Lab 9: Reading Data from Azure Data Lake Storage (ADLS) into Databricks Without Service Principal

Author: Dr. Sandeep Kumar Sharma

Learning Objective

In this lab, you will learn how to connect Azure Databricks with Azure Data Lake Storage (ADLS) using the simplest possible method (without service principals, without key vaults). You will read a file from ADLS, load it into a Spark DataFrame, and display it in Databricks.

Learning Outcome

After completing this lab, you will be able to: - Understand how Databricks interacts with external storage accounts - Access ADLS using a **direct storage account access key** - Mount ADLS to DBFS (basic, non-production method) - Read a CSV file stored in ADLS - Display the data inside Databricks

> ⚠️**Important Note (for students)**:
> This method is only for learning and POC purposes. In production, you must use Service Principal + OAuth, Managed Identity, or Unity Catalog.

Lab Scenario

Let's imagine: - Your customer data is collected through ADF pipelines. - All the raw data is stored inside an ADLS Gen2 storage account. - You want to pull that data from ADLS into Databricks, create a DataFrame, and start transformations.

For example, assume the file in ADLS is:

```
https://yourstorageaccount.dfs.core.windows.net/raw/employee.csv
```

We will mount this storage and read the employee.csv file.

Step-by-Step Lab

Step 1: Required Information

Before starting, collect the following details from Azure Portal: - **Storage Account Name** → e.g., `yourstorageaccount` - **Container Name** → e.g., `raw` - **File Path** → `employee.csv` - **Access Key:** Storage Account → Access Keys → Key1

Now construct your ADLS path:

```
adls_path = "wasbs://raw@yourstorageaccount.blob.core.windows.net/employee.csv"
```

Step 2: Configure Spark to Use Access Key

We will tell Spark how to authenticate using the storage account key.

```
spark.conf.set(
    "fs.azure.account.key.yourstorageaccount.blob.core.windows.net",
    "<YOUR_STORAGE_ACCOUNT_KEY>"
)
```

Replace `<YOUR_STORAGE_ACCOUNT_KEY>` with the actual key.

This gives Databricks direct access to the storage account.

Step 3: Read the CSV File from ADLS

Now read the file directly from ADLS into a DataFrame.

```
adls_csv_path = "wasbs://raw@yourstorageaccount.blob.core.windows.net/
employee.csv"

adls_df = spark.read.csv(adls_csv_path, header=True, inferSchema=True)
```

Step 4: Display the DataFrame

```
display(adls_df)
```

This will show the content of the employee.csv file inside Databricks.

Step 5: (Optional) Mount ADLS to DBFS

This step makes ADLS appear like a DBFS folder.

Step 5.1: Create a Mount Point

```
mount_point = "/mnt/adlsraw"
```

Step 5.2: Mount the ADLS Container

```
dbutils.fs.mount(
    source = "wasbs://raw@yourstorageaccount.blob.core.windows.net",
    mount_point = mount_point,
    extra_configs = {
        "fs.azure.account.key.yourstorageaccount.blob.core.windows.net":
"<YOUR_STORAGE_ACCOUNT_KEY>"
    }
)
```

Mounting creates a permanent link.

Step 6: Verify Mount

```
display(dbutils.fs.ls("/mnt/adlsraw"))
```

You should now see `employee.csv` inside this mounted folder.

Step 7: Read the File from the Mount Instead of the URL

```
mount_df = spark.read.csv("/mnt/adlsraw/employee.csv", header=True,
inferSchema=True)
display(mount_df)
```

This is simpler and preferred for notebooks.

Explanation

- Databricks normally reads from DBFS, but in real production workloads, data lives in ADLS.
- Accessing ADLS requires authentication. Here we used **storage account access key**, the simplest method.
- Spark understands `wasbs://` path and loads the file directly.
- Mounting ADLS makes your external storage look like DBFS.
- After mounting, reading becomes as easy as reading local files.

End of Lab 9

In this lab, you successfully: - Connected Databricks to ADLS using storage account access key - Read a CSV file stored in ADLS - Loaded the data into a DataFrame - Displayed it in Databricks - Mounted ADLS and accessed files through DBFS

Next we can build: **Lab 10 – Read/Write Data Back to ADLS After Transformation** or
Lab 11 – Using SAS Token Authentication with ADLS.