**ETL WORKFLOW AUTOMATION**

## TABLE OF CONTENTS :

* Project Statement
* Project Overview
* Prerequisites
* Azure Resources Used for the Project
* Project Objectives
* Tools Used
* Execution Overview
* Implementation-Tasks Performed
* Practical Implementation on Azure Portal
* Successful Output Generated
* Strategies that can be used in Optimising the Conversion process
* Conclusion

## Project Statement:

Build an end-to-end ETL (Extract, Transform, Load) workflow using Azure Data Factory to orchestrate data movement and transformation, and leverage Azure Databricks for complex data transformations using Spark.

## Project Overview:

This project demonstrates how to integrate multiple Azure services to create a scalable, reliable, and optimized ETL process. The raw dataset in .csv format undergoes various data cleaning steps such as removing duplicates, handling null values, trimming spaces, and parsing dates.

* **Raw Zone (Bronze Layer):** Stores unprocessed CSV files.
* **Cleaned Zone (Silver Layer):** Stores cleaned and standardized Parquet files.
* **Analytics Ready Zone (Gold Layer):** Stores aggregated/optimized tables for reporting.

This layered approach improves **data governance, quality, and accessibility** for downstream analytics.

## Prerequisites:

 **Azure Subscription:** Active Azure Subscription for resource management.

 **Resource Group:** Create a resource group for ETL resources.

 **Azure Storage Account:** Storage account with Data Lake Gen2 enabled.

 **Azure Data Factory (ADF):** Instance for pipeline orchestration.

 **Azure Databricks:** Workspace for Spark-based processing.

 **Databricks Cluster:** Cluster with PySpark and Spark SQL support.

 **Access Keys:** Use storage account access keys for authentication.

 **Data Source:**  Ensure the availability of the CSV file.

## Azure Resources Used for this Project:

* Azure Data Lake Storage Gen2
* Azure Data Factory
* Azure Storage Account
* Azure Databricks Workspace
* Copy Data Activity
* Databricks Notebook Activity

## Project Objectives:

* Build an automated ETL pipeline by leveraging Azure cloud services for seamless data integration and management.
* Apply data cleansing and transformation logic using PySpark to ensure accurate and reliable data processing.
* Optimize storage by saving transformed data in Parquet format, enabling efficient querying and analytics.
* Enforce strong data quality measures such as deduplication, null value handling, and schema validation.

## Tools Used:

**Azure Data Factory (Orchestrator):** ADF will serve as the orchestration tool, defining a pipeline with two stages:

**Copy Activity:** This stage will utilize the Azure Blob Storage or Azure Data Lake Storage linked service to copy the CSV files from the source container to a temporary folder in Azure Storage.

**Databricks Notebook Activity:** This stage will trigger a Databricks notebook that reads the copied CSV files, transforms them into Parquet format, and writes the new files to the final destination location (e.g., Azure Data Lake Storage Gen2).

**Azure Databricks (Data Processing Workspace):** Databricks notebooks are used to implement data cleansing, transformation, and schema validation using PySpark. The processed data is then written in Parquet format for optimized storage and analytics.

## Execution Overview:

1. **Upload the Dataset (Bronze Zone):**

The raw Netflix dataset in CSV format is uploaded into the **Bronze zone** of Azure Data Lake Storage Gen2. This zone stores raw, unprocessed data in its original format, acting as the single source of truth for future processing.

1. **Pipeline Orchestration with Azure Data Factory:**

An **Azure Data Factory pipeline** is created to automate the ETL process. Within the pipeline, a **Databricks Notebook Activity** is configured to trigger the execution of Databricks notebooks whenever the pipeline runs, ensuring smooth orchestration between services.

1. **Databricks Notebook – Data Ingestion:**

Inside Databricks, the raw CSV file from the Bronze zone is read into a **Spark DataFrame. A predefined schema** is applied during ingestion to enforce consistent data types, prevent schema drift, and ensure reliable downstream transformations.

1. **Data Cleansing and Transformation (Silver Zone):**

The ingested data is cleansed and transformed using **PySpark** operations:

* + Removal of duplicate entries.
  + Handling of null or missing values.
  + Standardization of formats (e.g., dates, text fields).
  + Schema validation and enforcement.

The curated output is stored in the **Silver zone** of the Data Lake in **Parquet format**, which provides optimized storage and faster querying for analytics.

* **Verification of Silver Output:**

The transformed Parquet files in the Silver zone are read back into a Spark DataFrame to validate the results. A sample of records is displayed, ensuring that cleansing, schema enforcement, and formatting were applied correctly.

* **Data Aggregation and Analytics Preparation (Gold Zone):**

From the curated Silver data, further transformations are applied in Databricks to prepare **business-ready datasets**:

* + Generating summary tables (e.g., Total movies by year, rating, country and category).
  + Performing aggregations, joins, or advanced transformations as required.
  + Structuring the data in a way that is optimized for reporting and BI tools.The final processed datasets are stored in the **Gold zone** of the Data Lake.

## Implementation-Tasks Performed:

 **Resource Setup:**

A dedicated resource group was created to manage project resources. Within this, a Storage Account with Data Lake Gen2 enabled was provisioned, and a container was structured into Bronze, Silver, and Gold zones. The raw Netflix dataset (CSV) was uploaded to the Bronze zone.

 **Databricks Configuration and Notebook Development:**

An Azure Databricks Workspace and Cluster were configured to serve as the processing environment. A Databricks Notebook was developed using PySpark SQL to ingest raw data, perform cleansing (duplicate removal, null handling, schema enforcement), and write the transformed data in Parquet format to the Silver zone.

 **Pipeline Orchestration with ADF:**

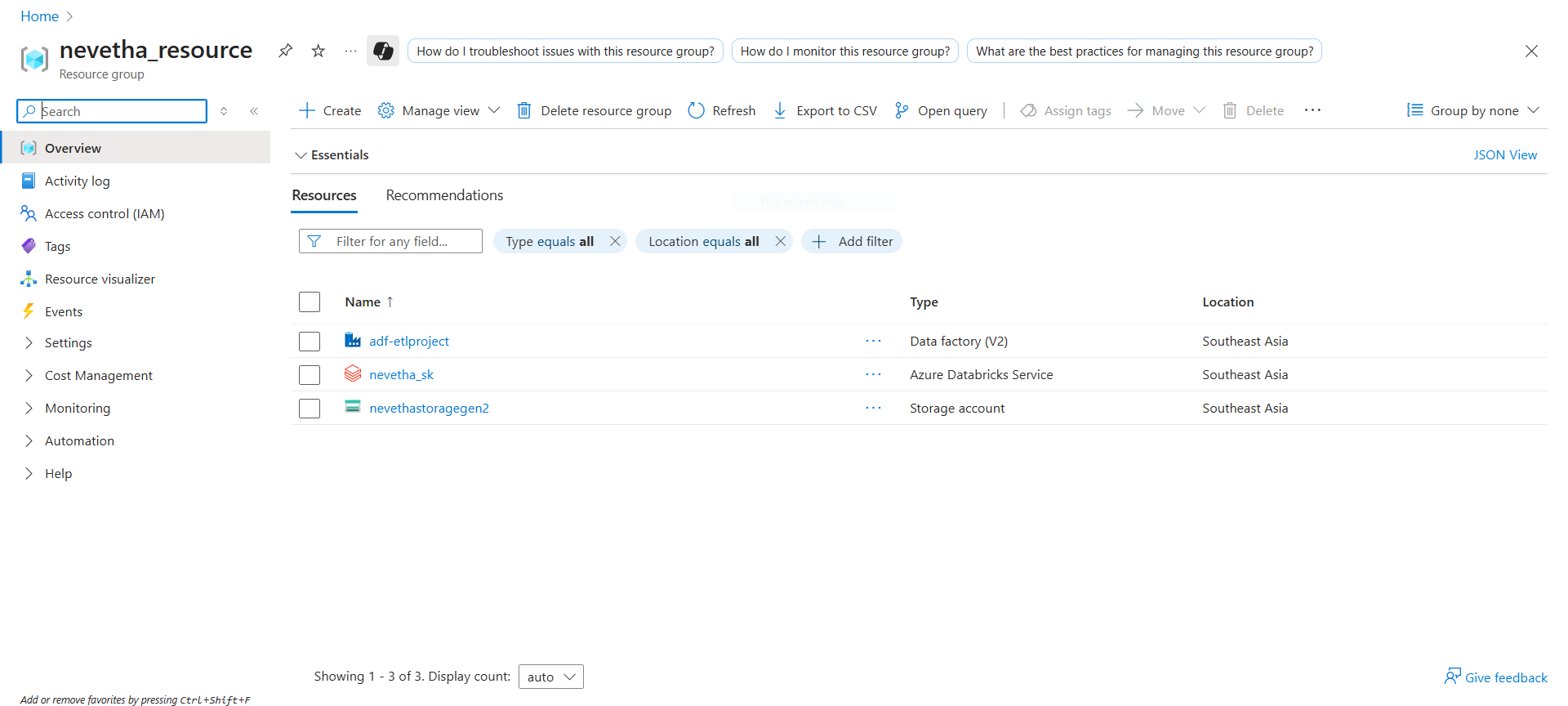
Azure Data Factory (ADF) was set up to orchestrate the ETL workflow. Linked Services were created to connect ADF with both Data Lake Storage and Databricks. An ADF Pipeline with a Databricks Notebook Activity was designed to automate the execution of the transformation logic.

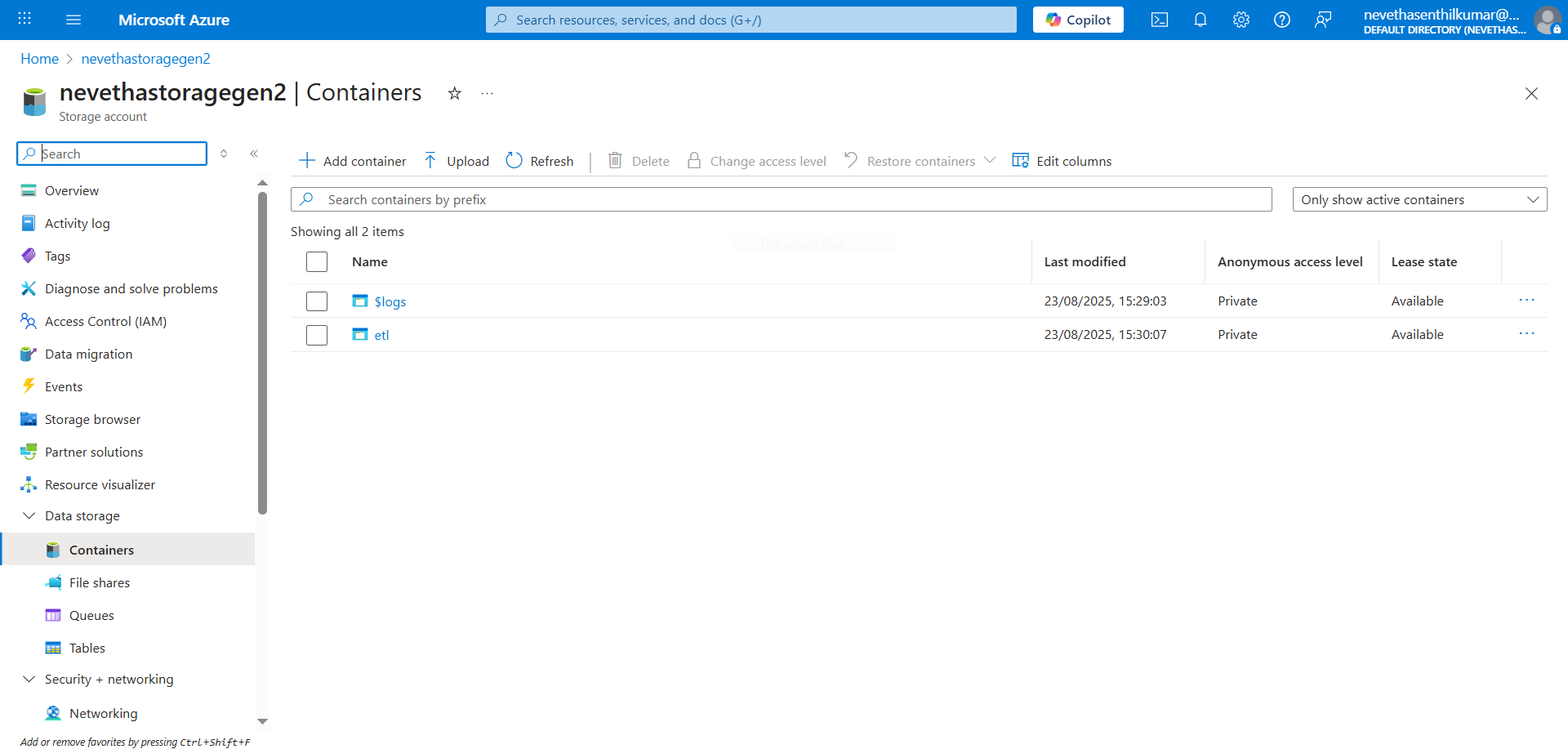
 **Execution and Validation:**

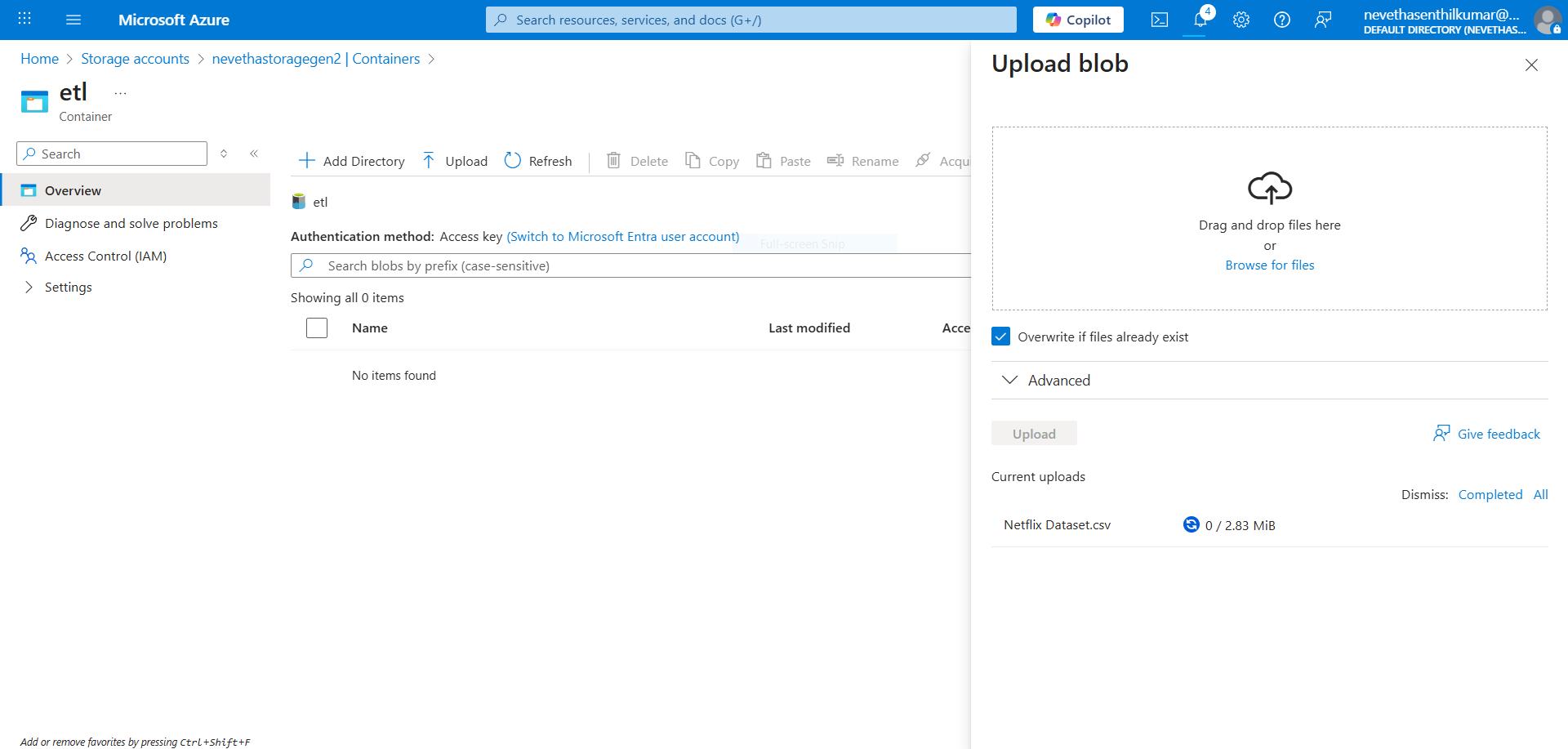
The pipeline was executed successfully, triggering the Databricks notebook. The output Parquet files were verified in the Silver zone by reading them into a Spark DataFrame and displaying sample records. This confirmed that the ETL pipeline executed as expected and produced curated, analytics-ready data.

## Practical Implementation on Azure Portal:

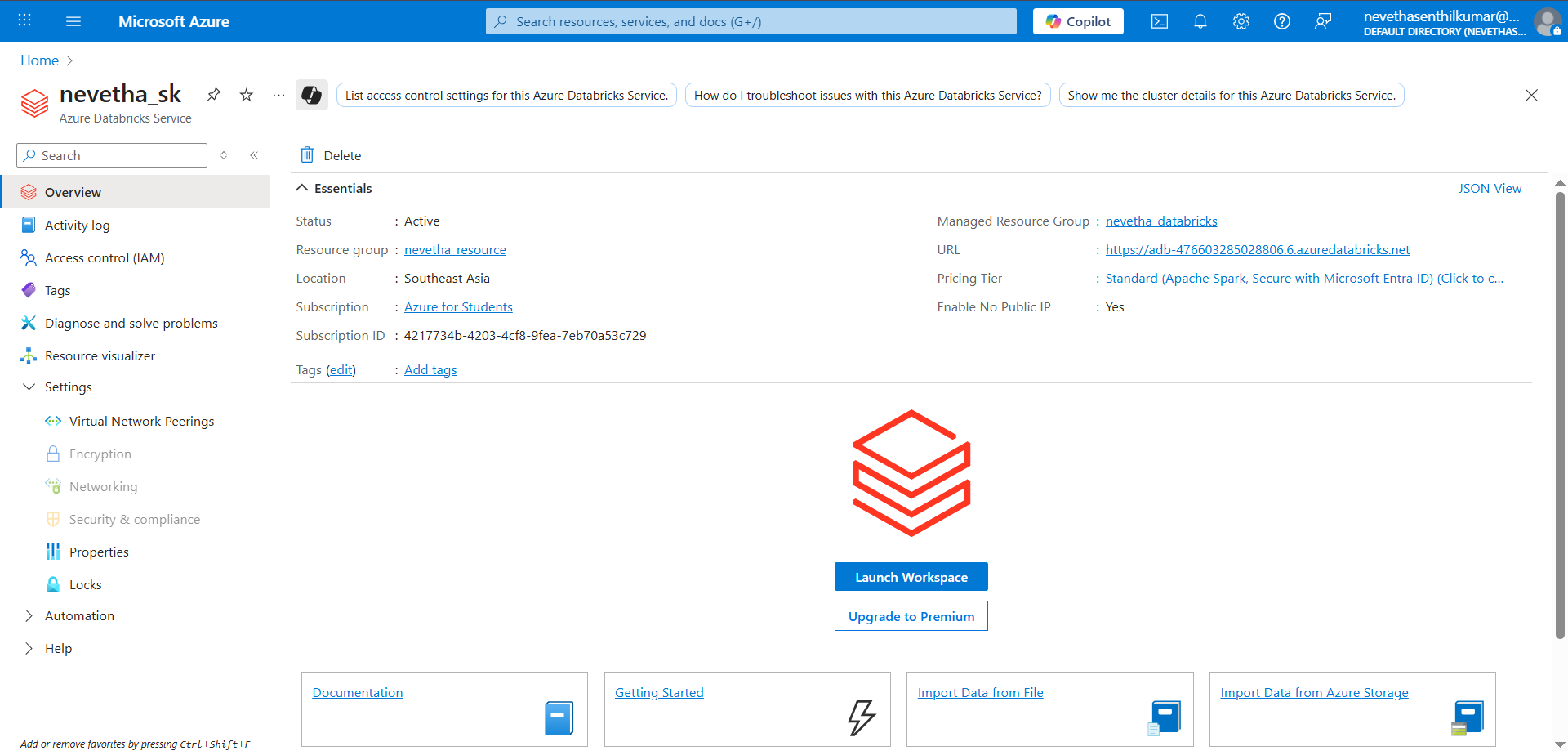
**STEP 1:** Create a Resource Group, Storage Account (with ADLS Gen2) and Upload the Raw Dataset (.csv file) in the newly created container

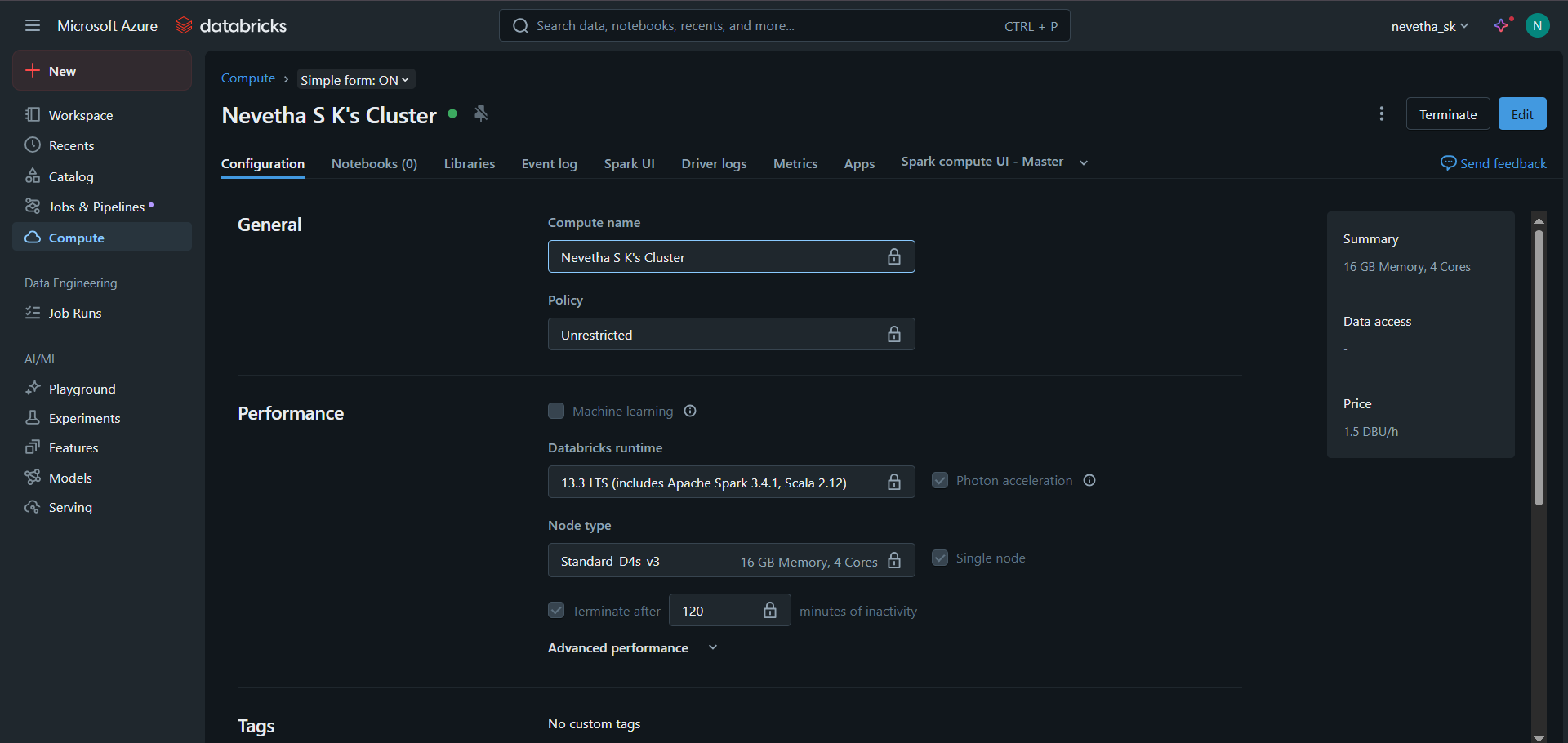






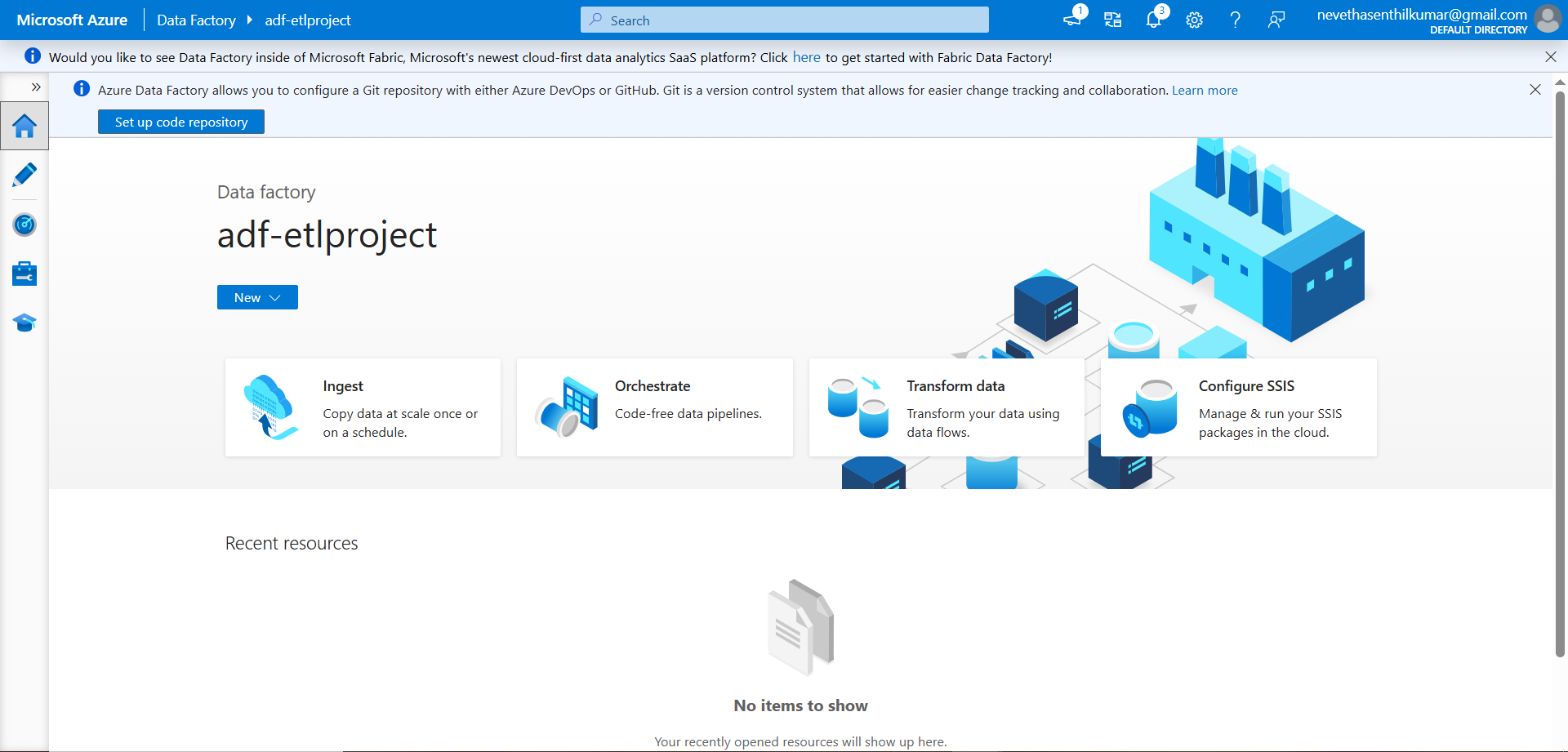
**STEP 2:** Create Azure Databricks Workspace & Cluster



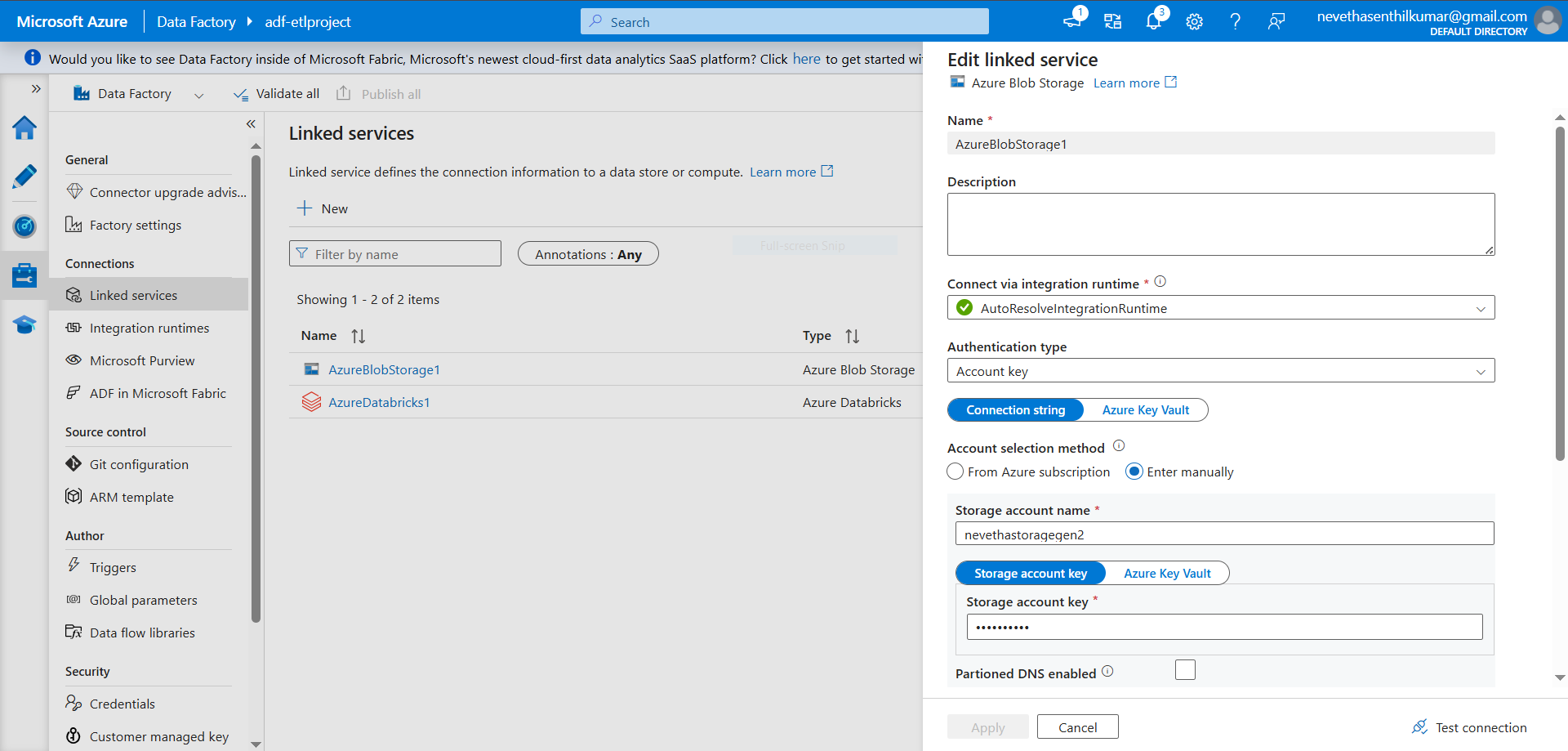


**STEP 3:** Orchestrate with Azure Data Factory (ADF)

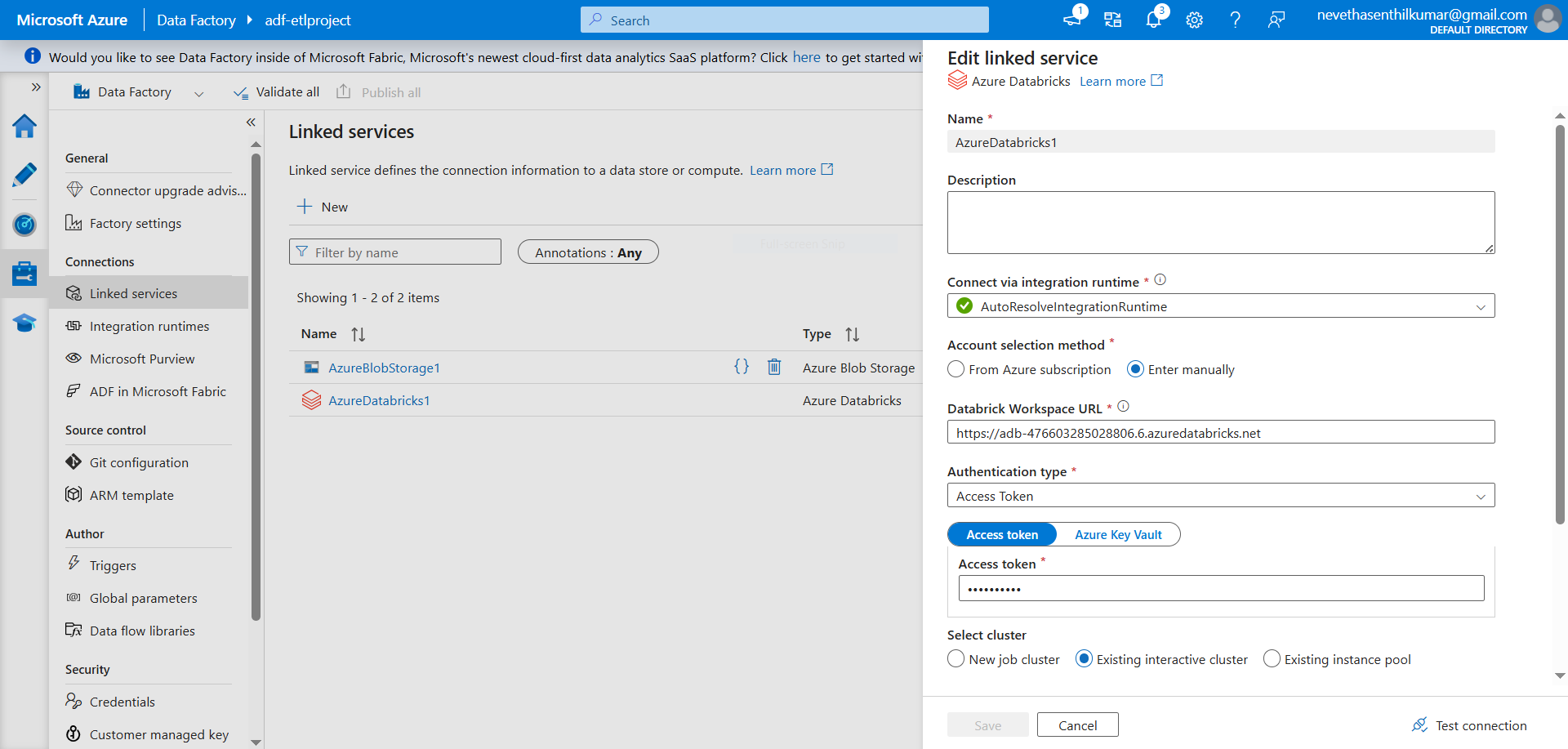
Create Data Factory



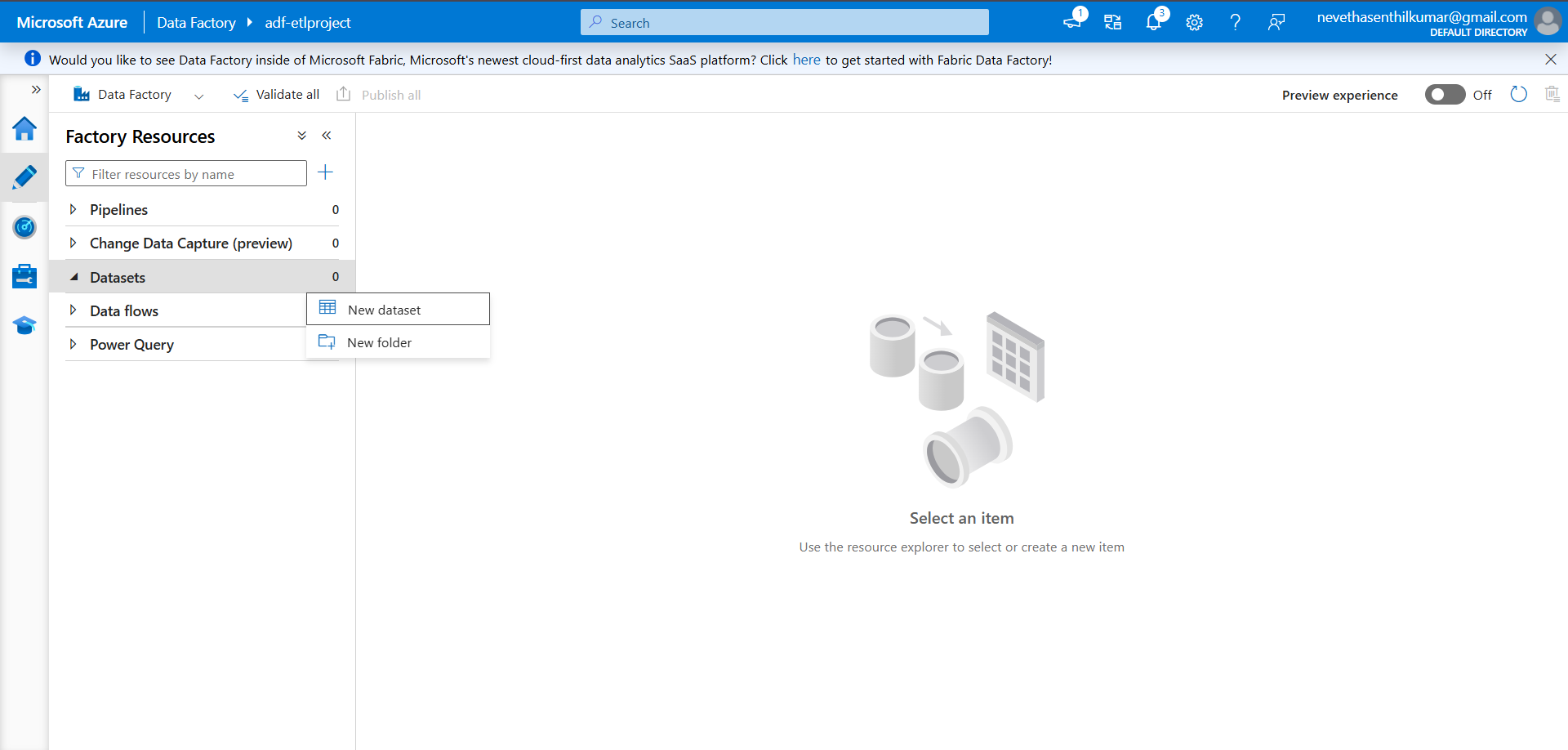
Create Linked Service: ADLS Gen2

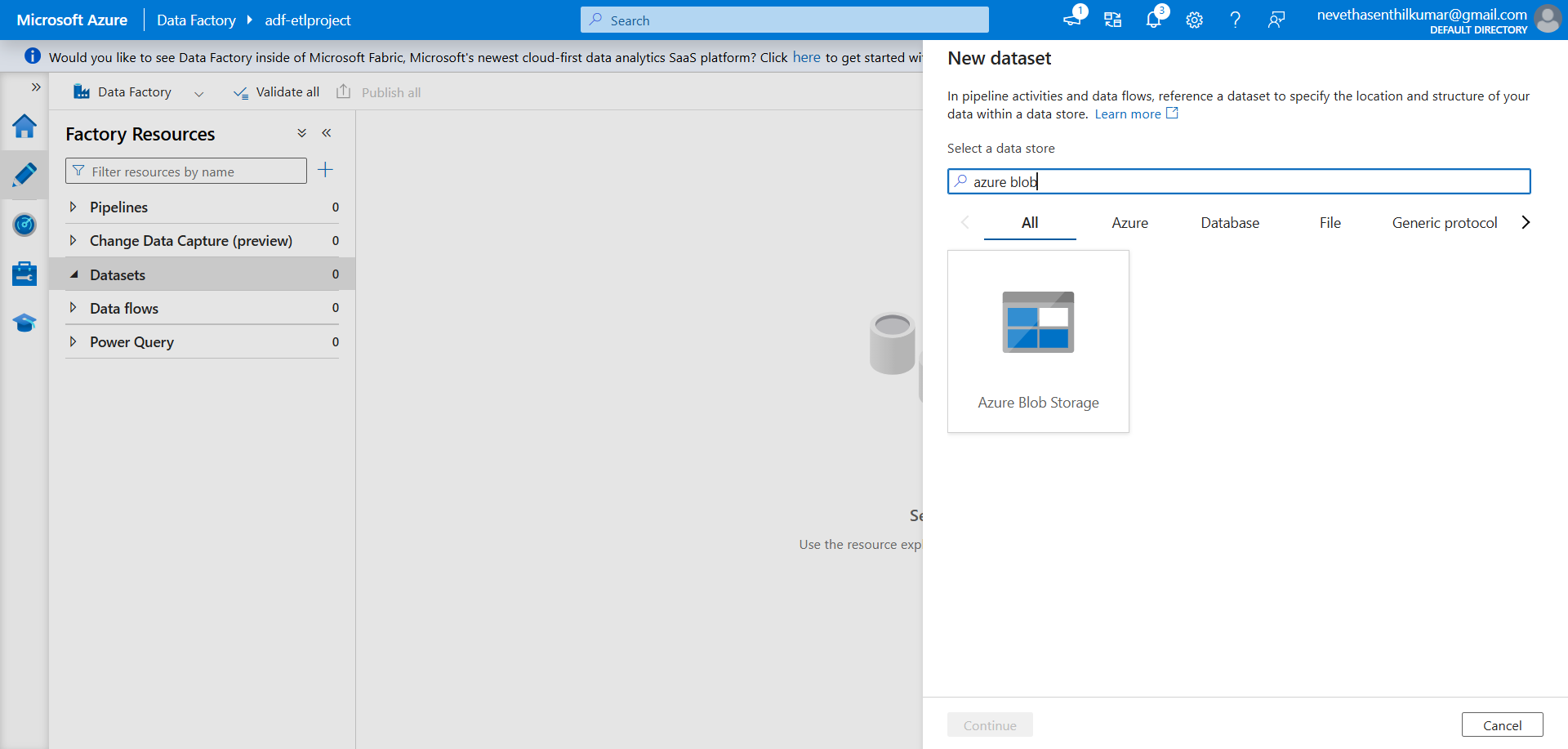


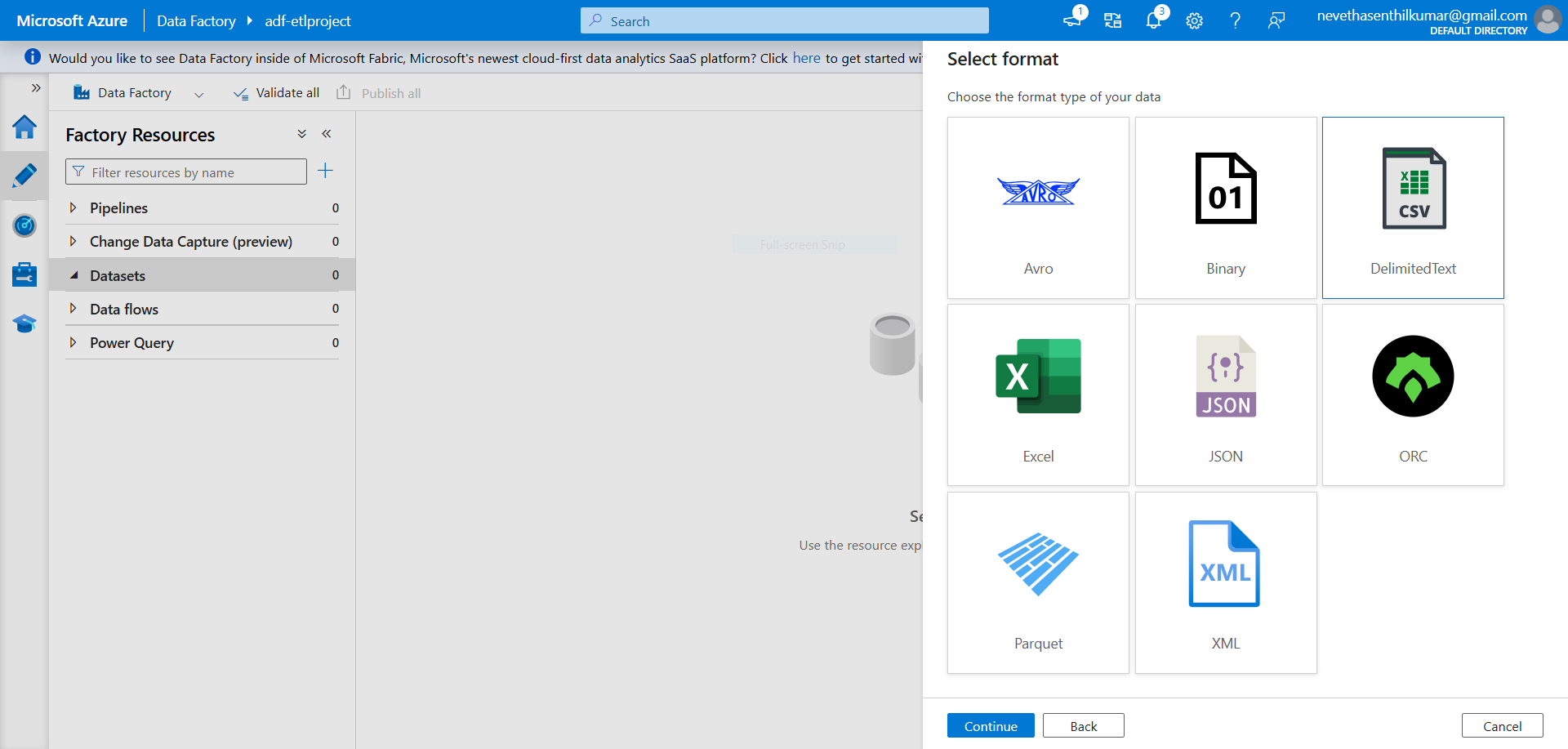
Create Linked Service: Azure Databricks

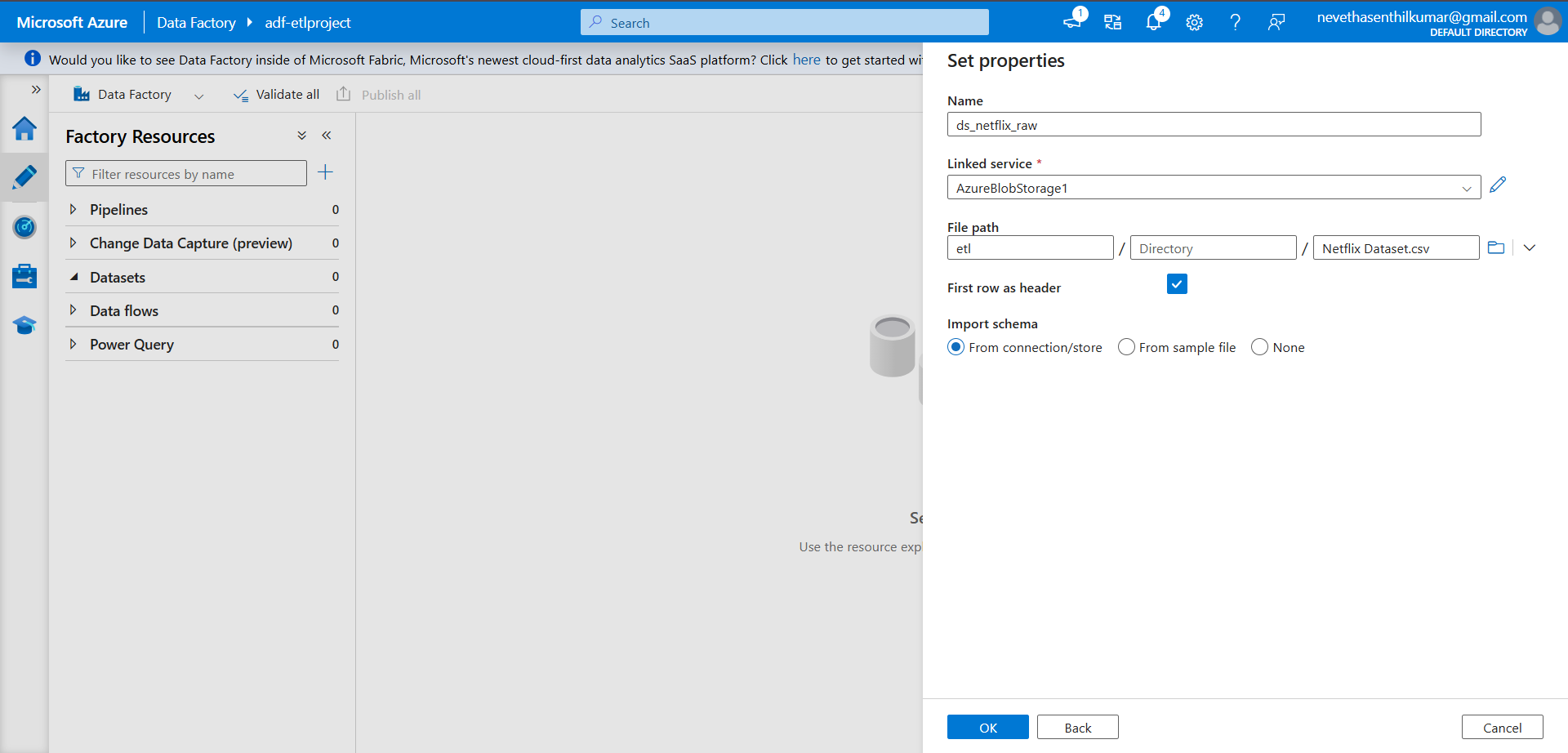


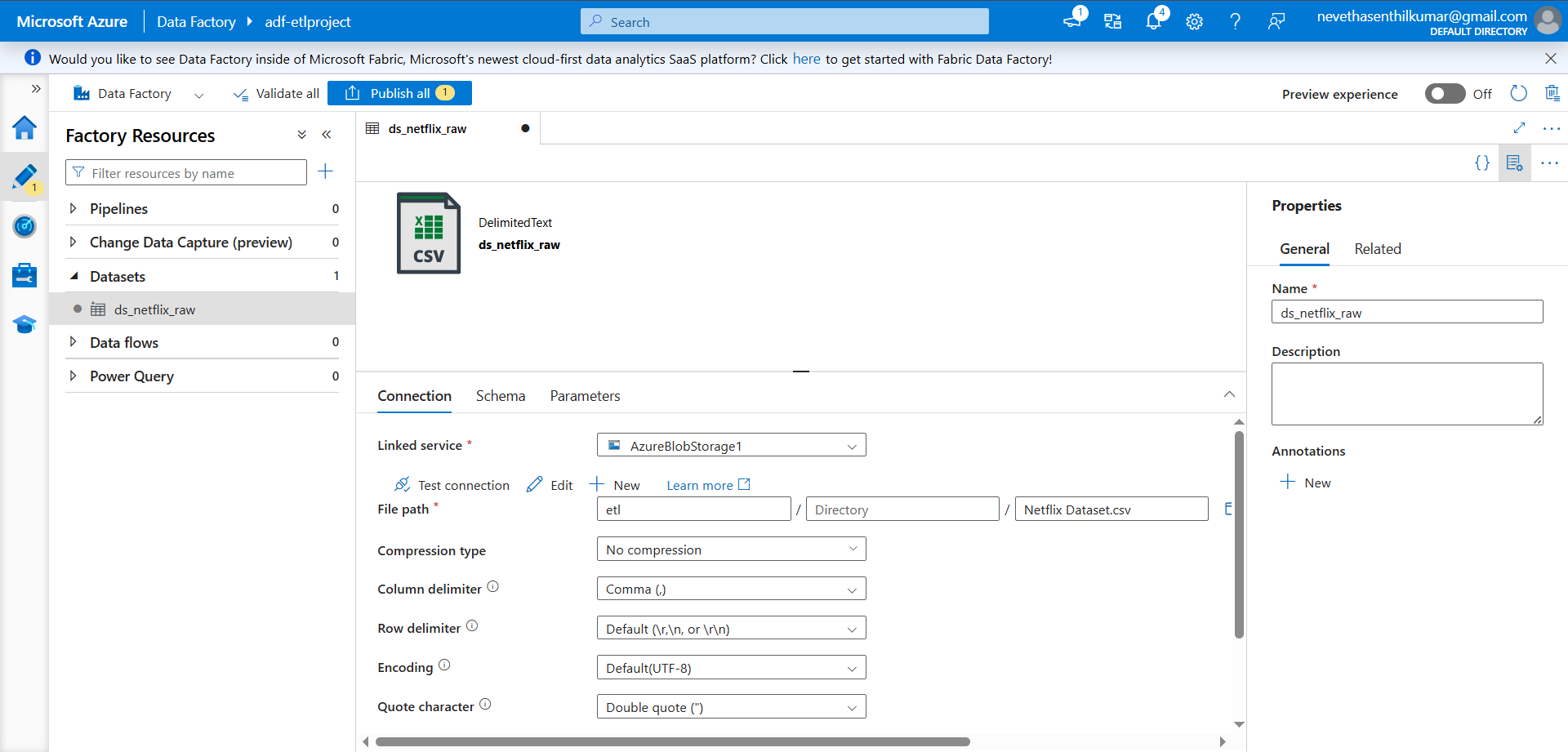
In Azure Data Factory, a dataset was created by linking to the delimited text file stored in Azure Blob Storage. This dataset acts as a reference, allowing ADF to read and process the data without re-uploading it.

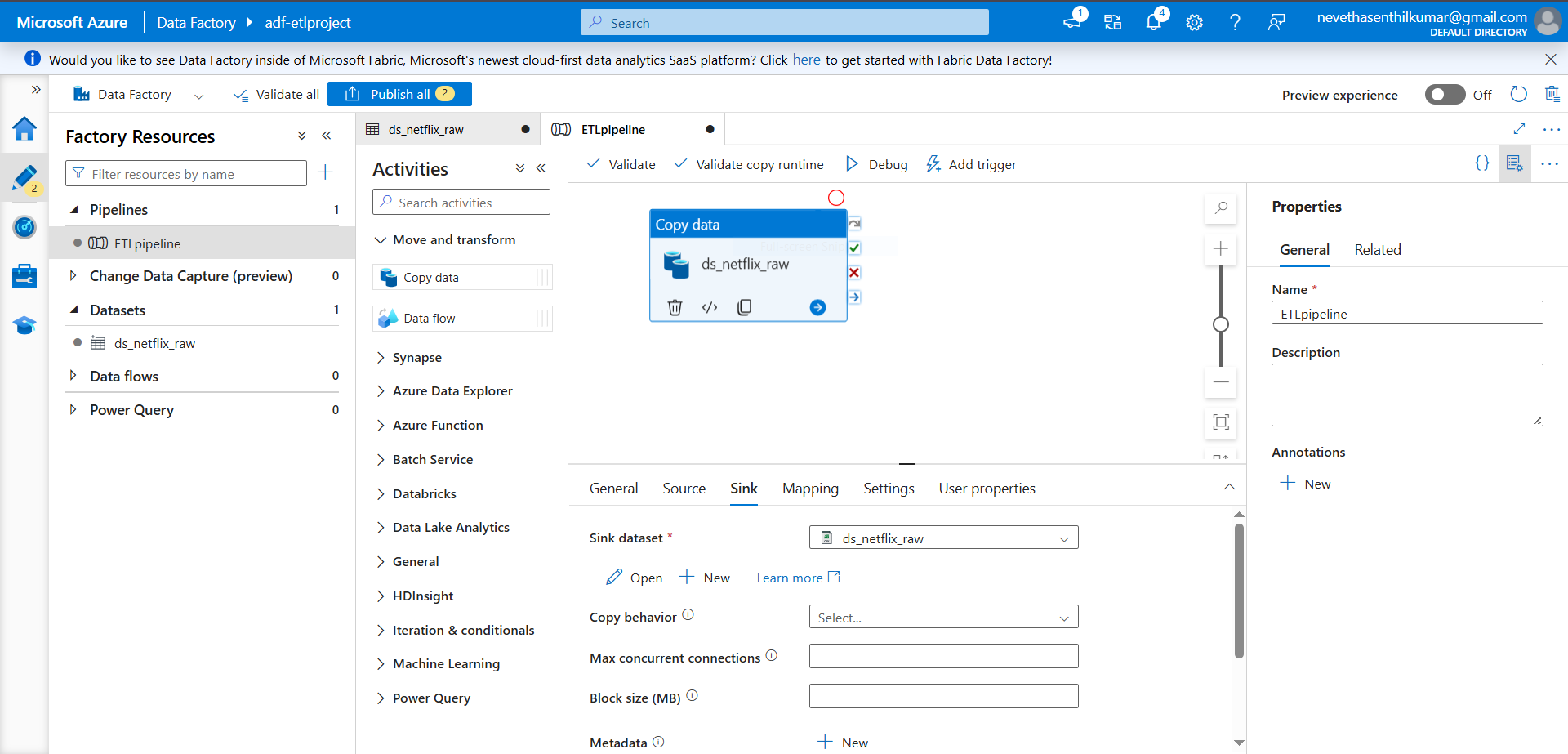




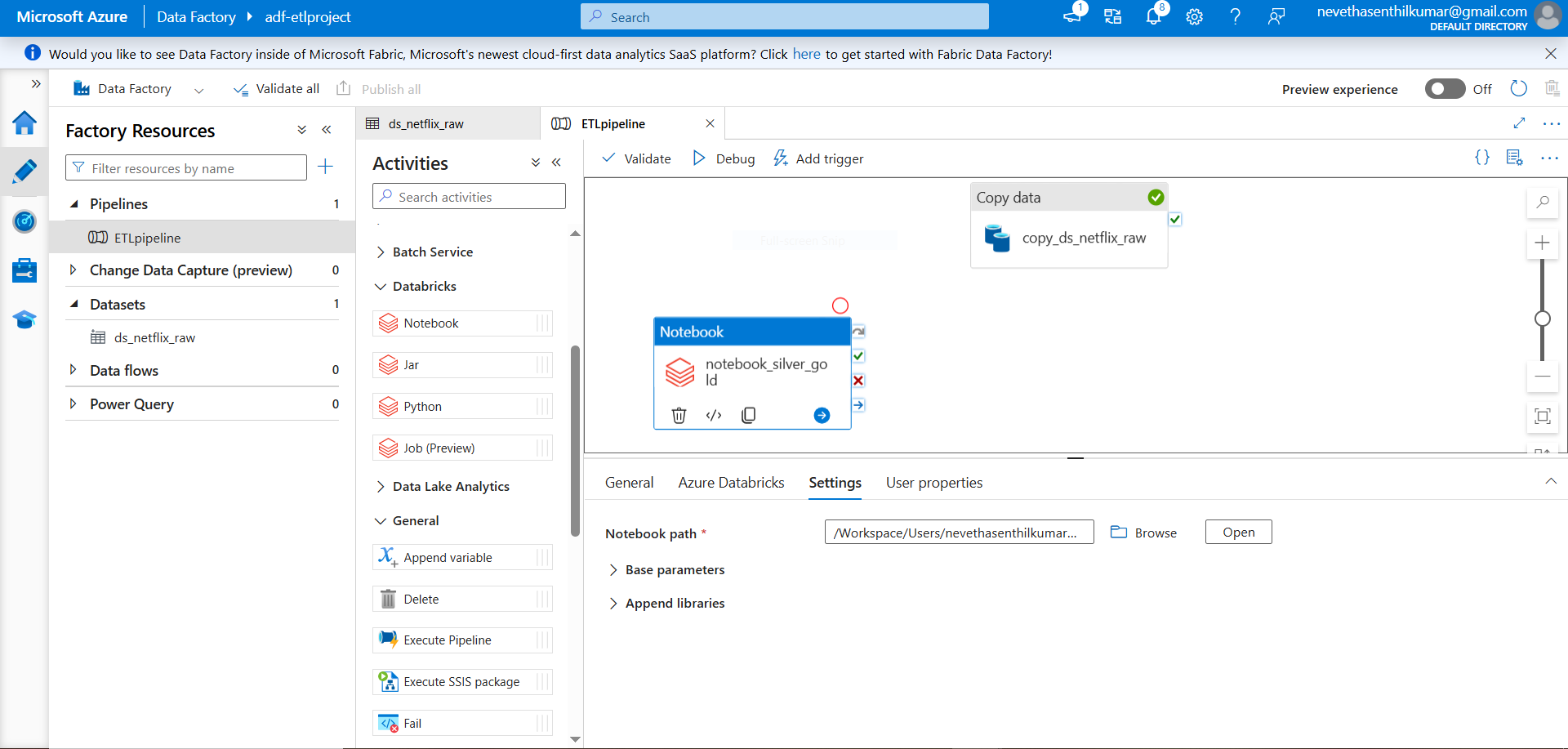






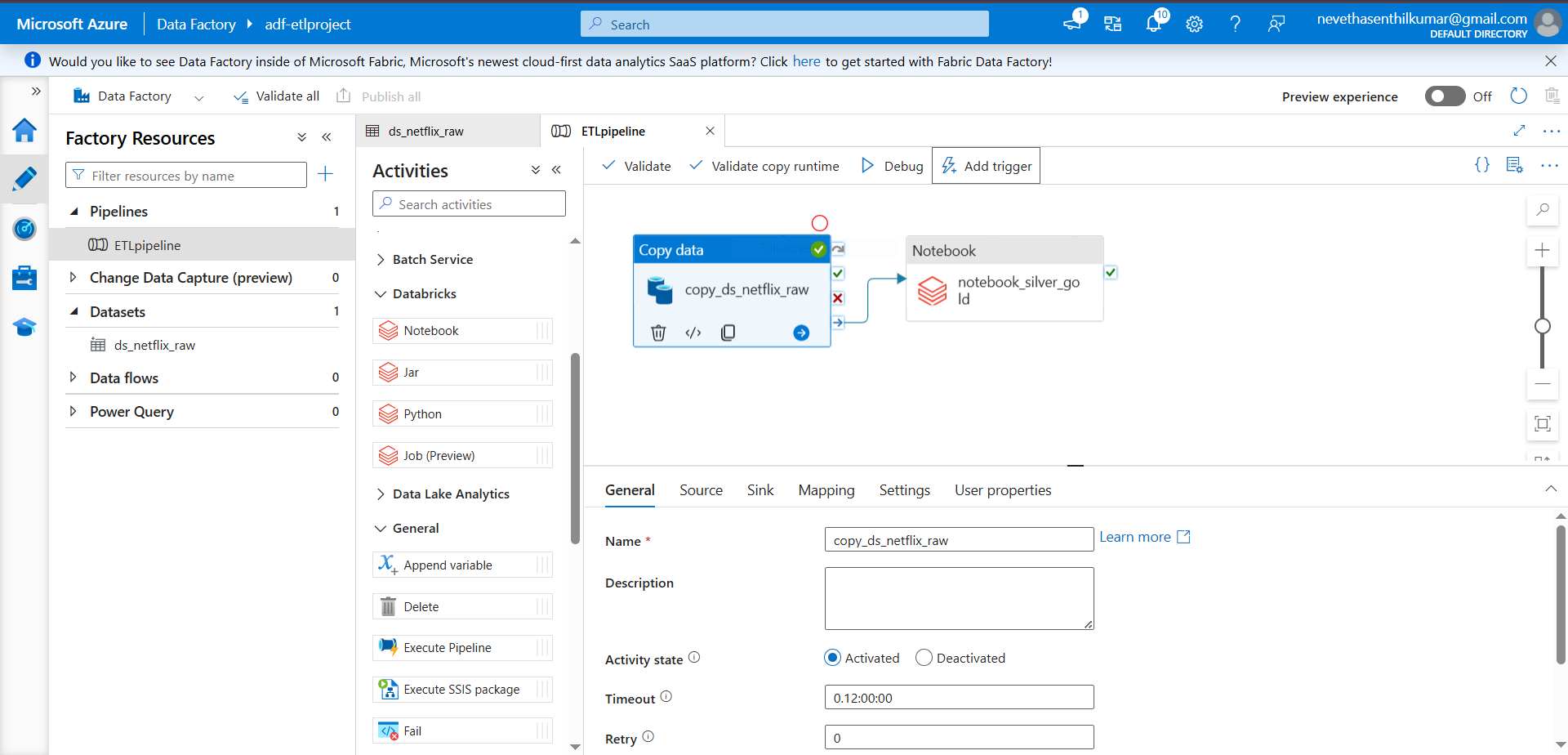


**Copy Data Activity** – Moves raw data from Blob Storage into the staging/target dataset.



**Notebook Activity** – Executes a Databricks notebook to transform and optimize the ingested data.

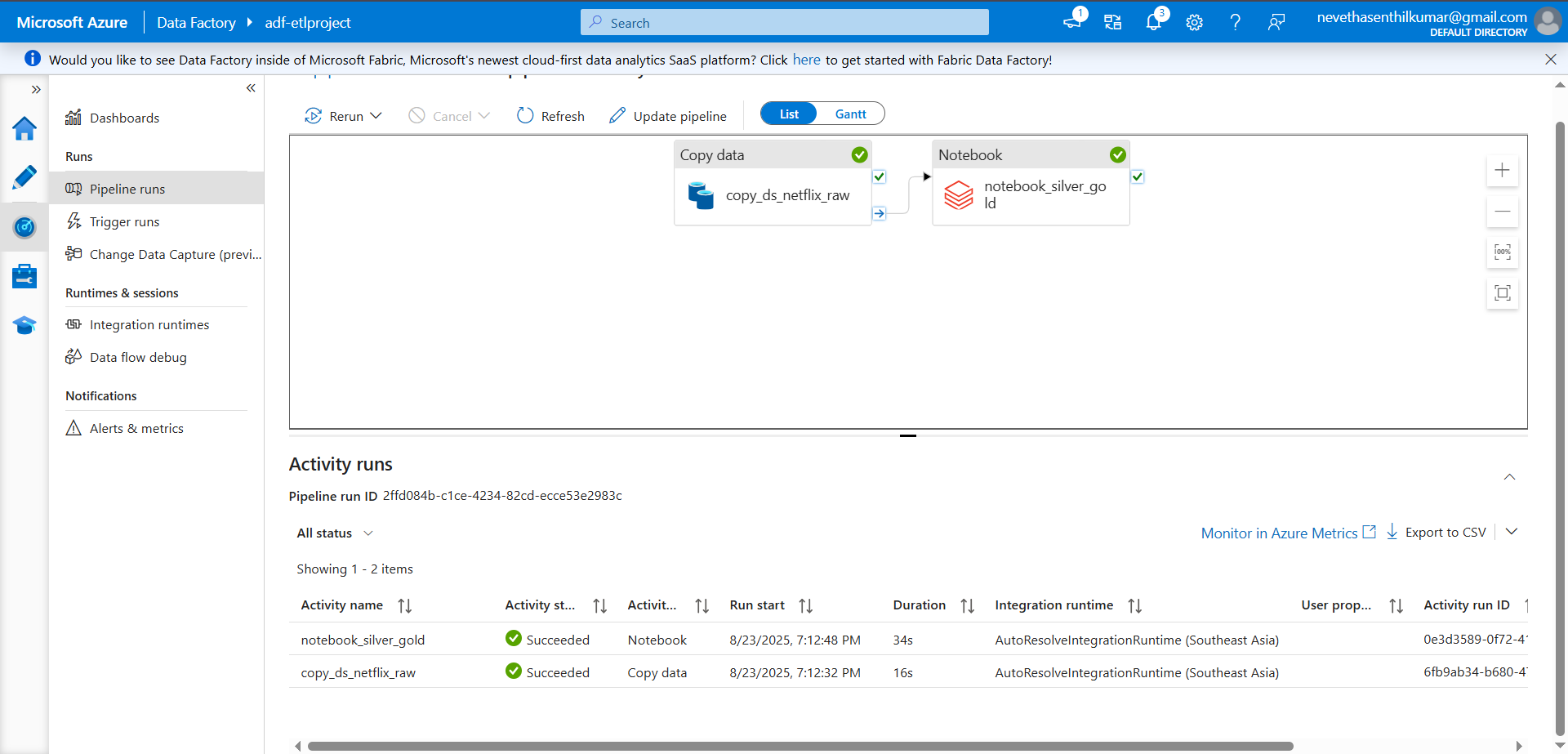
**Activity Connection** – A line is drawn between Copy Data and Notebook Activity to define execution order.

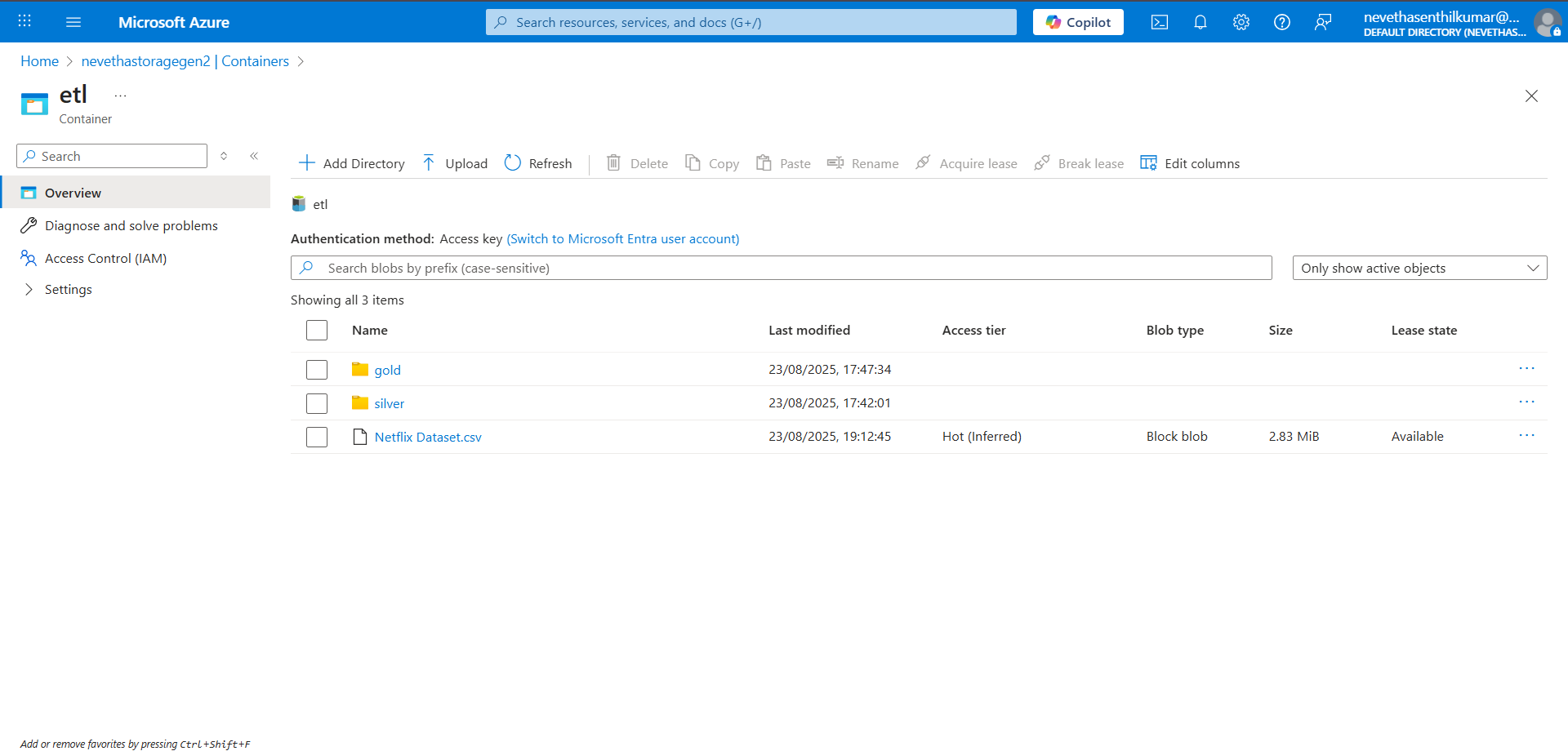


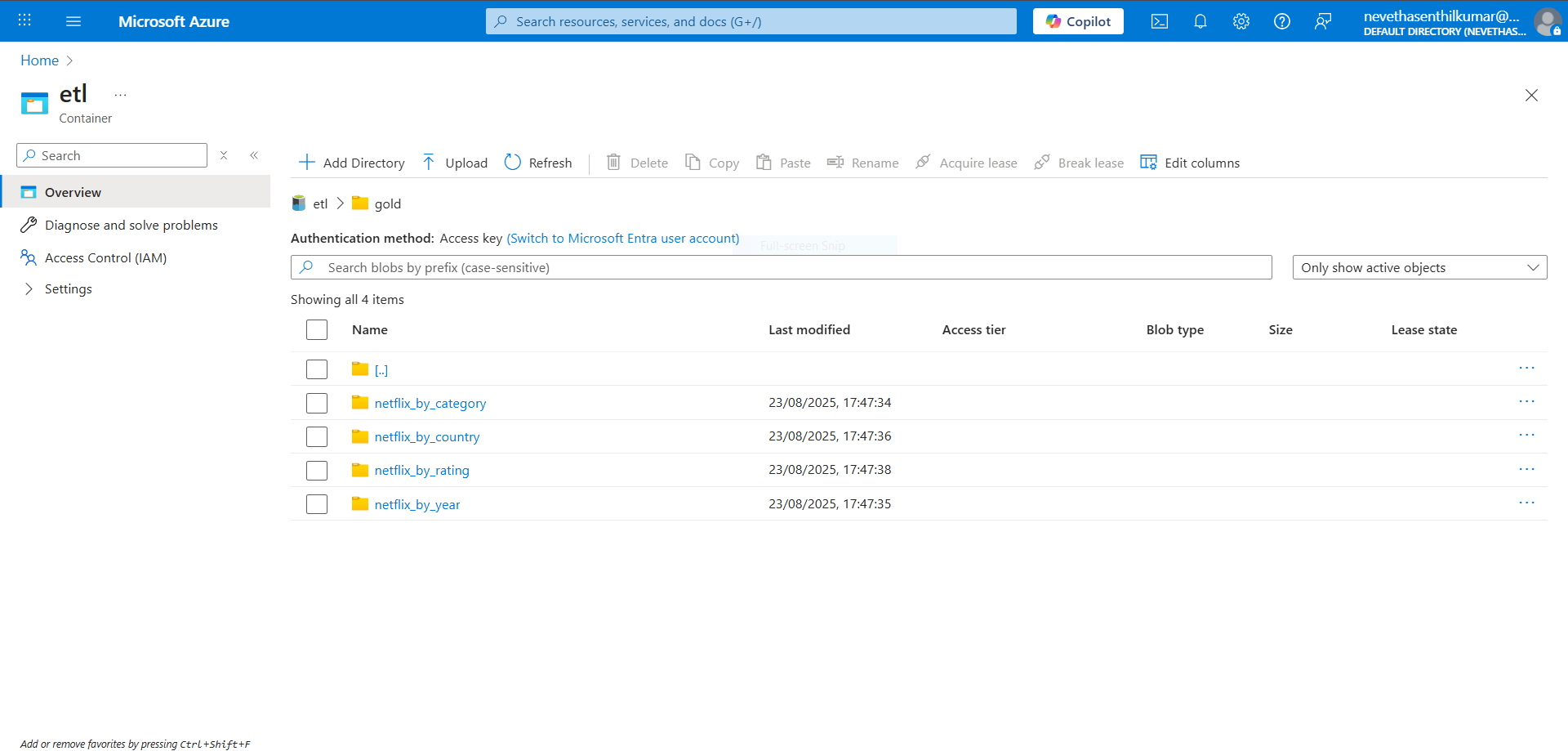
**Pipeline Trigger** – The pipeline is triggered manually to start the orchestration process.

## Successful Output Generated:

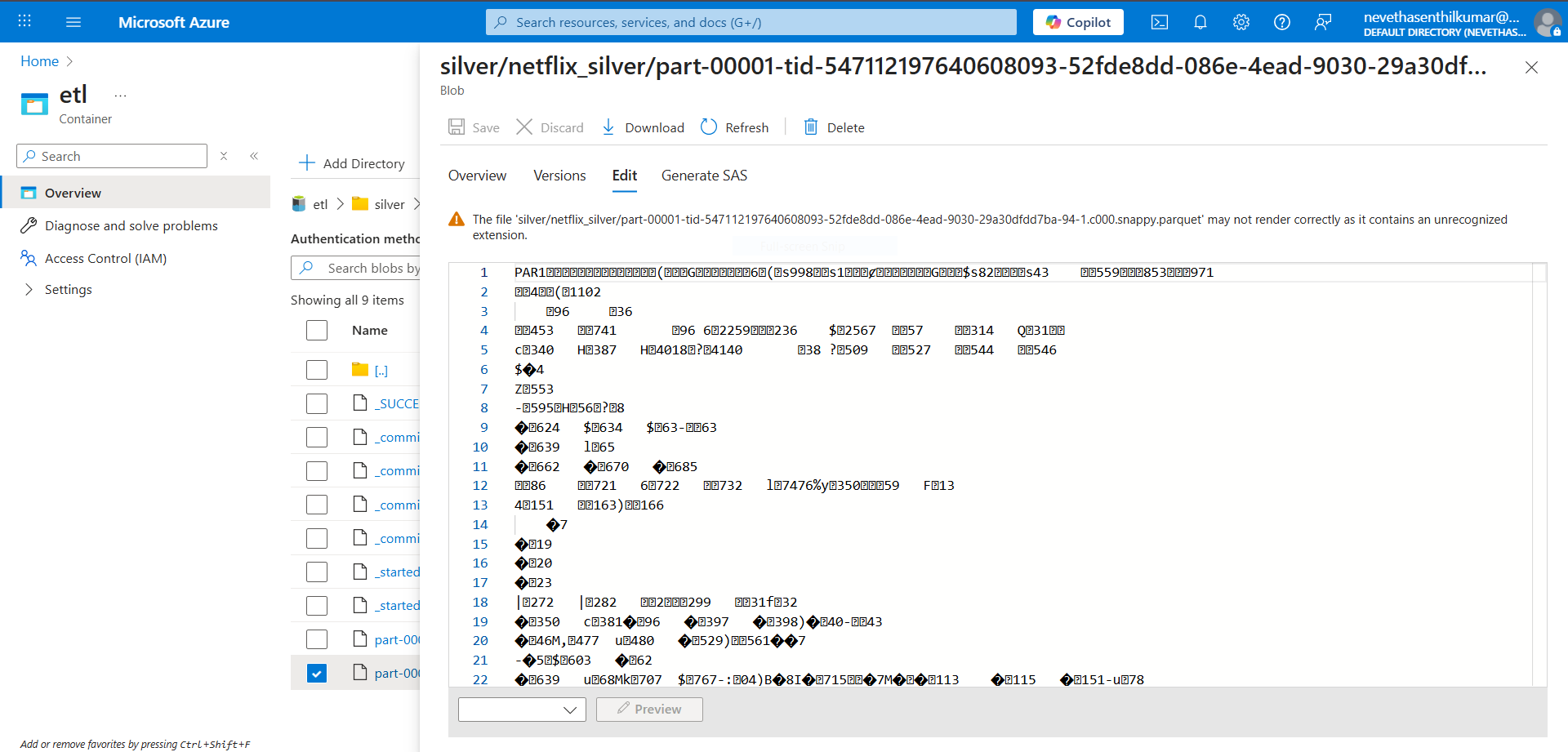
* Validate and Debug the pipeline created to see the results of the execution.
* It shows the Activity Status as Succeeded which means our CSV file is converted into the Parquet file in the destination folder after the successful ETL Process.







In the newly created folder, we can see files that are created as a result of the conversion process.



We can also see the data in the resultant file which is in parquet file format.

## Strategies that can be used in Optimising ETL Workflow:

### ****1. Efficient Data Ingestion****

* Use **incremental loading** instead of full loads (load only new/changed data using Watermarking or LastModifiedDate).
* Enable **parallelism** in ADF copy activities to ingest multiple partitions simultaneously.
* Compress input CSV files (Gzip/Snappy) to reduce network transfer time.

### 2. ****Data Transformation Optimization****

* Convert raw data into **Parquet or Delta Lake** instead of CSV for faster queries and reduced storage costs.
* Use **Databricks Auto Optimize** and **Z-Ordering** for better query performance.
* Push down transformations to Spark SQL instead of Python UDFs where possible (since SQL operations are optimized in Spark).

### 3. ****Pipeline Orchestration Best Practices****

* Break down the pipeline into **modular activities** (ingestion, staging, transformation, loading).
* Use **Data Factory triggers** for scheduled automation (daily/hourly runs).
* Enable **Retry policies & error handling** in ADF to ensure resilience.

### 4. ****Scalability and Performance****

* Use **Databricks autoscaling clusters** to optimize cost and performance during peak loads.
* Partition large datasets on frequently queried columns.
* Cache intermediate results in Databricks to avoid recomputation.

### 5. ****Monitoring and Debugging****

* Enable **ADF monitoring alerts** (via Azure Monitor/Log Analytics) to detect failed pipeline runs automatically.
* Store pipeline execution metadata (row counts, errors) for auditability.
* Implement **data validation checks** after transformations to ensure accuracy.

### 6. ****Cost Optimization****

* Shut down **Databricks clusters automatically** when idle.
* Use **reserved capacity pricing** for frequently used Azure resources.
* Optimize storage tiers (keep raw data in Hot tier, archived data in Cool/Archive).

### 7. ****Security and Governance****

* Use **Managed Identities** in ADF instead of storing credentials.
* Apply **RBAC (Role-Based Access Control)** to control access to ADLS and Databricks.
* Mask or encrypt sensitive data fields during transformation.

## Conclusion:

This project successfully demonstrated how to design and implement a **cloud-based ETL pipeline** using Azure resources. The integration of Azure Data Factory, Databricks, and Data Lake Storage ensures a scalable and efficient data engineering workflow. The raw Netflix dataset was transformed into a **clean, structured, and analytics-ready format,** making it suitable for advanced BI and ML use cases. Optimization strategies further improve performance and cost-effectiveness, making the solution production-ready for enterprise scenarios.