# Case Study: End-to-End Data Pipeline for COVID-19 Test Data Management

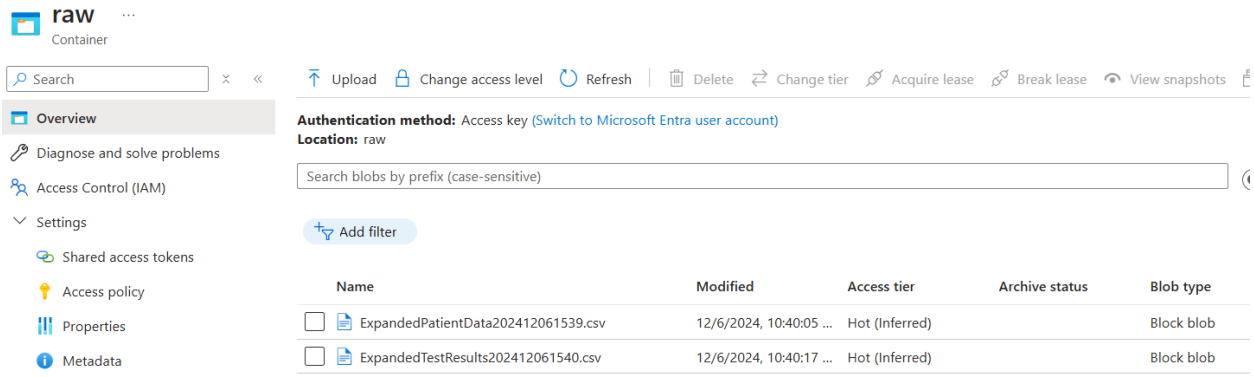
## Architecture

### Data Flow Steps:

1. **Data Storage**:

  - Raw data was stored in Azure Blob Storage (HTTP endpoint).

  - File types included CSVs containing patient and test details.




2. **Data Cleaning and ETL**:

  - Databricks was used to clean and transform the data using PySpark.

  - Key cleaning steps:

    - Removing rows with missing identifiers (e.g., `PatientID`, `TestID`).

    - Standardizing data types (e.g., casting `Age` to integer, `TestDate` to date).

    - Handling null values by replacing them with defaults like 'Unknown'.

  - ETL steps included joining patient and test data, deriving new columns such as `ResultStatus`, and selecting relevant fields for reporting.


3. **Data Loading**:

- Transformed data was saved in Delta format.

- Data was ingested into an Azure Synapse Analytics dedicated SQL pool for reporting and analytics.

---


## Implementation

### 1. Data Storage in Azure Blob Storage

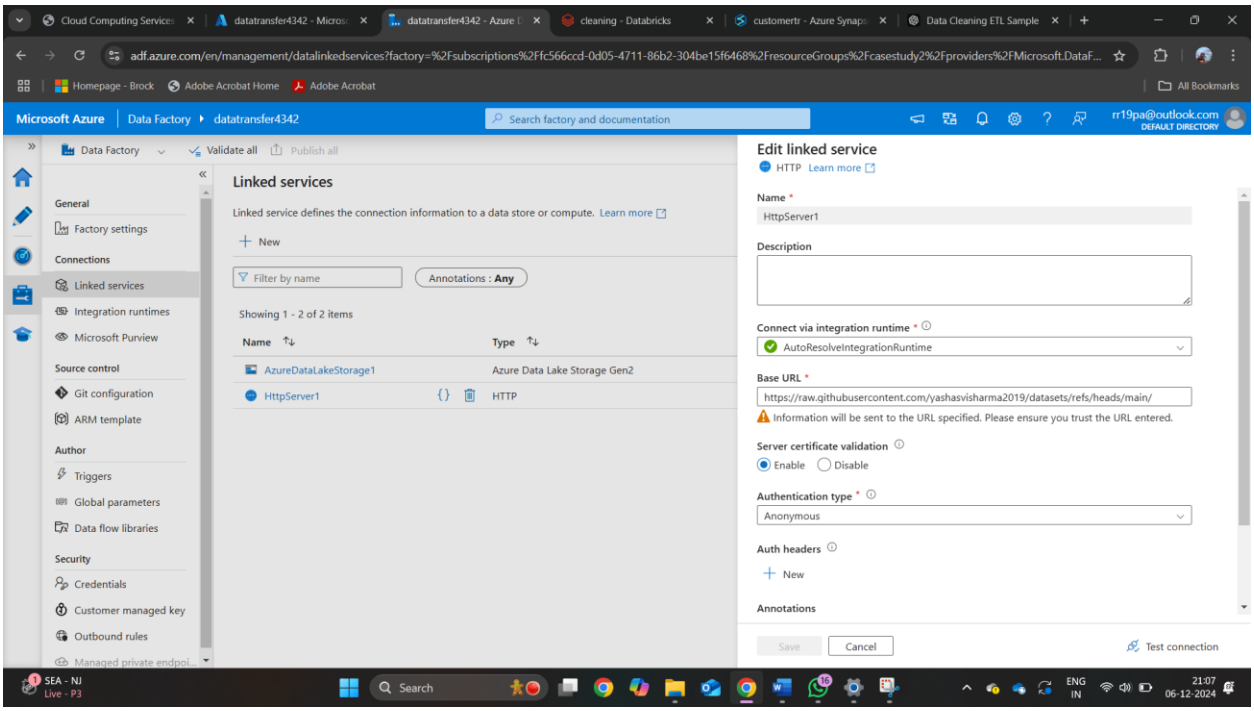
- **Source Data:**

  - Patient data: `ExpandedPatientData.csv`

  - Test results: `ExpandedTestResults.csv`

- **Storage Location:**

  - Data was uploaded to an Azure Blob Storage container via HTTP.




### 2. Data Cleaning and Transformation with Databricks

#### Data Cleaning

- **Patients Data**:

  - Dropped rows with null `PatientID`.

  - Cast `PatientID` and `Age` to `IntegerType` and `TestDate` to `DateType`.

  - Replaced null `COVIDStatus` values with 'Unknown'.

- **Tests Data**:

  - Removed rows with null `TestID` or `PatientID`.

  - Cast `TestID`, `PatientID` to `IntegerType` and `ResultDate` to `DateType`.

- Replaced null `LabName` values with 'Unknown'.

#### Data Transformation

- Joined patient and test data on `PatientID`.
- Added a derived column `ResultStatus` based on test results ('Positive' or 'Negative').
- Selected and renamed columns for analytics.

#### Error Handling

- Missing columns or unresolved variables were handled with validation steps and exceptions.
- Write operations to storage were encapsulated in try-except blocks to capture and log errors.

### 3. Data Loading to Synapse Analytics

- **Output Storage**: Transformed data was saved in Delta format in Azure Data Lake.
- **Synapse Pipeline**:
  - Data from Delta tables was ingested into an Azure Synapse Analytics dedicated SQL pool.

---

## Results

- Improved data quality by resolving inconsistencies and handling missing values.
- Achieved a centralized, optimized SQL-based data store for reporting and analytics.
- Enabled real-time insights for healthcare decision-making.

---

## Conclusion

This end-to-end pipeline effectively addressed the challenges of managing raw COVID-19 test data. The integration of Azure Blob Storage, Databricks, and Synapse Analytics streamlined the process from data ingestion to reporting. The scalable architecture supports future enhancements for additional datasets or analytics requirements.

## Future Work

- Automate the entire pipeline using Azure Data Factory.

- Implement advanced analytics on Synapse using machine learning models.