AWS ETL Pipeline Project

# Overview

This project demonstrates an ETL pipeline using AWS Glue for transformations, Amazon Redshift as the SQL data warehouse, and Managed Apache Airflow for orchestration.

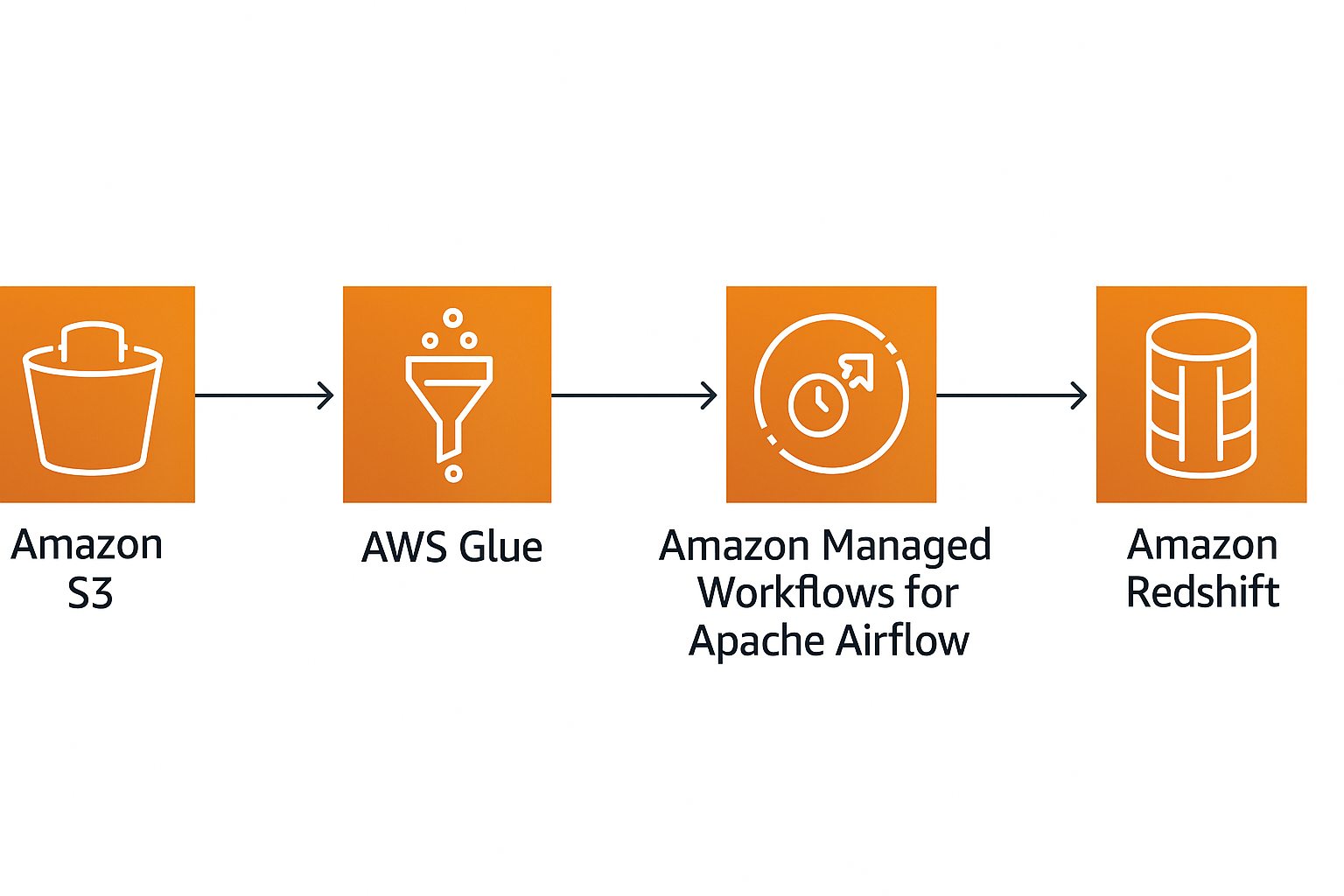
# Pipeline Steps

1. Raw CSV files are ingested into S3 (`input` folder).

2. Glue job reads the files, transforms them, and writes output to Amazon Redshift.

3. The pipeline is orchestrated using Apache Airflow DAG triggered on a schedule.

# Architecture Diagram



# AWS Services Used

- S3: Storage for raw input files

- Glue: Serverless ETL transformations

- Amazon Redshift: SQL-based data warehouse

- Managed Airflow: Orchestration and scheduling

## 1. Scaling Considerations

### 1.1 Horizontal Scaling

#### AWS Glue

- Auto-Scaling: Glue automatically scales Spark jobs based on the volume of data.

- Worker Configuration:  
 - NumberOfWorkers – Defines how many workers to use.  
 - WorkerType – Options include G.1X, G.2X for varying memory and compute needs.

- Optimization Features:  
 - Enable Job Bookmarks to avoid reprocessing data.  
 - Use Job Chunking for better parallelization on large datasets.

#### Amazon Redshift

- Use RA3 Nodes with managed storage for scalable performance.

- Enable Concurrency Scaling to handle peak-time query loads.

- Implement WLM (Workload Management) to allocate compute resources effectively across workloads.

#### Apache Airflow (MWAA)

- MWAA automatically scales workers based on DAG workload.

- Monitor resource usage to ensure efficient task distribution.

## 2. Cost Optimization Strategies

### 2.1 AWS Glue

- Use Job Bookmarks to process only incremental changes.

- Right-size workers based on job profiling and test runs.

- Schedule jobs during off-peak hours to reduce compute costs.

### 2.2 Amazon Redshift

- Pause/Resume Clusters in dev environments to save on compute costs.

- Use Compression Encodings to reduce storage and improve query performance.

- Leverage Materialized Views to cache frequent query results.

### 2.3 Apache Airflow (MWAA)

- Select the right Environment Class (small, medium, large) based on load.

- Monitor and fine-tune Max Workers to control cost and performance.

## 3. Future Optimizations

### 3.1 Incremental Processing

- Implement Change Data Capture (CDC) in source systems for real-time updates.

- Extend use of Glue Job Bookmarks to minimize redundant processing.

### 3.2 Data Quality Framework

- Integrate Great Expectations for comprehensive data validation checks.

- Add Automated Anomaly Detection to catch data issues early.

### 3.3 Performance Enhancements

- Use Redshift Spectrum to query data directly from S3 without loading into Redshift.

- Apply Materialized Views for frequently used aggregations.

- Use S3 Partitioning Strategies to speed up Glue and Spectrum queries.

### 3.4 Monitoring Enhancements

- Set up CloudWatch Dashboards to monitor job and pipeline health.

- Configure SNS Alerts for failures or anomalies.

- Define Custom Metrics to monitor data quality KPIs.

## 4. Deployment Instructions

### 4.1 Prerequisites

- Active AWS Account with required service permissions.

- Amazon Redshift Cluster provisioned and accessible.

- MWAA (Managed Workflows for Apache Airflow) environment configured.

- S3 Buckets prepared for scripts, intermediate data, and logs.