Migrating Data from GCP to Azure SQL Managed Instance (Incremental Load)

Priya Akunuri

GCP to Managed Instance (MI) Incremental Data Pipeline Using Azure Data Factory

Objective

This document outlines the steps for creating an incremental data pipeline to load the latest data from Google Cloud Big Query into an Azure Managed Instance (MI) via Blob Storage. The pipeline automates the process, ensuring that the latest file from Big Query is moved daily and loaded into MI while allowing schema drift during loading. The pipeline also includes optional file deletion from the staging area to save space.

Prerequisites

- Google Cloud Big Query account access.
- Azure Blob Storage set up as the staging area.
- Azure SQL Managed Instance (MI) for loading data.
- Azure Data Factory (ADF) with appropriate permissions.

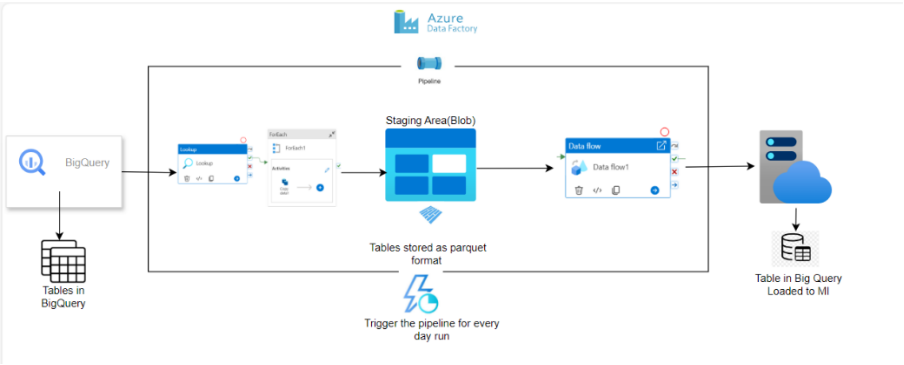
Source and Destination Details

Source : Big Query
Project ID: **partner-portal-426309**

Destination : Azure Managed Instance
Server name: smineuprdsqls01.09ed840669e8.database.windows.net
Database name: **DW**
Table name : **BigQ**

ETL Details :
Data Factory Name : **Adfbigqueryprod**
Storage account Name : **stoweuprdbigq01**

Architecture Diagram

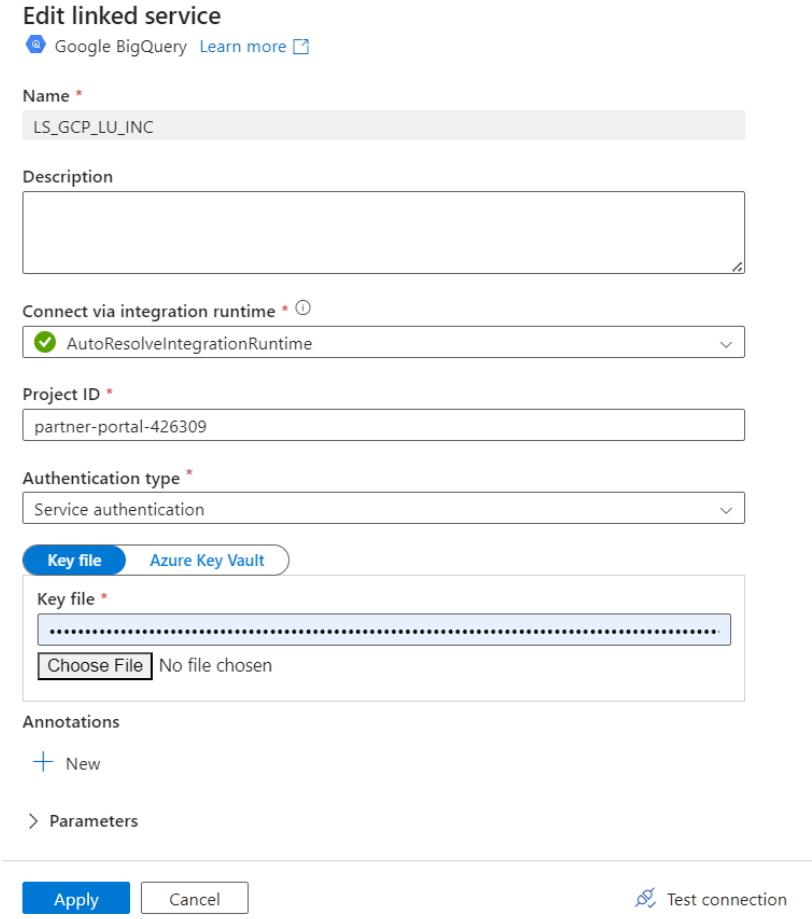


Step-by-Step Guide

Step 1: Create Linked Service for Big Query in Azure Data Factory
Go to Azure Data Factory.

- In Manage > Linked Services, click New.
- Select Google Big Query as the service.
- Configure the authentication settings to connect to your Big Query environment.

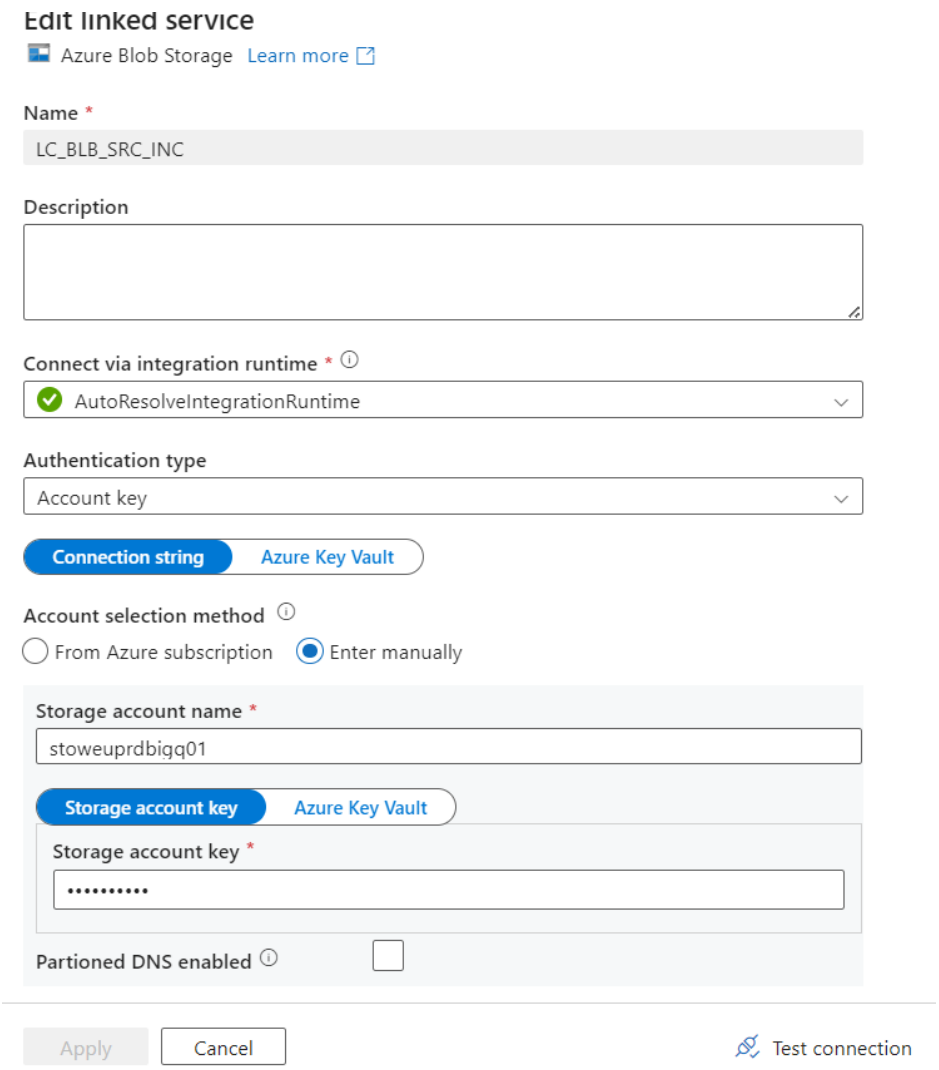
Usage: This linked service will be used in the Lookup and Copy activities to fetch the latest file from Big Query.



Step 2: Create Linked Service for Azure Blob Storage (Staging Area)

- Go to Manage > Linked Services > New.
- Select Azure Blob Storage as the service.
- Set up the connection to your Blob Storage account.

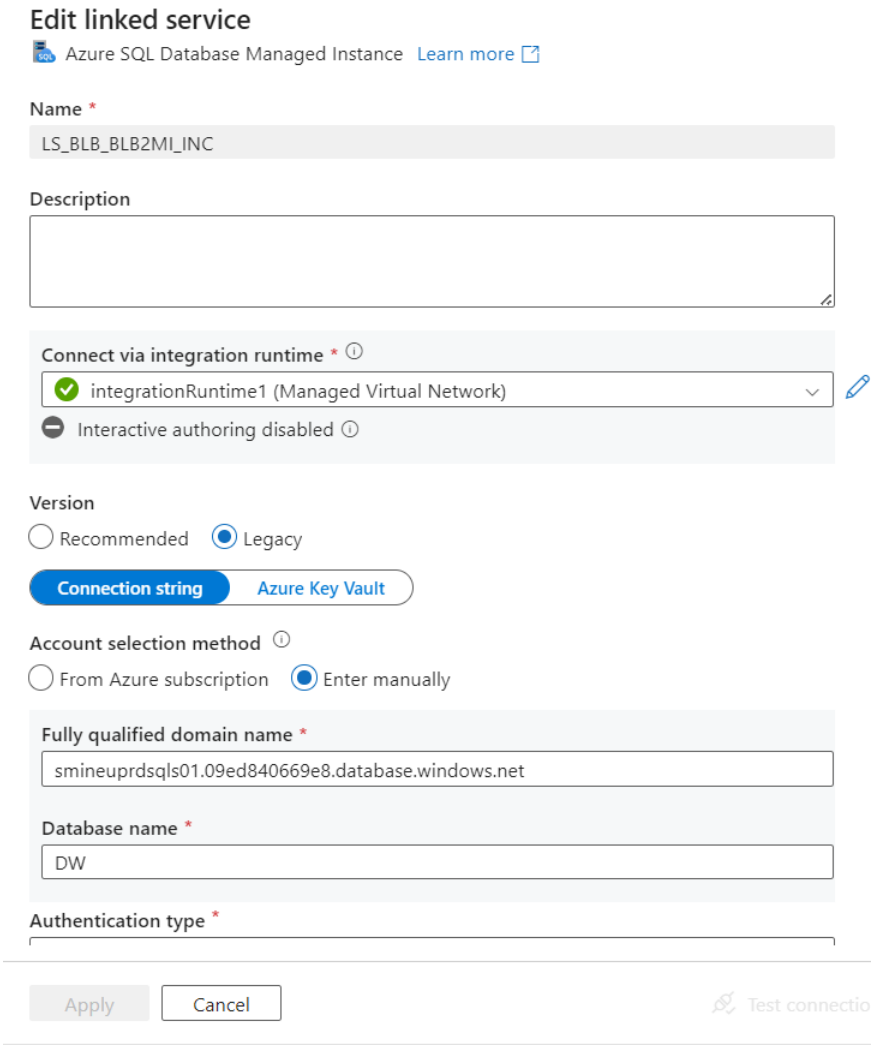
Usage: This linked service will store the file temporarily before loading it to MI using a Copy activity.



Step 3: Create Linked Service for Azure SQL Managed Instance

- Go to Manage > Linked Services > New.
- Select Azure SQL Managed Instance.
- Configure the connection details for the MI database.
- Create new integration run time because we are using Managed instance so auto integration runtime is not possible to use.
- Create the managed instance private end points and you can approve the end points from the managed instance
- Enable the interactive authoring

Usage: This linked service will be used in the Data Flow or Copy activity to load data into the MI table, allowing schema drift.



Step 4: Create Datasets for Google Cloud BigQuery (Source)

- Create two datasets for BigQuery:

For the Copy Activity:

This dataset will point to the BigQuery source, from which data will be copied to the staging area (Blob).

For the Lookup Activity:

This dataset will be used to fetch the latest file using a query.

Step 5: Create Dataset for Azure Blob Storage (Staging Area)
Create a dataset in ADF for the Blob Storage.

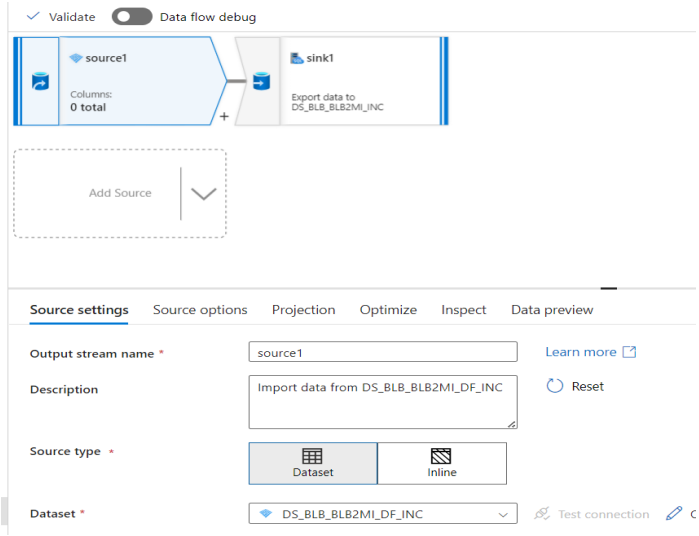
- This dataset will be used as the destination for the Copy Activity, which temporarily stores the data before transferring it to MI.
- I am using the parquet format.

Step 6: Create Dataset for Azure SQL Managed Instance (MI)

- Create a dataset for the SQL Managed Instance that will be used in the Data Flow activity to load the data from the Blob storage to MI.

Step 7: Create a Data Flow for Loading Data from Blob to MI

- In ADF, create a new Data Flow.
- Set Blob Storage as the Source.
- Set SQL Managed Instance as the Sink.
- Allow schema drift to accommodate any changes in the schema from the source.
  Enable Debug Mode to troubleshoot during development.



Step 8: Create the Pipeline

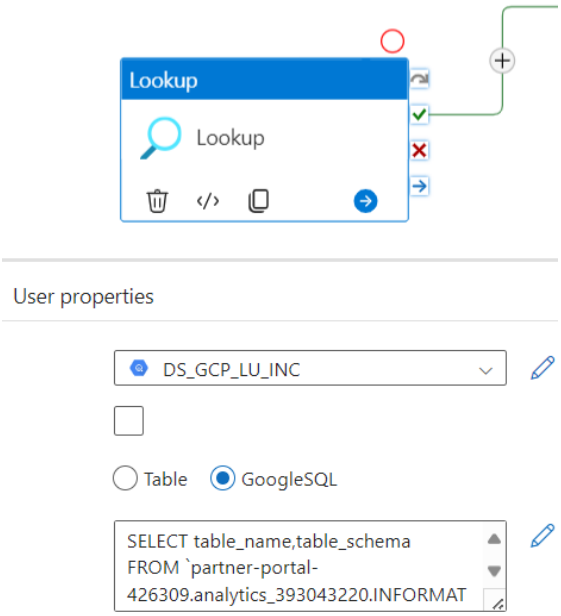
Add Lookup Activity:

- In your ADF pipeline, add a Lookup Activity.
- Use the dataset created for BigQuery Lookup.
- Write the following query to fetch the latest file from BigQuery:

SQL

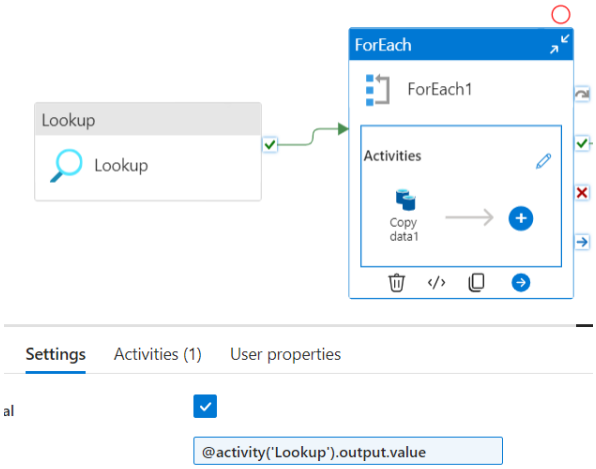
```
SELECT table_name, table_schema
FROM `partner-portal-426309.analytics_393043220.INFORMATION_SCHEMA.TABLES`
WHERE table_type = 'BASE TABLE'
 AND table_name LIKE 'events_%'
ORDER BY creation_time DESC
LIMIT 1;
```

This query will return the latest table.



Add For Each Activity:

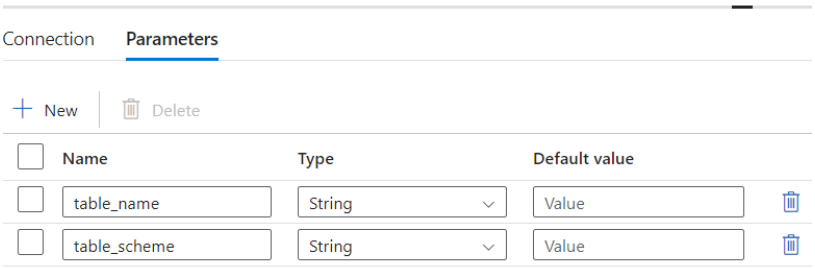
- Add a For Each Activity to loop through the results from the Lookup.
  [Use @activity('Lookup').output.value]
  to fetch the file name dynamically.
- Configure parameters for table_name and table_schema to fetch the latest file.
  Set the number of files to process, e.g., LIMIT 1 for the latest file.



Add Copy Activity:

- Inside the For Each Activity, add a Copy Activity to copy data from Big Query to Blob Storage.
- Use the dataset created for Big Query (Source).
- Configure two parameters, table_name and schema_name, and map them to:

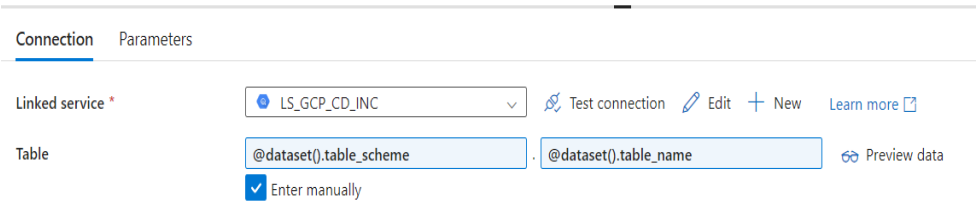
Google BigQuery
DS_GCP_CD_INC



@item().table_name

@item().table_schema

- In the Sink properties, use the Blob dataset and specify the file format (e.g., .parquet):



@concat(item().table_schema,'_',item().table_name,'.pqt')

Add Data Flow Activity:

- After the Copy Activity, add the Data Flow Activity.
- Connect the For Each loop to the Data Flow.
  This will load the data from Blob Storage to MI, allowing schema drift.

Optional: Add Delete Activity:

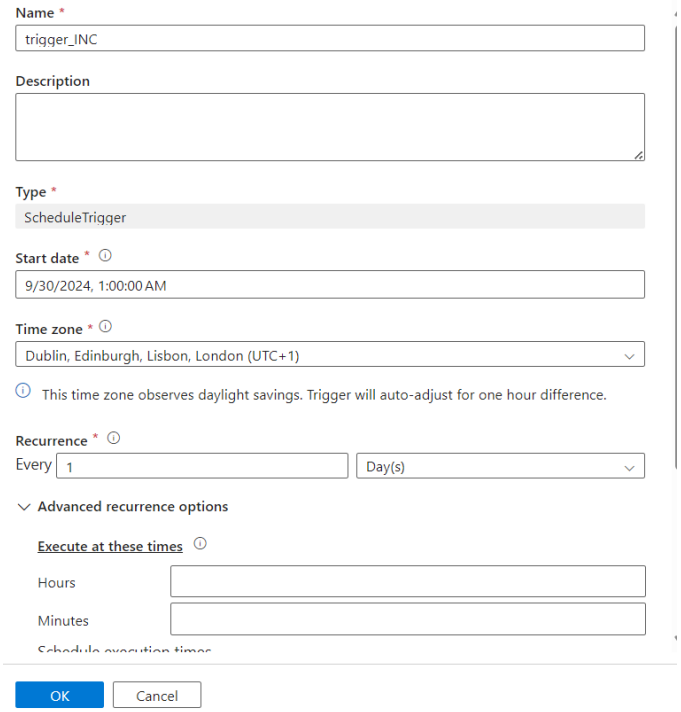
- Add a Delete Activity to remove the file from Blob Storage after the data is loaded into MI.
- This helps to save storage space by deleting the file after processing.

  Note: Skip this step if you want to retain the file in Blob Storage.

Step 9: Set Up Daily Trigger

- In Triggers, create a new trigger that runs the pipeline every day.
  Attach this trigger to the pipeline for automatic execution without manual

intervention.



Step 10: Debug and Run the Pipeline

- Enable Debug Mode in ADF to test the pipeline before deployment.

- After successful debugging, publish the pipeline.
  The pipeline will now run daily, fetching the latest file from BigQuery, loading it to Blob Storage, then transferring it to MI.

Conclusion:

By following the steps outlined above, we have successfully created an incremental data pipeline in Azure Data Factory that automates the process of fetching the latest data from Google Cloud Big Query and loading it into an Azure SQL Managed Instance (MI). This solution efficiently moves the latest files daily, utilizes Blob Storage as a staging area, and incorporates flexible data handling by allowing schema drift. Additionally, the optional file deletion step helps optimize storage usage.