

Data Engineering - Project 5

Migrating pipelines from ADF to Synapse & Databricks Notebook Migration to Fabric Workspace

Table of Contents

Flow Diagram	2
Method 1: Migration Using Azure PowerShell (ADF to Synapse)	2
Objective	2
Step 1: Prepare Synapse Environment	2
Step 2: Export ADF Pipeline	4
Step 3: Set PowerShell Environment	6
Step 4: Import ADF Artifacts into Synapse	7
Step 5: Validate and Run Pipeline in Synapse	8
Points to remember:	11
Conclusion:	11
Method 2: Migration Using JSON Code (ADF to Synapse)	12
Objective	12
Step 1: Validate Pipeline in Azure Data Factory	12
Step 2: Export JSON Definitions from ADF	12
Step 3: Open Azure Synapse Workspace	13
Step 4: Recreate Linked Services in Synapse	14
Step 5: Recreate Data Flow	15
Step 6: Test the Migrated Pipeline in Synapse	16
Points to Remember:	16
Conclusion:	16
Migrate Databricks Notebook to Fabric Workspace	17
Objective	17
Notebook 1	17
Notebook 2	20
Notebook 3	24
Notebook 4	25
Key Modifications Required	29
Conclusion	29

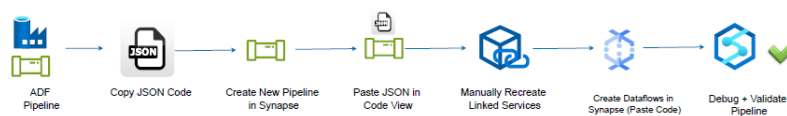
Flow Diagram

Project 5
Flow Diagram:

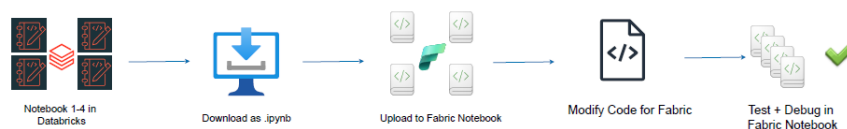
Method 1: ADF to Synapse using PowerShell + JSON Scripts



Method 2: ADF to Synapse using JSON Canvas Code



Databricks Notebooks (4) to Fabric Workspace



Method 1: Migration Using Azure PowerShell (ADF to Synapse)

Objective

The goal of this exercise is to migrate an Azure Data Factory (ADF) pipeline, along with its linked services and datasets, into an Azure Synapse Analytics workspace using a manual scripting approach. This ensures a deeper understanding of artifact structure, module usage, and workspace integration.

Tools used: Azure Cloud Shell, Az.Synapse PowerShell module, Synapse Studio

Step 1: Prepare Synapse Environment

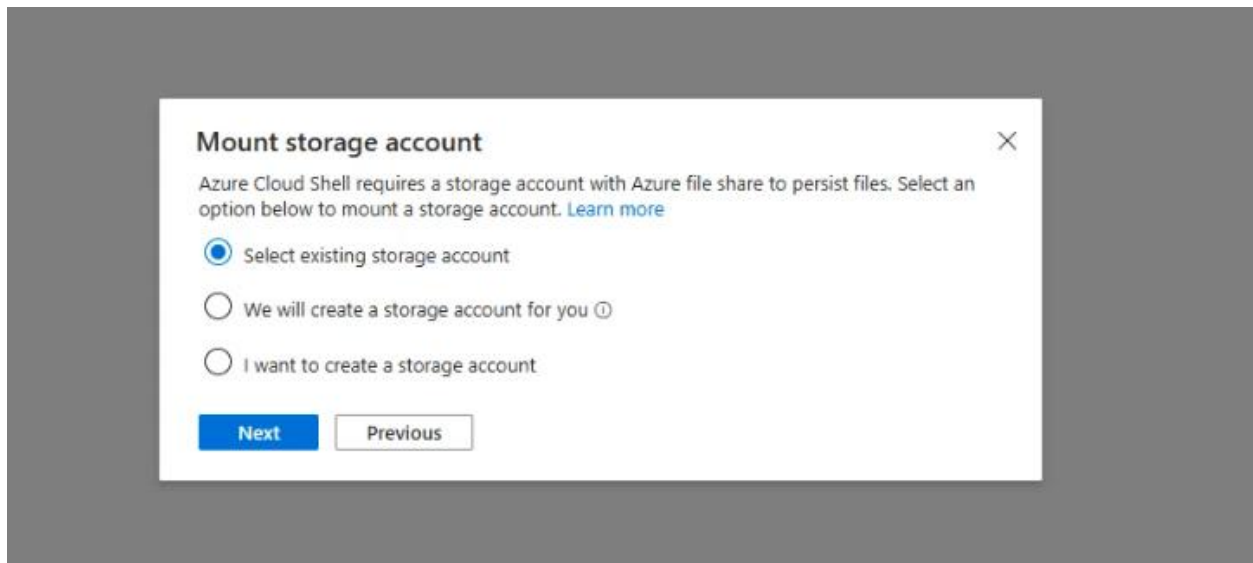
Step 1.1: Launch Cloud Shell

Go to the Azure Portal

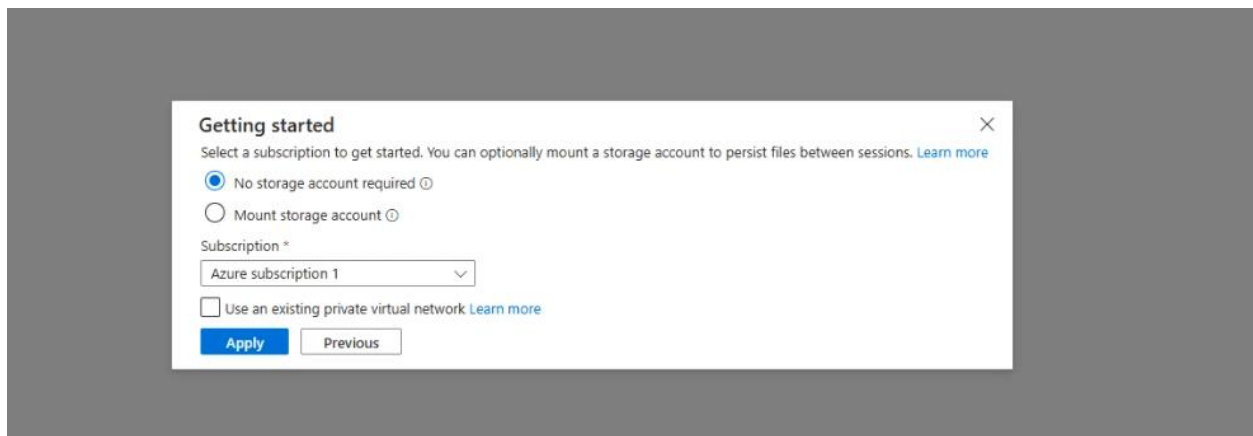
Click on the Cloud Shell icon on the top right

Choose PowerShell mode

Step 1.2: Mount Storage (if prompted)

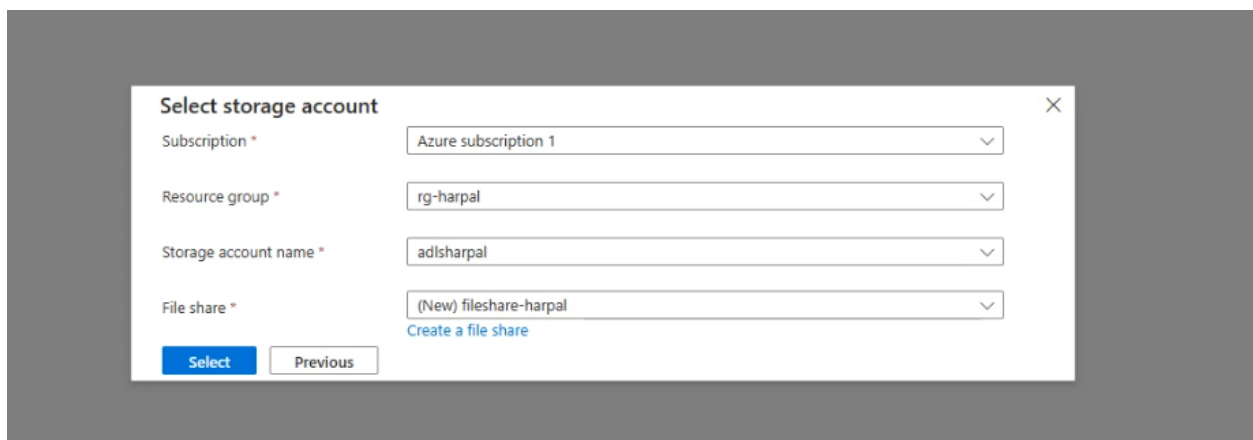


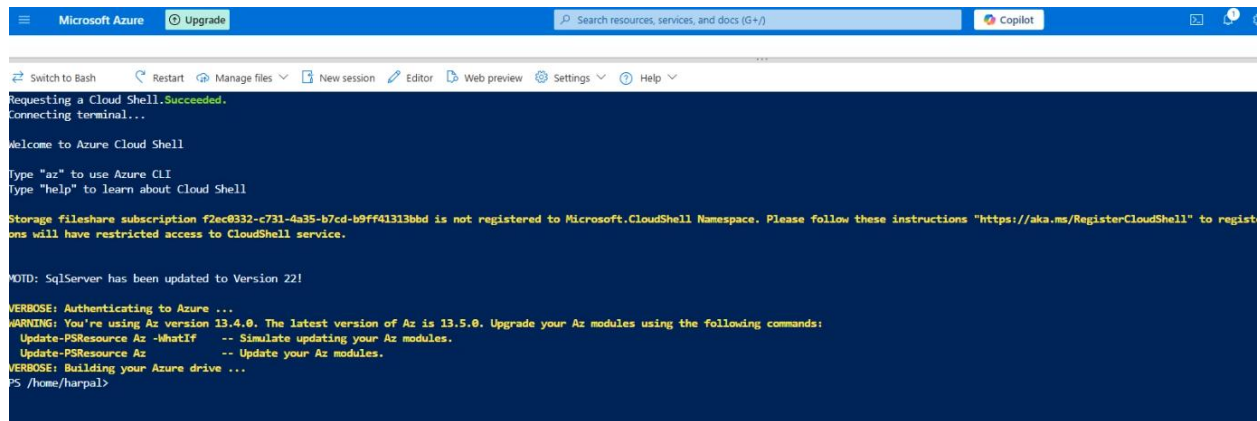
Select your Azure subscription



Create a new storage account or use an existing (e.g., adlsharpal)

Create a file share (e.g., fileshare-harpal)





```
Microsoft Azure Upgrade Search resources, services, and docs (G+) Copilot

Switch to Bash Restart Manage files New session Editor Web preview Settings Help

Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

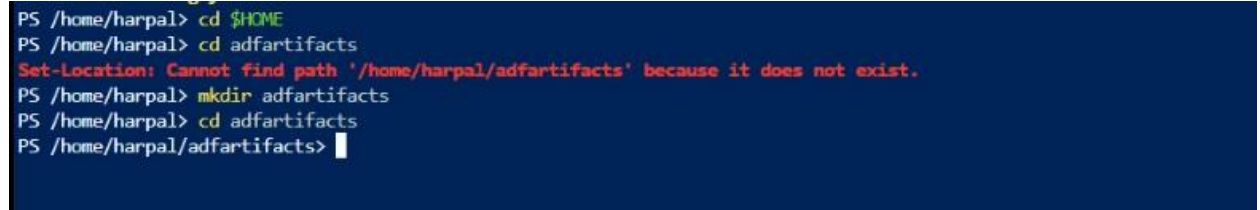
Storage fileshare subscription f2ec8332-c731-4a35-b7cd-b9ff41313bbd is not registered to Microsoft.CloudShell Namespace. Please follow these instructions "https://aka.ms/RegisterCloudShell" to register. Users will have restricted access to CloudShell service.

MOID: SqlServer has been updated to Version 221

VERBOSE: Authenticating to Azure ...
WARNING: You're using Az version 13.4.0. The latest version of Az is 13.5.0. Upgrade your Az modules using the following commands:
  Update-PSResource Az -WhatIf -- Simulate updating your Az modules.
  Update-PSResource Az -- Update your Az modules.
VERBOSE: Building your Azure drive ...
PS /home/harpal>
```

Step 1.3: Set Up Folder Structure in Cloud Shell

```
cd $HOME
mkdir adfartifacts
cd adfartifacts
```



```
PS /home/harpal> cd $HOME
PS /home/harpal> cd adfartifacts
Set-Location: Cannot find path '/home/harpal/adfartifacts' because it does not exist.
PS /home/harpal> mkdir adfartifacts
PS /home/harpal> cd adfartifacts
PS /home/harpal/adfartifacts>
```

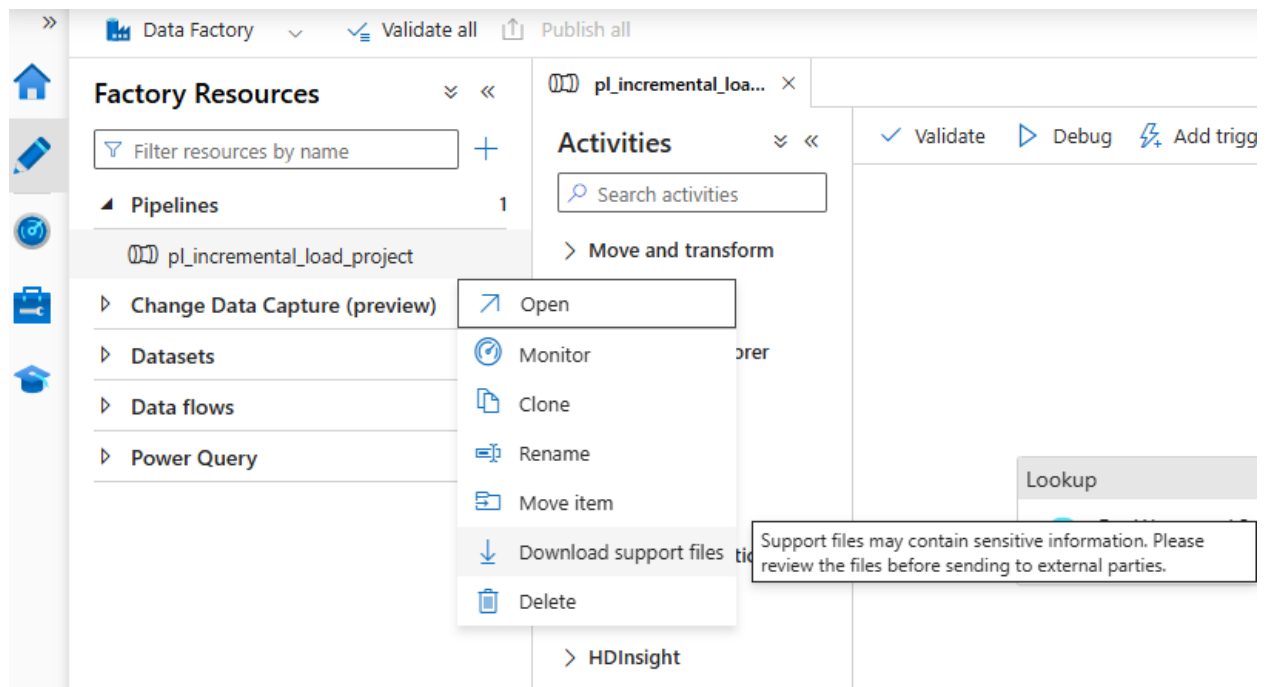
Step 2: Export ADF Pipeline

Step 2.1: Open ADF Studio

Go to your ADF resource > Author & Monitor

Locate pipeline pl_incremental_load_project

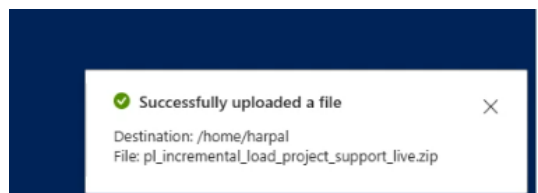
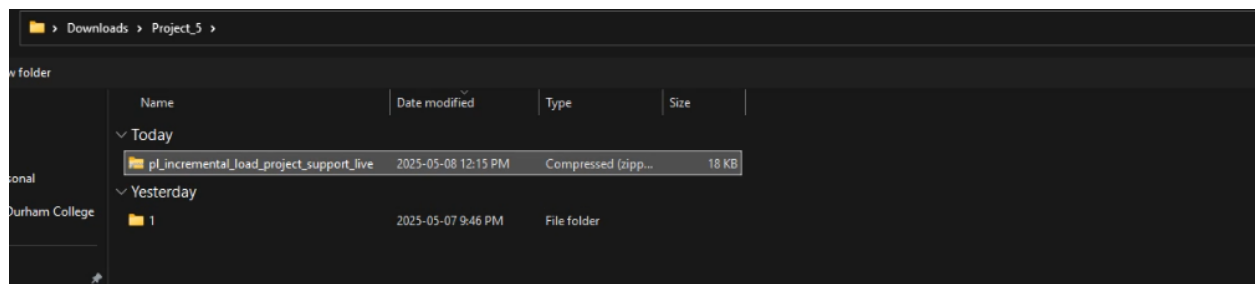
Right-click > Download support files



Step 2.2: Upload ZIP to Cloud Shell

Use Cloud Shell → Upload option

Upload the .zip file (e.g., pl_incremental_load_project_support_live.zip)



Moved that Zip file into newly created directory (adfartifacts)

```

PS /home/harpal/adfartifacts> ls
PS /home/harpal/adfartifacts> cd ~
PS /home/harpal> ls
adfartifacts clouddrive Microsoft pl_incremental_load_project_support_live.zip
PS /home/harpal> mv ./pl_incremental_load_project_support_live.zip ./adfartifacts/
PS /home/harpal> cd ./adfartifacts/
PS /home/harpal/adfartifacts> ls
pl_incremental_load_project_support_live.zip
PS /home/harpal/adfartifacts>

```

Step 2.3: Extract the ZIP

Expand-Archive "pl_incremental_load_project_support_live.zip" -DestinationPath "." -Force

```

ls
PS /home/harpal/adfartifacts> Expand-Archive "pl_incremental_load_project_support_live.zip" -DestinationPath "." -Force
PS /home/harpal/adfartifacts> ls
dataset diagnostic.json info.txt linkedService pipeline pl_incremental_load_project_support_live.zip

```

Step 3: Set PowerShell Environment

Step 3.1: Connect and Set Subscription

Connect-AzAccount

\$context = Select-AzSubscription -SubscriptionName "Azure subscription 1"

```

PS /home/harpal/adfartifacts> $context = Get-AzContext
PS /home/harpal/adfartifacts> $context

Tenant: b9b0d8ab-2a0a-4453-a4f5-34f2ed2382fd

SubscriptionName SubscriptionId Account Environment
-----
Azure subscription 1 f2ec0332-c731-4a35-b7cd-b9ff41313bbd MSI@50342 AzureCloud

```

Step 3.2: Import Az.Synapse Module

Install-Module Az.Synapse -Force

```

PS /home/harpal/adfartifacts>
PS /home/harpal/adfartifacts>
PS /home/harpal/adfartifacts> Install-Module -Name Az.Synapse -Force -AllowClobber

```

```

-----
PS /home/harpal/adfartifacts>
PS /home/harpal/adfartifacts>
Installing package 'Az.Synapse' [Installing dependent package 'Az.Accounts']

```

Import-Module Az.Synapse

```
PS /home/harpal/adfartifacts> Import-Module Az.Synapse
PS /home/harpal/adfartifacts> $workspace = "wsp-synapse-harpal"
```

Step 4: Import ADF Artifacts into Synapse

Set workspace name:

```
$workspace = "wsp-synapse-harpal"
```

```
PS /home/harpal/adfartifacts> Import-Module Az.Synapse
PS /home/harpal/adfartifacts> $workspace = "wsp-synapse-harpal"
```

Step 4.1: Import Linked Services

```
Set-AzSynapseLinkedService -WorkspaceName $workspace -Name azuresqllookup -DefinitionFile
"./linkedService/azuresqllookup.json" -DefaultProfile $context
```

```
Set-AzSynapseLinkedService -WorkspaceName $workspace -Name ls_csvfiles -DefinitionFile
"./linkedService/ls_csvfiles.json" -DefaultProfile $context
```

```
Set-AzSynapseLinkedService -WorkspaceName $workspace -Name ls_key_vault -DefinitionFile
"./linkedService/ls_key_vault.json" -DefaultProfile $context
```

Step 4.2: Import Datasets

```
Set-AzSynapseDataset -WorkspaceName $workspace -Name azuresqllookup -DefinitionFile
"./dataset/azuresqllookup.json" -DefaultProfile $context
```

```
Set-AzSynapseDataset -WorkspaceName $workspace -Name ds_csvfiles -DefinitionFile
"./dataset/ds_csvfiles.json" -DefaultProfile $context
```

```

PS /home/harpal/adfartifacts> Set-AzSynapseLinkedService -WorkspaceName $workspace -Name ls_csvfiles -DefinitionFile ".\linkedService\ls_csvfiles.json" -DefaultProfile $context

WorkspaceName : wsp-synapse-harpal
Properties     : Azure.Analytics.Synapse.Artifacts.Models.AzureBlobFSLinkedService
Id            : /subscriptions/f2ec0332-c731-4a35-b7cd-b9ff41313bbd/resourceGroups/rg-harpal/providers/Microsoft.Synapse/workspaces/wsp-synapse-harpal/linkedServices/ls_csvfiles
Name          : ls_csvfiles
Type          : Microsoft.Synapse/workspaces/linkedServices
Etag          : 21027d8b-0000-0a00-0000-681cde90000

PS /home/harpal/adfartifacts> Set-AzSynapseLinkedService -WorkspaceName $workspace -Name ls_key_vault -DefinitionFile ".\linkedService\ls_key_vault.json" -DefaultProfile $context

WorkspaceName : wsp-synapse-harpal
Properties     : Azure.Analytics.Synapse.Artifacts.Models.AzureKeyVaultLinkedService
Id            : /subscriptions/f2ec0332-c731-4a35-b7cd-b9ff41313bbd/resourceGroups/rg-harpal/providers/Microsoft.Synapse/workspaces/wsp-synapse-harpal/linkedServices/ls_key_vault
Name          : ls_key_vault
Type          : Microsoft.Synapse/workspaces/linkedServices
Etag          : 21028e0c-0000-0a00-0000-681cddf0000

PS /home/harpal/adfartifacts> Set-AzSynapseLinkedService -WorkspaceName $workspace -Name azuresqllookup -DefinitionFile ".\linkedService\azuresqllookup.json" -DefaultProfile $context

WorkspaceName : wsp-synapse-harpal
Properties     : Azure.Analytics.Synapse.Artifacts.Models.AzureSqlDatabaseLinkedService
Id            : /subscriptions/f2ec0332-c731-4a35-b7cd-b9ff41313bbd/resourceGroups/rg-harpal/providers/Microsoft.Synapse/workspaces/wsp-synapse-harpal/linkedServices/azuresqllookup
Name          : azuresqllookup
Type          : Microsoft.Synapse/workspaces/linkedServices
Etag          : 2102870d-0000-0a00-0000-681cde0c000

PS /home/harpal/adfartifacts> Set-AzSynapseDataset -WorkspaceName $workspace -Name azuresqllookup -DefinitionFile ".\dataset\azuresqllookup.json" -DefaultProfile $context

WorkspaceName : wsp-synapse-harpal
Properties     : Azure.Analytics.Synapse.Artifacts.Models.AzureSqlTableDataset
Id            : /subscriptions/f2ec0332-c731-4a35-b7cd-b9ff41313bbd/resourceGroups/rg-harpal/providers/Microsoft.Synapse/workspaces/wsp-synapse-harpal/datasets/azuresqllookup
Name          : azuresqllookup
Type          : Microsoft.Synapse/workspaces/datasets
Etag          : 2102e10e-0000-0a00-0000-681cde1b0000

PS /home/harpal/adfartifacts> Set-AzSynapseDataset -WorkspaceName $workspace -Name ds_csvfiles -DefinitionFile ".\dataset\ds_csvfiles.json" -DefaultProfile $context

```

Step 4.3: Import Pipeline

Set-AzSynapsePipeline -WorkspaceName \$workspace -Name "pl_incremental_load_project" -DefinitionFile ".\pipeline\pl_incremental_load_project.json" -DefaultProfile \$context

```

PS /home/harpal/adfartifacts> Set-AzSynapsePipeline -WorkspaceName $workspace -Name "pl_incremental_load_project" -DefinitionFile ".\pipeline\pl_incremental_load_project.json" -DefaultProfile $context

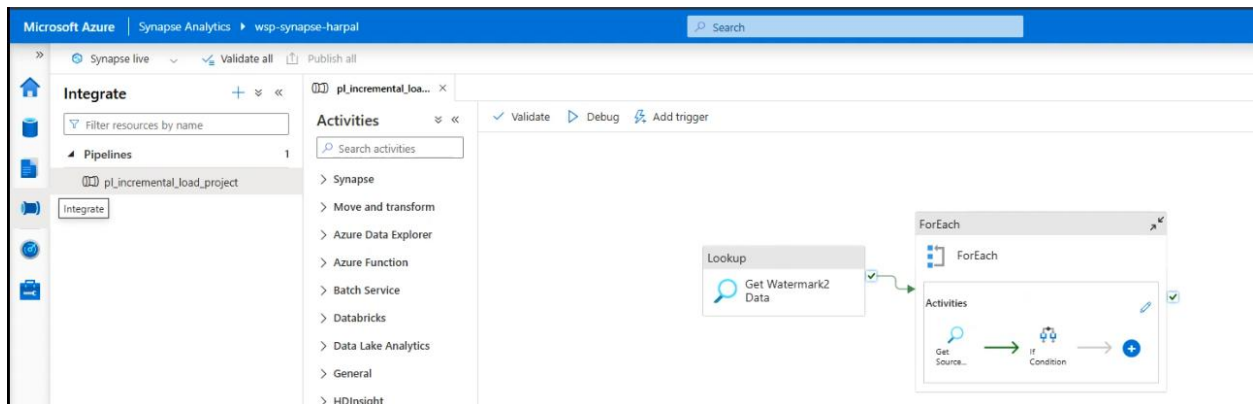
WorkspaceName : wsp-synapse-harpal
Description   :
Activities    : {Get Watermark2 Data, ForEach}
Variables     : {}
Concurrency   : {}
Annotations   : {}
RunDimensions : {}
Folder        : Microsoft.Azure.Commands.Synapse.Models.PSPipelineFolder
Parameters    : {}
AdditionalProperties : {}
Id            : /subscriptions/f2ec0332-c731-4a35-b7cd-b9ff41313bbd/resourceGroups/rg-harpal/providers/Microsoft.Synapse/workspaces/wsp-synapse-harpal/pipelines/pl_incremental_load_project
Name          : pl_incremental_load_project
Type          : Microsoft.Synapse/workspaces/pipelines
Etag          : 2102ef11-0000-0a00-0000-681cde370000

```

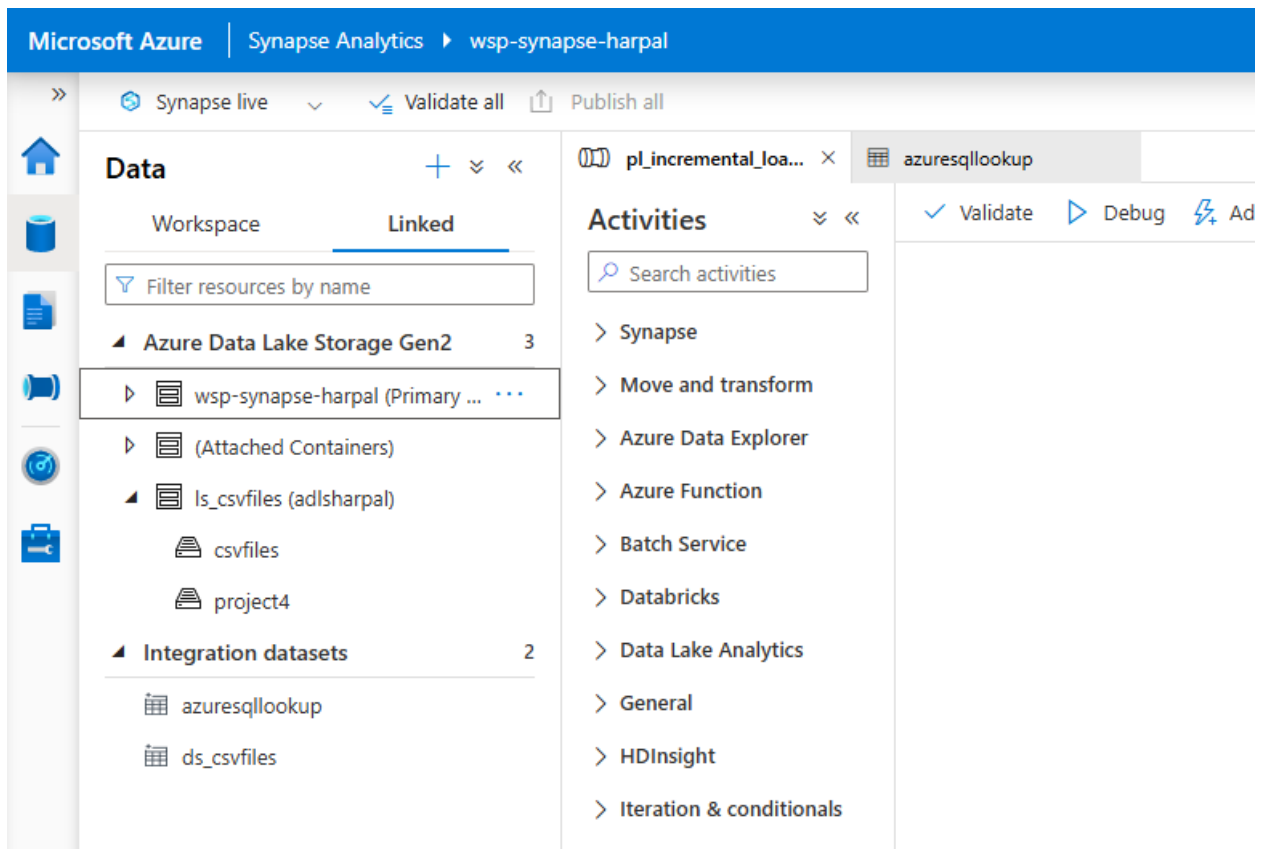
Step 5: Validate and Run Pipeline in Synapse

1. Open Synapse Studio
2. Go to Integrate > Verify pipeline pl_incremental_load_project exists
3. Click the pipeline > Click Validate (top bar)
4. Click Debug to test run the pipeline

Integrate Tab:



Check the Dataset in the Data Tab:



Manage Tab:

The screenshot shows the 'Manage' tab in the Microsoft Azure Synapse Analytics portal. The left sidebar contains navigation options like 'Connector upgrade advisor', 'Analytics pools', 'SQL pools', 'Apache Spark pools', 'Data Explorer pools (preview)', 'External connections', 'Linked services', 'Microsoft Purview', 'Integration', 'Triggers', 'Integration runtimes', 'Security', and 'Access control'. The main area is titled 'Linked services' and includes a '+ New' button, a search filter, and a table of existing services.

Name	Type	Related	Annotations
azuresqllookup	Azure SQL Database	2	
ls_csvfiles	Azure Data Lake Storage Gen2	1	
ls_key_vault	Azure Key Vault	1	
wsp-synapse-harpal-WorkspaceDefaultSqlServer	Azure Synapse Analytics	0	
wsp-synapse-harpal-WorkspaceDefaultStorage	Azure Data Lake Storage Gen2	0	

Key Vault Connection Checking in Linked Service in Synapse Workspace:

The screenshot shows the 'Edit linked service' dialog for the 'azuresqllookup' service. The 'Authentication type' is set to 'SQL authentication'. The 'User name' is 'harpalsqladmin'. The 'Secret name' is 'azuresqlpassword', which is linked to an 'Azure Key Vault' service named 'ls_key_vault'. The 'Secret version' is set to 'Latest version'. The 'Always encrypt' checkbox is unchecked. The 'Encrypt' dropdown is set to 'Mandatory'. The 'Trust server certificate' checkbox is unchecked. The 'Host name in certificate' field is empty. The 'Additional connection properties' section is expanded, showing a 'New' button. The 'Annotations' section is also expanded. At the bottom, there is a 'Save' button, a 'Cancel' button, and a green status message: 'Connection successful' with a 'Test connection' link.

Debug the pipeline:

The screenshot shows the 'Debug' tab in the Microsoft Azure Synapse Analytics pipeline editor. The left sidebar shows the 'Integrate' tab with a search filter and a list of activities. The main area displays a pipeline diagram with a 'Lookup' activity connected to a 'ForEach' activity. The 'Lookup' activity is configured to 'Get Watermark2 Data'. The 'ForEach' activity is configured to 'Get Source Max Value'. The pipeline status is 'Succeeded'. Below the diagram, there is a table showing the execution details of the pipeline run.

Activity name	Activity status	Activity ID	Run start	Duration	Integration runtime	User prop...	Activity run ID
If Condition	Succeeded	If Condition	5/8/2025, 12:40:46 PM	2s			b42d4919-9808-4b13-bec1-0bc8c
If Condition	Succeeded	If Condition	5/8/2025, 12:40:46 PM	2s			893d4772-7854-4420-b54c-893dc
If Condition	Succeeded	If Condition	5/8/2025, 12:40:43 PM	3s			cff92333-5e98-4954-a65e-e984ae
If Condition	Succeeded	If Condition	5/8/2025, 12:40:39 PM	2s			215dfab7-4aef-4667-8b50-36d9b
If Condition	Succeeded	If Condition	5/8/2025, 12:40:39 PM	3s			a80a2fc-5984-4a76-bf03-d87fcb
Get Source Max Value	Succeeded	Lookup	5/8/2025, 12:40:31 PM	7s	AutoResolveIntegrationRuntime (Canada Central)		7e393ab8-c206-4839-b487-4e75a
Get Source Max Value	Succeeded	Lookup	5/8/2025, 12:40:31 PM	14s	AutoResolveIntegrationRuntime (Canada Central)		04be959e-eae5-4843-9d79-861cb
Get Source Max Value	Succeeded	Lookup	5/8/2025, 12:40:31 PM	7s	AutoResolveIntegrationRuntime (Canada Central)		c2498662-9f25-4591-b448-68b3d

What to Do With /home/harpal/adfartifacts after the migration?

We can safely delete the contents after migration is 100% successful, because:

The artifacts (pipeline, datasets, linked services) have already been imported into Azure Synapse.

Synapse now stores its own copy, and Cloud Shell is no longer needed for them.

Run this in Cloud Shell to delete the folder:

```
rm -r ~/adfartifacts
```

Or remove the Zip file:

```
rm ~/pl_incremental_load_project_support_live.zip
```

```
PS /home/harpal/adfartifacts> rm -r ~/adfartifacts
PS /home/harpal/adfartifacts> ls
clouddrive Microsoft
PS /home/harpal/adfartifacts> cd..
PS /home/harpal> ls
clouddrive Microsoft
PS /home/harpal> 
```

Points to remember:

- Make sure you are in the correct working directory when running import commands
- Ensure file names and paths match exactly
- Reimporting with an incorrect workspace name won't show errors, but nothing will appear
- Validate linked services post-import (especially Key Vault and Integration Runtime)

Conclusion:

This manual migration approach helps build a strong understanding of how ADF and Synapse Pipelines share the same underlying schema. Using scripting provides flexibility and is ideal for troubleshooting, learning, or precise CI/CD setups. The project was successfully executed from end-to-end with the pipeline running in the Synapse environment.

Method 2: Migration Using JSON Code (ADF to Synapse)

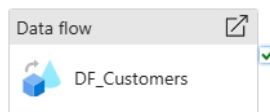
Objective

To migrate an existing pipeline and dataflow from Azure Data Factory (ADF) to Azure Synapse Analytics manually using the JSON definition of pipeline and dataflow artifacts.

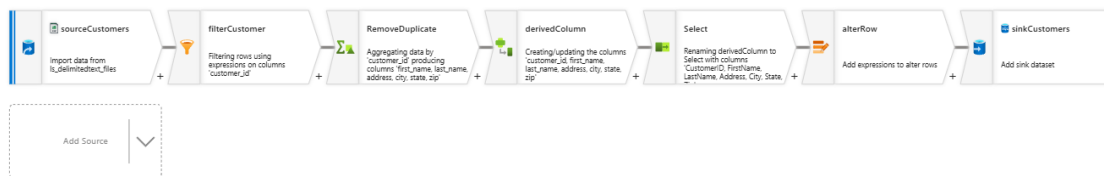
Step 1: Validate Pipeline in Azure Data Factory

We have this pipeline in ADF that cleans the customer's data from the ADLS Gen 2 storage account and loads it into the Azure SQL Database Table.

Open ADF workspace:



Dataflow Design:



Run the pipeline in Debug mode to ensure that it is working as expected

Parameters

Variables

Settings

Output

Pipeline run ID: 9c9818c4-42f1-4cb4-9eee-810df0368c96

Pipeline status

Succeeded

[View debug run consum](#)

All status

[Monitor in Azure Metrics](#)

[Export to CSV](#)

Showing 1 - 1 of 1 items

Activity name	Activity st...	Activit...	Run start	Duration	Integration runtime	User prop...	Activity run ID	Log
DF_Customers	<div><div></div>Succeeded</div>	Data flow	5/8/2025, 9:05:09 PM	59s	AutoResolveIntegrationRuntime (Canada Central)		8b9f7e07-b4e6-41b1-ac96-7cead920272a	

Step 2: Export JSON Definitions from ADF

Open the pipeline canvas in ADF

Click on the "Code" tab

Copy the entire JSON content of the pipeline design

Repeat the same process for the associated Data Flow

Dataflow Activity code at the pipeline level:

Pipeline name: pLadls_to_sql

Copy to clipboard

```

3  "properties": {
4    "activities": [
5      {
6        "name": "DF_Customers",
7        "type": "ExecuteDataFlow",
8        "dependsOn": [],
9        "policy": {
10         "timeout": "0.12:00:00",
11         "retry": 0,
12         "retryIntervalInSeconds": 30,
13         "secureOutput": false,
14         "secureInput": false
15       },
16       "userProperties": [],
17       "typeProperties": {
18         "dataflow": {
19           "referenceName": "df_customers_data",
20           "type": "DataFlowReference"
21         },
22         "compute": {
23           "coreCount": 8,
24           "computeType": "General"
25         },
26         "traceLevel": "None",
27         "cacheLinks": {
28           "firstRowOnly": true
29         }
30       }
31     ],
32     "annotations": [],
33     "lastPublishTime": "2025-05-09T00:52:42Z"
34   },
35   "type": "Microsoft.DataFactory/factories/pipelines"
36 }

```

OK Cancel

Dataflow Design JSON Code:

Copy to clipboard

```

1  {
2    "name": "df_customers_data",
3    "properties": {
4      "type": "MappingDataFlow",
5      "typeProperties": {
6        "sources": [
7          {
8            "linkedService": {
9              "referenceName": "ls_delimitedtext_files",
10             "type": "LinkedServiceReference"
11            },
12            "name": "sourceCustomers"
13          }
14        ],
15        "sinks": [
16          {
17            "linkedService": {
18              "referenceName": "ls_azure_sql_db",
19              "type": "LinkedServiceReference"
20            },
21            "name": "sinkCustomers"
22          }
23        ],
24        "transformations": [
25          {
26            "name": "filterCustomer"
27          },
28          {
29            "name": "RemoveDuplicate"
30          },
31          {
32            "name": "Select"
33          },
34          {
35            "name": "derivedColumn"
36          }
37        ]
38      }
39    }
40  }

```

Step 3: Open Azure Synapse Workspace

Launch Synapse Studio

Navigate to the Integrate hub

Click + Pipeline, then rename it exactly the same as the original ADF pipeline

Click on the "Code" tab and paste the copied JSON code from ADF

Confirm and save the pipeline.

Properties

General Related

Name *

pl_adls_to_sql

Description

Annotations

+ New

Step 4: Recreate Linked Services in Synapse

Go to the Manage tab in Synapse Studio

Manually create all Linked Services referenced in the original ADF pipeline

Azure SQL Database linked service

Azure Data Lake Gen2 linked service

Go to Synapse Workspace:

Azure SQL Database Linked Service:

Define the connection information needed for Azure Synapse Analytics to connect to external resources.

ny

Type	Rela
Azure SQL Database	2
Azure Data Lake Storage Gen2	1
Azure Key Vault	1
Azure Synapse Analytics	0
Azure Data Lake Storage Gen2	0

New linked service

☒ Azure SQL Database [Learn more](#)

Account selection method [ⓘ]

☒ From Azure subscription ☐ Enter manually

Azure subscription

Azure subscription 1 (f2ec0332-c731-4a35-b7cd-b9ff41313bbd)

Server name *

sqlservaghela

Database name *

harpalsqldb

Authentication type *

SQL authentication

User name *

harpalsqladmin

AKV linked service * [ⓘ]

ls_key_vault

Secret name *

azuresqldbpassword

☐ Edit

Secret version [ⓘ]

Latest version

☐ Edit

Always encrypted [ⓘ] ☐

Encrypt [ⓘ]

Mandatory

☒ Connection successful

Azure Data Lake Gen 2 Linked Service:

rich define the connection information needed for Azure Synapse Analytics to connect to external resources.

ns : Any

Type	Rela
Azure SQL Database	2
Azure SQL Database	0
Azure Data Lake Storage Gen2	1
Azure Key Vault	2
Azure Synapse Analytics	0
Azure Data Lake Storage Gen2	0

New linked service

Azure Data Lake Storage Gen2 [Learn more](#)

Choose a name for your linked service. This name cannot be updated later.

Name *

ls_delimitedtext_files

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime

Authentication type

Account key

Account selection method

From Azure subscription Enter manually

Azure subscription

Azure subscription 1 (f2ec0332-c731-4a35-b7cd-b9ff41313bbd)

Storage account name *

adlsharpal

Test connection

To linked service To file path

Annotations

+ New

> Parameters

> Advanced

Connection successful

Test connection Cancel

Creating... Back

Step 5: Recreate Data Flow

In the Develop hub of Synapse

Create a new Data Flow with the same name as in ADF

Open the "Code" view and paste the JSON from the ADF Data Flow canvas

Save and validate the design



Step 6: Test the Migrated Pipeline in Synapse

Open the new pipeline in Synapse

Click Debug to test the execution

Ensure data loads from ADLS to Azure SQL as expected

The screenshot shows the Microsoft Azure Synapse Analytics interface. The top navigation bar includes 'Microsoft Azure', 'Synapse Analytics', and the workspace name 'wsp-synapse-harpal'. The left sidebar shows the 'Develop' tab with a search bar and a list of resources, including 'df_customers_data'. The main area displays the 'Activities' tab for the selected pipeline, showing a 'Data flow' activity named 'DF_Customers'. Below the activities, the 'Output' tab shows the 'Pipeline run ID' as '29d470f6-7769-4c49-89f0-049f61c6afad' and the 'Pipeline status' as 'Succeeded'. A table below shows the execution details for the 'DF_Customers' activity.

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User properties	Actions
DF_Customers	Succeeded	Data flow	5/8/2025, 10:38:31 PM	2m 0s	AutoResolveIntegrationRuntime (Canada Central)		7et

Points to Remember:

- Ensure Linked Service names match exactly in Synapse or update references in JSON
- Validate the pipeline and dataflow in ADF before exporting
- Always test the pipeline in Debug mode post-migration
- Synapse supports most ADF components, but confirm compatibility for custom or preview features

Conclusion:

This manual JSON-based migration method successfully replicates an ADF pipeline and dataflow into Synapse Analytics. It avoids full Git-based export and gives granular control to migrate only what is needed.

Migrate Databricks Notebook to Fabric Workspace

Objective

To successfully migrate a set of 4 Databricks notebooks into Microsoft Fabric Workspace by modifying unsupported features such as mount points, key vault access, and storage paths. This includes reading data from Azure Data Lake Gen2, applying transformations, and saving results into Fabric Lakehouse tables.

Notebook 1

Go to Databricks Workspace and open/create a notebook:

```

Access ADLS Gen 2 Storage Account and Read File Python Tab
File Edit View Run Help Last edit was 5 hours ago

1
dbrutils.secrets.list("storage-account-scope")

[SecretMetadata(key='adls-stg-sas-token'),
 SecretMetadata(key='azuresqldbpassword'),
 SecretMetadata(key='blob-stg-sas-token')]

2
dbrutils.fs.mount(
  source = "wasbs://ctnadls@adlsharpal.blob.core.windows.net",
  mount_point = "/mnt/ctnadls",
  extra_configs = [{"fs.azure.sas.ctnadls.adlsharpal.blob.core.windows.net": dbrutils.secrets.get(scope = "storage-account-scope", key = "adls-stg-sas-token")}]
)
True

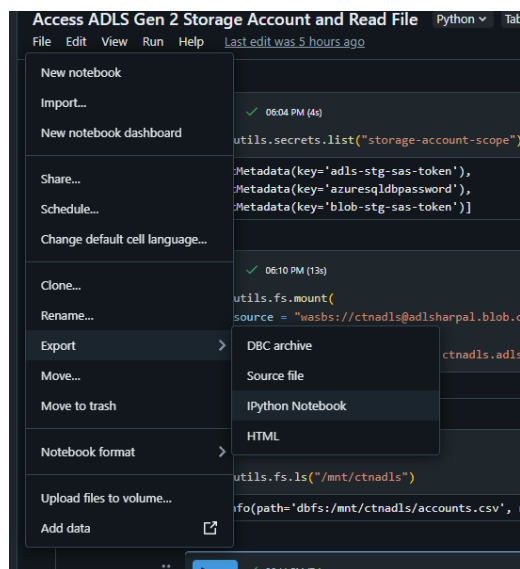
3
dbrutils.fs.ls("/mnt/ctnadls")

[FileInfo(path="/mnt/ctnadls/accounts.csv", name="accounts.csv", size=89, modificationTime=1746741810000)]

4
df_accounts = spark.read.csv("/mnt/ctnadls/accounts.csv", header=True)
display(df_accounts)
Python
(2) Spark Jobs

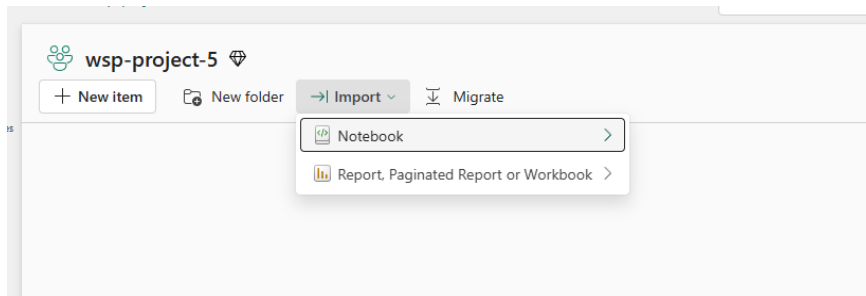
```

We will download this notebook in ipynb file format and migrate to the Fabric workspace.

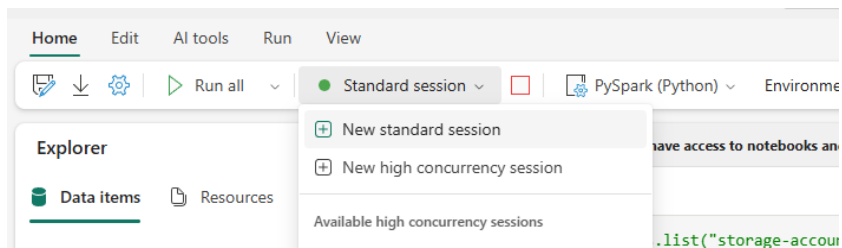


Turn on Fabric Capacity from the Azure Account

Now, open the Fabric Workspace and click on Import Notebook



Start the Spark Session



Make necessary changes for fabric notebook

Comment or remove code which is not needed



Add new code if we have used a mount point in Databricks, we have changed it here in the Fabric Notebook

```
1 sasToken = notebookutils.credentials.getSecret("https://keyvault-harpal.vault.azure.net/", "adls-stg-sas-token")
```

[1] ✓ - Command executed in 5 sec 811 ms by Fabric on 6:15:37 PM, 5/08/25

```
1 notebookutils.fs.mount(
2   "abfss://ctnadls@adlsharpal.dfs.core.windows.net",
3   "/test",
4   {"sasToken":sasToken}
5 )
```

[2] ✓ - Command executed in 876 ms by Fabric on 6:16:28 PM, 5/08/25

...

Table + New chart

Table view

	ABC localPath	ABC mountPoint	ABC scope	ABC source	ABC storageType
1	/synfs/notebo...	/test	job	abfss://ctna...	Data Lake Storag...

Display the csv file data from ADLS Gen 2:

```
1 containerPath = notebookutils.fs.getMountPath("/test")
```

[7] ✓ - Command executed in 297 ms by Fabric on 6:19:30 PM, 5/08/25

```
1 notebookutils.fs.ls(f"file://{containerPath}")
```

[6] ✓ - Command executed in 304 ms by Fabric on 6:18:39 PM, 5/08/25

... [FileInfo(path=file:/synfs/notebook/b5ba821f-52a8-4c15-aa80-34bf435152b4/test/accounts.csv, name=accounts.csv, size=89)]

```
1 df_accounts = spark.read.option("header", True).csv(f"file://{containerPath}/accounts.csv")
2 display(df_accounts)
3
```

[14] ✓ - Command executed in 1 sec 624 ms by Fabric on 6:25:35 PM, 5/08/25

Table + New chart

Table view

	ABC account_id	ABC customer_id	ABC account_type	ABC balance
1	100	50	Checking	1111
2	101	51	Checking	1111

Notebook 2

Read the CSV file, do the transformation and load it into ADLS Gen 2

N2_Read_Transform_Customer_File_And_Load Python ▾ Tabs: OFF ▾ ☆

File Edit View Run Help Last edit was 5 hours ago

▶ Run all Terminated ▾ Schedule

06:41 PM (9s) 1

```
df_customers = spark.read.csv("/mnt/ctnads/customers.csv", header=True)
display(df_customers)
```

▶ (2) Spark Jobs

df_customers: pyspark.sql.dataframe.DataFrame = [customer_id: string, first_name: string ... 5 more fields]

	customer_id	first_name	last_name	address	city	state	zip
1	1	John	Doe	123 Elm St	Toronto	ON	M4B1B3
2	2	Jane	Smith	456 Maple Ave	Ottawa	ON	K1A0B1
3	3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1
5	5	David	Wilson	202 Birch Blvd	Vancouver	BC	V5K0A1
6	6	Emma	Clark	505 Cedar St	Halifax	NS	B3H0A1
7	7	James	Martinez	606 Spruce Ln	Winnipeg	MB	R3C0A1
8	8	Olivia	Garcia	707 Fir St	Edmonton	AB	T5A0A1
9	9	William	Lopez	808 Redwood Dr	Victoria	BC	V8W0A1
10	10	Ava	Anderson	909 Cypress Ave	Quebec City	QC	G1A0A1
11	11	Alexander	Thomas	1010 Willow Rd	St. John's	NL	A1A0A1
12	12	Isabella	Lee	1111 Poplar St	Fredericton	NB	E3B0A1
13	13	Daniel	Harris	1212 Ash Blvd	Charlottetown	PE	C1A0A1
14	14	Sophia	Young	1313 Beech Dr	Yellowknife	NT	X1A0A1
15	15	Matthew	King	1414 Cedar Ln	Whitehorse	YT	Y1A0A1

87 rows | 8.62s runtime Refreshed 5 hours ago

Filter out the null customer ids:

06:41 PM (<1s) 2

```
from pyspark.sql.functions import col, when, lit
df_filtered = df_customers.filter((col("customer_id").isNull()))
display(df_filtered)
```

▶ (1) Spark Jobs

df_filtered: pyspark.sql.dataframe.DataFrame = [customer_id: string, first_name: string ... 5 more fields]

	customer_id	first_name	last_name	address	city	state	zip
1	1	John	Doe	123 Elm St	Toronto	ON	M4B1B3
2	2	Jane	Smith	456 Maple Ave	Ottawa	ON	K1A0B1
3	3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1

Drop the null records:

06:42 PM (<1s) 3

```
df_cleaned = df_filtered.na.drop()
display(df_cleaned)
```

▶ (1) Spark Jobs

df_cleaned: pyspark.sql.dataframe.DataFrame = [customer_id: string, first_name: string ... 5 more fields]

	customer_id	first_name	last_name	address	city	state	zip
68	68	Charlotte	Griffin	6767 Poplar St	Victoria Harbour	ON	L0K0A1
69	69	Joseph	Diaz	6868 Ash Blvd	Port McNicoll	ON	L0K0A1
70	70	Amelia	Hayes	6969 Beech Dr	Waubauskene	ON	L0K0A1
71	71	Christopher	Myers	7070 Cedar Ln	Coldwater	ON	L0K0A1

Rename the column name:

```

06:44 PM (1s) 4
df_renamed = df_cleaned.withColumnRenamed("customer_id", "CustomerID").withColumnRenamed("first_name", "FirstName").withColumnRenamed("last_name", "LastName").withColumnRenamed("address", "Address").withColumnRenamed("city", "City").withColumnRenamed("state", "State").withColumnRenamed("zip", "Zip")

display(df_renamed)
(1) Spark Jobs
df_renamed: pyspark.sql.dataframe.DataFrame = [CustomerID: string, FirstName: string ... 5 more fields]

```

	CustomerID	FirstName	LastName	Address	City	State	Zip
72	72	Mia	Ford	7171 Elm St	Orillia	ON	L3V0A1
73	73	Andrew	Hamilton	7272 Maple Ave	Gravenhurst	ON	P1P0A1
74	74	Harper	Graham	7373 Oak Dr	Bala	ON	P0C0A1

Writing it into ADLS Gen 2 storage account after cleaning:

```

06:47 PM (2s) 5
df_renamed.write.mode("overwrite").option("header", "true").csv("/mnt/ctnadls/customers_cleaned.csv")
(1) Spark Jobs
+ Code + Text
06:47 PM (1s) 6
df_customers_cleaned = spark.read.csv("/mnt/ctnadls/customers_cleaned.csv", header=True)
display(df_customers_cleaned)
(2) Spark Jobs
df_customers_cleaned: pyspark.sql.dataframe.DataFrame = [CustomerID: string, FirstName: string ... 5 more fields]

```

	CustomerID	FirstName	LastName	Address	City	State	Zip
72	72	Mia	Ford	7171 Elm St	Orillia	ON	L3V0A1
73	73	Andrew	Hamilton	7272 Maple Ave	Gravenhurst	ON	P1P0A1
74	74	Harper	Graham	7373 Oak Dr	Bala	ON	P0C0A1

Download this in ipynb file format

Now, open the Fabric Workspace and click on Import Notebook

Make necessary changes:

```

1 sasToken = notebookutils.credentials.getSecret("https://keyvault-harpal.vault.azure.net/", "adls-stg-sas-token")
2 notebookutils.fs.mount(
3     "abfss://ctnadls@adlsharpal.dfs.core.windows.net",
4     "/test",
5     {"sasToken": sasToken})
6 )
7 containerPath = notebookutils.fs.getMountPath("/test")

```

✓ - Command executed in 880 ms by Fabric on 7:15:53 PM, 5/08/25

Table	localPath	mountPoint	scope	source	storageType
1	/synfs/notebo...	/test	job	abfss://ctna...	Data Lake Storag...

```

1 df_customers = spark.read.csv(f"file://{containerPath}/customers.csv", header=True)
2 display(df_customers)

```

✓ - Command executed in 2 sec 494 ms by Fabric on 7:11:40 PM, 5/08/25

> Diagnostics 1

Table + New chart 7 col

Table view Download S

	ABC customer_id	ABC first_name	ABC last_name	ABC address	ABC city	ABC state	ABC zip
1	1	John	Doe	123 Elm St	Toronto	ON	M4B1B3
2	2	Jane	Smith	456 Maple A...	Ottawa	ON	K1A0B1
3	3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1
5	5	David	Wilson	202 Birch Blvd	Vancouver	BC	V5K0A1
6	6	Emma	Clark	505 Cedar St	Halifax	NS	B3H0A1
7	7	James	Martinez	606 Spruce Ln	Winnipeg	MB	R3C0A1
8	8	Olivia	Garcia	707 Fir St	Edmonton	AB	T5A0A1

Filter Null Values:

```

1 from pyspark.sql.functions import col, when, lit
2 df_filtered = df_customers.filter((col("customer_id").isNotNull()))
3 display(df_filtered)

```

✓ - Command executed in 832 ms by Fabric on 7:15:58 PM, 5/08/25

> Diagnostics 1

Table + New chart

Table view Download

	ABC customer_id	ABC first_name	ABC last_name	ABC address	ABC city	ABC state	ABC zip
--	-----------------	----------------	---------------	-------------	----------	-----------	---------

Drop Null records:

```

1 df_cleaned = df_filtered.na.drop()
2 display(df_cleaned)

```

✓ - Command executed in 821 ms by Fabric on 7:11:52 PM, 5/08/25

> Diagnostics 1

Table + New chart

Table view

	ABC customer_id	ABC first_name	ABC last_name	ABC address	ABC city	ABC state	ABC zip
1	1	John	Doe	123 Elm St	Toronto	ON	M4B1B3
2	2	Jane	Smith	456 Maple A...	Ottawa	ON	K1A0B1
3	3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1

Rename Column Name:

```

1 df_renamed = df_cleaned.withColumnRenamed("customer_id", "CustomerID").withColumnRenamed("first_name", "First
2
3 display(df_renamed)

```

✓ - Command executed in 882 ms by Fabric on 7:16:01 PM, 5/08/25

>  Diagnostics  1 

Table  New chart

Table view

	ABC CustomerID	ABC FirstName	ABC LastName	ABC Address	ABC City	ABC State	ABC Zip
1	1	John	Doe	123 Elm St	Toronto	ON	M4B1B3
2	2	Jane	Smith	456 Maple Ave	Ottawa	ON	K1A0B1
3	3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1
5	5	David	Wilson	202 Birch Blvd	Vancouver	BC	V5K0A1
6	6	Emma	Clark	505 Cedar St	Halifax	NS	B3H0A1
7	7	James	Martinez	606 Spruce Ln	Winnipeg	MB	R3C0A1




Write cleaned data into the ADLS Gen 2 storage account:

```

1 #df_renamed.write.mode("overwrite").option("header", "true").csv("/mnt/ctnadls/customers_cleaned.csv")
2
3 df_renamed.write \
4   .mode("overwrite") \
5   .option("header", "true") \
6   .csv(f"file://{containerPath}/customers_cleaned_from_fabric.csv")

```

14] ✓ - Command executed in 4 sec 704 ms by Fabric on 7:16:55 PM, 5/08/25

>  Diagnostics  1 

```

1 df_customers_cleaned = spark.read.csv(f"file://{containerPath}/customers_cleaned_from_fabric.csv", header=True)
2 display(df_customers_cleaned)

```

15] ✓ - Command executed in 1 sec 490 ms by Fabric on 7:17:00 PM, 5/08/25



Table  New chart

Table view  Down

	ABC CustomerID	ABC FirstName	ABC LastName	ABC Address	ABC City	ABC State	ABC Zip
1	1	John	Doe	123 Elm St	Toronto	ON	M4B1B3
2	2	Jane	Smith	456 Maple Ave	Ottawa	ON	K1A0B1
3	3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1
5	5	David	Wilson	202 Birch Blvd	Vancouver	BC	V5K0A1

Notebook 3

We will create a function to read tables and SQL queries in Databricks:

```

Weekend_Project_Data_Handling_with_SQL_Database_ADLS_Gen2_and_Databricks Python Tabs: OFF
File Edit View Run Help Run all Connect Schedule

1
def read_sql(table_name):
    df = (spark.read
        .format("jdbc")
        .option("url", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqldb-url'))
        .option("dbtable", table_name)
        .option("user", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqluser'))
        .option("password", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqlpassword'))
        .load()
    )
    return df

2
def read_sql_query(query):
    df = (spark.read
        .format("jdbc")
        .option("url", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqldb-url'))
        .option("query", query)
        .option("user", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqluser'))
        .option("password", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqlpassword'))
        .load()
    )
    return df

3
def write_data_sql(table_name, dataframe, mode):
    (dataframe.write
        .format("jdbc")
        .mode(mode)
        .option("url", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqldb-url'))
        .option("dbtable", table_name)
        .option("user", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqluser'))
        .option("password", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqlpassword'))
        .save()
    )
  
```

Download it and upload it into the Fabric Workspace

Make necessary changes

```

1 sqlpassword = notebookutils.credentials.getSecret("https://keyvault-harpal.vault.azure.net/", "azuresqldbpassword")
2 sqluser = notebookutils.credentials.getSecret("https://keyvault-harpal.vault.azure.net/", "azuresqluser")
3 url = notebookutils.credentials.getSecret("https://keyvault-harpal.vault.azure.net/", "azuresqldb-url")

- Command executed in 4 sec 70 ms by Fabric on 11:49:52 PM, 5/08/25 PySpark (Python)

1
2 def read_sql(table_name):
3     df = (spark.read
4         .format("jdbc")
5         .option("url", url)
6         .option("dbtable", table_name)
7         .option("user", sqluser)
8         .option("password", sqlpassword)
9         .load()
10    )
11    return df

[0] - Waiting PySpark (Python)
  
```


Notebook 4

Notebook 3 function we will use in this notebook

Databricks workspace:

```

1: Run Another notebook from this using magic command %run
%run "/Workspace/Harpal_Work_Files/Weekend_Project_Data_Handling_with_SQL_Database_ADLs_Gen2_and_Databricks"

2
df_table = read_sql(table_name="salesit.Product")
display(df_table)

3: Remove Duplicate Record If Any
df_table_unique= df_table.dropDuplicates()
display(df_table_unique)

4: Remove Null Column
#Remove Null Column
df_table = df_table.drop("Weight","SellStartDate", "SellEndDate", "ThumbNailPhoto", "DiscontinuedDate", "ThumbNailPhotoFileName", "rowguid", "ModifiedDate")

5
display(df_table)

6
df_table.printSchema()

```

```

from pyspark.sql.functions import col

fill_values = {
    'color': 'Unknown',
    'Size': 'Not Specified'
}

df_table_replace = df_table.na.fill(fill_values)
display(df_table_replace)

8: Filter Data based on filter condition
df_table_replace = df_table_replace.filter((col("Size") > '45') & (col("Size").isin("M", "L", "XL")))
df_table_replace = df_table_replace.filter(~col("Color").isin("Unknown"))
display(df_table_replace)

9
df_table_col_renamed = df_table_replace.withColumnRenamed("Name", "ProductName") #.withColumnRenamed("rowguid", "RowGUID")
display(df_table_col_renamed)

10
dbutils.fs.ls("/mnt/csvfiles")

```

11: Write Data to ADLS

```
df_table_col_renamed.write.format("delta").mode("overwrite").save("/mnt/csvfiles/ProductDetails")
```

12: Read Data from ADLS Delta Folder

```
df_productData = spark.read.format("delta").option("header", "true").option("inferSchema", "true").load("/mnt/csvfiles/ProductDetails")
display(df_productData)
```

13: SCD Type 1 Logic start here:

```
%sql
--drop table hive_metastore.default.ProductDetailSCD1
```

14

```
%sql
CREATE TABLE IF NOT EXISTS hive_metastore.default.ProductDetailSCD1 (
  ProductID INT,
  ProductName STRING,
  ProductNumber STRING,
  Color STRING,
  StandardCost DECIMAL(10, 4),
  ListPrice DECIMAL(10, 4),
  Size STRING,
  ProductCategoryID INT,
  ProductModelID INT,
  HashKey BIGINT,
  CreatedDate TIMESTAMP,
  UpdatedDate TIMESTAMP,
  CreatedBy STRING,
  UpdatedBy STRING
)
USING DELTA
LOCATION '/mnt/csvfiles/scdtvpe1/gold/ProductDetailSCD1'
```

15

```
from pyspark.sql.functions import *
df_hash_derived = df_productData.withColumn("HashKey", crc32(concat(*df_productData.columns)))
display(df_hash_derived)
```

16

```
from delta.tables import DeltaTable
deltaTable = DeltaTable.forPath(spark, "/mnt/csvfiles/scdtvpe1/gold/ProductDetailSCD1")
deltaTable.toDF().show() #convert delta table to dataframe and display the data
```

17

```
df_src = (
  df_hash_derived.alias("src")
  .join(
    deltaTable.toDF().alias("tgt"),
    (col("src.ProductID") == col("tgt.ProductID")), # Join only on id
    "left"
  )
  .filter( (col("tgt.ProductID").isNull()) | (col("src.HashKey") != col("tgt.HashKey")) )
  .select("src.*")
)
display(df_src)
```

```

18
deltaTable.alias("tgt").merge(
  df_src.alias("src"), "tgt.ProductID = src.ProductID"
).whenMatchedUpdate(
  set={
    "tgt.ProductID": "src.ProductID",
    "tgt.ProductName": "src.ProductName",
    "tgt.ProductNumber": "src.ProductNumber",
    "tgt.Color": "src.Color",
    "tgt.StandardCost": "src.StandardCost",
    "tgt.ListPrice": "src.ListPrice",
    "tgt.Size": "src.Size",
    "tgt.ProductCategoryID": "src.ProductCategoryID",
    "tgt.ProductModelID": "src.ProductModelID",
    "tgt.HashKey": "src.HashKey",
    "tgt.UpdatedBy": lit("Harpal-Updated"),
    "tgt.UpdatedDate": current_timestamp(),
  }
).whenNotMatchedInsert(
  values={
    "tgt.ProductID": "src.ProductID",
    "tgt.ProductName": "src.ProductName",
    "tgt.ProductNumber": "src.ProductNumber",
    "tgt.Color": "src.Color",
    "tgt.StandardCost": "src.StandardCost",
    "tgt.ListPrice": "src.ListPrice",
    "tgt.Size": "src.Size",
    "tgt.ProductCategoryID": "src.ProductCategoryID",
    "tgt.ProductModelID": "src.ProductModelID",
    "tgt.HashKey": "src.HashKey",
    "tgt.CreatedBy": lit("Harpal"),
    "tgt.CreatedDate": current_timestamp(),
    "tgt.UpdatedBy": lit("Harpal"),
    "tgt.UpdatedDate": current_timestamp(),
  }
).execute()

```

```

19
%sql
select * from hive_metastore.default.ProductDetailsCD1;

20
%sql
describe history hive_metastore.default.ProductDetailsCD1;

```

Fabric Workspace steps:

Modify the code according to the support for Fabric Notebook

```

1 %run Weekend_Project_Data_Handling_with_SQL_Database_ADLS_Gen2_and_Databricks

```



[1] ✓ 14 sec - Command executed in 7 sec 34 ms by Fabric on 12:02:10 AM, 5/09/25

```

1 df_table = read_sql(table_name="salesLt.Product")
2 display(df_table)

```

[2] ✓ 4 sec - Command executed in 4 sec 868 ms by Fabric on 12:02:15 AM, 5/09/25

>  Spark jobs (1 of 1 succeeded)  Resources

Modified code:

Write data into Lakehouse Table

```

1 #df_table_col_renamed.write.format("delta").mode("overwrite").save("/mnt/csvfiles/ProductDetails")
2 df_table_col_renamed.write.mode("overwrite").saveAsTable("ProductDetails")

```

[11] ✓ 29 sec - Command executed in 29 sec 135 ms by Fabric on 12:02:53 AM, 5/09/25 PySpark (Python)

> Spark jobs (3 of 3 succeeded) Resources

Read Data from Lakehouse

```

1 df_productdetails = spark.sql("SELECT * FROM lakehouse_hv.productdetails LIMIT 1000")
2 display(df_productdetails)

```

[14] ✓ 1 sec - Command executed in 1 sec 534 ms by Fabric on 12:03:58 AM, 5/09/25 PySpark (Python)

> Spark jobs (1 of 1 succeeded) Resources

```

1 %%sql
2 CREATE TABLE IF NOT EXISTS ProductDetailsCD1 (
3     ProductID INT,
4     ProductName STRING,
5     ProductNumber STRING,
6     Color STRING,
7     StandardCost DECIMAL(10, 4),
8     ListPrice DECIMAL(10, 4),
9     Size STRING,
10    ProductCategoryID INT,
11    ProductModelID INT,
12    HashKey BIGINT,
13    CreatedDate TIMESTAMP,
14    UpdatedDate TIMESTAMP,
15    CreatedBy STRING,
16    UpdatedBy STRING
17 )

```

✓ 3 sec - Command executed in 3 sec 462 ms by Fabric on 12:05:38 AM, 5/09/25

> Spark jobs (1 of 1 succeeded) Resources Log

```

1 from pyspark.sql.functions import *
2 df_hash_derived = df_productdetails.withColumn("HashKey", crc32(concat(*df_productdetails.columns)))
3 display(df_hash_derived)

```

✓ <1 sec - Command executed in 904 ms by Fabric on 12:06:05 AM, 5/09/25

```

1 from delta.tables import DeltaTable
2 #deltaTable = DeltaTable.forPath(spark, "lakehouse_hv/Tables/productdetailscd1")
3
4 deltaTable = DeltaTable.forName(spark, "productdetailscd1")
5 deltaTable.toDF().show() #convert delta table to dataframe and display the data

```

✓ 1 sec - Command executed in 1 sec 521 ms by Fabric on 12:10:09 AM, 5/09/25 PySpark (Python)

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|ProductID|ProductName|ProductNumber|Color|StandardCost|ListPrice|Size|ProductCategoryID|ProductModelID|HashKey|CreatedDate|UpdatedDate|CreatedBy|UpdatedBy|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

1

%%sql

2

describe ProductDetailSCD1;

✓

1 sec - Command executed in 1 sec 459 ms by Fabric on 12:10:52 AM, 5/09/25

Table

+ New chart

Table view

	ABC col_name	ABC data_type	ABC comment	
1	ProductID	int	NULL	
2	ProductName	string	NULL	
3	ProductNumber	string	NULL	
4	Color	string	NULL	
5	StandardCost	decimal(10,4)	NULL	
6	ListPrice	decimal(10,4)	NULL	
7	Size	string	NULL	
8	ProductCateg...	int	NULL	
9	ProductModelID	int	NULL	
10	HashKey	bigint	NULL	
11	CreatedDate	timestamp	NULL	

Key Modifications Required

Feature	Databricks Code	Fabric Code
Mount Point	/mnt/...	Files/...
Key Vault	dbutils.secrets.get(...)	notebookutils.credentials.getSecret(...)
File Write	.write.csv("/mnt/...")`	.write.csv("Files/...")`
Table Read	.load("/mnt/table")	.read.table("tableName")

Conclusion

The migration from Databricks to Microsoft Fabric required rewriting some logic, removing mount dependencies, adjusting secret access methods, and ensuring compatibility with Fabric’s OneLake file model. After applying these changes, all four notebooks were successfully executed in the Fabric Workspace using Delta tables and Lakehouse-backed CSV storage.