# Migration Strategy for Moving 70 TB Data from AWS Glue to Apache Iceberg Tables

## Project Overview

This migration strategy is designed to transition 70 TB of data from AWS Glue to Apache Iceberg tables, which will enhance data management capabilities, improve query performance, and support advanced features like versioning and time-travel queries.

## 1. Assessment Phase

1. Data Profiling and Inventory

* - Perform a comprehensive data assessment on the existing Glue tables, including schema structure, data formats, partitioning, and file sizes.  
  - List Glue tables with their metadata, row counts, and dependencies.

2. Identify Data Transformation Needs

* - Assess the necessary transformations, such as data format conversions, partition adjustments, and schema alignment for Iceberg compatibility.

3. Infrastructure and Resource Evaluation

* - Estimate storage and compute resources to ensure Iceberg can handle the data volume efficiently.  
  - Confirm that the necessary resources are available, especially if integrating with Spark or any other computational engine during the migration.

## 2. Migration Design

1. Define Partition Strategy for Iceberg Tables

* - Use partitioning for Iceberg based on access patterns (e.g., date or time-based) to optimize read and write performance.  
  - Document the partition strategy for each table, including field names and partition types.

2. Data Format and Schema Alignment

* - Convert data formats from Glue-compatible formats to Iceberg-compatible formats, if necessary.  
  - Standardize schema definitions to ensure consistency and avoid schema drift issues in Iceberg.

3. Handling Incremental Data

* - Design an approach for managing new data written to Glue during migration, possibly using Change Data Capture (CDC) or freeze periods for data writes.

## 3. Migration Phases

1. Initial Full Load Migration

* - Extract full datasets from Glue tables using Spark jobs.  
  - Write data to Iceberg tables in defined partitions and format.  
  - Validate with row counts and data quality checks.

2. Incremental Load and Synchronization

* - Set up a CDC process or periodic synchronization to migrate only the data that changed since the initial load.

3. Final Cut-over and Data Validation

* - Perform a final incremental load just before cut-over.  
  - Validate data completeness and consistency.  
  - Update downstream applications and queries to use Iceberg tables instead of Glue.

## 4. Testing and Validation

1. Validation Plan

* - Develop a validation strategy to compare Glue and Iceberg tables, including data integrity checks, schema comparisons, and record count verification.

2. Performance Testing

* - Conduct performance benchmarks to evaluate the query efficiency of Iceberg tables.  
  - Test partition pruning, predicate pushdown, and other Iceberg-specific optimizations.

3. Rollback Plan

* - Design a rollback procedure in case issues arise, allowing reversion to Glue tables without data loss.

## 5. Deployment and Monitoring

1. Cut-over

* - Schedule a final migration window for the transition.  
  - Ensure all downstream systems are switched to query the Iceberg tables.

2. Post-Migration Monitoring

* - Set up monitoring for Iceberg tables to track performance, data consistency, and resource utilization.  
  - Implement logging for error tracking and monitoring read/write patterns.

## 6. Documentation and Knowledge Transfer

1. Document Changes

* - Update data catalogs, data dictionaries, and documentation for downstream users.

2. Training Sessions

* - Conduct training for the team on Iceberg management, query optimizations, and troubleshooting Iceberg tables.

## Cost Estimation for 70 TB Migration Using AWS Glue

The estimated cost for migrating 70 TB of data with AWS Glue includes data processing, storage, and additional resource costs.

### 1. AWS Glue ETL Job Costs

- Estimated cost for Glue ETL Jobs using 10 DPUs: $7,040 for full migration (0.44 per DPU-hour x 10 DPUs x 160 hours).

### 2. Data Storage Costs

- Estimated S3 storage: $1,610/month (70 TB x $23/TB/month).

### 3. Additional Data Transfer Costs

- Data Transfer to S3 (same AWS region) is generally free. Inter-region or outbound transfers, if needed, may add additional costs.

\*\*Total Estimated Cost\*\*: Approximately $8,650, excluding potential inter-region transfers.

## Timeline and Milestones

- Assessment Phase: Week 1-2  
- Migration Design and Development: Week 3-5  
- Initial Full Load Migration: Week 6-7  
- Incremental Load Migration: Week 8-10  
- Final Cut-over and Post-Migration Testing: Week 11-12

## Time Estimation for Incremental Migration

To process 70 TB of data in 5-day increments over a 5-year period (1,825 days), the following time estimation provides a breakdown based on batch processing time.

### 1. Total Number of Batches

- Given a 5-year dataset and processing 5-day batches, the total number of batches required is:  
 Total Batches = 1,825 days / 5 days per batch = 365 batches.

### 2. Time Per Batch

- Each 5-day batch is estimated to take approximately 0.45 hours (27 minutes) to process using a 10-DPU Glue job.  
- Processing all 365 batches sequentially results in:  
 Total Processing Time = 365 batches \* 0.45 hours = 164.25 hours.

### 3. Total Days Required

- Total Days Required (Continuous Processing): 164.25 hours / 24 hours per day ≈ 7 days.  
- With potential breaks or resource constraints, the timeline could extend up to 10-14 days.

### Summary:

|  |  |
| --- | --- |
| Parameter | Estimate |
| Total Number of Batches | 365 batches |
| Processing Time Per Batch | 0.45 hours (27 minutes) |
| Total Processing Hours | 164.25 hours |
| Total Days (Continuous Processing) | 7 days |
| Total Days (With Breaks/Buffer) | 10-14 days |

## Technical Design for Glue with Checkpointing Implementation

To ensure reliability during the migration process and allow for re-ingestion of specific date ranges in the event of job failures, a checkpointing mechanism is implemented in the AWS Glue job. This section outlines the technical design of the Glue job with a checkpointing file approach, enabling tracking of successfully processed batches and supporting error recovery.

### 1. Checkpointing Strategy

The checkpointing mechanism leverages a checkpoint file stored in Amazon S3. This file records the date ranges of each successfully processed batch. If a Glue job fails for a specific 5-day batch, the checkpoint file enables re-processing only for the failed batch without duplicating or skipping other batches.

### 2. Implementation Steps

* The following steps detail the checkpointing implementation:

- \*\*Initialize Checkpoint File\*\*: When the Glue job starts, it checks if the checkpoint file exists in S3. If not, a new file is created to track the migration batches.  
- \*\*Batch Processing\*\*: For each 5-day batch, the job reads data, processes transformations, and writes to the Iceberg table.  
- \*\*Update Checkpoint File\*\*: After a batch is successfully processed, the job updates the checkpoint file with the processed date range.  
- \*\*Error Handling\*\*: If a batch processing job fails, the checkpoint file is left unchanged, marking the batch as incomplete.  
- \*\*Re-ingestion for Failed Batches\*\*: Upon restart, the job reads the checkpoint file to identify any unprocessed date ranges and resumes migration from the last successfully processed batch.

### 3. Handling Failures and Re-Ingestion Process

* The Glue job includes error handling and re-ingestion mechanisms for reliable processing:

- \*\*Retry Mechanism\*\*: For transient errors, a retry mechanism is applied with configurable retry counts and intervals.  
- \*\*Manual Restart\*\*: If a failure is detected for a specific date range, the Glue job can be restarted to process only the incomplete batches, based on the last checkpointed date range.  
- \*\*Automated Notifications\*\*: Set up notifications for job failures to trigger alerts, allowing prompt responses for failed batch recovery.

### 4. Benefits of Checkpointing Implementation

* This checkpointing approach provides several key benefits:

- \*\*Improved Reliability\*\*: Avoids re-processing of successfully completed batches, ensuring only failed batches are re-ingested.  
- \*\*Enhanced Efficiency\*\*: Saves processing time and reduces resource costs by allowing targeted recovery.  
- \*\*Simplified Recovery Process\*\*: Facilitates straightforward recovery and minimal manual intervention in case of job failures.