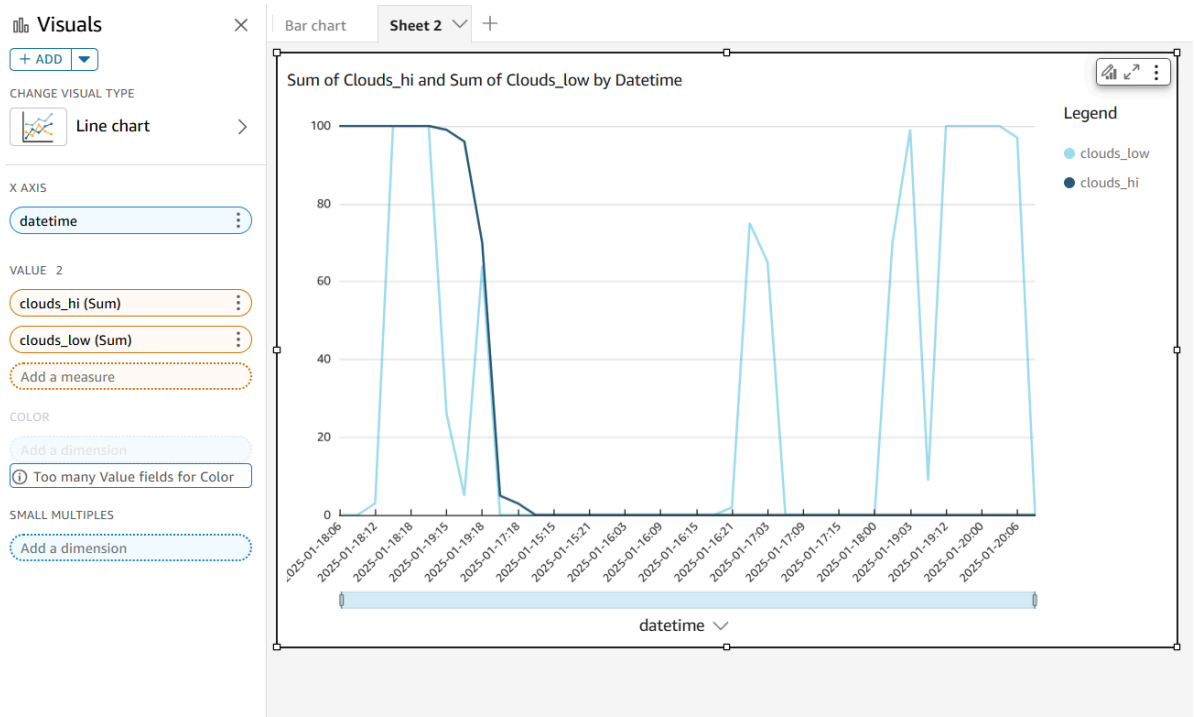
Let’s create a dashboard to visualize the insights from the transformed CSV data.

- Click **Add** to create a new analysis.
- Select the `ProcessedCSVData` dataset.



- Build the following visualizations:
  - **Bar Chart:** Compare values from different columns.
  - **Line Chart:** Analyze trends over time.

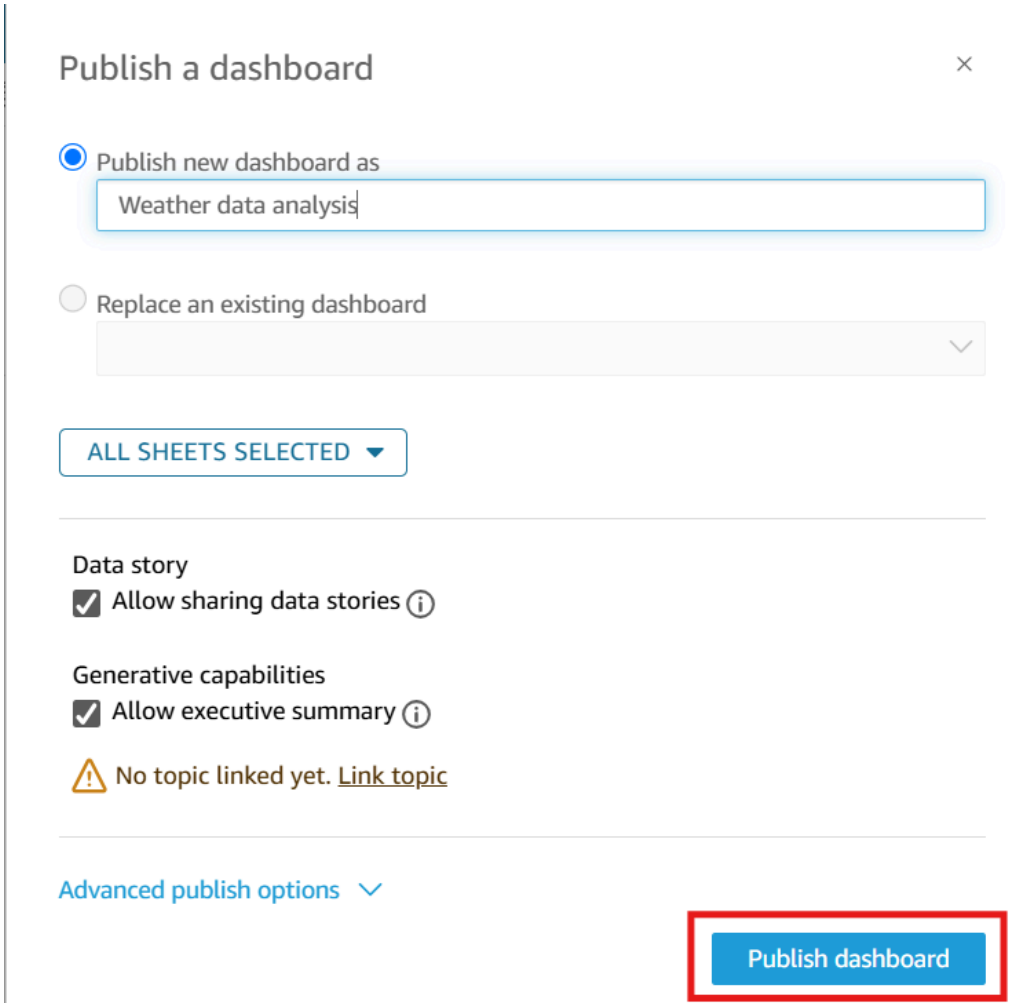




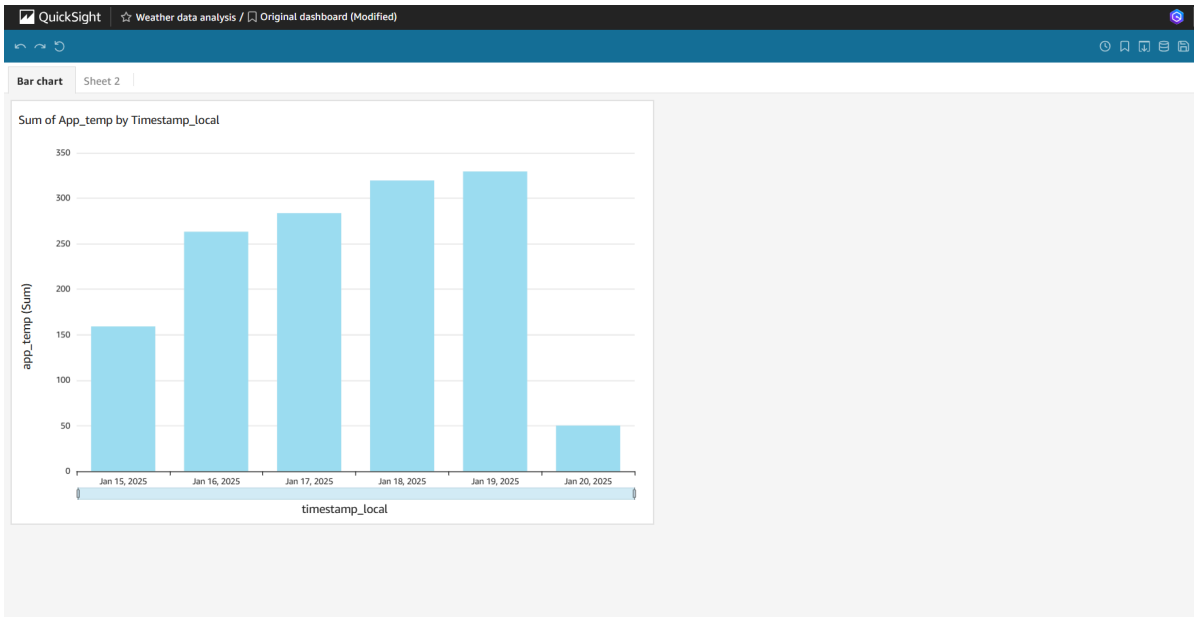
- You can try out different charts by using the columns available in the dataset.

#3 Share and Publish the Dashboard

- Select **Publish Dashboard** to make it accessible to other users.



-



- QuickSight account set up and configured.
- Data source connected and loaded into QuickSight.
- Dashboard created with meaningful visualizations.
- Pipeline tested with new data, ensuring dynamic updates.