

Azure Data Factory and Databricks ETL Pipeline

Project Overview

This repository documents the creation and configuration of an Azure Data Factory (ADF) pipeline integrated with Databricks for an end-to-end ETL process. It leverages Azure Storage Accounts for Medallion architecture, Databricks notebooks for transformations, and Azure Key Vault for secure integration.

Business Scenario:

The pipeline processes retail sales data from various regions, transforming and organizing it into structured layers (Raw → Curated → Staging) for business insights.

Detailed Project Description

This project involves setting up a comprehensive ETL pipeline using Azure Data Factory (ADF) and Databricks. The pipeline processes sales data through various transformation stages based on the Medallion Architecture, enhancing data quality and usability. The project ensures secure management of credentials using Azure Key Vault and efficient data movement between Azure Storage Accounts.

Key Features:

- **Scalability:** Handles large datasets with partitioning and dynamic paths.
- **Security:** Uses Azure Key Vault for secure integration.
- **Adaptability:** Parameterized pipeline allows reusability across different datasets.

Architecture Components:

1. **Azure Data Factory:** Orchestrates data workflows.
2. **Databricks:** Executes PySpark scripts for transformations.
3. **Azure Storage Accounts:** Organized into Raw, Curated, and Staging layers.
4. **Azure Key Vault:** Secures sensitive credentials.

Well-Defined Workflow and Methodology

Step 1: Setting Up Storage Accounts for Medallion Architecture

1. **Source Storage Account (sourceingestion):**
 - **Create a Storage Account:**

- Go to the Azure Portal and create a storage account named sourceingestion.
- **Create a Container:**
 - Inside the storage account, create a container named sales-data.
- **Upload Source File:**
 - Upload your source file (e.g., sales_data.csv) to the sales-data container.

2. Destination Storage Account (projectstorage997):

- **Create a Storage Account:**
 - Create another storage account named projectstorage997.
- **Create Containers:**
 - Add three containers:
 - raw-data for storing unprocessed data.
 - curated for storing processed intermediate data.
 - staging for storing the final, transformed data.

Explanation: This structure follows the Medallion Architecture (Raw → Curated → Staging), organizing data flow into layers for better processing and management.

Step 2: Creating and Configuring ADF

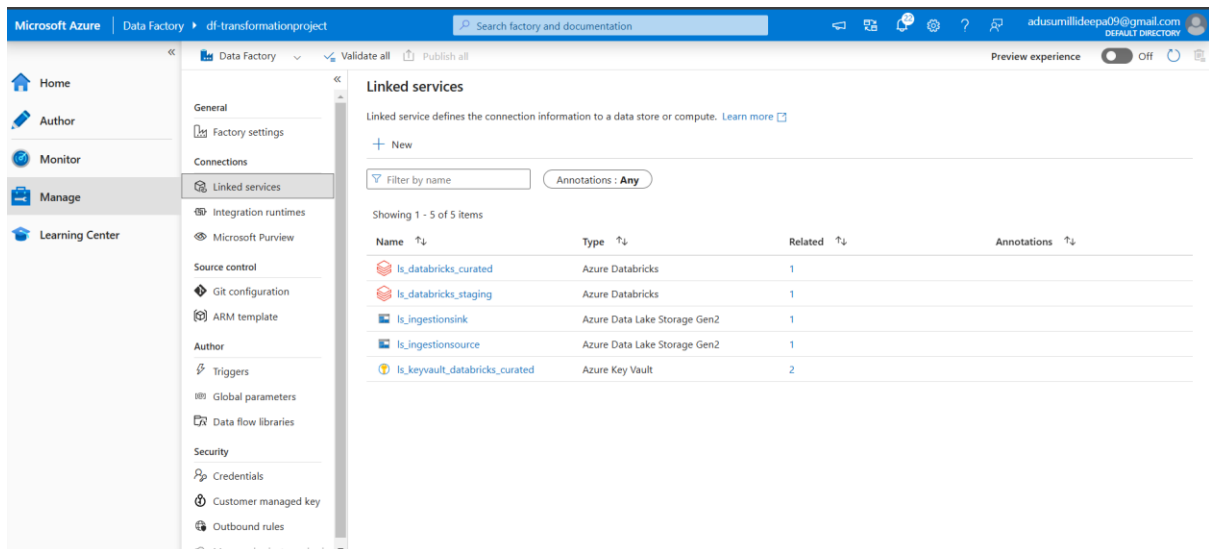
1. Create the ADF Instance:

- Go to the Azure Portal and create an instance of Azure Data Factory.
- Launch the Data Factory Studio for designing pipelines.

2. Linked Services Configuration:

- Navigate to Manage > Linked Services in ADF Studio.
- **Add Linked Services:**
 - For Source Storage: Connect to sourceingestion.
 - For Destination Storage: Connect to projectstorage997.
 - For Databricks: Use Azure Key Vault for secrets (explained in Step 5).

Explanation: Linked Services act as connectors, allowing ADF to interact with storage accounts, Databricks, and other resources.



3. Pipeline Design - Copy Data Activity:

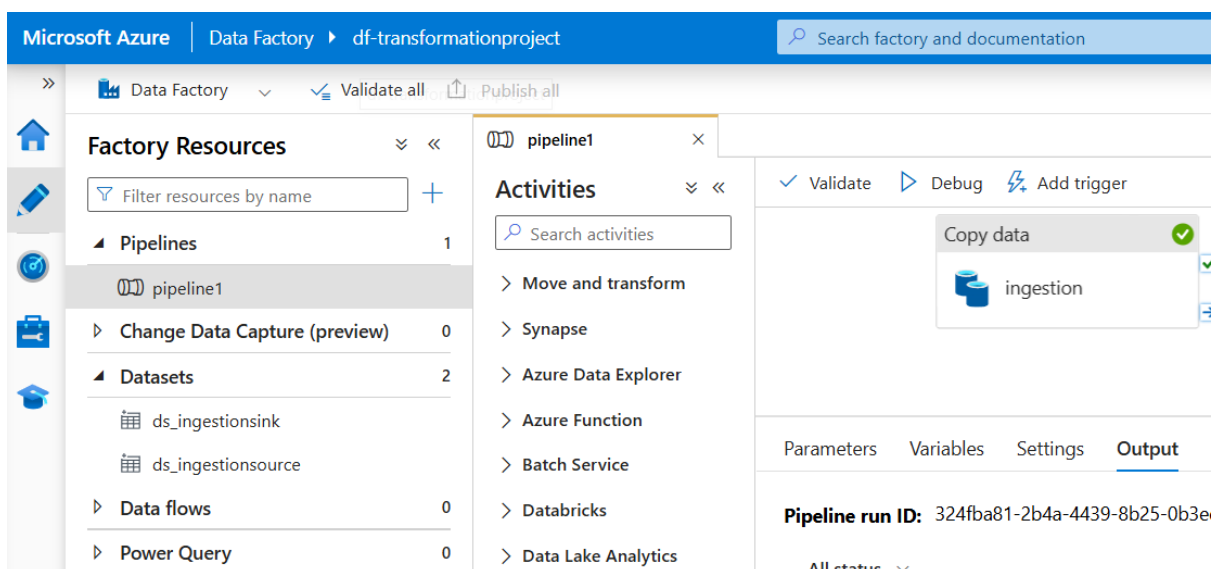
- **Add a Copy Data Activity:**

- Add a Copy Data activity to copy sales_data.csv from the source ingestion storage to the raw-data container in projectstorage997.

- **Configure Dynamic Parameters:**

- Configure dynamic parameters for the source and sink paths, enabling reusability of the pipeline.

Explanation: Copy Data is the initial step that moves raw data into the raw layer for further processing.



Step 3: Setting Up Databricks Notebooks

1. Prepare Databricks Workspaces:

- **Create a Databricks Workspace:**
 - Create a Databricks workspace in Azure.
- **Create Notebooks:**
 - Inside Databricks, create two notebooks:
 - Curated Notebook for intermediate processing.
 - Staging Notebook for final transformations.

2. Notebook for Curated Data:

- **Apply Data Cleaning Activities:**
 - Handle null values using `isnull()`.
 - Convert quantity values to positive using `abs()`.
 - Capitalize column data with `initcap()`.
 - Perform data cleansing with `regexp()`.
 - Convert dates with `to_date()` and `coalesce()` to unify different date formats in transaction dates.
 - Rename columns using `columnrenamed()`.
- **Write Transformed Data:**
 - Write the transformed data back to the curated container in CSV format.

The screenshot shows the Databricks workspace interface. The left sidebar contains navigation options: Workspace, Recents, Catalog, Workflows, Compute, SQL, SQL Editor, Queries, Dashboards, Genie, Alerts, Query History, SQL Warehouses, Data Engineering, Job Runs, Data Ingestion, Delta Live Tables, Machine Learning, and Playground. The main area displays a notebook titled 'curated' with Python code. The code sets up authentication for Azure Databricks using environment variables and then lists files in a specific directory. Below the code, a table shows the results of the file listing operation.

```
application_id="38ef420a-2ea5-4336-9ebe-71ed745351fd"
directory_id="f3e00db4-0a83-491b-bce4-7ba2fa1212ea"
service_credential = dbutils.secrets.get('projectscope', 'service-credential')

spark.conf.set("fs.azure.account.auth.type.projectstorage997.dfs.core.windows.net", "OAuth")
spark.conf.set("fs.azure.account.oauth.provider.type.projectstorage997.dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")
spark.conf.set("fs.azure.account.oauth2.client.id.projectstorage997.dfs.core.windows.net", application_id)
spark.conf.set("fs.azure.account.oauth2.client.secret.projectstorage997.dfs.core.windows.net", service_credential)
spark.conf.set("fs.azure.account.oauth2.client.endpoint.projectstorage997.dfs.core.windows.net", f"https://login.microsoftonline.com/{directory_id}/oauth2/token")
```

2 days ago (1)

```
display(dbutils.fs.ls("abfss://raw-data@projectstorage997.dfs.core.windows.net/"))
```

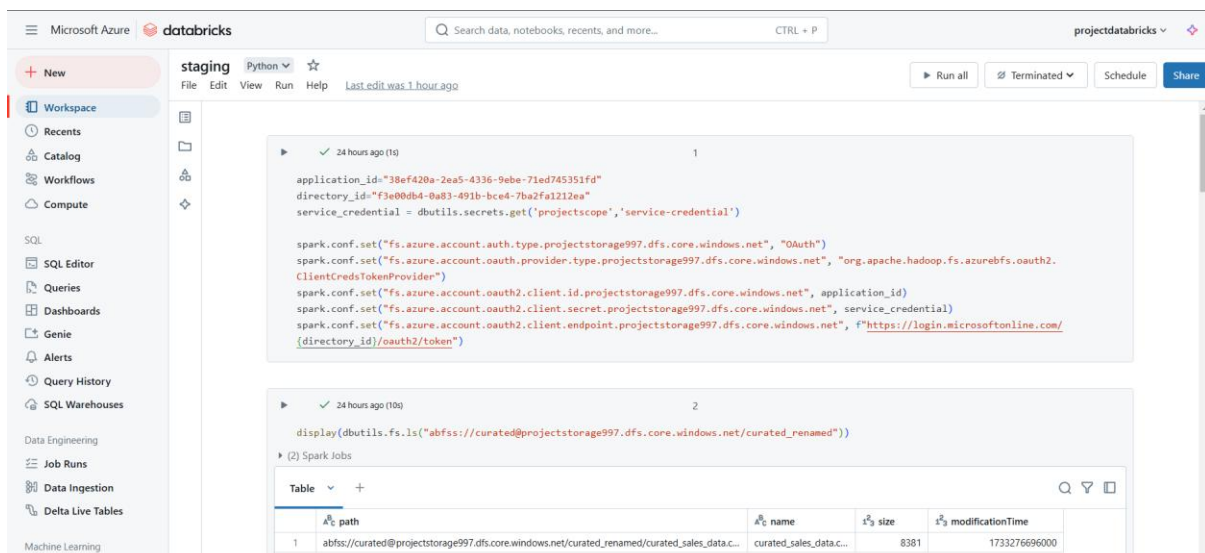
(2) Spark Jobs

Table	path	name	size	modificationTime
1	abfss://raw-data@projectstorage997.dfs.core.windows.net/dat...	data/	0	1733181737000

3. Notebook for Staging Data:

- **Apply Additional Transformations:**
 - Define schema using StructType().
 - Extract year from date fields.
 - Create a new column (e.g., total_quantity).
 - Partition data based on product type for efficient querying.
- **Write Final Output:**
 - Write the final output to the staging container in Parquet format.

Explanation: Each notebook processes the data to refine and enhance its usability as it moves through the Medallion Architecture.



```
application_id="38ef420a-2ea5-4336-9ebe-71ed745351fd"
directory_id="f3e00db4-0a83-491b-bce4-7ba2fa1212ea"
service_credential = dbutils.secrets.get('projectscope', 'service-credential')

spark.conf.set("fs.azure.account.auth.type.projectstorage997.dfs.core.windows.net", "OAuth")
spark.conf.set("fs.azure.account.oauth.provider.type.projectstorage997.dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")
spark.conf.set("fs.azure.account.oauth2.client.id.projectstorage997.dfs.core.windows.net", application_id)
spark.conf.set("fs.azure.account.oauth2.client.secret.projectstorage997.dfs.core.windows.net", service_credential)
spark.conf.set("fs.azure.account.oauth2.client.endpoint.projectstorage997.dfs.core.windows.net", f"https://login.microsoftonline.com/{directory_id}/oauth2/token")
```

```
display(dbutils.fs.ls("abfss://curated@projectstorage997.dfs.core.windows.net/curated_renamed"))
```

	path	name	size	modificationTime
1	abfss://curated@projectstorage997.dfs.core.windows.net/curated_renamed/curated_sales_data.c...	curated_sales_data.c...	8381	1733276696000

Step 4: Pipeline Integration with Databricks

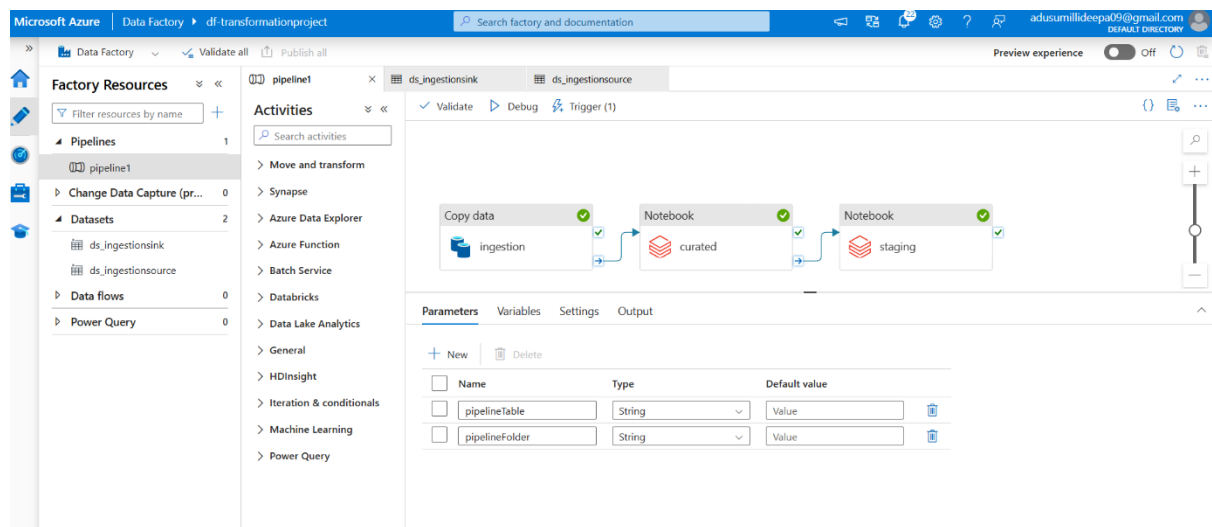
1. Databricks Linked Service:

- **Use Azure Key Vault:**
 - Store credentials (access token and scope details) securely.
- **Create a Databricks Linked Service:**
 - In ADF, reference these secrets to create a Databricks linked service.

2. Pipeline Design:

- **Add Databricks Notebook Activities:**
 - Add two Databricks Notebook activities to the pipeline:
 - One for the curated stage.
 - One for the staging stage.
- **Configure Activities:**
 - Configure the activities to call the respective Databricks notebooks.

Explanation: This step integrates Databricks with ADF, allowing it to execute the transformations defined in the notebooks.



Step 5: Setting Up Azure Key Vault

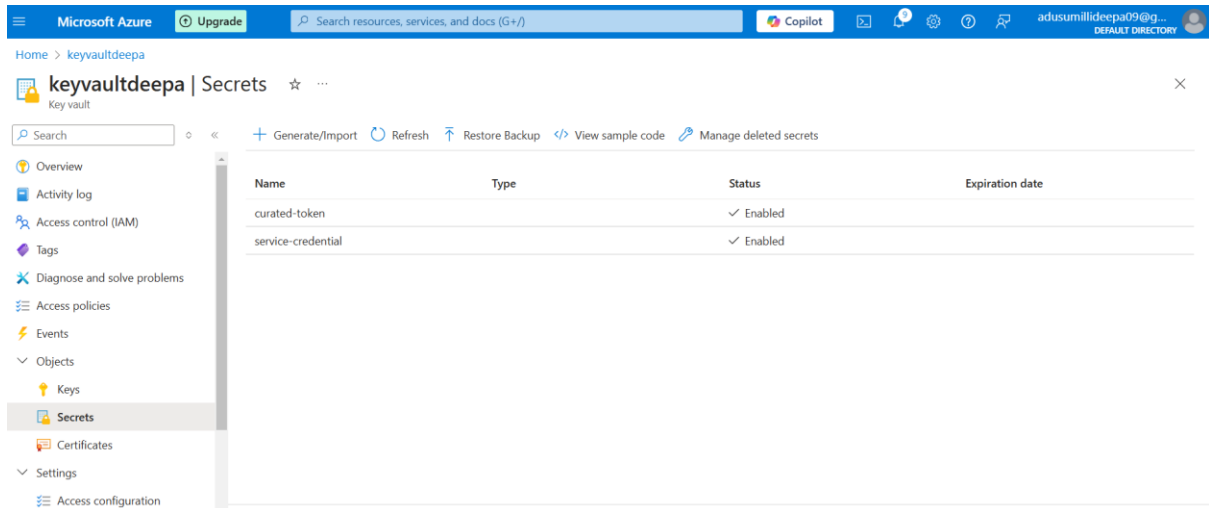
1. Microsoft Entra ID (AAD) Creation:

- **Create an App:**
 - Use app registrations to create an app.
- **Create a Secret:**
 - Create the secret for the app and copy the value.
- **Add a Role Assignment:**
 - Assign the app as a storage Blob contributor and attach it to the storage account (projectstorage997).

2. Key Vault Creation:

- **Create an Azure Key Vault:**

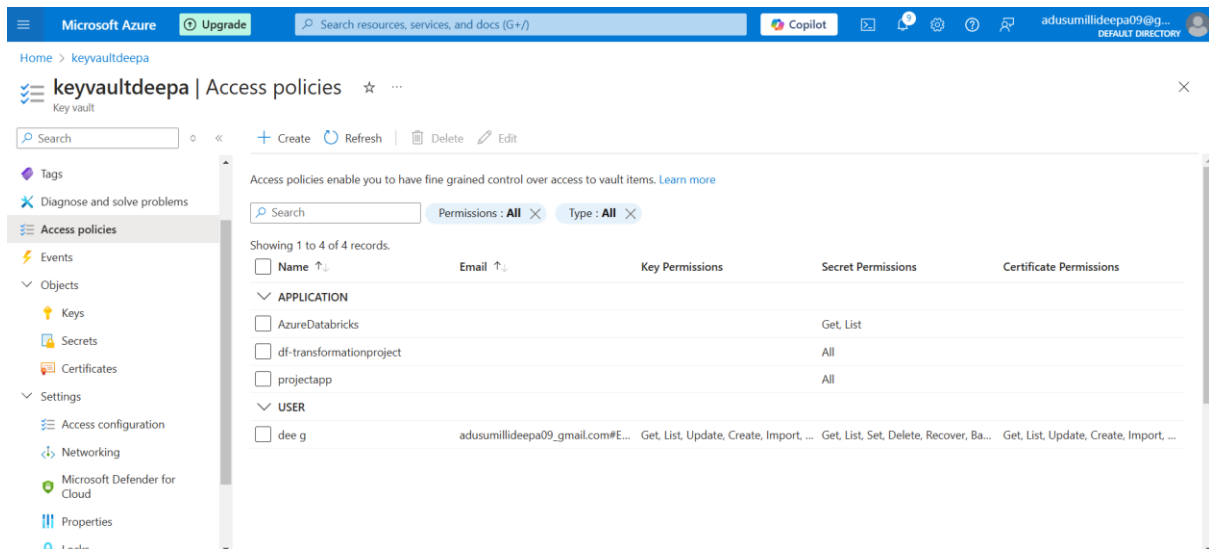
- Create an Azure Key Vault.
- **Add Secrets:**
 - Copy the secret value from the app and create a secret.
 - Add a Databricks access token.



3. Access Policies:

- **Grant Access:**
 - Grant access to:
 - The app to configure the storage account with Databricks.
 - ADF to fetch secrets.
 - Databricks to use these secrets.

Explanation: Key Vault ensures secure and centralized management of sensitive information like credentials and tokens.



Step 6: Implementing Dynamic Parameters

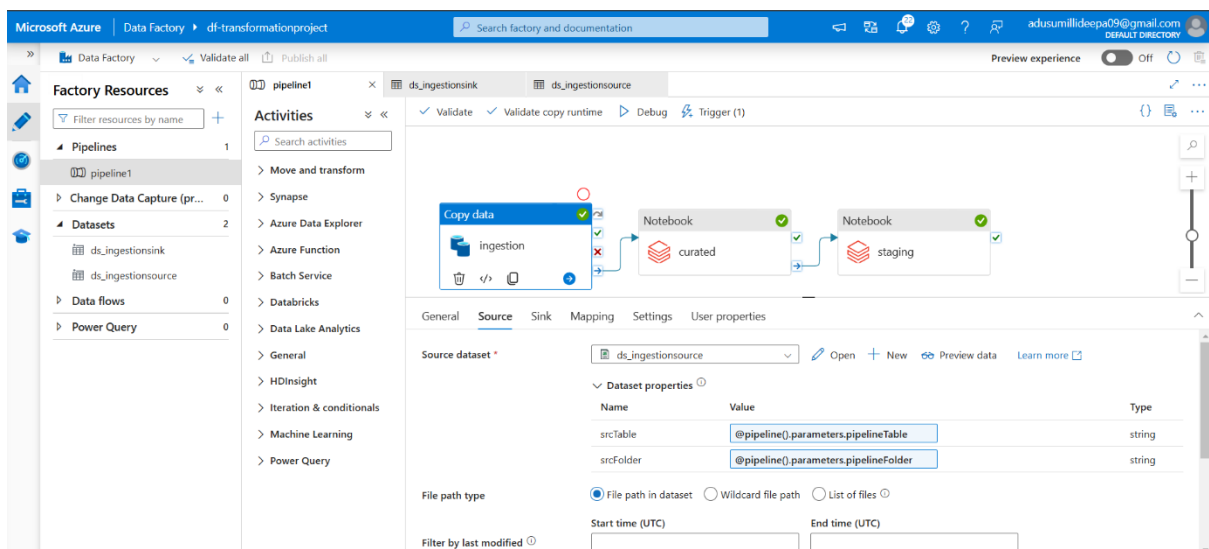
1. Create Parameters in the Pipeline:

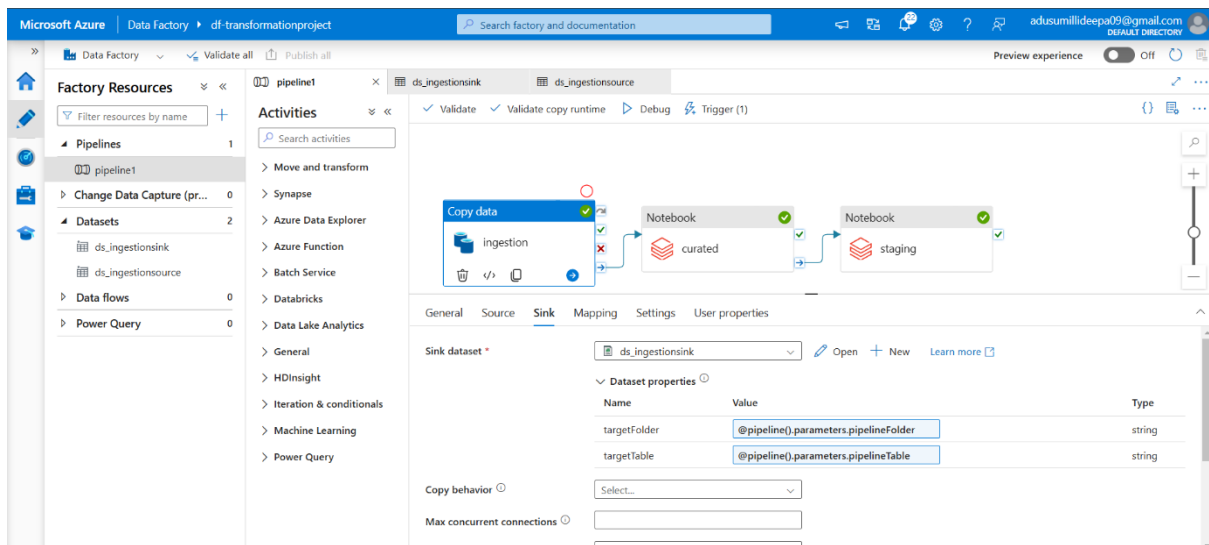
○ Add Parameters:

- Add parameters for dynamic source and sink paths in the Copy Data activity.

○ Use Parameters in Notebooks:

- Use these parameters in the Databricks Notebook activities for flexibility.





2. Parameterize Linked Services:

- **Allow Easy Changes:**

- Use parameterized Linked Services to allow easy changes in storage or Databricks configurations without modifying the pipeline.

Explanation: Dynamic parameters make the pipeline reusable and adaptable to different datasets and configurations.

Step 7: Testing and Validation

1. Run the Pipeline:

- **Test the Pipeline:**

- Trigger the pipeline with the source file (sales_data.csv).

- **Verify Data Flow:**

- Verify that the data flows through the raw, curated, and staging containers as expected.

Microsoft Azure | Data Factory | df-transformationproject

Search factory and documentation

Preview experience Off

Factory Resources

- Pipelines 1
 - pipeline1
- Datasets 2
 - ds_ingestionsink
 - ds_ingestionsource
- Data flows 0
- Power Query 0

Activities

- Move and transform
- Synapse
- Azure Data Explorer
- Azure Function
- Batch Service
- Databricks
- Data Lake Analytics
- General
- HDInsight
- Iteration & conditionals
- Machine Learning
- Power Query

pipeline1

Copy data ingestion Notebook curated Notebook staging

Parameters Variables Settings Output

Pipeline run ID: 324fba81-2b4a-4439-8b25-0b3ed327fc4d Pipeline status: Succeeded

All status

Showing 1 - 3 of 3 items

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime
staging	Succeeded	Notebook	12/3/2024, 8:45:05 PM	19s	AutoResolveIntegrator
curated	Succeeded	Notebook	12/3/2024, 8:40:28 PM	4m 36s	AutoResolveIntegrator
ingestion	Succeeded	Copy data	12/3/2024, 8:40:16 PM	12s	AutoResolveIntegrator

Microsoft Azure | Data Factory | df-transformationproject

Search factory and documentation

Pipeline runs

Triggered Debug Rerun Cancel options Refresh Edit columns List Gantt

Filter by run ID or name Eastern Time (US & C... Last 24 hours Pipeline name: All Status: All Runs: Latest runs

Triggered by: All Add filter

Showing 1 - 9 items

Pipeline name	Run start	Run end	Duration	Triggered by	Status	Run	Parameters
pipeline1	12/4/2024, 9:24:00 PM	12/4/2024, 9:25:06 PM	1m 6s	trigger_ingestion	Succeeded	Original	
pipeline1	12/4/2024, 9:22:00 PM	12/4/2024, 9:23:09 PM	1m 10s	trigger_ingestion	Succeeded	Original	
pipeline1	12/4/2024, 9:20:01 PM	12/4/2024, 9:21:08 PM	1m 8s	trigger_ingestion	Succeeded	Original	
pipeline1	12/4/2024, 9:17:59 PM	12/4/2024, 9:19:08 PM	1m 9s	trigger_ingestion	Succeeded	Original	
pipeline1	12/4/2024, 9:16:01 PM	12/4/2024, 9:17:22 PM	1m 22s	trigger_ingestion	Succeeded	Original	
pipeline1	12/4/2024, 9:14:00 PM	12/4/2024, 9:14:56 PM	56s	trigger_ingestion	Succeeded	Original	
pipeline1	12/4/2024, 9:12:00 PM	12/4/2024, 9:13:21 PM	1m 22s	trigger_ingestion	Failed	Original	
pipeline1	12/4/2024, 9:12:00 PM	12/4/2024, 9:13:22 PM	1m 22s	trigger_ingestion	Succeeded	Original	
pipeline1	12/4/2024, 4:02:56 PM	12/4/2024, 4:06:10 PM	3m 14s	Manual trigger	Canceled	Original	

Microsoft Azure | Data Factory | df-transformationproject

Search factory and documentation

Trigger runs

All Schedule Tumbling window Storage events Custom events Refresh Edit columns

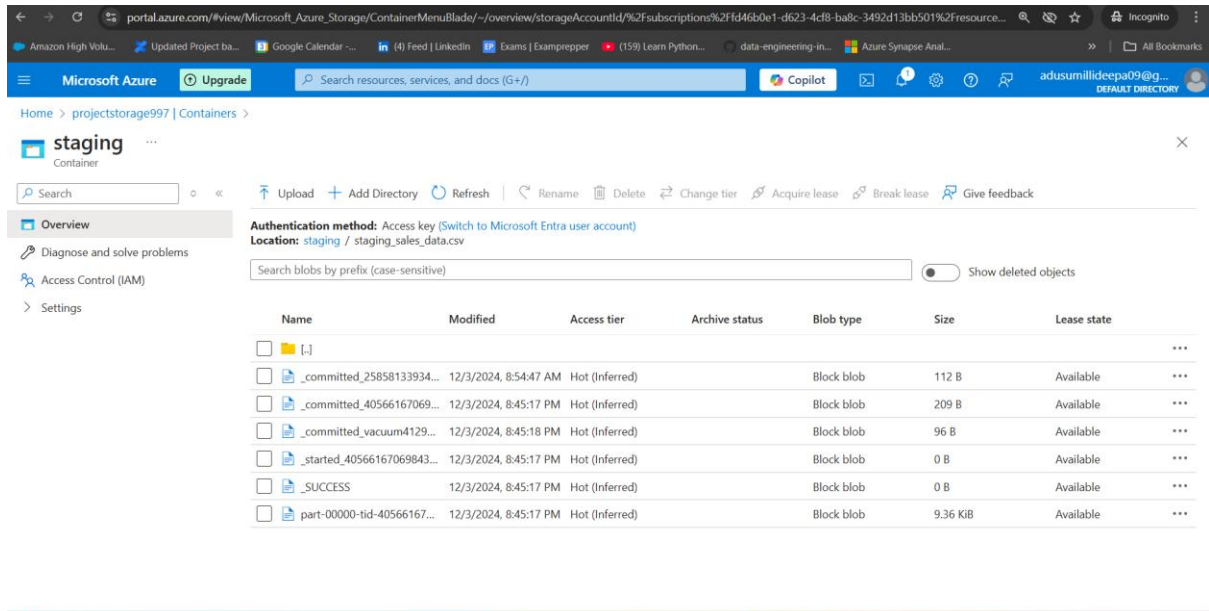
Eastern Time (US & C... Last 24 hours Trigger name: All Status: All

Showing 1 - 8 items

Trigger name	Scheduled time	Trigger time	Status	Pipelines	Message	Properties	Run ID
trigger_ingestion	12/4/2024, 9:24:00 PM	12/4/2024, 9:24:00 PM	Succeeded	1			0858468241445109294f
trigger_ingestion	12/4/2024, 9:22:00 PM	12/4/2024, 9:21:59 PM	Succeeded	1			0858468241565812241f
trigger_ingestion	12/4/2024, 9:20:00 PM	12/4/2024, 9:19:59 PM	Succeeded	1			0858468241685573773f
trigger_ingestion	12/4/2024, 9:18:00 PM	12/4/2024, 9:17:59 PM	Succeeded	1			0858468241805857404f
trigger_ingestion	12/4/2024, 9:16:00 PM	12/4/2024, 9:16:00 PM	Succeeded	1			0858468241925035700f
trigger_ingestion	12/4/2024, 9:14:00 PM	12/4/2024, 9:14:00 PM	Succeeded	1			0858468242045245083f
trigger_ingestion	12/4/2024, 9:11:59 PM	12/4/2024, 9:11:59 PM	Succeeded	1			0858468242165594068f
trigger_ingestion	12/4/2024, 9:11:59 PM	12/4/2024, 9:12:00 PM	Succeeded	1			0858468242165400046f

2. Validate Outputs:

- **Check Curated Container:**
 - Check the curated container for processed CSV files.
- **Check Staging Container:**
 - Check the staging container for Parquet files with applied transformations and partitions.



Documentation of Challenges and Errors

Common Challenges Encountered:

1. Authentication Issues:

- **Problem:** Difficulty in authenticating Linked Services to access Azure Storage or Databricks.
- **Cause:** Incorrect configuration of secrets in Azure Key Vault or missing access policies.

2. Data Movement Errors:

- **Problem:** Failures in the Copy Data activity.
- **Cause:** Incorrect source/sink paths, missing permissions, or data format issues.

3. Transformation Failures:

- **Problem:** Errors in Databricks notebooks during transformations, Mixed date formats

- **Cause:** Syntax errors in notebook code, incorrect data types, or missing columns.

4. Performance Bottlenecks:

- **Problem:** Slow execution of pipelines.
- **Cause:** Large datasets, insufficiently optimized transformations, or network latency.

Error Handling:

1. Logging:

- Ensure all ADF activities and Databricks notebooks are configured to log detailed information about errors.

2. Alerts:

- Set up alerts in ADF to notify pipeline failures.

3. Retries:

- Configure retry policies for critical pipeline activities.

Thorough Troubleshooting Steps

1. Authentication Issues:

- **Verify Secrets Configuration:** Ensure all secrets in Azure Key Vault are correctly configured and accessible.
- **Check Access Policies:** Verify access policies in Azure Key Vault to ensure ADF and Databricks have the necessary permissions.
- **Review Linked Service Credentials:** Ensure the Databricks linked service is using the correct credentials and token.

2. Data Movement Errors:

- **Validate Paths:** Ensure the source and sink paths in the Copy Data activity are correct.
- **Check Permissions:** Verify access permissions on Azure Storage containers.
- **Review Data Format:** Ensure compatibility between source and destination data formats.

3. Transformation Failures:

- **Debug Notebooks:** Run Databricks notebook cells individually to pinpoint errors.
- **Check Schema and Data Types:** Ensure the schema and data types in the source data match the expected format in the notebooks.
- **Verify Column Names:** Ensure all necessary columns are present and correctly named.
- **Mixed date formats:** Standardized with PySpark's coalesce function to handle multiple formats.

4. Performance Bottlenecks:

- **Optimize Transformations:** Minimize data shuffles and use efficient operations in Databricks transformations.
- **Partition Data:** Partition large datasets to parallelize processing.
- **Evaluate Network Performance:** Consider using premium storage tiers for better throughput.

5. General Troubleshooting:

- **Review Logs:** Regularly review pipeline run logs and activity details in ADF.
- **Monitor Databricks:** Utilize Databricks' built-in tools for monitoring and performance tuning.
- **Implement Version Control:** Use version control for Databricks notebooks to track changes and revert if necessary.

Lessons Learned and Future Improvements

1. Lessons Learned:

- **Effective Use of Medallion Architecture:**
 - This project underscored the importance of organizing data flows into layers (Raw → Curated → Staging) to improve data processing and management.
- **Importance of Secure Credential Management:**
 - Leveraging Azure Key Vault ensured the secure handling of sensitive information and streamlined integration with other services.
- **Optimization for Large Datasets:**

- Addressing performance bottlenecks through optimized transformations and efficient data partitioning proved crucial for handling large datasets.

2. Future Improvements:

- **Enhanced Monitoring and Alerting:**
 - Implement more granular monitoring and alerting mechanisms to detect issues early and mitigate potential problems.
- **Scalability Enhancements:**
 - Explore ways to further enhance the scalability of the pipeline to handle larger datasets and more complex transformations.
- **Continuous Integration and Deployment (CI/CD):**
 - Implement CI/CD pipelines for automated testing, validation, and deployment of changes to the ETL process.

Conclusion

This detailed pipeline demonstrates the integration of Azure Data Factory (ADF) and Databricks to implement a robust ETL process based on Medallion Architecture. Through the effective use of Azure Storage Accounts, Databricks notebooks, and Azure Key Vault, the project ensures scalability, flexibility, and secure management of credentials. The pipeline is designed to handle large datasets efficiently, providing a structured workflow for data processing and transformation.

Key Takeaways:

- The Medallion Architecture effectively organizes data flow into layers (Raw → Curated → Staging) for better processing and management.
- Secure credential management using Azure Key Vault ensures the protection of sensitive information and simplifies integration with other services.
- Dynamic parameters and linked services enhance the reusability and adaptability of the pipeline to different datasets and configurations.
- Comprehensive testing, validation, and troubleshooting steps are essential for maintaining the pipeline's reliability and performance.

GitHub repository with code and documentation

You can find the complete code and detailed documentation for this project on GitHub at the following link:

<https://github.com/DeepaManasa/Projects/blob/ADF-pipeline-using-Databricks>

Documentation Link:

<https://github.com/DeepaManasa/Projects/blob/ADF-pipeline-using-Databricks/README.md>

Code Links for Databricks notebooks:

1)Curated Notebook:

<https://github.com/DeepaManasa/Projects/blob/ADF-pipeline-using-Databricks/curated.py>

2)Staging Notebook:

<https://github.com/DeepaManasa/Projects/blob/ADF-pipeline-using-Databricks/staging.py>