# Schedule Databricks Notebook Using JobCluster Through ADF

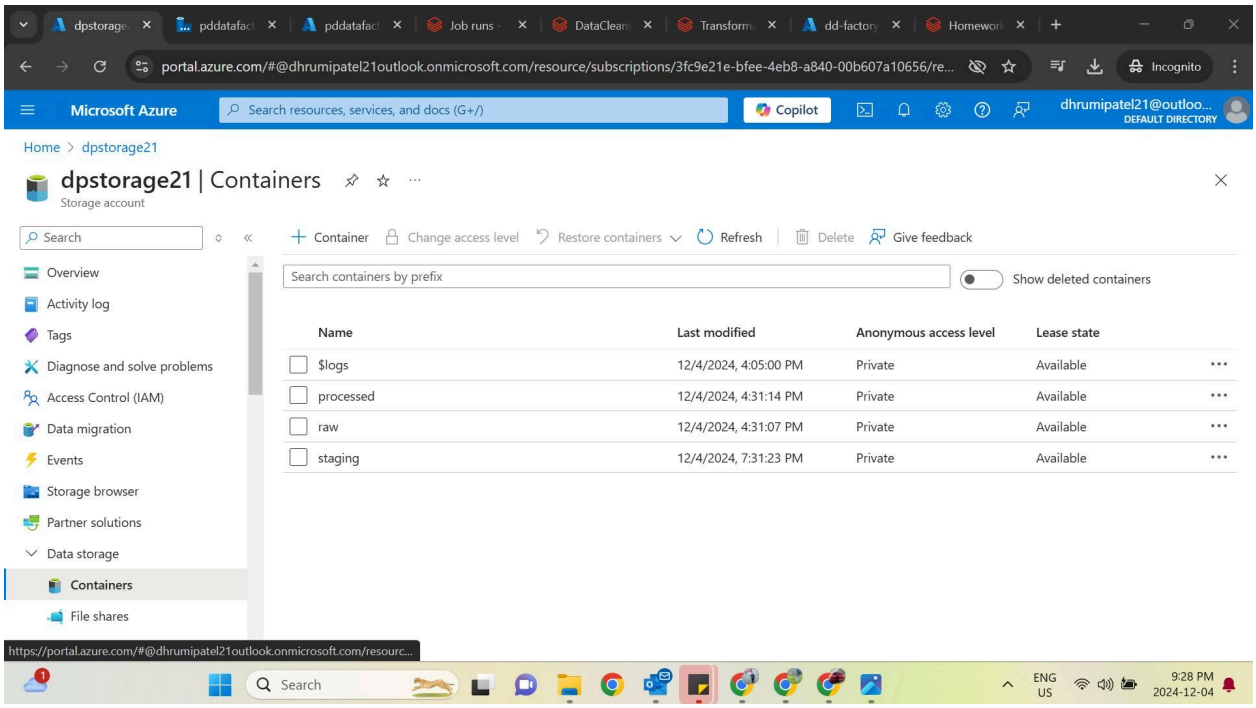
## Project Overview

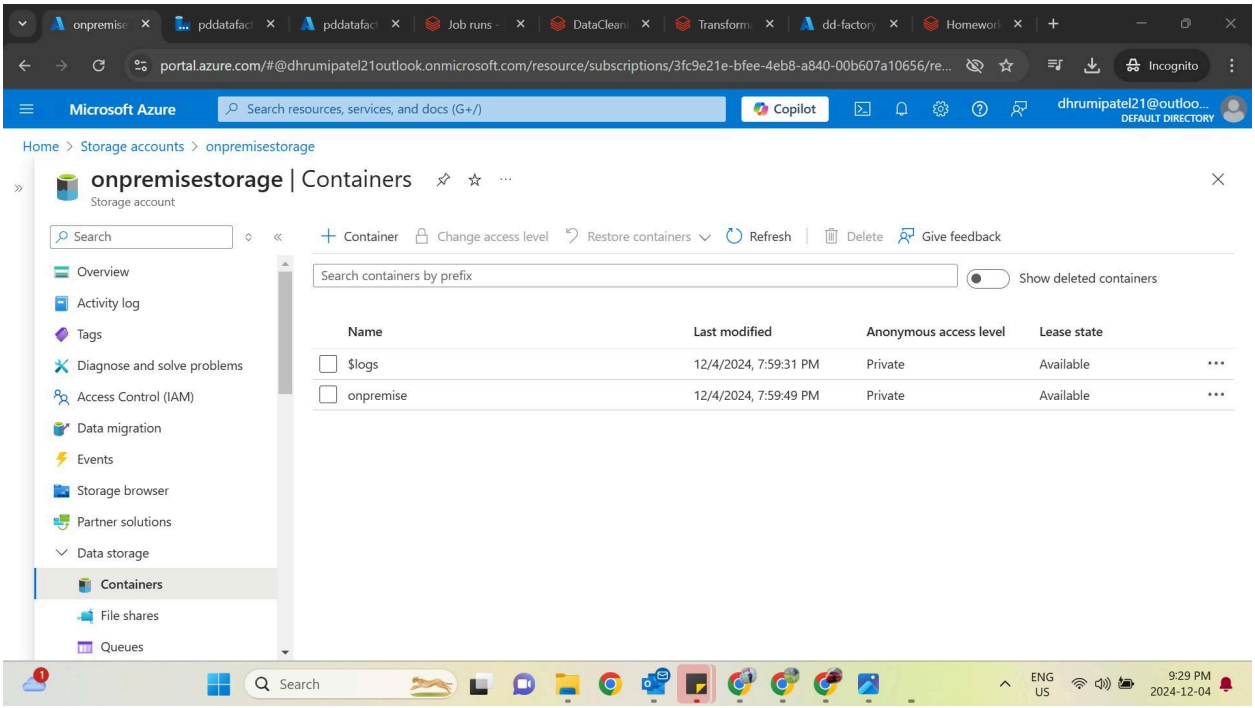
The objective of this project is to automate the execution of a Databricks notebook using a job cluster through Azure Data Factory (ADF). The workflow involves the movement and processing of data between multiple storage accounts and containers, with a focus on data cleaning, transformation, and error handling.

## Workflow Steps

1. **Storage Account Utilization**
   - Utilized two Azure Storage Accounts for data operations:
     - **Source Storage Account:** Data is copied from the source container.
     - **Destination Storage Account:** Processed data is stored here in designated containers.

2. **Data Pipeline Stages**
   - **On Premise Container:** Incoming raw data is transferred to this container.
   - **Raw Container:** Data cleaning and transformation tasks are performed here.
   - **Processed Container:** Finalized and validated data is stored.
   - **Staging Container:** Required data is copied for further analysis and operations.



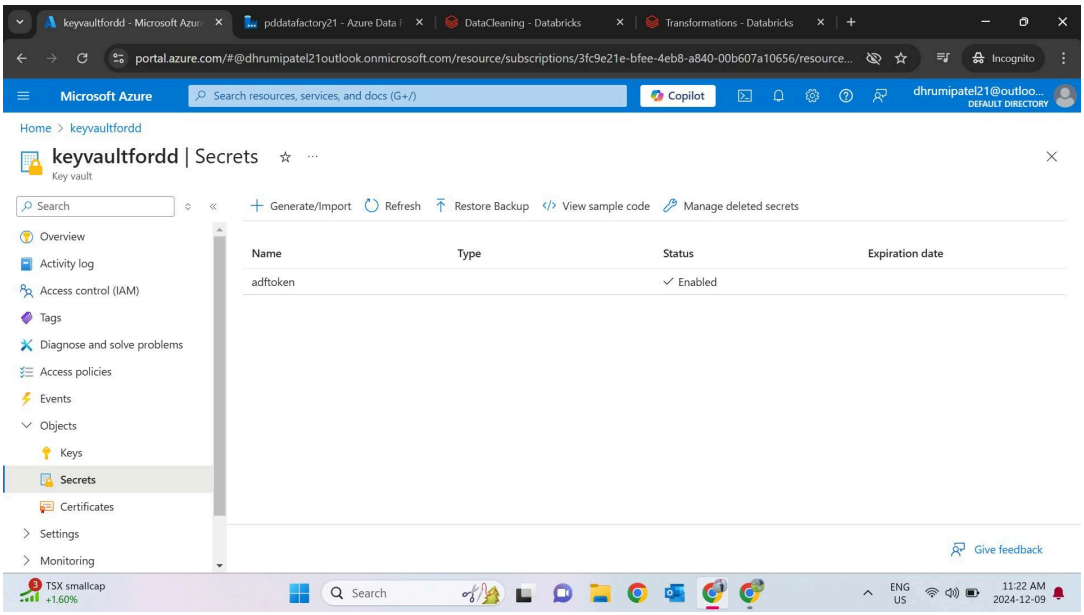



3. **Dynamic Parameterization**
   - Implemented dynamic parameters in Azure Data Factory to enhance the flexibility and scalability of the pipelines.

4. **Data Transformation**
   - **Python Scripting:**
     - Performed data cleaning using Python scripts.
     - Implemented mounting and unmounting of storage containers in Databricks for efficient data access.
   - Data was transformed and organized into the required format.

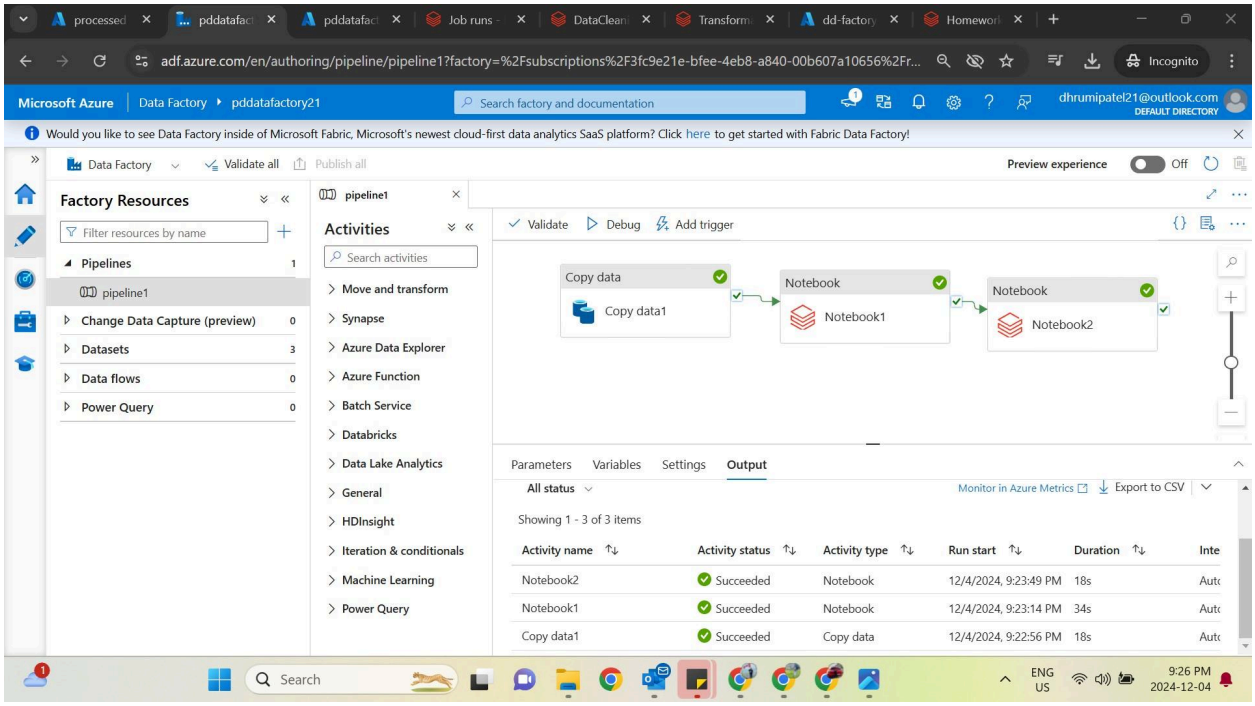
5. **Version Control & Error Logging**
   - Ensured version control of scripts and notebooks using Databricks Git integration.
   - Error logs were implemented at each stage of the pipeline to track and troubleshoot issues efficiently.

## Technical Components

1. **Azure Data Factory (ADF)**
    - Pipeline created for dynamic data movement between storage accounts and containers.
    - Parameters were utilized to dynamically control the file paths and dataset configurations.
2. **Databricks**
    - A Databricks job cluster was scheduled through ADF for running the notebook.
    - Notebook Tasks:
        - Data Mounting
        - Cleaning and Transformation
        - Data Writing to Containers
3. **Key Vault Integration**
    - Integrated Azure Key Vault to securely manage sensitive information such as tokens and credentials.
    - 


## Outputs

- Cleaned and transformed datasets in the final stage container.
- Consolidated data moved to the staging container for further utilization.




## Error Handling

- Configured pipelines to log errors during each phase of execution.
- Implemented retry mechanisms in ADF for transient failures.

## Supporting Files

- **HTML Output:** Data cleaning result files (DataCleaning_outputfile.html).
- **JSON Configuration:** Pipeline definition for ADF ([Json file for pipeline.txt](mnt/data/Json file for pipeline.txt)).
- **Notebook:** Transformations and Mounting tasks (TransformationswithMounting.ipynb).

## Conclusion

This project successfully demonstrates the end-to-end automation of data processing using Azure Data Factory and Databricks, incorporating dynamic parameters, secure token management, and error logging mechanisms.