**Connecting Azure Data Factory (ADF) with Azure Databricks**

**Objective**

* Learn how to integrate ADF with Databricks for data pipelines.
* Move raw data from **Azure Data Lake Storage Gen2 (ADLS)**, process it with Databricks, and save results back.

**Pre-requisites**

1. Azure Subscription.
2. Resource Group (e.g., rg-adf-dbx-lab).
3. Services:
   * **Azure Data Factory** (ADF v2).
   * **Azure Databricks Workspace**.
   * **Azure Data Lake Storage Gen2** (with hierarchical namespace enabled).
4. Datasets for testing:
   * orders.csv (raw transactional data).
   * products.json (product reference data).
   * customers.csv (customer master data).

**Step 1 – Setup Storage & Upload Data**

1. Create an **ADLS Gen2 storage account** → container: raw.
2. Upload sample datasets:

**orders.csv**

OrderID,CustomerID,ProductID,Quantity,OrderDate

1001,C001,P001,2,2025-01-10

1002,C002,P002,1,2025-01-12

1003,C001,P003,4,2025-01-15

**products.json**

[

{"ProductID":"P001","ProductName":"Laptop","Category":"Electronics"},

{"ProductID":"P002","ProductName":"Chair","Category":"Furniture"},

{"ProductID":"P003","ProductName":"Headphones","Category":"Electronics"}

]

**customers.csv**

CustomerID,CustomerName,Region

C001,John Smith,North

C002,Sarah Lee,West

**Step 2 – Provision Databricks & Notebook**

1. In Azure Portal → Create **Azure Databricks workspace**.
2. Launch workspace → Create **cluster** (can use small autoscaling cluster).
3. In **Workspace → Notebooks**, create a new notebook ProcessOrders.

Add PySpark code:

from pyspark.sql.functions import col

# Paths

raw\_path = "abfss://raw@<storageaccount>.dfs.core.windows.net/"

output\_path = "abfss://curated@<storageaccount>.dfs.core.windows.net/"

# Load datasets

orders = spark.read.option("header", True).csv(raw\_path + "orders.csv")

customers = spark.read.option("header", True).csv(raw\_path + "customers.csv")

products = spark.read.json(raw\_path + "products.json")

# Join orders with customer and product info

result = (orders

.join(customers, "CustomerID")

.join(products, "ProductID")

.withColumn("Quantity", col("Quantity").cast("int")))

# Aggregate sales per region

sales\_by\_region = result.groupBy("Region").sum("Quantity")

# Write output

sales\_by\_region.write.mode("overwrite").parquet(output\_path + "sales\_by\_region")

Save & test inside Databricks.

**Step 3 – Configure Linked Services in ADF**

1. Go to **ADF Studio** → Manage → Linked Services.
2. Add:
   * **Azure Data Lake Storage Gen2** (use account key or managed identity).
   * **Azure Databricks** (provide workspace URL, choose Interactive Cluster or Job Cluster).

**Step 4 – Create Datasets in ADF**

* Dataset 1: **Orders** → points to orders.csv in ADLS.
* Dataset 2: **Customers** → points to customers.csv.
* Dataset 3: **Products** → points to products.json.
* Dataset 4: **SalesByRegion** → points to curated/sales\_by\_region (parquet output).

**Step 5 – Create Pipeline in ADF**

1. Create new pipeline → Add activity **Notebook**.
   * Link to **Databricks Linked Service**.
   * Select notebook ProcessOrders.
   * Optionally pass parameters (e.g., input path, output path).
2. (Optional) Add **Copy Activity** before Databricks to ingest data into raw container from Blob/SQL.
3. Add a **Data Flow or Copy Activity** after Databricks to load processed parquet into SQL Database/Synapse.

**Step 6 – Trigger & Monitor**

1. Debug pipeline to test end-to-end.
2. Publish & set **schedule trigger** (e.g., daily at 1 AM).
3. Monitor run in **ADF → Monitor** tab.
4. Verify output parquet in curated/sales\_by\_region.

**Expected Output**

sales\_by\_region parquet dataset:

| **Region** | **sum(Quantity)** |
| --- | --- |
| North | 6 |
| West | 1 |