

Fabric Data Engineer

DP-700

*Total : **50-60** Questions*

(including one case study of around 10 Questions)

(~ 15-20 Questions from Each Section)

Implement and Manage an Analytics Solution

- Workspace settings
- Version Control & Deployment
- Access Control
- Data Governance
- Orchestration

Ingest and Transform Data

- Data ingestion
- Full and incremental loading
- Data stores: Lakehouse, Data Warehouse, Eventhouse
- Data transformation
- Streaming data
- Languages: T-SQL, KQL, PySpark

Monitor and Optimize an Analytics Solution

- Monitoring of Fabric items
- Error handling
- Optimizing solutions in Fabric

FABRIC; It weaves together modern Data Stack

Microsoft Fabric is an end-to-end data platform by Microsoft that unifies data engineering, data science, real-time analytics, business intelligence, and data governance in one SaaS environment. Think of it as the next generation of Power BI + Azure Synapse + Data Factory + OneLake, all rolled into one seamless experience.

FABRIC CAPACITY; Create capacity

Create Fabric capacity ...

Welcome to Microsoft Fabric

Fabric delivers an end-to-end analytics platform from the data lake to the business user.

Basics

Create Fabric capacity that you can use with your Fabric workspaces.

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all of your resources.

Subscription * CIC-Prod

Resource group * Tikku

Capacity details

Name your Capacity and select a location.

Capacity name * fabriccapacity011

Region * Central India

Size * F2

Fabric Capacity Units

Plans

SKU	Capacity Units	COST (ESTIMATED/MONTH)
F2	2	\$23,136.84
F4	4	\$46,273.68
F8	8	\$92,547.37
F16	16	\$181,094.74
F32	32	\$370,189.48
F64	64	\$740,378.95
F128	128	\$14,807,577.90
F256	256	\$29,615,158.00
F512	512	\$59,230,016.51
F1024	1024	\$1,384,063.21
F2048	2048	\$2,369,126.43

Let's create a Fabric workspace.

PRO premium / capacity → Power BI Only.

Build your first report

Add data to start building a report

Embedding

Fabric capacity

Semantic model storage format

Capacity * fabriccapacity011 - Central India

Template apps

Apply Cancel

Types of Licences

License Model	Acronym	What You Can Use	Note
Free	-	Cannot view shared Power BI objects	Limited to personal use
Pro	-	Power BI	Can share and collaborate with other Pro users
Premium Per User	PPU	Power BI	Can share with other PPU users, but not Free or Pro users
Premium Per Capacity	P	Power BI, Fabric	Free users can view shared Power BI objects (up to June 2018) → Mostly Used
Embedded	A / EM	Power BI	For embedding reports in apps, requires separate setup
Fabric Capacity	F	Fabric	- If F32 or below, you need a Power BI paid license to view shared objects - If F64 or above, free users can view shared Power BI objects → Mostly Used

F-SKU

Important = pause when Fabric is not in use

Tour TO FABRIC

LAKEHOUSE; is used to store large amounts of data

- Cleaning, Querying, Reporting & Sharing

A Data Lake is like a giant storage space for all types of raw data—structured, semi-structured, and unstructured—without strict organization.

A Data Lakehouse combines the flexibility of a Data Lake with the structured querying and performance of a Data Warehouse, making it easier to analyze and process data efficiently.

The screenshot shows the Microsoft Fabric interface for a 'DemoLakehouse'. On the left, the 'Explorer' sidebar has 'Tables' selected. A tooltip over a button in the top right corner says 'SQL analytics endpoint' with the subtext 'Query data using SQL'. A red arrow points from the text 'There is NO DML for SQL' to this tooltip. Below the sidebar, there's a section titled 'Get data in your lakehouse' with five buttons: 'Upload files', 'Start with sample data', 'New shortcut', 'New Dataflow Gen2', and 'New data pipeline'.

Lakehouse,
SQL is Read only
which means DML
won't work
DML works in DW

NOTEBOOK; to write code in PySpark (Python + Spark), SQL to analyze, visualize data. (VSCode Notebooks)

Pipeline; Set of Activities (Transform & Transport data)

Shortcut; Access Data residing in an External Lake

Data Flow Gen 2; Power Query

WAREHOUSE; Business Analytics on multiple Data sources

LAKEHOUSE VS **WAREHOUSE**;

Feature	Data Lakehouse	Data Warehouse
Primary Skill Set	PySpark, Spark SQL	SQL
Data Structure	Folders, files, databases, tables	Databases, schemas, tables
Supported Data Types	✓ Structured, ✓ Semi-structured, ✓ Unstructured	✓ Structured only
Read Operations	Spark, T-SQL NO DML	T-SQL YES DML
Write Operations	Spark	T-SQL
Multi-Table Transactions	✗ No	✓ Yes
Security	Row-level, table-level	Object-level, column-level, row-level, DDL/DML, dynamic data masking
Access Data via Shortcuts?	✓ Yes	✗ Indirectly through Lakehouse

F62 to view
Power BI
Objects.

more
security

DATAFLOW Gen 2 ; Data Ingestion (100 Data Sources)

Ctrl + Shift → Multiple Column Selection

The screenshot shows the Power Query Editor interface. A red box highlights the 'Data Schema view' button in the top-left corner. The main area displays a table with columns: ProductKey, ProductAlternateKey, ProductSubcategoryKey, WeightUnitMeasureCode, and SpanishProductName. A red box highlights the 'Promoted headers' section where 'ProductKey' is selected. The 'Query settings' pane on the right shows 'DimProduct' as the name and 'Source' as the type. The status bar at the bottom indicates 'Columns: 36 Rows: 99+'.

ADD DESTINATION ; Important Step

The screenshot shows the Power Query Editor with a red box highlighting the 'Data destination' section. It lists 'Default destination' (No default destination available) and 'Query destination' (Lakehouse, Warehouse, SQL database, Azure SQL database, Azure Data Explorer (Kusto)). The status bar at the bottom indicates 'Columns: 36 Rows: 99+'.

workspace

↓
Destination

Choose destination settings

To improve the performance of the data load into the destination, we are going to disable staging for the source query.

→ Manual Transformations

→ Existing Measures Lost

→ Existing Measures Retained

The screenshot shows the 'Choose destination settings' dialog. A red box highlights the 'Use automatic settings' checkbox. Below it, 'Update method' options are shown: Existing data, New data, Append, and Replace. An arrow points from 'Append' to 'Schema options on publish' which includes 'Existing schema', 'Dynamic schema', and 'Fixed schema'. A red arrow points from 'Replace' to the text 'Drop & New'.

SAVE DATA FLOW AS TEMPLATE Parquet format

Export template → Saved as **.POT** → Go to Import Gen 2

POT File Options:

- Import from Excel
- Import from SQL Server
- Import from a Text/CSV file
- Import from dataflows

Get data from another source → Import from a Power Query template → open download .pot file

REFRESH DATAFLOW

Configure a refresh schedule
Define a data refresh schedule to import data from the data source into the semantic model. [Learn more](#)

On
Weekly
Daily

Time
Add another time

Send refresh failure notifications to
Dataflow owner
These contacts:
Enter email addresses

Apply **Discard**

Every 30 Minutes

IMPLEMENT USING FAST COPY IN DATAFLOWS

FastCopy is designed to optimize the copy of large Data

The screenshot shows the 'Options' dialog in Power BI. Under the 'Global' section, there is a 'Fast Copy' group with a checked checkbox for 'Allow use of fast copy connectors'. Below this, there is a 'Concurrency' section with a slider and a checkbox for 'Limit number of concurrent evaluations'. A handwritten note 'Data flow' is written above the 'Fast Copy' section, and a handwritten note 'options' is written below it.

The screenshot shows the 'Queries [4]' pane in Power BI. A context menu is open over a query named 'DimProductCategory'. The menu includes options like 'Copy', 'Paste', 'Delete', 'Rename', 'Enable staging', 'Incremental refresh', and 'Require fast copy'. A checkmark is next to 'Require fast copy'. A handwritten note '123 ProductCategoryKey' is written above the menu, and a handwritten note '1 2 3' is written next to the first three rows of the table.

Limitations:

- **Data Size:** The maximum file size supported for Fast Copy is typically **100 MB per file**.
- **Rows:** It is optimized to handle up to **5 million rows** in a single copy operation efficiently.

MONITOR DATA TRANSFORMATIONS IN DATAFLOWS;

Dataflow → Go to Refresh History Details.

Optimize DATAFLOWS; For Better Performance

Technique	Description	Best Practice
Query Folding	Push computations back to the data source to leverage database optimizations.	Use native queries at the source for filtering, aggregating, etc.
Separate Load and Transform Dataflows	Split data loading and transformation into separate flows for better manageability and performance.	Run separate Dataflows: one for loading, one for transforming.
Incremental Refresh	Only refresh data that has changed, rather than the entire dataset.	Use Change Data Capture (CDC) or time-based filters for incremental loads.
Minimize Data Movement	Reduce unnecessary data movement by applying filters early in the flow.	Filter data at the source to move only relevant data.
Use Multiple Data Flows in Parallel	Run multiple Dataflows in parallel to process large volumes of data simultaneously. <i>using pipelines</i>	Break tasks into smaller, parallelized jobs to improve speed.

TRANSFORMING DATA USING DATAFLOW;

Tips; • Use Locale (Data type) in case of Error

• Ctrl + Shift + Select (multiple columns)

↳ can be used to change Dtype also

Choose columns

• Explore Date time Filters

• Merge → Join → Add Columns

• Append → Union → Add Rows.

Imp. ↳ populates Rows with Same Column Names

Column Names should Match.

MERGE TWO TABLES; Imp:

Merge

Select tables and matching columns to create a merged table.

Left table for merge: DimProductCategory

ProductCategoryKey	ProductCategoryAlternateKey	EnglishProductName
1		Bikes
2		Components
3		Clothing
4		Accessories

Right table for merge: DimProductSubcategory

subcategoryKey	ProductSubcategoryAlternateKey	EnglishProductSubcategoryName
1		Mountain Bikes
2		Road Bikes
3		Touring Bikes
4		Handlebars

Join kind: Left outer

Use fuzzy matching to perform the merge

The selection matches 4 of 4 rows from the first table

Cancel OK

Important;

- we can merge while selecting multiple columns

Composite Key Concept

→ Joins

Left Anti → Exclude from R.

Right Anti → Exclude from L.

Add etc.

Statistics Standard Scientific

Add

Multiply

Subtract

Divide

Divide (Integer)

Modulo

Percentage

Percent of

Number column

ProductSubcategoryKey

EnglishProductName

Spanish

Valid 100%

Error 0%

Empty 0%

Distinct, 427 unique

238 distinct

- Mark as key → Mark column as Primary Key
- Custom column → Column using M code.
- Duplicate Query → Duplicate of Query
No changes if made in Original
- Reference Query → Shortens Steps
Changes if made in Original

DIFFERENCE BETWEEN DUPLICATE & REFERENCE

DUPLICATE = No changes to original!

REFERENCE = Changes reflect, if done in parent table.

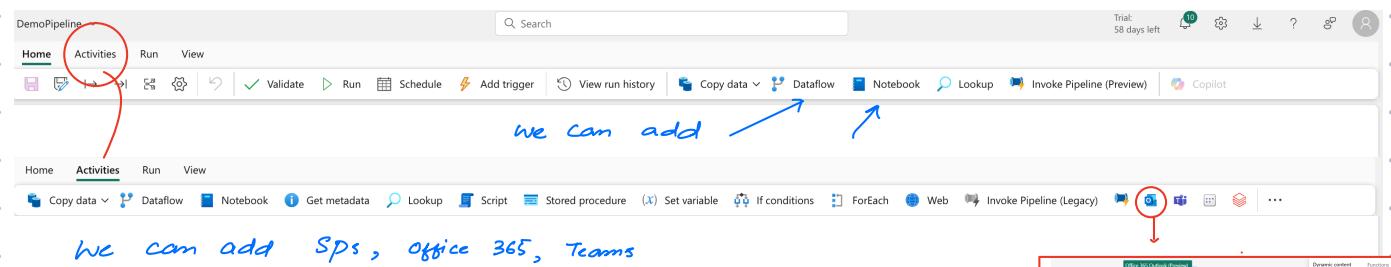
PIPELINES; Set of Activities

is a series of Data related tasks (activities)

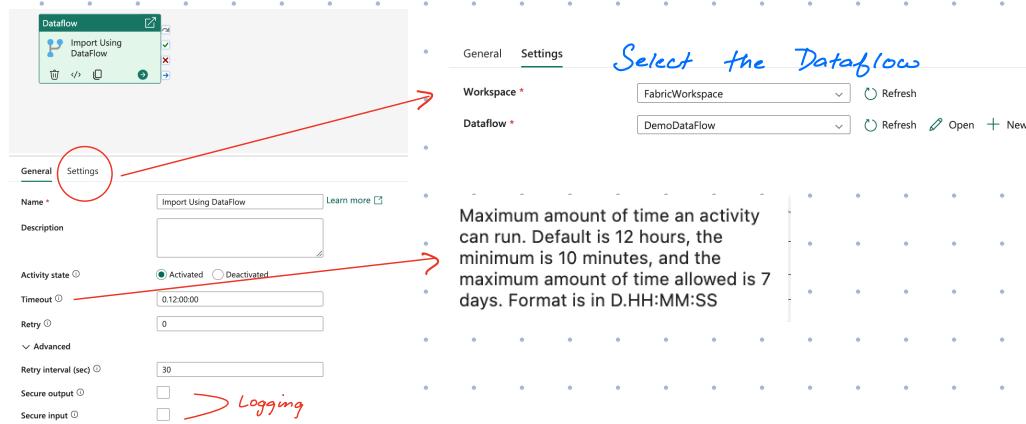
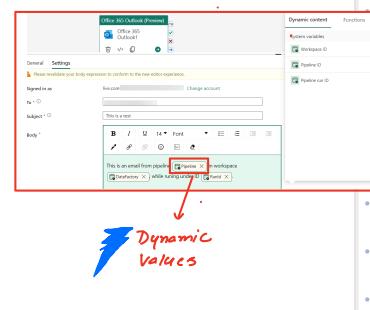
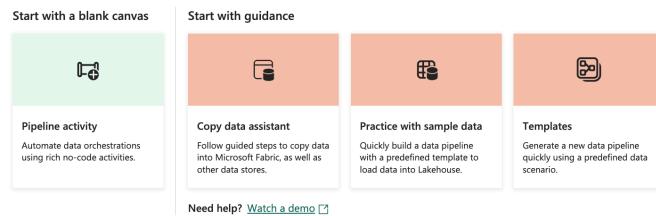
- Where to get the Data (Extract)
- What to do with Data (Transform)
- Where to send Data (Load)

Or.

Pipeline Automates the movement and Transformation of data from one place to another



Build a data pipeline to organize and move your data



Maximum amount of time an activity can run. Default is 12 hours, the minimum is 10 minutes, and the maximum amount of time allowed is 7 days. Format is in D.HH:MM:SS

CONNECTING ACTIVITIES;

Once the Data flow is run successfully, it will then go to another activity.



COPYING DATA IN PIPELINE;

To Copy the Data, we have 2 options

1. Using Copy Assistant (Allows multiple Tables)
2. Using copy Activity (Allows Single Table)

1. Using Assistant

Build a data pipeline to organize and move your data

The screenshot shows the 'Copy data' interface in the Azure Data Factory Copy Data Assistant. It includes steps for connecting to a data source (DemoLakehouse) and a data destination (FactInternetSales). The 'Column mappings' section is highlighted, showing how columns from the source schema are mapped to the destination schema. A red arrow points to the 'Destination' dropdown in the mapping table, with the handwritten note 'can change names'.

2. Using Copy Activity

The screenshot shows the 'Copy data' activity configuration in the Azure Data Factory Activities tab. It includes fields for General, Source, Destination, Mapping, and Settings. In the Mapping section, a checkbox for 'Convert dec - INT' is checked. Handwritten notes include 'Convert dec - INT' and 'EN-US etc.' in the Culture field. The 'Settings' tab is also visible.

What is Throughput in a Data Pipeline? Imp:

Throughput in a data pipeline refers to the amount of data that flows through the pipeline in a given time period. It is typically measured in bytes per second (Bps), records per second, or events per second and is a key performance metric for evaluating data pipeline efficiency.

Degree of Copy Parallelism?

In Azure Data Factory (ADF), when moving data from SQL Server to Blob Storage, setting **Degree of Copy Parallelism = 4** means ADF will run 4 parallel data streams to speed up the copy process.

SCHEDULE & MONITOR DATA PIPELINES

SCHEDULE PIPELINE ;

DemoPipeline
Data pipeline

Last success is in
3 April 2025 at 2:46:13 am
(UTC) Coordinated Universal Time

The scheduled refresh is turned off

Schedule

Run

Schedule
Scheduled run
On Off

Repeat
By the minute
By the minute
Hourly
Daily
Weekly
Monthly

End date and time
dd/mm/yyyy, --::--

Time zone
(UTC +05:30) Chennai, Kolkata, Mumbai, Ne

Apply Discard

PIPELINES



Schedule

MONITOR PIPELINE ;

View run history

Monitor

View and track the status of the activities across all the workspaces for which you have permissions within Microsoft Fabric.

Historical runs of DemoPipeline

Activity name Status Item type Start time Submitted by Location

DemoPipeline Succeeded Data pipeline 03/04/2025, 8:16 am Nadia Khan FabricWorkspace

Open Retry View detail

Monitoring hub > DemoPipeline

Rerun Cancel Refresh Update pipeline List Gantt

Dataflow Import Using Dataflow → Office 365 Outlook (Preview) → Pipeline Success

Export to CSV Filter Column Options

Pipeliness Success

Import Using Dataflow

Apr 03 08:16:20 Apr 03 08:16:25 Apr 03 08:16:30 Apr 03 08:16:35 Apr 03 08:16:40 Apr 03 08:16:45 Apr 03 08:16:50 Apr 03 08:16:55 Apr 03 08:17:00 Apr 03 08:17:05 Apr 03 08:17:10 Apr 03 08:17:20

→ List or Gant view

RESOLVE PIPELINE ISSUES ;

Copy data

Copy data1

Error details

Error code DWCopyCommandOperationFailed

Failure type User configuration issue

Details ErrorCode=DWCopyCommandOperationFailed,Type=Microsoft.DataTransfer.Common.Shared.HybridDeliveryException.Message='DataWarehouse' Copy Command operation failed with error "String or binary data would be truncated while reading column of type 'VARCHAR(8000)'. Check ANSI_WARNINGS option. Underlying data description: file https://olionkav13v4tf7hu84.efs.core.windows.net/74

How helpful or unhelpful was this error message?

Parameters Variables Settings On

Pipeline run ID: 15030f24-1881-4566-a7:

Activity name

ForEach_guw

Copy_guw

Filter by keyword Showing 8 it

1. New Mapping in Mapping

Import schemas Preview source New mapping Reset Delete Clear all unmapped mappings

Source Destination Type

Edit column Edit column

Mapping source is empty.

2. Fault Tolerance

General Source Destination Mapping Settings

Intelligent throughput optimization Auto Use custom value

Degree of copy parallelism Auto

Fault tolerance Select all

Enable logging 3

Enable staging 4

Data store type

Select all Skip incompatible rows Skip missing files Skip forbidden files Skip files with invalid names

Important; warehouse doesn't support varchar (max)

Azure SQL DB / SA = port 1433

Sample Data in Pipelines

Build a data pipeline to organize and move your data

Start with a blank canvas	Start with guidance		
Pipeline activity Automate data orchestrations using rich no-code activities.	Copy data assistant Follow guided steps to copy data into Microsoft Fabric, as well as other data stores.	Practice with sample data Quickly build a data pipeline with a predefined template to load data into Lakehouse.	Templates Generate a new data pipeline quickly using a predefined data scenario.
Need help? Watch a demo			

Only NY Taxi Data

The screenshot shows the Microsoft Fabric interface with the "Sample data" tab selected. It displays several datasets: "NYC Taxi - Green" (2 GB Parquet), "Diabetes" (14 KB Parquet), "Public Holidays" (500 KB Parquet), and "Retail Data Model from Wide World Importers" (352MB Parquet). Each dataset has a brief description and a link to view more details.

PIPELINE TEMPLATES ;

Pipeline templates

<input type="text"/>	Import template		
 Bulk Copy from Database by Microsoft Use this template to copy data in bulk from a database using an external control table to store the partition list of your source tables....	 Bulk Copy from Files to Database by Microsoft Use this template to copy data in bulk from Azure Data Lake Storage Gen2 to Azure SQL Database. ...	 Copy data from ADLS Gen2 to Lakehouse file by Microsoft Use this template to copy data from ADLS Gen2 to a specified file location in your Lakehouse. ...	 Copy data from ADLS Gen2 to Lakehouse Table by Microsoft Use this template to copy data from ADLS Gen2 to a specified table in your Lakehouse. ...
 Copy data from Azure SQL DB to Lakehouse Table by Microsoft Use this template to copy data from your Azure SQL database to a specified table in your Lakehouse. ...	 Copy data from sample data to Lakehouse file by Microsoft Use this template to copy data from sample data (NYC Taxi - Green) to Lakehouse file. ...	 Copy data from sample data to Warehouse by Microsoft Use this template to copy data from sample data (Retail Data Model from Wide World Importers) to your Fabric Warehouse....	 Copy multiple files containers between File Stores by Microsoft Use this template to leverage multiple copy activities to copy containers or folders between file based stores, where each copy...
 Copy new files only by LastModifiedDate by Microsoft Use this template to copy new or changed files only by using LastModifiedDate. ...	 Data anonymization with Presidio as an HTTP service by Microsoft This template uses Presidio as an HTTP service to anonymize PII data in text files during copy between one storage account to another....	 Delete files older than 30 days by Microsoft Use this template to delete files that have been modified more than 30 days ago from storage stores. ...	 Delta copy from Database by Microsoft Use this template to copy new or updated rows only from a database using a high-watermark stored in an external control table....
 Move files	 Unlock SharePoint Oversharing		

Important You cannot use MySQL Database as an incoming connector for "Fast copy"

NOTEBOOKS; is just like a Jupyter Notebook where we can run + write code but it is fully integrated with Fabric.

Supported Languages:

Fabric notebooks are built on Apache Spark and support:

- Python (most common for data science & ETL)
- SQL
- Scala
- Spark R (optional)

What Can You Do in a Fabric Notebook?

Task	Example
✓ Data Cleaning	Use Python (Pandas/Spark) to clean messy datasets
🔍 Data Exploration	Run queries on Lakehouse tables using SparkSQL
📊 Visualize Data	Use <code>matplotlib</code> , <code>seaborn</code> , or built-in Fabric charts
📦 ML/Modeling	Train models with <code>scikit-learn</code> , <code>MLLib</code> , etc.
⚙️ Pipeline Integration	Run as part of a pipeline activity (trigger notebooks in Fabric pipelines)

INGEST DATA USING LOCAL UPLOAD

The screenshot shows the Fabric interface with the 'DemoLakehouse' project selected. In the 'Tables' section, the 'Files' folder is expanded, and a context menu is open with 'Upload' highlighted. An 'Upload files' dialog is displayed, showing a file selection input and an 'Upload' button. Below it, a list of 'Current uploads' shows two CSV files: 'MtoMTarget.csv' and 'MtoMActual.csv', both with green checkmarks indicating successful upload. The sizes are listed as 173 B / 173 B and 152 B / 152 B respectively.

we can load these files as Tables

The screenshot shows the 'Load to Tables' process. A context menu is open over a CSV file ('MtoMActual.csv') with 'Load to Tables' selected. A sub-menu shows options for 'New table' and 'Existing table'. To the right, a 'Load file to new table' dialog is open, showing fields for 'New table name' (mtmtarget), 'Column header' (radio button selected), and 'Separator' (radio button selected). The dialog also includes a note about separator characters and 'Load' and 'Cancel' buttons.

Important; Does it make any difference if Data sits inside Tables or Files; (in Lakehouse)

Answer Yes, because Tables are processed Data
Tables will be Quicker

The screenshot shows the Microsoft OneLake Explorer interface. It has two main sections: 'Lakehouse' (Explore your data files and folders) and 'SQL analytics endpoint' (Query data using SQL). A red box highlights the 'SQL analytics endpoint' section.

Only Tables are Accessible.

Microsoft One Lake Explorer



- ✓ → Success
- ⌚ → Sync
- ☁️ → Online Only

A software that can be used locally to upload files (on windows)

Appropriate Method To Copy To LH or WH

It depends on multiple factors.

- Small files → Local copy or One Lake Explorer (Directly)
- Small files → Dataflows (Gen 2) (with Transform)
- Large Data → Data pipeline (Copy Tool) (without Transformations)
- Data → Notebook Code (Complex Transform)

INGEST DATA USING NOTEBOOK

Create Notebook → Add Destination.

Will focus on pyspark → Python API for Spark

In short, use spark using Python

Use Pandas if you're working on local datasets (like Excel/CSV files under a few GB).

Use Spark if you're dealing with huge datasets (logs, transactions, sensor data, etc.) or working in a distributed environment.

ABFS → Azure Blob File System

READ CSV

Loading data from csv file into a dataframe using Spark

```
df = spark.read.option("header", "true").format("csv").load("Files/MtoMActual.csv")  
  
df2 =  
spark.read.csv("abfss://FabricWorkspace@onelake.dfs.fabric.microsoft.com/DemoLakehouse.Lakehouse/Files/MtoMActual.csv")  
  
df2 =  
spark.read.load("abfss://FabricWorkspace@onelake.dfs.fabric.microsoft.com/DemoLakehouse.Lakehouse/Files/MtoMActual.csv",  
format='csv', header=True)
```

Loading data from csv file into a dataframe using Pandas

```
import pandas as pd  
# Load data into pandas DataFrame from "/lakehouse/default/" + "Files/MtoMActual.csv"  
df = pd.read_csv("/lakehouse/default/Files/MtoMActual.csv")  
display(df)
```

Saving data from a dataframe to a csv file or table

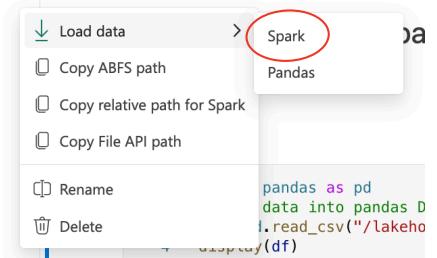
```
df.write.mode("overwrite").format("csv").save("Files/MtoMActual2.csv")  
  
df.write.mode("overwrite").format("delta").saveAsTable("MtoMActual")
```

Loading data from a table

```
df = spark.read.table("Mtomactual")  
display(df)  
  
df2 = spark.read.format("delta").load("Tables/mtomactual")
```

USE

display(df)



CREATING SEMANTIC MODEL Tables, Data model

The screenshot shows the Azure Data Explorer interface. On the left, the 'Explorer' pane displays a tree structure of databases and tables. In the center, a 'New semantic model' dialog box is open, prompting for a semantic model name and workspace. Below the dialog, a list of tables from the 'DimProduct' database is shown. A red circle highlights the 'Tables' section in the 'Select or deselect tables for the semantic model' dropdown.

LOADING DATA From TABLE IN PySPARK & SQL

In Notebooks, SQL is Read only (Lake house)

For manipulations,

use spark or pandas

A screenshot of a Jupyter Notebook cell. The code cell contains the following SQL command: `%%sql
SELECT * FROM mtomactual`. To the right of the cell, the handwritten note "Read Only." is written. In the top right corner of the cell, there is a dropdown menu labeled "Spark SQL" with a red circle around it.

Process to load Data into Tables

LAKEHOUSE → Files → UPLOAD CSV → LOAD AS TABLE

A screenshot of the Azure Data Explorer interface. The 'Tables' pane shows a list of tables, with 'DimCustomer' and 'DimSalesTerritory' selected. A red circle highlights the 'Load data' button next to 'DimCustomer'. The ribbon bar at the top has a 'Spark' tab selected, indicated by a red circle. A red arrow points from the 'Load data' button to the 'Spark' tab.

From Tables, Data can be loaded by SQL Spark by default.

TIP: Always upload Tabular files in Tables so they serve as processed data for analytics SQL Endpoint

A screenshot of the Azure Data Explorer interface. The 'Files' pane shows a single file named 'DimGeography.csv'. A red circle highlights this file. A context menu is open over the file, with a red circle highlighting the 'Load to Tables' option. Other options in the menu include 'New table' and 'Existing table'.

DATE TIME FORMATS IN PYSPARK

Symbol	Meaning	Example
yyyy	Year	2025
MM	Month (2 digits)	01 to 12
MMM	Month (short name)	Jan., Feb.
MMMM	Month (full name)	January, February
dd	Day of month (2 digits)	01 to 31
E / EEEE	Day of week	Tue., Tuesday
HH	Hour (24-hour format)	00 to 23
hh	Hour (12-hour format)	01 to 12
mm	Minutes	00 to 59
ss	Seconds	00 to 59
a	AM / PM	AM, PM
S / SSS	Fraction of second (ms)	123 (milliseconds)
z	Time zone offset	+0530, -0800
'text'	Escape text	'at' shows as at

Concat in Pandas / PySpark

Pandas → Doesn't need similar structure

Pyspark → Needs similar structure

Fillna → Coalesce → IF(NA, fill) in SQL

OPTIMIZE NOTEBOOKS

- we can schedule runs on our notebook

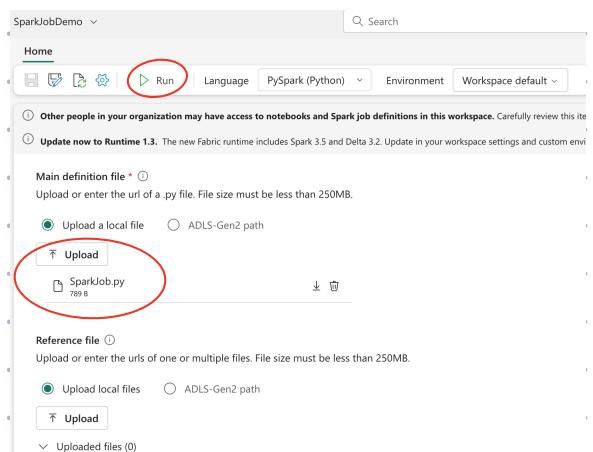
SPARK JOB DEFINITION Set of tasks that we want to apply on data

```
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType
spark = SparkSession.builder.getOrCreate()

schema = StructType([StructField('Date', StringType(), True),
StructField('Subtype', StringType(), True),
StructField('PurchaseMethod', StringType(), True),
StructField('Out', StringType(), True)])

dfs = spark.readStream.option("header", "true").schema(schema).format("csv").load("Files/Shop*.csv")
deltatablepath = "Tables/Shop_table"
query = dfs.writeStream.format("delta").outputMode("append") \
.option("checkpointLocation", deltatablepath + '/_checkpoint') \
.option("path", deltatablepath).start()
query.awaitTermination()
```

Save this code in .py file which will later be used to create Spark Job definition



IMPORTANT; Spark Jobs help run spark code continuously.

Choose Between Pipeline, Dataflow Gen2, Notebook

Feature	Pipeline (Copy Activity)	No CODE / Low CODE Dataflow Gen2	Notebook
Use Case	Data ingestion, warehouse migration, basic transformation	Data lake transformation, data wrangling	Complex data transformation, profiling, machine learning, large-scale processing
Primary Developer Persona	Data Engineer, Data Integrator <i>Migration</i>	Data Engineer, Business Analyst <i>Data Transformation</i>	Data Scientist, Data Engineer, Developer <i>Complex Transforms</i>
Skill Set Required	ETL tools, SQL, JSON	ETL, Power Query (M), SQL	Python, Scala, Spark SQL, R
Development Interface	Visual Wizard, Canvas	No-code / Low-code Graphical Interface	Code-based Notebook Interface
Sources Supported	30+ built-in connectors	150+ built-in connectors	100s of Spark-compatible libraries
Destinations Supported	18+ connectors	Limited connectors	100s of Spark-compatible output formats
Complexity & Transformation	Low — lightweight tasks like column mapping, type conversion, flattening	Medium — 300+ built-in transformations, native & custom logic	Low to High — native Spark transformations, custom logic, advanced processing

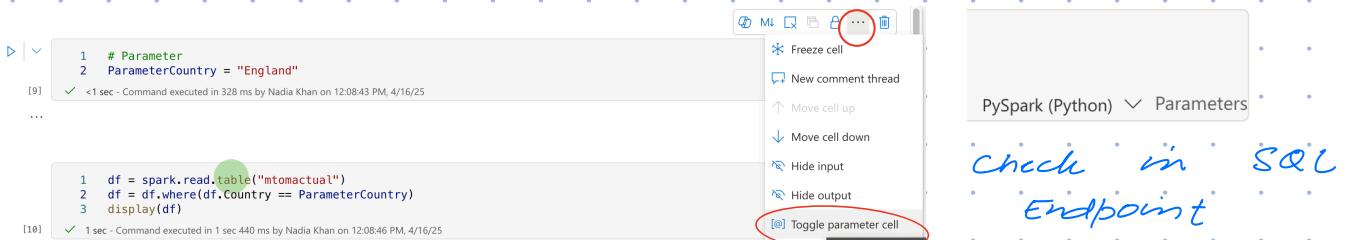
Pipelines → Simple Data movement / light Trans.

Dataflow Gen2 → Transformation (Less sources)

Notebook → Complex logic / MC etc.

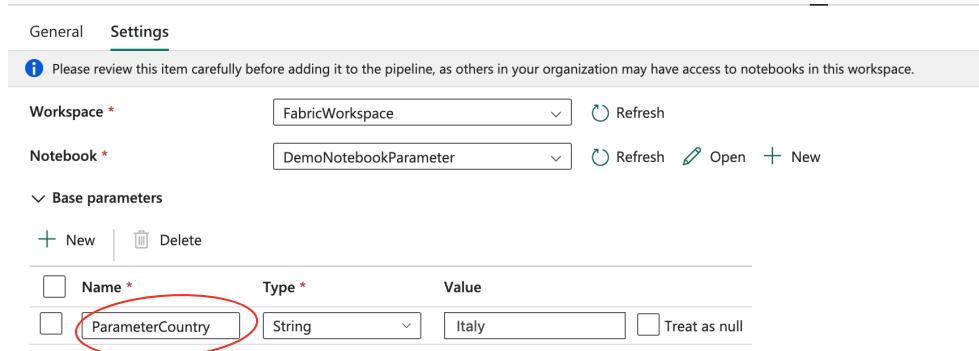
PARAMETERS IN NOTEBOOK & PIPELINES

STEP I; Notebook Parameters;



STEP II; PIPELINE PARAMETER

New Pipeline → Notebook



↳ we can Run and change values
and it will reflect in SQL End point

Using Dynamic Expressions

Name *	Type *	Value
<input checked="" type="checkbox"/> ParameterCountry	String	France <input type="checkbox"/> Treat as null

Add dynamic content [Alt+Shift+D]

New parameter

Name *	<input type="text"/>
Type	String
Default value	<input type="text"/>

Pipeline expression builder

Add dynamic content below using any combination of expressions, functions and system variables.

@pipeline().parameters.CountryParameter

Clear contents

Parameters System variables Trigger parameters Functions ...

Search

CountryParameter
Pipeline parameter

Many Options

So next time I will get this prompt to enter Parameter on Run

Pipeline run

Parameters

Name	Type	Value
CountryParameter	String	England

Parameters System variables Trigger parameters Functions ...

Search

- ▽ Expand all
- > Collection Functions
- > Conversion Functions
- > Date Functions
- > Logical Functions
- > Math Functions
- > String Functions

Important

Item Snapshots = It shows all the code and any output in the code

Spark Job definition; allows to Run code continuously.

SHORTCUTS

The screenshot shows the OneLake Explorer interface. On the left, there's a tree view under 'DemoLakehouse' with 'Tables' expanded, showing 'dfactual', 'dftarget', 'DimCustomer', 'dimgeography', and 'DimProduct'. A context menu is open over the 'dfactual' table, with the 'New shortcut' option highlighted and circled in red.

Can create shortcuts from OneLake

Doesn't copy Data but only creates shortcut

OneLake workspace Settings

The screenshot shows the 'Workspace settings' page. Under 'OneLake Settings', it says 'Configure and manage settings for OneLake in this workspace'. It includes sections for 'OneLake File Explorer' (integrating with Windows File Explorer) and 'Enable cache for shortcuts' (which is turned off). A red arrow points from the 'Cache' section of the notes to this setting.

Caches Data from other sources (Amazon S3)
cache upto 1 GB

Partition Data from DB

The screenshot shows the OneLake UI with a 'Copy data' pipeline step. Below it, the 'Destination' tab is selected, showing 'Advanced' settings. Under 'Table action', 'Append' is selected. In the 'Partition columns' section, 'Country' is listed with a note: 'partition can be more than one'. A red circle highlights the 'Country' column.

Note; Partitioning will allow to save data in partitions (parquet)
Folders / files / country = India
spark.read.parquet (path)

we can query whole table and also partitions

PARTITION DATA USING PYSPARK CODE IN LAKEHOUSE

python

```
df.write.mode("overwrite").format("delta") \
    .partitionBy("Color", "ProductSubcategoryKey").saveAsTable("DimProductPartitioned")
```

Organisational = Extra ID

MIRROR EXTERNAL DB; Azure mirroring external DB simply helps in copying and syncing data from external data sources into Azure (SQL, Synapse etc.) for Analytics & Realtime Insights

New item

Favorites All items

Get data

Ingest batch and real-time data into a single location within your Fabric workspace.

- Mirrored Azure Cosmos DB (preview)
- Mirrored Azure Database for Po... (preview)
- Mirrored Azure Databricks catalog (preview)
- Mirrored Azure SQL Database (preview)
- Mirrored Azure SQL Managed In... (preview)
- Mirrored database (preview)
- Mirrored Snowflake

NOTE It will create

- Mirrored DB
- Semantic Model

Imp; Dropbox is not supported by shortcuts

NOTEBOOK PERFORMANCE OPTIMISATION

1. Partitioning Data before Querying
2. Same Spark Session for multiple Notebooks

Standard session

New standard session

New high concurrency session

Available high concurrency sessions

No available sessions found

Fast

For Dedicated Compute

For multiple notebooks we can use high concurrency session for fast processing

View session information

3. Optimize write: write fewer / larger files (~128mb & 1GB) Fast Reads; few small files

```
python
spark.conf.set("spark.databricks.delta.optimizeWrite.enabled", "true")
```

4. Vorder; (vertical order) for column store index
 - Fast Scanning / Read
 - Better compression
 - Faster scans.
 - By default, no manual work

IDENTIFY & RESOLVE ISSUES WITH DELTA TABLE FILES

The screenshot shows the Delta Lake UI interface. On the left, there's an 'Explorer' sidebar with a tree view of tables under 'DemoLakehouse'. A red arrow points from the text 'Maintenance' to a three-dot menu icon next to the 'DimProduct' table. The main area shows a list of files in 'DimProduct (file view)'. On the right, a modal window titled 'Run maintenance commands' is open, with a red circle highlighting the 'Run' button at the bottom.

OPTIMIZE SPARK PERFORMANCE ++

1. When importing data, schema should be defined explicitly

PySpark Data Types – Quick Reference Table

Data Type	PySpark Type	Description	Example
String	<code>StringType()</code>	Text values	<code>"Tajamul"</code>
Integer	<code>IntegerType()</code>	32-bit integer	<code>25</code>
Long	<code>LongType()</code>	64-bit integer	<code>123456789012</code>
Float	<code>FloatType()</code>	32-bit floating point	<code>3.14</code>
Double	<code>DoubleType()</code>	64-bit floating point	<code>3.1415926535</code>
Boolean	<code>BooleanType()</code>	True/False values	<code>True, False</code>
Date	<code>DateType()</code>	Date only (yyyy-mm-dd)	<code>2025-04-16</code>
Timestamp	<code>TimestampType()</code>	Date and time	<code>2025-04-16 14:22:00</code>
Decimal	<code>DecimalType(p, s)</code>	Fixed-precision decimal	<code>Decimal("12.34")</code>
ArrayList	<code>ArrayType(elementType)</code>	Array of values	<code>[1, 2, 3]</code>
Map/Dict	<code>MapType(keyType, valueType)</code>	Key-value pairs	<code>{"a": 1, "b": 2}</code>
Struct (Row/Record)	<code>StructType([fields...])</code>	Nested fields/records (like JSON)	<code>{name: "Taj", age: 30}</code>

2. Avoid * and use select specific columns needed for the query
 3. df.explain(Truc) → Details of Query Execution
 4. Cache frequently used tables;
`spark.sql("CACHE TABLE mtomactual")`
 5. Enable AutoTune → Automatic Performance Tuning
↳ enabled by default that dynamically adjusts config.
- Important** "Enable Staging" for a query may speed up or slow down loading performance
- Optimum file size in Optimize write;
128 MB → 1 Gi

TRANSFORMING DATA IN DATAWAREHOUSE

Important; we can't create a table in Lakehouse (SQL Endpoint) or Notebook because SQL is Read only.

→ However Tables can be created in warehouse

DATA TYPES IN PYSPARK VS SQL

Data Type Category	Range/Description	PySpark Data Type	SQL Data Type
Tiny Integer	0 to 255 (SQL unsigned) / -128 to 127 (PySpark)	<code>tinyint</code>	<code>tinyint (unsigned)</code>
Small Integer	-32,768 to +32,767	<code>smallint</code>	<code>smallint</code>
Integer	-2,147,483,648 to +2,147,483,647	<code>int</code>	<code>int</code>
Big Integer	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807	<code>bigint</code>	<code>bigint</code>
Fixed-point Numbers	Specified precision and scale	<code>decimal(p, s)</code>	<code>decimal(p, s)</code> or <code>numeric(p, s)</code>
Floating-point Numbers	Approximate real numbers	<code>float</code> , <code>double</code>	<code>float</code> , <code>real</code>
Strings	Text values	<code>string</code> , <code>char(n)</code> , <code>varchar(n)</code>	<code>char(n)</code> , <code>varchar(n)</code>
Boolean	True or False	<code>boolean</code>	<code>bit</code>
Date and Time	Date and time (with timestamp)	<code>timestamp</code>	<code>datetime2</code>
Date Only	Calendar date only	<code>date</code>	<code>date</code>
Time Only	Time without date	<code>time</code>	<code>time</code>

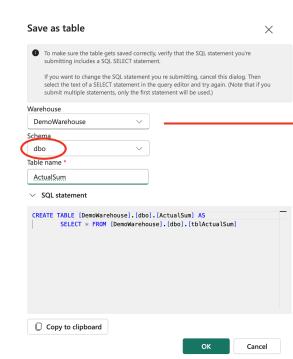
SAVING TABLES IN WAREHOUSE

```
45 -- Query Tables
46 SELECT * FROM [DemoWarehouse].[dbo].[tblActualSum]
47 SELECT * FROM [DemoWarehouse].[dbo].[tblTargetSum]
48 Save as table. Save the full results as a new table.
49
```

Select Query

Messages Results

ABC	Country
1	England
2	France
3	Italy



Important

Select * from Table
↓ (X)

Save as Demo

Takes more
Time

Select # from Table 1
into Table

↓
Faster and efficient

How to choose One?

Feature	Dataflow Gen2	Notebook	T-SQL (Warehouse)
Skill Set	Power Query (M), SQL	Spark (Python, SQL, Scala, R)	SQL
Use Case	Ingestion, transformation, profiling	Advanced transformation & processing	Reporting, transformation, analytics
User Persona	Data engineers, analysts	Data scientists, engineers	Data warehouse developers, DBAs
Interface	No-code/low-code (graphical)	Code-based (Notebook)	SQL scripts
Transformation Power	300+ built-in functions	Spark + open-source libraries	SQL-based for structured data
Connectivity	150+ connectors	100s of Spark libraries	DW, Power BI, Pipelines

Slowly changing Dimensions

When over the months names of products slowly changes.

Month 1 → P1 Type 0 = No updates

Month 2 → P1a Type 1 = All updates only

Month 3 → P1b Type 2 = Allows current & historical data.

SCD Type	Description	Example
Type 0	No changes are tracked; data remains static	Birthdate, National ID
Type 1	Overwrites old data with new data <u>(no history maintained)</u>	Correcting a misspelled name
Type 2	Keeps full history by creating a new record with a new surrogate key	Changing address, new row for each change with effective date/version

INCREMENTAL DATA LOAD;

Loading only the changes.

- Using Dataflow Gen2 Not Pipeline

Power Query IncrementalDataflow Dataflow saved

Home Transform Add column View Help

Queries [1]

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

ProductKey OrderDateKey DueDateKey

Incremental Refresh

Enable incremental refresh

Choose a Date/DateTime/TimeZone column to filter by: OrderDate

Extract data from the past: Days

Bucket size: Day

Day → 7 days → 7 Queries
Week → 1 week > 1 Query

Large Data
Small Data

Finally Destination → Warehouse

Lakehouse doesn't support incremental refresh.

Completed (0.44 s) Columns: 26 Rows: 99+ Add default destination...

MANAGE & OPTIMIZE DATA WAREHOUSE ;

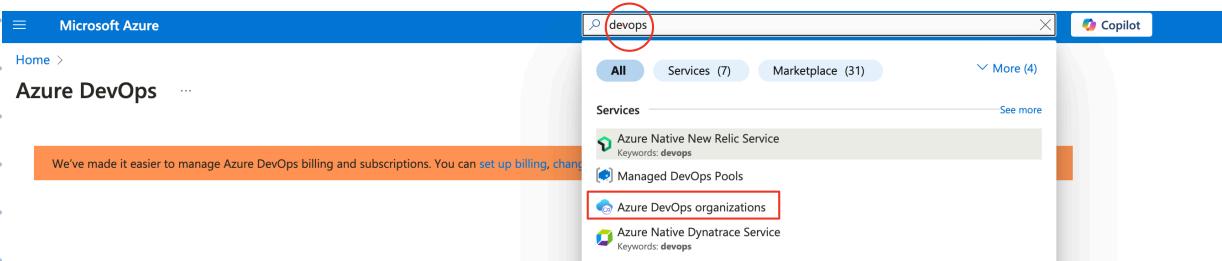
GIT INTEGRATION: connecting fabric workspaces with a Git repository (Github or Azure Devops Git) to enable version control, collaborative development and tracking changes for

- Notebooks
- Dataflows
- Pipelines
- Semantic Models
- Reports & Dashboards.

Typical Workflow:

1. Connect your Fabric workspace to a Git repo (Azure DevOps or GitHub).
2. Select a branch (usually `main` or `dev`).
3. Work on items like notebooks, reports, and pipelines.
4. Commit changes to the Git branch directly from Fabric.
5. Use pull requests for code review and collaboration.
6. Merge changes into `main` when ready for production.

CREATE AZURE DEVOPS ACCOUNT ;

Two screenshots illustrating the Azure DevOps organization creation process. The left screenshot shows the initial setup step where the user is prompted to name their organization as "dev.azure.com/ PremiumDemo" (with the entire input field circled in red). The right screenshot shows the "Create a project to get started" step, where the user can choose a project name ("Premium Workspace Proj") and set visibility ("Public" or "Private"). A small illustration of a person sitting with a dog is visible in the background of the right screenshot.

Azure DevOps Git is mostly preferred because it has tight integration with Azure Active Directory / Access Control. GitHub is Open Source / personal.

Final Take:

- Use **Azure DevOps Git** when you need a **secure, integrated enterprise DevOps platform**, especially with **Azure**.
- Use **GitHub** when you're focusing on **open-source**, community, or need **collaborative CI/CD** with **GitHub Actions**.

CREATING Repos : Initialize first Repository

Important = Development / Testing in Branch Repo
Production in Main Repo

GIT INTEGRATION

Inside Power BI Premium workspace

Choose from predesigned tasks

Power BI Delegated Settings

Workspace settings

Git integration

Provider: Azure DevOps AAD account

Connect Git repository and branch

Organization: PremiumDemo

Project: Premium Workspace Pro (project name)

Git repository: Premium Workspace Pro (git repository)

Branch: main (main / branch)

Git folder: Enter name of folder (optional)

Connect and sync Cancel

Important; workspace can be connected to one branch and to one folder at once

Commit IN WORKSPACE

Choose from predesigned task flows or add a task to build one (preview)

Source control

Current branch: main

Changes Updates

Optional Commit Message

Item Status

Bank_Churn_Analysis

Bank_Churn_Analysis.pbix

To commit changes, we need in workspace Contributor Rights Atleast.

Commit Undo

Once version control is setup, any files added will show here as they need to be committed to show in Azure Devops Repo

Note; Dashboards are not supported for GIT INTEGRATION

Premium Workspace Pro

Bank_Churn_Analysis.Report

StaticResources

.platform

definition.pbcr

report.json

Bank_Churn_Analysis.Semar

definition

.platform

definition.pbism

README.md

NOTE: In Azure Devops Git, only the schema will be copied over not the Data; which ensures Data Security

- Changing Branch → Needs Admin Rights
- Disconnect → Does not require ↗

CREATE Quick Reports; click on semantic model and Create Quick Report.

AZURE DevOps GIT:

Azure DevOps Premium Demo / Premium Workspace Pro > Repos > Files > Premium Workspace Pro

main / Type to find a file or folder...

Files

Contents History

Name Last change Commits

- AutoReport1.Report 6m ago 81ed0778 Two new auto reports Nadia Khan
- Bank_Churn_Analysis.Report 6m ago 81ed0778 Two new auto reports Nadia Khan
- Bank_Churn_Analysis.SemanticModel 42m ago ee71fce8 Committing 2 items from workspace 910de28e-d218-4cd2-bc88-a2285dd242...
- README.md 1h ago 22a8abef Added README.md Nadia Khan

Introduction
TODO: Give a short introduction of your project. Let this section explain the objectives or the motivation behind this project.

Getting Started
TODO: Guide users through getting your code up and running on their own system. In this section you can talk about:

1. Installation process
2. Software dependencies
3. Latest releases
4. API references

ADD MORE PROJECTS;

Azure DevOps PremiumDemo

PremiumDemo

Projects My work items My pull requests

+ New project

Incremental Project Premium Workspace Pro

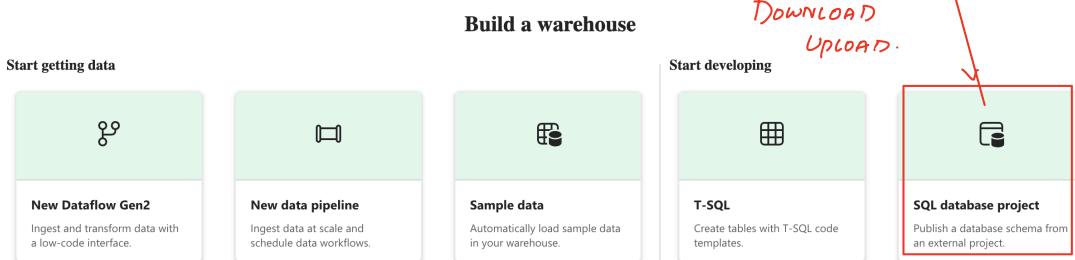
IMPLEMENT DB PROJECTS; Copy schema from one DW → other

DemoWarehouse

Home Reporting

Get data New SQL query SQL templates Query activity Model layouts Download SQL database project Copilot

This warehouse has a default Power BI semantic model. To automatically add objects, go to warehouse settings. To manually add objects, use Manage default semantic model. Learn more



Important; Database project includes The Schema

DYNAMIC DATA MASKING IN DW; HIDING DATA

is applicable for viewer mode

The screenshot shows the Snowflake UI. On the left, there's a sidebar with options like 'New item', 'New folder', 'Import', and 'Migrate'. Below it is a list of warehouses and semantic models. On the right, a 'Grant people access' dialog is open, asking for a name or email address to share a warehouse with. It also lists additional permissions, with 'Read all data using SQL (ReadData)' checked and 'Read all OneLake data (ReadAll) and subscribe to events (SubscribeOneLakeEvents)' highlighted with a blue box.

While sharing Masked Data, only use this one
Can be bypassed, so avoid using for Masked Data.

DYNAMIC DATA MASKING CODE;

```
CREATE TABLE DynamicDataMasking
(ID int MASKED WITH (FUNCTION = 'random(1,10)'), → 11-20 → 1-10
EmailAddress varchar(50) MASKED WITH (FUNCTION = 'email()'), → Khan@gmail.in → kxxxx@xxx.com
FamilyName varchar(30) MASKED WITH (FUNCTION = 'partial(1, "XXXXXX", 2)'), → TAJAMULKHAN → TXXXXXXXAN
Department varchar(30) MASKED WITH (FUNCTION = 'default()'), → xxxx
DateOfBirth date MASKED WITH (FUNCTION = 'default()')) → Default → 1900/01/01
```

MODIFY DATA MASKING

```
GRANT SELECT ON DynamicDataMasking TO [jane@Filecats.onmicrosoft.com] → Grant permission
```

```
ALTER TABLE DynamicDataMasking
ALTER COLUMN ID DROP MASKED; → Drop masking
```

```
ALTER TABLE DynamicDataMasking
ALTER COLUMN ID ADD MASKED WITH (FUNCTION = 'random(21, 30)') →
```

Note; Dynamic Masking is not enough because people can still query (select * from table) and it will return on ID=1

So, we should use Dynamic Masking in conjunction with OLS and RLS.

OPTIMIZE DATA WAREHOUSE;

- Use copy into then insert (one by one)
- Star instead of Snowflake
- Reduce No. of Columns / Rows
- Use tiny/small int / varchar
- Store Data as Numbers instead of char
- Use Direct Lake Mode when possible

NOTE;
SQL Server shows maximum of 10K rows, for more use SSMS

KUSTO SQL

is read only and used for Large/Big data exploration,
named after deep diver "Jack Kusto"

vs Kusto vs Traditional SQL

Feature	Kusto (KQL)	Traditional SQL
Primary use case	Log & telemetry analysis	Relational data (OLTP/OLAP)
Write operations	✗ (Read-only)	✓ (SELECT, INSERT, UPDATE, DELETE)
Syntax style	SQL-like but with custom functions	ANSI SQL
Performance focus	High-speed analytics on time-series data	Transactional + Analytical
Supported in	Azure Monitor, ADX, Sentinel, etc.	SQL Server, PostgreSQL, MySQL, etc.

iot, timeseries data

CREATING AN EVENTHOUSE ; using KQL

Eventhouse = Fast log insight..

Lakehouse = Deep Data Analysis

Feature	Eventhouse	Lakehouse
Use Case	Real-time log & event analytics	BI, reporting, ML on big data
Speed	Real-time (ms to seconds)	Near real-time or batch (minutes+)
Data Type	Logs, metrics, telemetry	Structured + unstructured data
Query Lang	KQL (Kusto Query Language)	SQL, Delta SQL, PySpark
Example Tools	Azure Data Explorer, Log Analytics	Databricks, Snowflake, Synapse

CREATE EVENTHOUSE DB

The screenshot shows the Azure Data Explorer interface. On the left, there's a sidebar with navigation links like Home, Create, OneLake catalog, Workspaces, and KQL Workspace. The main area shows a database named 'firsteventhouse' with a table 'EventHouseDB'. A red circle highlights the 'EventHouseDB' entry in the 'KQL databases' list. A red arrow points from the text 'create DB here' to this highlighted entry.

POPULATE WITH DATA

This screenshot shows the 'EventHouseDB' table in the 'firsteventhouse' database. A red circle highlights the 'Get data' button in the 'Database data' dropdown menu. To the right, a 'Real-Time Intelligence Sample Gallery' window is open, displaying various sample scenarios such as Stock analytics, Weather analytics, IoT analytics, Log analytics, Metrics analytics, and Automotive operations analytics. Each scenario includes a brief description and a preview icon.

Note; KQL is mostly designed for Eventhouse, we can also use SQL but KQL is more preferred,

- Power BI Reports ; **KQL supports** (✓)
(Create) **SQL doesn't** (✗)

- KQL is way faster than SQL

-- **KQL** **SQL = More Time**
explain

```
select top 10 State, count(*) as NumberOfRows from Weather  
group by State  
order by NumberOfRows
```

Weather **KQL = Less Time**

```
| summarize NumberOfRows=toint(count()) by State  
| project State, NumberOfRows  
| sort by NumberOfRows asc nulls first  
| take int(10)
```

The screenshot shows the Azure Data Explorer interface. On the left, there's a sidebar with 'EventHouseDB' selected, showing 'firsteventhouse' and 'EventHouseDB'. Under 'EventHouseDB', there are 'Tables', 'Shortcuts', 'Materialized views', 'Functions', and 'Data streams'. A 'Queryset' tab is open. In the main area, a query is running:

```
1 --  
2 | explain  
3 select top 10 State, count(*) as NumberOfRows from Weather  
4 group by State  
5 order by NumberOfRows  
6  
7 Weather  
8 | summarize NumberOfRows=toint(count()) by State  
9 | project State, NumberOfRows  
10 | sort by NumberOfRows asc nulls first  
11 | take int(10)
```

Below the query, a bar chart titled 'Table 1' displays the number of rows for various states. The chart has 'State' on the Y-axis and 'NumberOfRows' on the X-axis (0 to 34). The data points are:

State	NumberOfRows
HAWAII WATERS	2
GUU OF ALASKA	4
GUAM	4
LAKE ONTARIO	7
E PACIFIC	10
VIRGIN ISLANDS	12
AMERICAN SAMOA	16
DISTRICT OF COLUMBIA	22
LAKE ERIE	28
LAKE ST CLAIR	32

Annotations in red highlight several features:

- 'Preview 50 Rows' (near the Run button)
- 'Bundo' (near the preview button)
- 'Pin to dashboard' (circled)
- 'Create Power BI report' (circled)
- 'Different to power BI' (near the chart)
- 'can create visuals Directly' (near the 'Edit visual' button)

ADx ; **AZURE DATA EXPLORER** is a fast analytics service for large volumes of Data. It uses KQL. Quick insights from massive Data Streams.

- Used for monitoring apps, analyzing IoT data etc.

The screenshot shows the Azure Data Explorer interface. On the left, there's a sidebar with 'Home', 'Query', 'Connections', 'Dashboards', and 'My cluster'. A 'New tab' button is at the top. Below it, 'Connections' and 'Favorites' are listed. A query is running:

```
1 --  
2 | explain  
3 USE KQL  
4  
5 Weather  
6 | summarize NumberOfRows=toint(count()) by State  
7 | project State, NumberOfRows  
8 | sort by NumberOfRows asc nulls first  
9 | take int(10)
```

NOTE; we can use KQL inside eventhouse & ADX.

KQL; Start Basic KQL Queries.

Note In KQL, order by is Desc by default.

- Extend is used to add calculated columns

1 | Personnel
2 | project FamilyName = LastName, GivenName
That's right. There would be 2 columns: FamilyName and GivenName.
Correct ✓

+ Explain with AI

New Old Name

1 | Personnel | top 10
That's right. I could use "| top 10 by LastName", which would then order by LastName descending, and then take the first 10 rows. However, using "| top 10" by itself would result in an error.
Correct ✓

+ Explain with AI

SELECT DATA IN KQL

```
// Selecting Columns in KQL
// Use // for comments
// Use | to separate operators in the pipeline

Weather // Table name
| project State, EpisodeId, StartDate=StartTime, EndTime, EpisodeNarrative, EventNarrative // Select columns with optional aliasing (StartDate = StartTime)
| extend Duration = EndTime - StartDate // Add calculated column
| extend EpisodeNarrative = "hi" // Overwrite values in the column (not renaming rows)
| order by State asc nulls last // Order results; asc by default, 'nulls last' is valid
| project-rename Narrative = EpisodeNarrative // Rename column: new name = old name
| project-away EventNarrative // Remove column
```

// Limit Number of Rows

```
Weather // Table name
| distinct State
| top 10 by State asc nulls last
```

```
Weather
| sample 10
| limit 5
```

CREATE STRING LITERALS

```
// Creating string literals

print passcode = 1 // Number
print 'He is know by the name "Roy"'
```

```
// Problem with Backslash use Double Backslash
print "He is a good Husband\\Son "
```

```
// Multi line Code - Use Apostrophe
print ``He is very well know \
for his character, \
He is very Kind``
```

Multi Line String

FILTERING WITH STRING where

NOTE ; when renaming , use only one = sign
when comparing , use == sign

```
Weather
| project Narrative=EpisodeNarrative, EventType
| where EventType == "Heavy Rain" // case sensitive

Weather
| project Narrative=EpisodeNarrative, EventType
| where EventType =~ "heavy rain" // case insensitive

Weather
| project Narrative=EpisodeNarrative, EventType
| where EventType != "Heavy Rain" // case sensitive

Weather
| project Narrative=EpisodeNarrative, EventType
| where EventType !~ "heavy rain" // case insensitive

Weather
| project Narrative=EpisodeNarrative, EventType
| where not (EventType == "Heavy Rain") // case sensitive

Weather
| project Narrative=EpisodeNarrative, EventType
| where EventType == "Heavy Rain" or EventType == "Blizzard" // case sensitive

Weather
| project Narrative=EpisodeNarrative, EventType, State
| where State == "TEXAS" and (EventType == "Heavy Rain" or EventType == "Blizzard")
// case sensitive

Weather
| project Narrative=EpisodeNarrative, EventType
| where EventType in ("Heavy Rain", "Blizzard") // case sensitive

Weather
| project Narrative=EpisodeNarrative, EventType
| where not(EventType =~ "heavy rain" or EventType =~ "blizzard") // case insensitive

Weather
| project Narrative=EpisodeNarrative, EventType
| where EventType !in~ ("heavy rain", "blizzard") // case insensitive
```

FILTERING for Part of a String

HAS
Full word

CONTAINS
only letters

Advanced

```
Weather
| project EpisodeNarrative
| where EpisodeNarrative has "county" // case insensitive

Weather
| project EpisodeNarrative
| where EpisodeNarrative contains "county" // case insensitive

Weather
| project EpisodeNarrative
| where EpisodeNarrative has_cs "county" // case sensitive

Weather
| project EpisodeNarrative
| where EpisodeNarrative has "cou"

Weather
| project EpisodeNarrative
| where EpisodeNarrative contains "cou"

Weather
| project EpisodeNarrative
| where EpisodeNarrative hasprefix "cou"
| where not (EpisodeNarrative hasprefix "count")
| where EpisodeNarrative !hasprefix "couple"

Weather
| project EpisodeNarrative
| where EpisodeNarrative startswith "showers"
```

```
Weather
| project EpisodeNarrative
| where EpisodeNarrative has "showers" or EpisodeNarrative has "county"

Weather
| project EpisodeNarrative
| where EpisodeNarrative has "showers" and EpisodeNarrative has "county"

Weather
| project EpisodeNarrative
| where EpisodeNarrative has_any ("showers", "county")

Weather
| project EpisodeNarrative
| where EpisodeNarrative has_all ("showers", "county")
```

DATA AGGREGATION

```

Weather
| summarize Calc = count() by State, EventType
    ↗ Think of this as Group by

Weather
| summarize Calc = countif(EventType == "Flood") by State, EventType

Weather
| summarize Calc = countif(EventType != "Flood") by State, EventType

Weather
| where State in~ ("Texas", "Kansas", "Alaska")
| summarize Calc = countif(EventType =~ "flood") by State, EventType
| where Calc >= 100

Weather
| summarize Calc = sum(InjuriesDirect) by State

Weather
| summarize Calc = sumif(InjuriesDirect, EventType == "Flood") by State

Weather
| summarize Calc = avg(InjuriesDirect) by State

Weather
| summarize Calc = max(InjuriesDirect) by State

Weather
| summarize Calc = min(InjuriesDirect) by State

Weather
| summarize Calc = maxif(InjuriesDirect, EventType == "Flood") by State

Weather
| summarize Calc = dcount(EventType) by State

```

No having
can use
multiple where

ADVANCE
· dcount

→ Distinct count

KQL FUNCTIONS

- Empty strings (note: also can use isnotempty, isnull and isnotnull)

```

Weather
| project State, EpisodeNarrative, EventType, BeginLocation
| where isempty(BeginLocation) // or == ""
| limit 20

```

- Combining strings together and trimming the result

```

Weather
| project State, EpisodeNarrative, EventType, BeginLocation
| extend Calc = strcat(State, ":", EventType, ":", EpisodeNarrative) → simple concat
| limit 20

Weather
| project State, EpisodeNarrative, EventType, BeginLocation
| extend Calc = strcat_delim(":", State, EventType, EpisodeNarrative) → concat with Delim
| limit 20
    ↗ (Both strings + number)

Weather
| project State, EpisodeNarrative, EventType, BeginLocation, EventId
| extend EventType = toupper(EventType) // and tolower
| extend Calc = strcat_delim(":", EventId, State, EventType, EpisodeNarrative) → UPPER LOWER
| limit 20

Weather
| project State, BeginLocation, EndLocation
| extend Calc = trim(' ', strcat_delim(" ", BeginLocation, State, EndLocation)) → Trim (Remove, where)
| limit 20

```

→ we can also use trim-start or end

MANIPULATING STRINGS

```

Weather
| project EventType
| extend FindSpace = indexof(EventType, " ") // gives -1 if there is not a space
| limit 20

| extend FindSpace = indexof(strcat(EventType, " "), " ")
| extend BeforeSpace = substring(EventType, 0, FindSpace)
| project FindSpace, BeforeSpace, State, EpisodeNarrative, EventType, BeginLocation

```

Substring is used to extract part of string

Substring

Syntax

(column, start index, No. of characters)

String = Length of String

REPLACE

STRING

```
| extend EventType = replace_string(EventType, "heavy", "huge") // does not change "Heavy".  
| extend EventType = replace_string(EventType, "Heavy", "Huge")
```

IMPORTANT



| extend DepLoc = strcat(Department, " ", Location)

Correct ✓

That's correct. The strcat function concatenates strings together.

+ Explain with AI



print indexof("Family Name", " ") will return the answer 6.

Correct ✓

That's correct. As KQL is a zero-based language, the 1st character is character number zero, and the 7th is character number 6.

+ Explain with AI

Important ; KQL follows BODMAS RULE

NUMBER DATA TYPES;

- Real, Decimal = has fractions
- int, long = do not have fractions

Example

print 1/2 = 0 → floor Division
1.0/2 or 1/2.0 = 0.5

Data Type	Example	Description
whole {	int 8	Signed 32-bit integer (e.g., whole numbers)
	long 1234567890123 (Default)	Signed 64-bit integer for larger whole numbers
floats {	real 10.5 (Default)	64-bit double-precision floating-point number
	decimal decimal(11,99) slower compared to real exact division	Signed 128-bit high-precision decimal number
string	"hello"	Sequence of characters (text)
bool	true / false	Boolean value (true or false)
datetime	datetime(2024-01-01)	Date and time value
timespan	1d, 5h, 30m	Time difference (duration)

$$\frac{1}{2} = 0$$

$$\frac{1}{2} = 0$$

$$\frac{1}{2} = 0.50000$$

$$\frac{1}{2} = 0.5$$

MATH FUNCTIONS

```
| extend Calc = round(EpisodeId - EventId/ real(5), 0) // and ,1) for 1 decimal place
| extend Calc = ceiling(EpisodeId - EventId/ real(5), 0) rounds up
```

Other math functions

```
| extend Calc2 = abs(Calc)
| extend Calc3 = sign(Calc)
| extend Calc = EpisodeId % (EventId/10) // delete Calc2 and Calc3.
| extend Calc = pow(EpisodeId, 2)
| extend Calc = sqrt(EpisodeId)

| extend Calc = rand()
| extend Calc = rand(4)
// int      - signed 32-bit integer
// long     - signed 64-bit integer
// real     - signed 64-bit double-precision, floating-point number
// decimal - signed 128-bit decimal number.
```

DATE & TIMESTAMP DATATYPES

```
Weather
| distinct StartTime, EndTime
| extend Duration = EndTime - StartTime, RevisedEndTime = EndTime + 1d
// d h m s ms microsecond tick

print datetime(2030-02-03 01:23:45.6);
print todatetime("2030-02-03 01:23:45.6");
print make_datetime(2030, 2, 3, 1, 23, 45.6);

print timespan(1d);
print timespan(3);
print timespan(40 seconds);
print timespan(1.23:45:17.8);
print timespan(1.23:45:17.8) - 3h;

print timespan(1.23:45:17.8) * 3;
print timespan(1.23:45:17.8) / 3h;

print now() + 3h;
print now(3h);
print now(-3h);
print ago(3h)
```

Imp.

now ()
ago (3h)

DATE & TIMESTAMP Functions

```
| where StartTime between (datetime(2007-03-26) .. datetime(2007-04-15))
| extend Calc = startofday(StartTime) // and endofday → Important

Weather
| distinct StartDate = startofday(StartTime)
| extend Calc = startofweek(StartDate) // start of the week = Sunday.
| where StartDate between (datetime(2007-03-26) .. datetime(2007-04-15))
// also startofmonth, startofyear, endofweek, endofmonth, endofyear

| extend Calc = dayofweek(StartDate)
// and also hourofday, dayofmonth, dayofyear, weekofyear, monthofyear, getyear

| extend Calc = datetime_part("week_of_year", StartDate)
// start of the week = Sunday. First week includes the first Thursday.
// you can extract Year, Quarter, Month, Day, DayOfYear, Hour, Minute, Second, Millisecond, Microsecond and Nanosecond

Date manipulation
Weather
| distinct StartTime
| where StartTime between (datetime(2007-03-26) .. datetime(2007-04-15))

| extend Calc = StartTime+2d
| Calc2= datetime_add('day', 2, StartTime) → Important
// The period can be Year, Quarter, Month, Week, Day, Hour, Minute, Second, Millisecond, Microsecond and Nanosecond

| extend Calc = EndTime-StartTime
| Calc2= datetime_diff('day', EndTime, StartTime)
// datetime_diff only uses the relevant period.
// 26th 00:00 to 26th 23:00 is 0 days

Converting dates and timestamps
Weather
| distinct StartTime, EndTime
| extend Calc = format_datetime(StartTime, 'd MM yyyy')
, Calc2 = format_timespan(EndTime - StartTime, 'ddd.hh:mm:ss')
```

Format Specifier	Data Type	Description
yyyy	datetime	4-digit year (e.g., 2025)
MM	datetime	Month with leading zero (01-12)
dd	datetime	Day of the month with leading zero (01-31)
HH	Both	Hour in 24-hour format (00-23)
mm	Both	Minutes with leading zero (00-59)
ss	Both	Seconds with leading zero (00-59)
f to ffffffff	Both	Fractions of a second (up to 7 digits)
tt	datetime	AM/PM designator
ddddddd	timespan	Total days in duration (with padding if needed)



1 | | extend EndTime = datetime_add('day', 1, StartTime)

That's right. 'day' needs to be quotation or speech marks, and then the number of days and the datetime.

Correct ✓

+ Explain with AI

UNION THE DATA ;

One Query & second should be brackets

```
Weather
| distinct State, EventType
| union kind = outer withsource = TableSource
(Weather
| distinct State, EventNarrative)

union withsource = AdditionalColumn Wea*
```

Default Union = Outer

Inner = Only columns which are common in both tables

JOIN IN KUSTO SQL

```
Weather
| join
(Region | extend State = toupper(State))
on State
```

Default join = innerunique

```
Weather
| join kind = fullouter
(Region | extend State = toupper(State))
on ($left.State == $right.State)
```

→ Can use
on State and on

Remember =
syntax

Imp; Keep small table on left side for efficient joins

Join Type	Description
inner (default)	Returns only matching rows from both tables.
leftouter	Returns all rows from the left table, plus matching rows from the right.
rightouter	Returns all rows from the right table, plus matching rows from the left.
fullouter	Returns all rows from both tables, with nulls where there is no match.
leftanti	Returns rows from the left table that have no match in the right table.
rightanti	Returns rows from the right table that have no match in the left table.
innerunique	Like inner, but each match in the right table is used at most once.

IDENTIFY NULL & DUPLICATE VALUES

```
Weather
| join kind = fullouter
(Region | extend State = toupper(State))
on ($left.State == $right.State)
| project State, Region
| summarize Count = count() by State, Region
| where Count > 1
```

```
Weather
| join kind = fullouter
(Region | extend State = toupper(State))
on ($left.State == $right.State)
//| where State1 == "" or isnull(State1)
| where coalesce(State1, "") == ""
```

Mostly isnull (col) or col == ""

```
Weather
| join kind = rightanti
(Region | extend State = toupper(State))
on ($left.State == $right.State)
```

CONDITIONAL STATEMENTS ;

```
| summarize NumberOfEvents = count() by State  
| extend TexasOrFlorida = iif(State == "TEXAS" or State == "FLORIDA", "Texas/Florida", "Other")  
  
Weather  
| summarize NumberOfEvents = count() by State  
| extend TexasOrFlorida = iif(State in ("TEXAS", "FLORIDA"), "Texas/Florida", "Other")  
  
Weather  
| summarize NumberOfEvents = count() by State  
| extend TexasOrFlorida = case(State == "TEXAS", "Texas/Florida", State == "FLORIDA", "Texas/Florida", "Other")  
  
// the Else argument is required.
```

Note : iif and iff are same

- Case statement acts as Nested if

ORDERED FLOW OF DATA IN FABRIC

Step	Component	Purpose / Description
1	Data Sources	External data (APIs, databases, files, IoT, logs, etc.) collected from different systems.
2	OneLake (Data hub)	Centralized data lake storage for all data in Fabric (like a single data lake for all).
3	Pipelines	Used to orchestrate data movement and transformations (ETL/ELT).
4	Dataflow Gen2	No-code/low-code data prep tool for cleaning, shaping, and transforming data.
5	Lakehouse	Combines lake storage + structured access using Delta format (best for analytics).
6	Eventhouse	Specialized for ingesting and analyzing real-time, event-based data (e.g., logs/telemetry).
7	Notebooks	For data engineering, Python/Spark/SQL-based processing and advanced transformations.
8	SQL Endpoints	Query interface to access and analyze data from Lakehouse/Eventhouse.
9	Warehouse	Structured data warehouse optimized for BI and analytics (relational model).
10	Power BI	Data visualization and reporting layer to build dashboards and reports.

One LAKE ; one source of all data (Data hub) where we see every data (semantic models, eventhouse, Lakehouse, DB, warehouses)

Real Time HUB : Here we will see tables from

eventhouse DB's (NOT DBs)

Real-Time

Note ; we can create copy of EH Tables in one lake by using option in EH

Avoid this during development as it won't let us delete the data.

EventHouseDB

Database details

- Compressed: 12.1MB
- Original: 62.6MB
- OneLake: Enabled (button circled)
- Availability: Disabled (switch circled)

OneLake

When enabled, tables in this database are available in OneLake. [Learn more](#)

ENABLE ONE LAKE INTEGRATION FOR SEMANTIC MODELS

Power BI FabricWorkspace

IncrementalWarehouse

Describe the contents of this semantic model.

500 characters left

Apply Discard

Gateway and cloud connections

Data source credentials

Parameters

Query Caching

Refresh

Server settings

Data connections

Q&A

Endorsement and discovery

Request access

Large semantic model storage format

Query scale-out

OneLake Integration

You can automatically write data imported into your semantic model tables to delta tables in OneLake. Make sure your semantic model includes one or more import tables. [Learn more](#)

Off

CHOOSE APPROPRIATE STREAMING

Aspect	Eventstream	KQL DB
Purpose	Ingest real-time data	Store & query streams
Processing	Capture & transform events	Complex queries, alerts
Integration	Works with Fabric sources/sinks	Tied to Eventstream & Fabric
Use Case	IoT, logs, apps	Monitoring, alerting
Query	Routing rules	KQL
Scale	Millions/sec, low latency	High concurrency

EVENT STREAM

DATA SOURCES ;

we can use sample data (like Bicycles) for practice

Recommended View all sources

- Azure Event Hubs Azure resources Connect
- Azure IoT Hub Azure resources Connect
- OneLake events Fabric events Connect

Pull data from your Azure Event Hubs and stream it into Fabric.

Pull data from your Azure IoT Hub and stream it into Fabric.

Route your Fabric OneLake events, such as creation or deletion, to Fabric.

Search your Microsoft sources

Source Subscription Resource group Region

Source	Subscription	Resource group	Region
Azure SQL DB (CDC)	CIC-Prod	clinic-interconnect-prod-us2e...	East US 2
Azure SQL DB (CDC)	CIC-Prod	synapseworkspace-managedr...	West Europe
Azure Service Bus	CIC-Prod	clinic-interconnect-prod-us2e...	East US 2
Azure Cosmos DB (CDC)	CIC-Prod	tikku	West US 2
Azure SQL DB (CDC)	CIC-Prod	clinic-interconnect-prod-us2e...	East US 2

Find source

- Azure Cosmos DB (CDC)
- Azure Data Explorer
- Azure Event Hubs Namespace
- Azure IoT Hub
- Azure MySQL DB (CDC)
- Azure Service Bus
- Azure SQL DB (CDC)
- Azure SQL MI DB (CDC)
- PostgreSQL DB (CDC)

Clear all

DATA PROCESSING BY EVENT STREAM

DemoEventstream

Bikes

DemoEventstream

Filter

Set-up required

Predefined operation

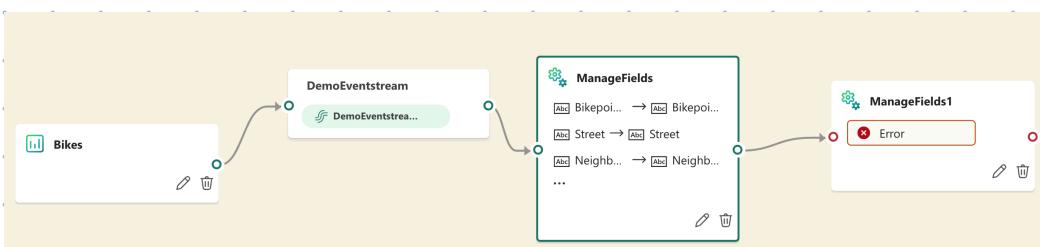
- Aggregate
- Expand
- Filter
- Group by
- Join
- Manage fields
- Union

Destinations

- Lakehouse
- Eventhouse
- Activator
- Stream

Used to Edit Fields

NOTE ; Double Datatype is used to store Decimal values



New field

Field ①

Select field

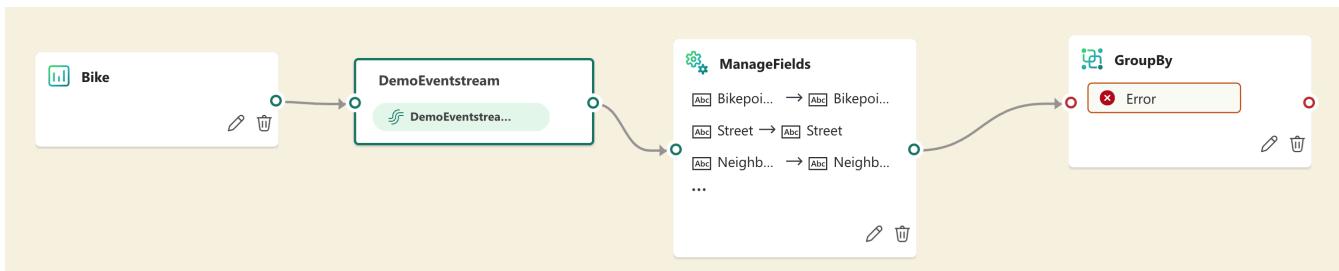
Search

- > Imported Schema
- > Built-in Date Time Function
- > Built-in String Function
- > Built-in Mathematical Function

Inbuilt functions

Note ; Groupby is important

Group BY TRANSFORM EVENT



Settings

Group aggregations by (optional)

Time window [Learn more](#)

Tumbling

Duration *① Of Data*

5 Second

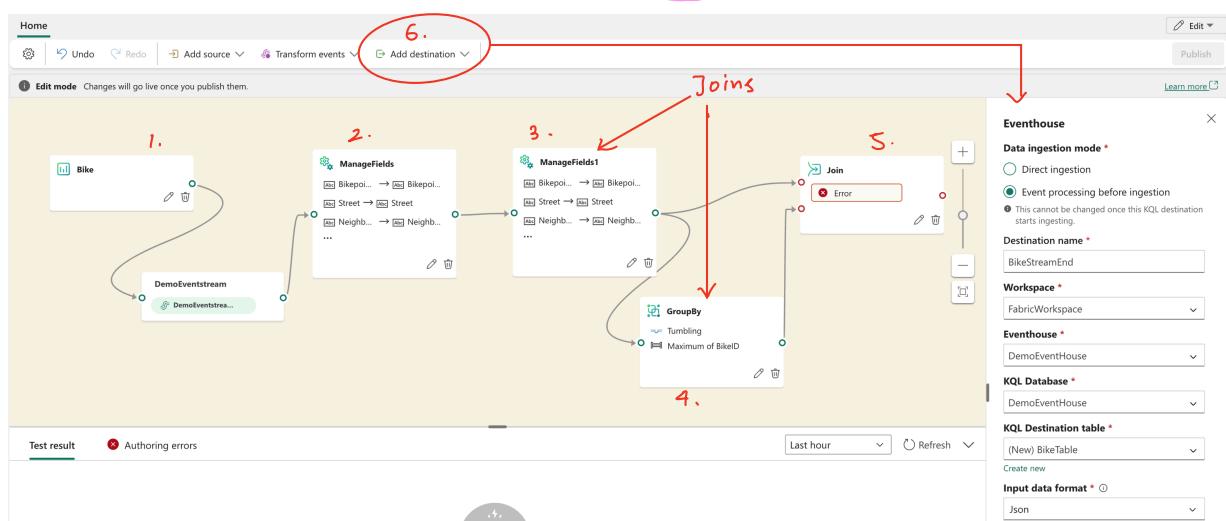
Offset *② except*

0 Second

Window Type	Description	Overlap	Example Use Case
Tumbling Window	Fixed-size, non-overlapping time-based windows.	No	Count events every 5 minutes.
Sliding Window	Fixed-size windows that slide over time, allowing overlaps.	Yes	Compute moving average over last 10 minutes, every 1 min.
Session Window	Dynamically sized based on inactivity gaps between events.	Varies	Group user sessions with 15-min inactivity timeout.
Hopping Window	Fixed-size windows that start at regular intervals and may overlap.	Yes	Count overlapping metrics every 5 mins over 15-min data.
Snapshot Window	Captures a "snapshot" of the stream at a particular point (e.g., time/event).	No	Capture metrics exactly at end-of-day or a trigger event.

Same Timestamp

Completing EventStream



Note: once destinations is added we can click on publish we can Activate / Deactivate stream.

PUBLISHED DATA IN EventHOUSE

The screenshot shows the EventHouse interface. On the left, there's a sidebar with 'DemoEventHouse' selected. Under 'KQL databases', 'DemoEventHouse' is listed. Under 'Tables', 'BikeTable' is selected, showing its schema: left_BikepointID, left_Street, left_Neighbourhood, left_Latitude, left_Longitude, left_No_Bikes, left_No_Empty_Docks, left_BikeID, right_Neighbourhood, right_MAX_BikeID, and right_Window_End_Time. The main area displays the results of a KQL query:

```
(Shift+Enter) +  
1 BikeTable  
2 | project left_Street
```

The results show a single row of data for the 'BikeTable' table.

ACTIVATOR

Monitor Data & activate alerts & actions

ALERTS IN FABRIC

Component	Alert Capability	Trigger Type	Integration/Output
Power BI Dashboards	Tile-based alerts (KPI, card, gauge)	Value thresholds	Email, Power Automate
Eventhouse (KQL DB)	Alerts on query results using KQL & schedules	Query evaluation	Azure Monitor, Email, Logic Apps
Eventstream	Real-time conditions & routing logic	Incoming event conditions	Logic Apps, Eventhouse, Lakehouse, Functions
Data Activator	Real-time data condition monitoring	Any data source change	Email, Teams, Power Automate, Webhook

Also, we can set in KQL

TRIGGER ON BLOB STORAGE

Pipeline → Trigger → Blob Events

OPTIMIZE EVENTSTREAMS can change throughput



TestEventStream

Eventstream

About

Endorsement

Retention

Event throughput

Event throughput

Select the event throughput level for this eventstream's ingress and egress. The eventstream will optimize performance for its sources and destinations based on the selected level. [Learn more](#)

Low (< 10MB/s)

Medium (10MB/s - 100 MB/s)

High (> 100 MB/s)

ⓘ Throughput level can only be upgraded into a higher level. Once applied, it cannot be downgraded.

Apply

Optimize EventHouse; can change consumption

The screenshot shows the EventHouse portal interface. On the left, there's a sidebar with 'Eventhouse' selected. In the center, there's a 'System overview' section with various metrics like storage usage (Original size: 1 GB, Compressed size: 352.2 MB) and ingestion rates (52 databases). A red arrow points from the 'Minimum consumption' link in the top navigation bar to a modal window titled 'Minimum consumption'. The modal explains what it does and lists options for 'Minimum available capacity unit (CU)' sizes, ranging from 'On demand (Charging)' to 'Extra extra large (50 CUs)'.

CHOOSE BETWEEN NATIVE, FOLLOWED STORAGE & SHORTCUT

For Real time Intelligence Data.

ADD OTHER INTERNAL DB (shortcut)

The screenshot shows the KQL databases page with 'EventHouseDB' selected. A 'New Database' dialog is open, showing options for 'Type' (selected: 'New database (default)', others: 'New database (follower)', 'New shortcut database (follower)'). A red arrow points from the 'follower' option to a note: 'Either KQL DB ADX.' To the right, a 'New database shortcut / New' dialog is shown with 'Source cluster URI' set to 'https://trd-xvdwd8q3n1j5r41th.z5.kusto.fabric.microsoft.com', 'Database' set to 'DemoEventHouse', and 'Cache policy (days)' set to '36500' (labeled as 'Century').

Query Both new/old KQL DBs

Automotive
| distinct pickup_datetime, pickup_longitude

database('EventHouseDB').Table
| project columns

Important

- New DB is also Read only
- Data is Referenced not copied

ADD EXTERNAL DB;

The screenshot shows the 'Database' tab in the EventHouse portal. Under 'firsteventhouse', 'OneLake shortcut' is circled in red. On the right, a 'New shortcut' dialog is open, showing 'Internal sources' (Microsoft OneLake Fabric) and 'External sources' (Amazon S3, Amazon S3 Compatible, Azure Data Lake Storage Gen2, Google Cloud Storage, Dataaverse). A red arrow points from the 'External sources' section to a note: 'External tables will be added as shortcut and can be KQL = externalTable ("Dimproduct")'.

Important; External Tables will be added as shortcut and can be KQL = externalTable ("Dimproduct")

Question 1:

I want to create an Activator.

What data source can I NOT use in the Activator?

- Files being added into a SharePoint folder.**

Correct ✓

That's right. The data sources include various databases using CDC, Azure Blob Storage events and Fabric Events (including OneLake).

[+ Explain with AI](#)

Question 3:

What is the default retention period for an eventstream?

- 1 hour

- 1 day**

Correct ✓

That's right. You can set it anywhere between 1 and 90 days.

[+ Explain with AI](#)

Question 4:

I want to adjust the estimated event throughput for an eventstream.

I click on the ... next to the eventstream in the Workspace, go to Settings, and then click on the "Event throughput" tab.

What are the options?

- Up to 10 Kb/s, 10-100 Kb/s, and over 100 Kb/s

- Up to 1 Mb/s, 1-10 Mb/s, and over 10 Mb/s

- Up to 10 Mb/s, 10-100 Mb/s, and over 100 Mb/s**

Correct ✓

[+ Explain with AI](#)

Question 2:

I am in a pipeline.

I click on Home - Add trigger.

What data source do I use?

- An Azure SQL Database using Change Data Capture (CDC)

- An Azure Blob storage account**

Correct ✓

That's correct. Among other events, you can check whether a blob or directory has been created, deleted, renamed, or a file has been put into another tier.

[+ Explain with AI](#)

WORKSPACE

SETTINGS

&

MONITORING

SPARK WORKSPACE

SETTINGS

Workspace settings

Spark settings

Configure and manage settings for Spark workloads and the default environment for the workspace.

Default pool for workspace

Use the automatically created starter pool or create custom pools for workspaces and items in the capacity if the setting Customize compute configurations for items is turned off, this pool will be used for all environments in this workspace.

Starter Pool Default and faster

Pool details

- Node family: Memory optimized
- Node size: Medium
- Number of nodes: 1-2

Customize compute configurations for Items

When turned on, users can adjust compute configuration for individual items such as notebooks and spark job definitions. Learn more about Compute configurations for items.

Power BI

Delegated Settings

Data Engineering/Science

Spark settings

Workspace settings

Edit pool

Spark pool name: DemoPool

Node family: Memory optimized

Node size: Medium

Autoscale: If enabled, your Apache Spark pool will automatically scale up and down based on the amount of activity.

Dynamically allocate executors: Enable dynamic allocation

SKU Name	Default Max Nodes	Max Number of Nodes
F2	1	1
F4	1	1
F8	2	2
F16	3	4
F32	8	8
F64 or Trial	10	16
F128	10	32
F256	10	64
F512	10	128
F1024	10	200
F2048	10	200

Important If I change settings for workspace, then these settings won't affect active sessions

CREATE NEW POOL

Workspace settings

Create new pool

Spark pool name: DemoPool

Node family: Memory optimized

Node size: Small

Autoscale: If enabled, your Apache Spark pool will automatically scale up and down based on the amount of activity.

Dynamically allocate executors: Enable dynamic allocation

Spark settings

Configure and manage settings for Spark workloads and the default environment for the workspace.

Default pool for workspace

Use the automatically created starter pool or create custom pools for workspaces and items in the capacity if the setting Customize compute configurations for items is turned off, this pool will be used for all environments in this workspace.

Starter pool: DemoPool

Number of nodes: 1-2

Workspace pools

- New pool
- DemoPool: Node family: Memory optimized, Node size: Small

NOTE: Once pool is setup, we can start standard session in PySpark

Usually takes 10 Sec
Usually takes 3 mins

How to Allow people to change the pool?

we need to set up an environment

ENVIRONMENT;

You have unpublished changes. To apply these changes to notebooks and Spark job definition run in this environment, select Publish. To save your changes without updating the environment, select Save.

Save **Publish**

Libraries

- Built-in Library
- Public Library
- Custom Library

Spark compute

Acceleration

Compute

Spark properties

Storage

Resources

Spark compute configuration

This configuration applies to all notebooks and spark job definitions in this environment. When you select a pool, the settings in that pool serve as limits for any environment-level pool details you customize.

Environment pool: DemoPool

Pool details

- Node family: Auto (Memory optimized)
- Node size: Small
- Number of nodes: 1-2

Compute

Spark driver core: 4

Spark driver memory: 28GB

Spark executor core: 4

Spark executor memory: 28GB

Dynamically allocate executors: Enable dynamic allocation

Use custom pool

Save & publish

Inside Spark, change env which uses custom pool

OneLake catalog

Choose an environment for this notebook

All My data Endorsed in your org Favorites

Filter by keyword

Name	Owner	Refreshed	Location	Endorsement	Sensitivity
DemoEnvironment	Nadia Khan	—	FabricWorkspace	—	—

Run all Connect PySpark (Python) Environment DemoEnvironment Data Wrangler Copilot

3m

OTHER SPARK SETTINGS

Workspace settings

The screenshot shows the 'Spark settings' section of the workspace settings. It includes tabs for Pool, Environment, Jobs (selected), and High concurrency. A red circle highlights the 'High concurrency' tab. A note at the top says 'This section contains unsaved changes.' Below the tabs, there's a toggle switch labeled 'Off' with a red circle around it. A callout box points to this switch with the text: 'To reduce Spark session start times for individual notebooks, turn on high concurrency settings for notebooks and pipelines in the **High concurrency** tab.' Another callout box points to a 'Time Limit' input field with the text: 'Set Spark session timeout'. The input field shows '20' minutes.

High concurrency means system is designed to handle multiple users or parallel notebook executions efficiently.

DOMAIN WORKSPACE SETTINGS;

Domains are used to organise workspaces.

Settings → Admin Portal → Domains.

NOTE: These settings are specific to Fabric Admin

Important: In Domain we assign workspace or subworkspaces to Domain

example Sales workspace → Sales Domain

we can also use workspace setting to map our workspace to specific Domain

The screenshot shows the 'General' settings for a workspace. It includes sections for License info, Azure connections, System storage, Git integration, OneLake, Workspace identity, Network security, Monitoring, Power BI, Delegated Settings, Data Engineering/Science, and Data Factory. Under the 'Domain' section, there is a note: 'Assign this workspace to a relevant domain to help people discover the content inside it. Each workspace can be assigned to one domain.' A red circle highlights the 'Domain' link. A callout arrow points from this link to the text: 'Assign workspace to Domain.'

APACHE WORKSPACE SETTINGS

Admin → Allow Apache Airflow Jobs

Workspace settings

- General
 - License info
 - Azure connections
 - System storage
 - Git integration
 - OneLake
 - Workspace identity
 - Network security
 - Monitoring
- Power BI
- Delegated Settings
- Data Engineering/Science
- Data Factory
- Apache Airflow runtime settings

Create new pool

Name

Setting name cannot be empty.

Compute node size

Large

Enable autoscale

Extra nodes

0

Create

Cancel

WORKSPACE

ROLES

IN

FABRIC

Role	Can View Content	Can Edit Content	Can Share/Publish	Can Manage Permissions	Can Delete Workspace
Viewer	✓ Yes	✗ No	✗ No	✗ No	✗ No
Contributor	✓ Yes	✓ Yes	✗ No	✗ No	✗ No
Member	✓ Yes	✓ Yes	✓ Yes	✗ No	✗ No
Admin	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes

Add people

FabricWorkspace

Admins, members, and contributors have edit and view access. Viewers only have view access. [Learn more](#)

Enter name or email

Viewer

Add

Admin

Member

Contributor

Viewer

Important Notes.

Notes:

- Viewer:** Can only view reports, dashboards, and other items.
- Contributor:** Can edit or create items but can't publish apps or change permissions.
- Member:** Can do everything except manage access or delete the workspace.
- Admin:** Full control over everything in the workspace.

MANAGE

Access

TO

FABRIC

ITEMS;

LAKEHOUSE

WAREHOUSE

NOTEBOOK



Manage Access

Grant people access

DemoLakehouse

People you share this Lakehouse with can open it and its SQL endpoint and read the default dataset. To allow them to read directly in the Lakehouse, grant additional permissions.

Enter a name or email address

Additional permissions

- Read all SQL endpoint data
- Read all Apache Spark and subscribe to events
- Build reports on the default semantic model

Notification Options

- Notify recipients by email

Grant people access

DemoWarehouse

People you share this warehouse with can connect to it. To give additional permissions, select them from the list.

Enter a name or email address

Additional permissions

- Read all data using SQL (ReadData)
- Read all OneLake data (ReadAll) and subscribe to events (SubscribeOneLakeEvents)
- Build reports on the default semantic model (Build)
- Monitor queries (Monitor)
- Audit queries (Audit)
- Share granted permissions (Reshare)

Grant people access

DemoNotebook

You are granting read permissions to this notebook to the following recipients.

Enter a name or email address

Additional permissions

- Share
- Edit
- Run

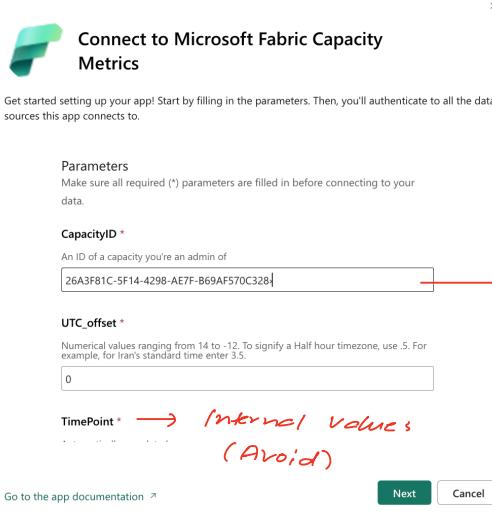
Notification Options

- Notify recipients by email

FABRIC CAPACITY METRICS APP

Power BI App - Fabric Capacity Metrics

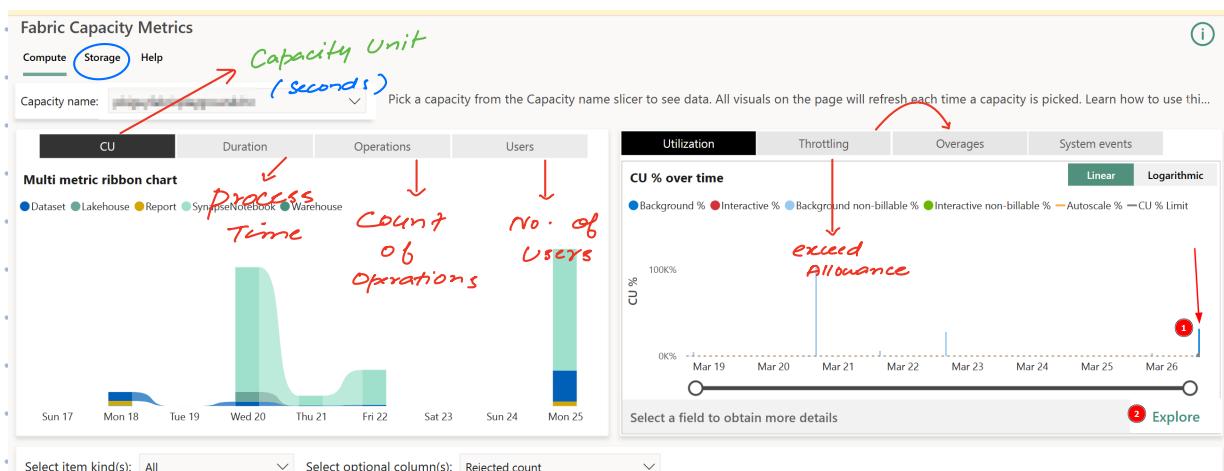
→ This allows to choose plan resp.



Admin portal
↓
capacity settings
↓
Action

Capacity ID

APP INTERFACE:



SEMANTIC MODEL REFRESH

→ Semantic Models → Refresh History

Refresh history

Scheduled	OneDrive	Direct Lake	OneLake Integration		
Details	Type	Start	End	Status	Message

Refresh → This is used to check refresh history of SharePoint files

Close

IMPLEMENT WORKSPACE LOGGING

Workspace settings

Successfully added the monitoring Eventhouse and turned on logging.

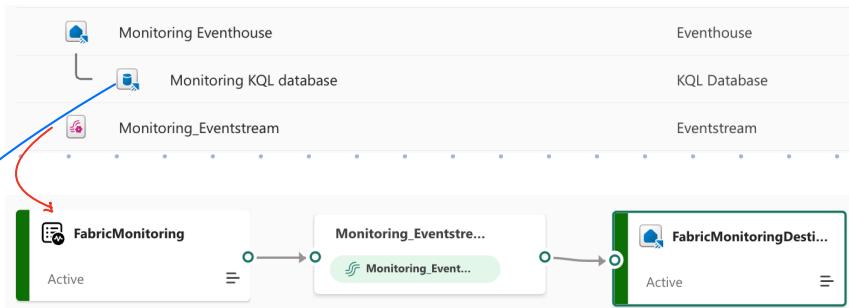
Monitoring

Monitor workspace activity to gain insights into workspace performance.

Log workspace activity On
With logging on, workspace activity data is collected and stored in the read-only monitoring QL database within the monitoring Eventhouse. Users can query the database for performance and diagnostic information.

Monitoring database link
Select the link to open the monitoring QL database within the monitoring Eventhouse.
[Monitoring database](#)

Delete monitoring Eventhouse
Deleting the monitoring Eventhouse also deletes the monitoring QL database and any previously collected data. To monitor workspace activity again, add a read-only monitoring Eventhouse to the workspace and turn on logging. The QL database is automatically created in this process.
[Delete database](#)



MONITOR KQL DATABASE ;

Monitoring Eventhouse

- System overview
- Databases
- Monitoring (Coming soon)

Monitoring KQL database

Data Activity Tracker

Ingestion 7d: Ingestions: 62, Inqueries: 9, Queries: 9, Last run: 7:11 PM

Tables Data preview Query insights - top 500 queries

Table	Ingestion 7d						
RawLogs	N/A	N/A	5x	N/A	12.3x	N/A	12.3x
SemanticModelLogs	0	0	0	0	0	0	0
EventhouseMetrics	0	0	40	0	0	0	0
GraphQLMetrics	0	0	0	0	0	0	0
EventhouseDataOperationsLogs	0	0	0	0	0	0	0
EventhouseCommandLogs	0	0	0	0	0	0	0

NOTE we can use KQL to query workspace logs

WORKSPACE LOGGING DASHBOARDS

Download

tajamulkhan / Azure-Data-Engineer

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main Azure-Data-Engineer / DP 700 - Fabric Data Engineer / Fabric Workspace Monitoring /

Type ↗ to search

6ae7efb · 23 minutes ago History

Last commit message Last commit date

Name	Last commit message	Last commit date
..		
Fabric Workspace Monitoring Dashboard.json	add	23 minutes ago
Fabric Workspace Monitoring.pb1	add	23 minutes ago

REAL TIME DASHBOARD

Here use json file

Add Data Source
(KQL Database)

change to
KQL Database

2nd Option

Use parquet file, and then populate with

URI ID

The data will be loaded and
visuals will appear.

IMPORTANT

1. workspace Retention period
Default = 7 Days
2. Contributor Role = To Read Lakehouse
Viewer Role = To Read SQL Endpoint
3. CU in monitoring app = capacity unit
4. workspace monitoring logs are retained for 30 Days

5.
 - I am using a KQL Database.
 - I create a database shortcut (follower) to the database `OtherDB` and the table `OtherTable`.
 - What KQL code can I use to query the `OtherTable`?

`OtherDB.OtherTable`

`database('OtherDB').table('OtherTable')`

That's right.

+ Explain with AI

Correct ✓

SECURITY

&

GOVERN

Data Governance & Security is covered by Microsoft Purview

APPLY SENSITIVITY LABELS

By default we have, Confidential, No sensitivity or None. No sensitivity is default label

Label	Description	Encryption	Access Restriction
Highly Confidential	Top-level protection for critical or regulated data.	<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> Strict (No external share)
Confidential	For sensitive internal data.	<input checked="" type="checkbox"/> Optional	<input checked="" type="checkbox"/> Limited (Internal only)
No Sensitivity	Default label; no special protection.	<input type="checkbox"/> No	<input type="checkbox"/> None
None	No label assigned.	<input type="checkbox"/> No	<input type="checkbox"/> None

Q,, How to add custom sensitivity labels

Purview - Information - sensitivity - Add +

(choose level of access)

Q,, Where can we apply labels

- Power BI Desktop
- Power BI Service

ENDORSEMENT

Endorsement Type	Purpose	Visibility Boost	Who Can Set It	Usage Example
Promoted	Highlights content as useful or recommended	<input checked="" type="checkbox"/> Yes (search ranking)	Workspace Admins, Contributors	A well-built report or reusable dataset
Certified	Officially approved content, organization-wide	<input checked="" type="checkbox"/> Highest	Admins or Designated Certifiers	Company-wide sales dashboard
None	Default state with no endorsement	<input type="checkbox"/> No boost	—	Work-in-progress or ad-hoc content

SECURITY IN DATAWAREHOUSE

Create Warehouse → Create Function

Row LEVEL SECURITY

```
CREATE TABLE tblActual
(
Country VARCHAR(20),
Location VARCHAR(20),
Actual INT,
UserName VARCHAR(20)
)

INSERT INTO tblActual (Country, Location, Actual, UserName) VALUES
('England', 'London', 5000, 'Susan'),
('England', 'Birmingham', 7000, 'Susan'),
('England', 'Manchester', 11000, 'Susan'),
('France', 'Paris', 4000, 'Jane'),
('Italy', 'Milan', 3000, 'Jane'),
('Italy', 'Rome', 13000, 'Thomas');

SELECT USER_NAME() → This gives logged in Username

CREATE FUNCTION securityfunction(@UserName AS VARCHAR(50))
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN
SELECT 1 AS securityfunction
WHERE @UserName = LEFT(USER_NAME(), LEN(@UserName)) OR LEFT(USER_NAME(), 4) = 'Jane'

CREATE SECURITY POLICY SecurityPolicy
ADD FILTER PREDICATE dbo.securityfunction(UserName)
ON dbo.tblActual
WITH (STATE = ON);

SELECT * FROM tblActual
```

COLUMN LEVEL SECURITY

```
CREATE TABLE tblActual
(
Country VARCHAR(20),
Location VARCHAR(20),
Actual INT,
UserName VARCHAR(20)
)

INSERT INTO tblActual (Country, Location, Actual, UserName) VALUES
('England', 'London', 5000, 'Susan'),
('England', 'Birmingham', 7000, 'Susan'),
('England', 'Manchester', 11000, 'Susan'),
('France', 'Paris', 4000, 'Jane'),
('Italy', 'Milan', 3000, 'Jane'),
('Italy', 'Rome', 13000, 'Thomas');

SELECT * FROM tblActual;

GRANT SELECT ON tblActual TO [jane@Filecats.onmicrosoft.com]
GRANT SELECT ON tblActual(Country, Location) TO [susan@Filecats.onmicrosoft.com]
```

In production, we will rather create Views CLS X

OBJECT LEVEL SECURITY

Grant
Revoke

We can Grant Select on Tables, View, Proc etc.

```
GRANT SELECT ON tblActual TO [jane@Filecats.onmicrosoft.com]
GRANT SELECT ON tblActual(Country, Location) TO [susan@Filecats.onmicrosoft.com]
REVOKE SELECT ON tblActual TO [susan@Filecats.onmicrosoft.com]

DENY SELECT ON tblActual TO [Microsoft@Filecats.onmicrosoft.com]

SELECT * FROM sys.database_principals → check Users

GRANT SELECT ON tblActual TO [New Group 2]
```

Important To undo Deny, use Revoke Then, Grant.

We can't deny permissions to object owners

Q1 How to Add Security Groups;

Microsoft Azure

Home > Groups | Overview

Quattro Business Support Solutions Pvt Ltd

New group Download groups Preview features

Overview Tutorials

Search your tenant

All groups Deleted groups Diagnose and solve problems

Settings Activity Troubleshooting + Support

Basic information

Total groups	4,401	Dynamic groups	0
M365 groups	1,453	Cloud groups	2,340
Security groups	890	On-premises groups	2,061

Important; we can share our objects (DW's) etc with these Groups and then use Grant, Revoke, Deny on Groups

SECURITY IN LAKEHOUSE

Manage OneLake data access (preview)

Turn on data access roles (preview)

You are about to turn on OneLake data access roles (preview) for this item. Using this capability, workspace admins, members, and contributors are permitted to change OneLake data access permissions at the folder level. Learn more

After OneLake data access roles is applied to DemoLakehouse only users with Write permission can share the data in this item using External data sharing. Any existing external data shares may stop working. Learn more

Continue Cancel

Role name DefaultReader

New role (preview)

Assign role Role name * Included folders All folders Selected folders Tables Folder Demo Shortcut DimCustomer DimProduct DimProductCategory

Assign NewRole (preview)

DemoLakehouse

Assign users to this role. Learn more

Add users

Add people or groups

Enter a name or email address

Add Cancel

Add users based on Lakehouse permissions

Select permissions

Permission	User Count
Read	1 user
Write	1 user
Reshare	1 user
Execute	1 user
ReadAll	1 user

permissions.

DEPLOYMENT PIPELINES;

lets you deploy your data assets (like reports, datasets, notebooks) through 3 stages

- Development
- Test
- Production

CREATE Deployment Pipeline ;

workspaces → Deployment Pipeline

Deployment pipelines

+ New workspace



Add a new deployment pipeline

Use a pipeline to manage your workspace content through the deployment stages, continuously delivering the latest content to your users.

Get started

New pipeline

Start by creating the pipeline you'll use for managing and deploying the workspace content.

Assign 1 workspace

Select your workspace and assign it to the deployment stage development, test, or production.

Develop and 3 your content

Manage, review, and compare your workspace content until the time for releasing to users.

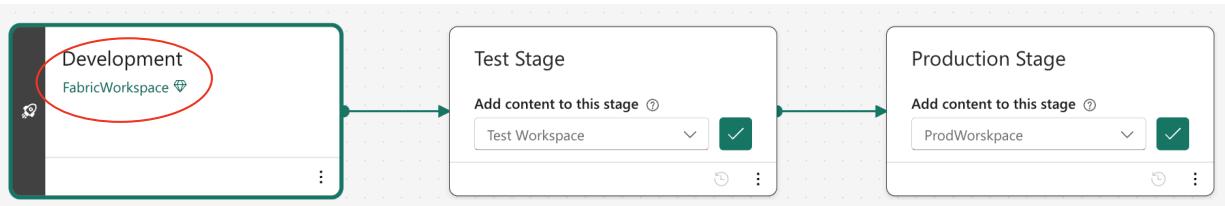
Share 4 your users

Deploy your content to your organization and share it with the team for reviewing and release.



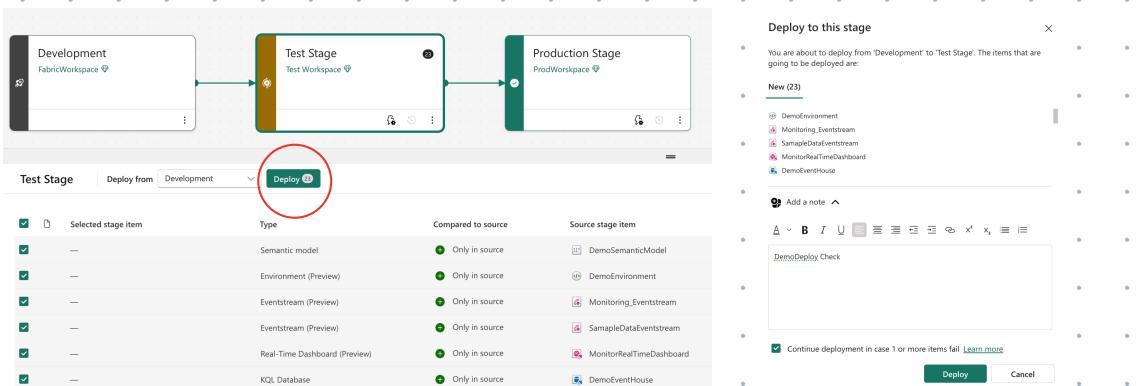
NOTE; we can have upto 10 workspaces and minimum of 2

ASSIGN WORKSPACES



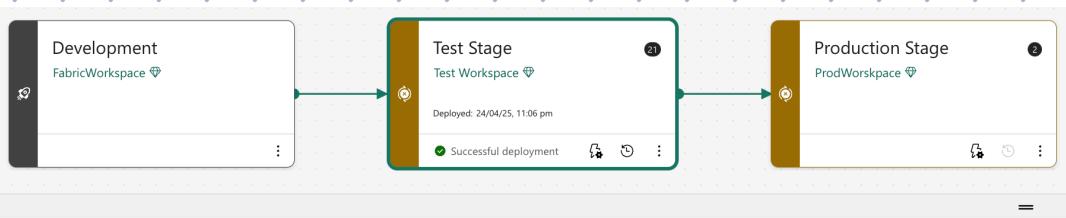
NOTE; It is preferred to have multiple workspaces (Dev, Test, Prod)

DEPLOY FROM DEV TO TEST STAGE



NOTE; Gen2 Dataflow etc is not supported

After Deployment



Test Stage Deploy from Development Deploy

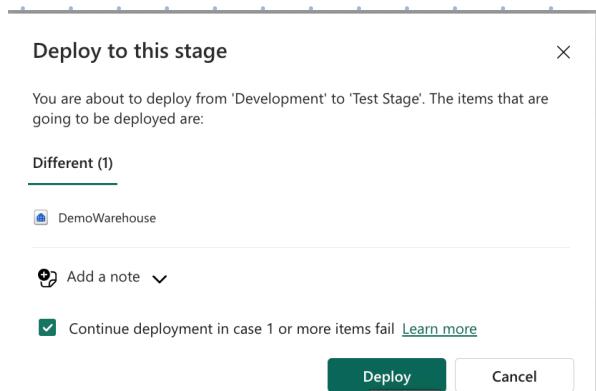
Selected stage item	Type	Compared to source	Source stage item
DemoSemanticModel	Semantic model	Same as source	DemoSemanticModel
DemoWarehouse	Warehouse (Preview)	Same as source	DemoWarehouse

Note; If I make any changes, then

Test Stage Deploy from Development Deploy

Selected stage item	Type	Compared to source	Source stage item
DemoSemanticModel	Semantic model	Same as source	DemoSemanticModel
DemoWarehouse	Warehouse (Preview)	Different from source	DemoWarehouse

while Redeploying only changes will be deployed



Note; Any changes like Rename won't affect as it will be automatically mapped

Configure Deployment Pipelines;

1. Deployment History;

Deployment history					
Select a stage to view deployment history:					
Deployed to	Date and time	Deployed by	Items	Note	ID
Test Stage	24/04/25, 11:06 pm	[Redacted]	[Redacted] + 1	[Redacted]	[Redacted] Added
Test Stage	24/04/25, 11:03 pm	[Redacted]	[Redacted] + 1	[Redacted] Note	[Redacted]

Note; we can't download microsoft pbix file after deployment.

MANAGE Access To Pipeline

The screenshot shows the Power BI Deployment pipelines interface. At the top, there's a navigation bar with 'Power BI Deployment pipelines', a search bar, and various icons. Below the navigation is a pipeline diagram with three stages: 'Development FabricWorkspace', 'Test Stage Test Workspace', and 'Production Stage ProdWorkspace'. Arrows indicate the flow from Development to Test and from Test to Production. The Test stage has a green success status bar at the bottom. On the right side of the screen, a modal window titled 'Add people' is open, showing dropdown menus for selecting workspaces for each stage: 'Development', 'Test Stage', and 'Production Stage'. There are also sections for 'Add or update workspace permissions' and a toggle switch.

Q, What can Pipeline Admin. Do ?

Plan & Implement Deployment Solutions

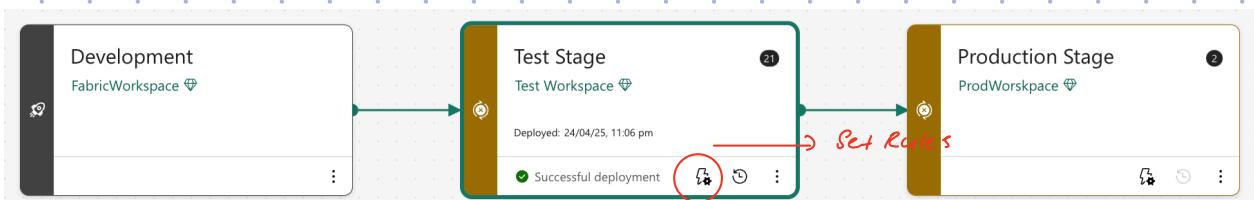
Pipeline Admin can:

- Create, view, edit, share, delete pipelines
- Assign/unassign workspaces to stages
- View deployment history & stage items
- Manage settings & users

If also a Workspace Contributor, they can:

- Compare stages
- Set rules
- Deploy to next stage (if contributor/member/admin of both workspaces)

Setup Rules



Rules can help changing Data sources

THE END

Key Concepts;

1. Warehouses can store both Relational and Standalone tables given they have fixed Schema.

2.

Lakehouse	Warehouse	Eventhouse (KQL Database)
Store and analyze large, mixed data (structured + semi-structured)	Store and query pure structured, relational data	Store and query streaming, time-series, or log data
Structured + Semi-structured (e.g., CSV, Parquet, JSON)	Strictly structured (tables, columns, SQL schema)	Structured event data (logs, metrics, telemetry)
SQL (Apache Spark SQL)	T-SQL (Transact-SQL, like Azure Synapse)	KQL (Kusto Query Language)
Moderate (through Eventstreams or external tools)	Low (batch-oriented, scheduled refreshes)	High (real-time ingestion and querying)
Big Data, AI/ML, combining batch + streaming data	Reporting, BI dashboards, transactional analytics	Monitoring, alerting, real-time dashboards
Optimized for big data workloads (may have some latency)	Optimized for high concurrency, fast SQL analytics	Optimized for lightning-fast query over event data
Cheaper for massive volumes of raw data	Costlier due to high performance/compute for structured queries	Cost-effective for huge volumes of log/telemetry data

3. **MASKING**; Column MASKED WITH (FUNCTION = "default")

4. **QUERY FOLDING**;

Yes — query folding can still happen even if your initial query in Power BI is just:

```
sql          ⌂ Copy ⌂ Edit  
SELECT * FROM TableName
```

Then, if you apply filters in Power Query (like filtering dates or removing nulls), Power BI tries to fold those steps back into a single SQL query like:

```
sql          ⌂ Copy ⌂ Edit  
SELECT * FROM TableName WHERE OrderDate >= '2024-01-01'
```

So even though your original query was `SELECT *`, Power Query is smart enough to push your filter step back to the source — as long as the transformations are foldable.

5. **SQl READ OR WRITE**

Lakehouse = Read only (even in notebook SQL)

Warehouse = Read + write

6. **MERGE**

Dataflows Gen 2 support Merge on multi columns

7. Warehouse doesn't support Varchar (max)

8. In Lakehouse, always upload table data to tables, for efficient retrieval and they can be accessed through SQL Endpoint

9. Shortcut;

Doesn't copy Data but only creates shortcut

10. Save Table in DW;

Select * from Table

↓ (X)

Save as Demo

↓

Takes more
Time

Select * from Table 1

into Table

↓

Faster and efficient

11. Incremental Load;

Small Data = 1 week / 1 Query

Big Data = 7 days / 7 Queries

12. GIT INTEGRATION

Azure Devops Git = Official (more Secure)

GITHUB = Open Source

13. AZURE DEVOPS GIT;

Dashboard = NOT Supported

Data = only Meta Data Copied.

14. KQL = Orders by Desc default

Faster than SQL

Used in ADX or Eventhouse

```
// Selecting Columns in KQL
// Use // for comments
// Use | to separate operators in the pipeline
```

Limit 5 = Limits to 5 Rows

```
Weather // Table name
| project State, EpisodeId, StartTime=StartTime, EndTime, EpisodeNarrative, EventNarrative // Select columns with optional aliasing (StartTime = StartTime)
| extend Duration = EndTime - StartTime // Add calculated column
| extend EpisodeNarrative = "hi" // Overwrite values in the column (not renaming rows)
| order by State asc nulls last // Order results; asc by default, 'nulls last' is valid
| project-rename Narrative = EpisodeNarrative // Rename column: new name = old name
| project-away EventNarrative // Remove column
```

* has is used to match word & contains for string

15. Numeric D-Typos in KQL

`int = 1/2 = 0 (Floor Division)`

`Decimal = 1/2 = 0.5`

16. UNION THE DATA ;

One Query & second should be brackets

```
Weather
| distinct State, EventType
| union kind = outer withsource = TableSource
(Weather
| distinct State, EventNarrative)

union withsource = AdditionalColumn Wea*
```

Default Union = Outer

Inner = Only Column which are common in both tables

17. JOIN IN KUSTO SQL

```
Weather
| join
(Region | extend State = toupper(State))
on State
-----> Can use
on State and on

Weather
| join kind = fullouter
(Region | extend State = toupper(State))
on ($left.State == $right.State)
```

Default join = inner unique

Imp: Keep small table on left side for efficient join

18. ORDERED FLOW OF DATA IN FABRIC

Step	Component	Purpose / Description
1	Data Sources	External data (APIs, databases, files, IoT, logs, etc.) collected from different systems.
2	OneLake	Centralized data lake storage for all data in Fabric (like a single data lake for all).
3	Pipelines <i>(Data hub)</i>	Used to orchestrate data movement and transformations (ETL/ELT).
4	Dataflow Gen2	No-code/low-code data prep tool for cleaning, shaping, and transforming data.
5	Lakehouse	Combines lake storage + structured access using Delta format (best for analytics).
6	Eventhouse	Specialized for ingesting and analyzing real-time, event-based data (e.g., logs/telemetry).
7	Notebooks	For data engineering, Python/Spark/SQL-based processing and advanced transformations.
8	SQL Endpoints	Query interface to access and analyze data from Lakehouse/Eventhouse.
9	Warehouse	Structured data warehouse optimized for BI and analytics (relational model).
10	Power BI	Data visualization and reporting layer to build dashboards and reports.

19. How they work together:

- EventStream brings in real-time events (clicks, purchases, actions)
- EventDB stores these events efficiently for fast querying and reporting
- EventHouse provides the full infrastructure to manage ingestion, storage, and analysis

20. Eventstream = Pipeline for Streaming Data.
21. Eventstream Double Data Type is used to store Decimal.
22. Retention period for Eventstream = 1 Day.
23. Workspace Retention period
Default = 7 Days
24. Contributor Role = To Read Lakehouse
Viewer Role = To Read SQL Endpoint
25. CU in monitoring app = Capacity Unit
26. workspace monitoring logs are retained for 30 days

- 27.
- I am using a KQL Database.
 - I create a database shortcut (follower) to the database `OtherDB` and the table `OtherTable`.
 - What KQL code can I use to query the `OtherTable`?
- `OtherDB.OtherTable`
- `database('OtherDB').table('OtherTable')` Correct ✓
That's right.
[+ Explain with AI](#)

28. How to add custom sensitivity labels
Purview - Information - sensitivity - Add Labels +
29. **Endorsements** ;
- Promoted** : workspace Admin
- Certified** : official Certifiers
30. **Dataflow Gen2** is not supported in deployment pipelines
31. Any changes like Rename won't affect as it will be automatically mapped
32. we can't download microsoft .pbix file after deployment.

33. V-Order improves Read & write speeds.
34. Activator can't be created on Lakehouse.
35. Dataverse is not a destination for Dataflow Gen2
36. Delete Data Activity is unavailable in Incremental Pipeline
37. Refresh can be set Daily or weekly
38. In Maintenance, we can run VACUUM command
39. Easy way to remember windows

Type	Think of it like...
Tumbling	Tiles neatly placed on a floor
Hopping	Overlapping rugs
Sliding	A sliding glass door, moving bit by bit
Session	A casual meetup that lasts until everyone leaves

40. To start Spark Job stream
- df = spark.readStream

41. Below F64 → we can Run Power BI Report

42.

Your selection is correct

The quickest you can schedule a Dataflow Gen2 to run is every 30 minutes.

Your selection is correct

You can schedule a Data Pipeline to run every minute.

43. To Block users from sharing access to personal workspace

Microsoft 365 Admin

43. Minimum Access Level to complete 2 stages in DP

Contributor

44. I want to copy Large Data without Transformation

Copy Tool in Pipeline

45. Importance of Cache & Shortcuts

Simple Analogy:

Imagine you're watching a YouTube video.

- Without caching: You stream it fresh every time → uses a lot of data.
- With caching: It saves the video locally once → next time, no extra internet use.

And **shortcuts** are like **bookmarks** — they point to the real video, but don't take up extra space.

Quick Summary:

- Use **Eventstream + KQL** for **real-time monitoring**.
- Use **Data Pipeline** for **general ETL tasks** (batch or near real-time).
- Use **Apache Spark** for **large data volumes, ML, or custom logic**.
- Use **Dataflow Gen2** for **drag-and-drop transformations**.
- Use **Streaming Dataflow** for **simple streaming data transformation**.
- Use **Lakehouse** for **unified analytics storage**.
- Use **Warehouse** for **Power BI + SQL analytics**.

Simplified Rules of Thumb:

-  **Dataflow Gen2** = No-code batch ETL
-  **Streaming Dataflow** = Low-code **real-time** stream processing
-  **Data Pipeline** = Orchestrating batch or near real-time pipelines
-  **Apache Spark** = Big data, ML, or custom logic (code-heavy)
-  **Eventstream** = Real-time ingestion from external systems
-  **KQL** = Real-time querying/log analytics
-  **Lakehouse** = Store everything (files + metadata)
-  **Warehouse** = SQL + BI-ready structured analytics