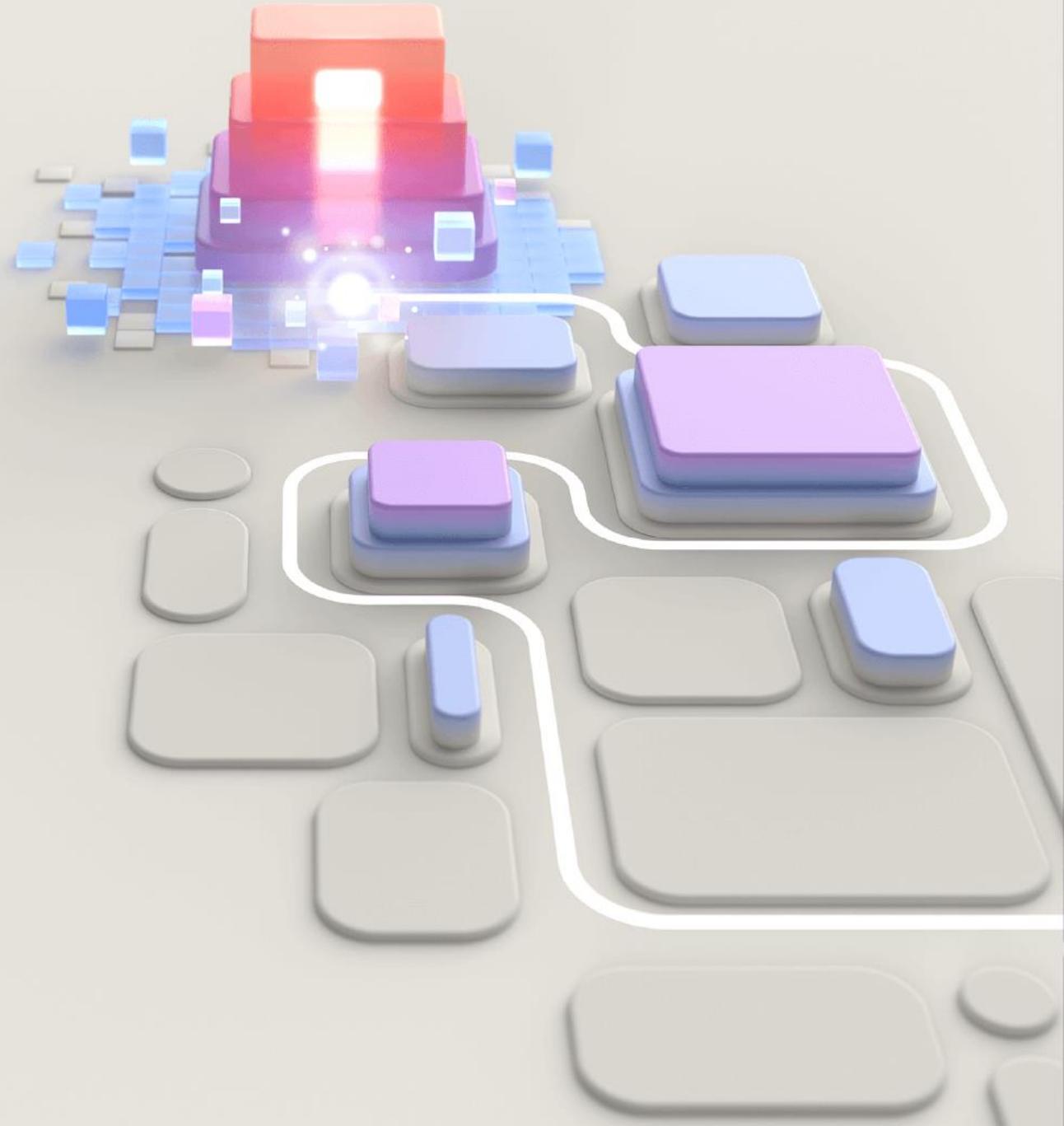


# DP-601: Implement a lakehouse with Microsoft Fabric



# Presenter

## Mohammed Arif, PhD GenAI Architect & Data Scientist



Mohammed Arif has more than eighteen (18+) years of working experience in Information Communication and Technology (ICT) industry. The highlights of his career are more than nine (9) years of holding various senior management and/or C-Level and had six (6) years of international ICT consultancy exposure in various countries (APAC and Australia), specially on Big Data, Data Engineering, Machine Learning and AI arena.

He is also Certified Trainer for Microsoft & Cloudera.





<https://arif.works/dp601>

Explore end-to-end analytics  
with Microsoft Fabric



# Learning objectives

- Describe end-to-end analytics in Microsoft Fabric
- Understand data teams and roles that use Fabric
- Describe how to enable and use Fabric

**But First ..**



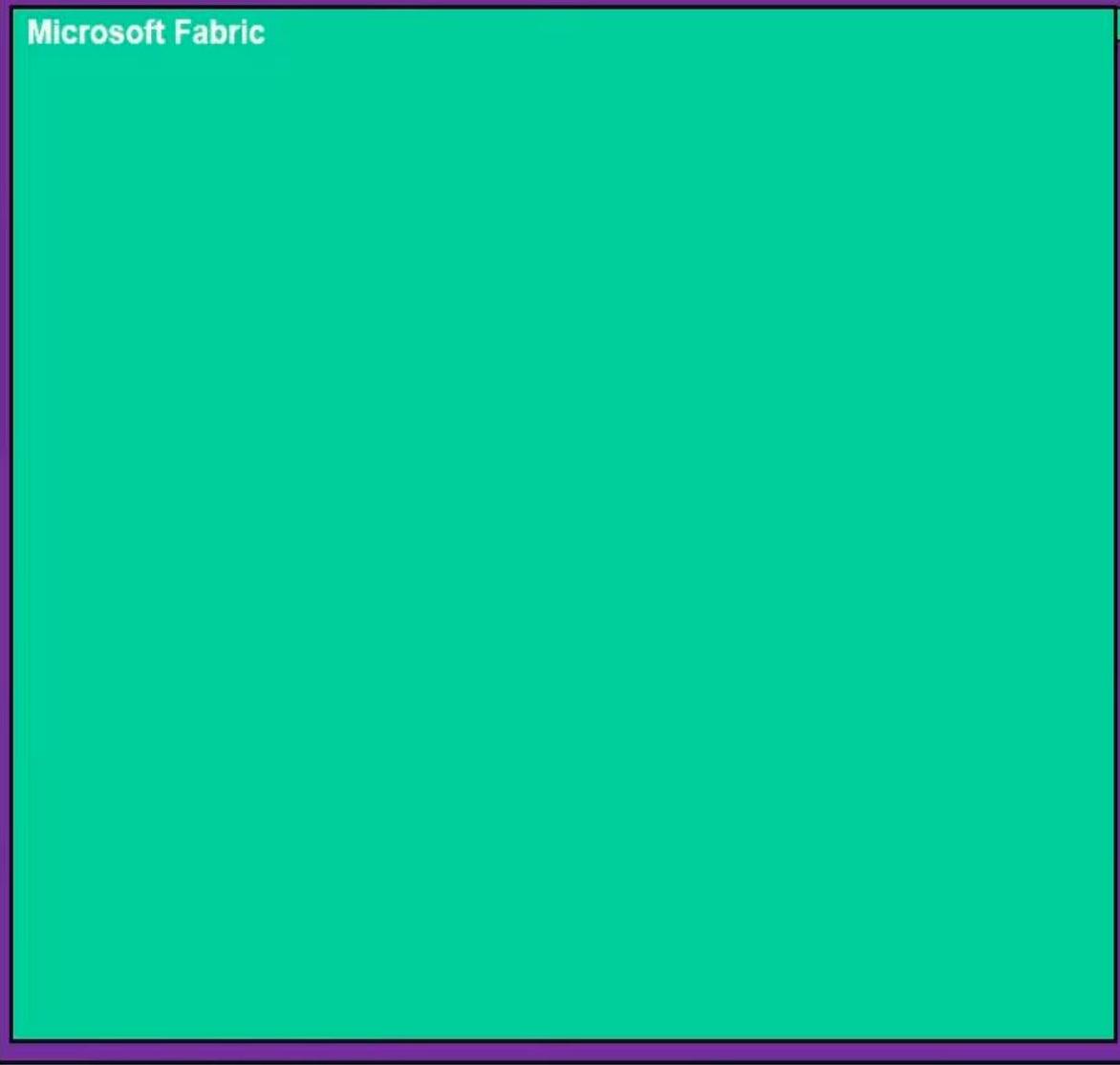
Microsoft Fabric Structure

## Organizational Microsoft Entra Tenant

## Microsoft Entra Tenant

- Identity & access management
- Control center for managing users and security across Microsoft Cloud services

**Organizational Microsoft Entra Tenant**



**Microsoft Fabric**

**Microsoft Fabric**

- Deployed/enabled per Tenant
- Data platform SaaS

**Microsoft Entra Tenant**

- Identity & access management
- Control center for managing users and security across Microsoft Cloud services

# Organizational Microsoft Entra Tenant

## Microsoft Fabric



### Microsoft Fabric

- Deployed/enabled per Tenant
- Data platform SaaS

### Microsoft Entra Tenant

- Identity & access management
- Control center for managing users and security across Microsoft Cloud services

### Workspace

- Containers for different Fabric items
- Uses Capacity for compute

**Organizational Microsoft Entra Tenant**



**Capacity**

- Compute resources
- Main costs

**Microsoft Fabric**

- Deployed/enabled per Tenant
- Data platform SaaS

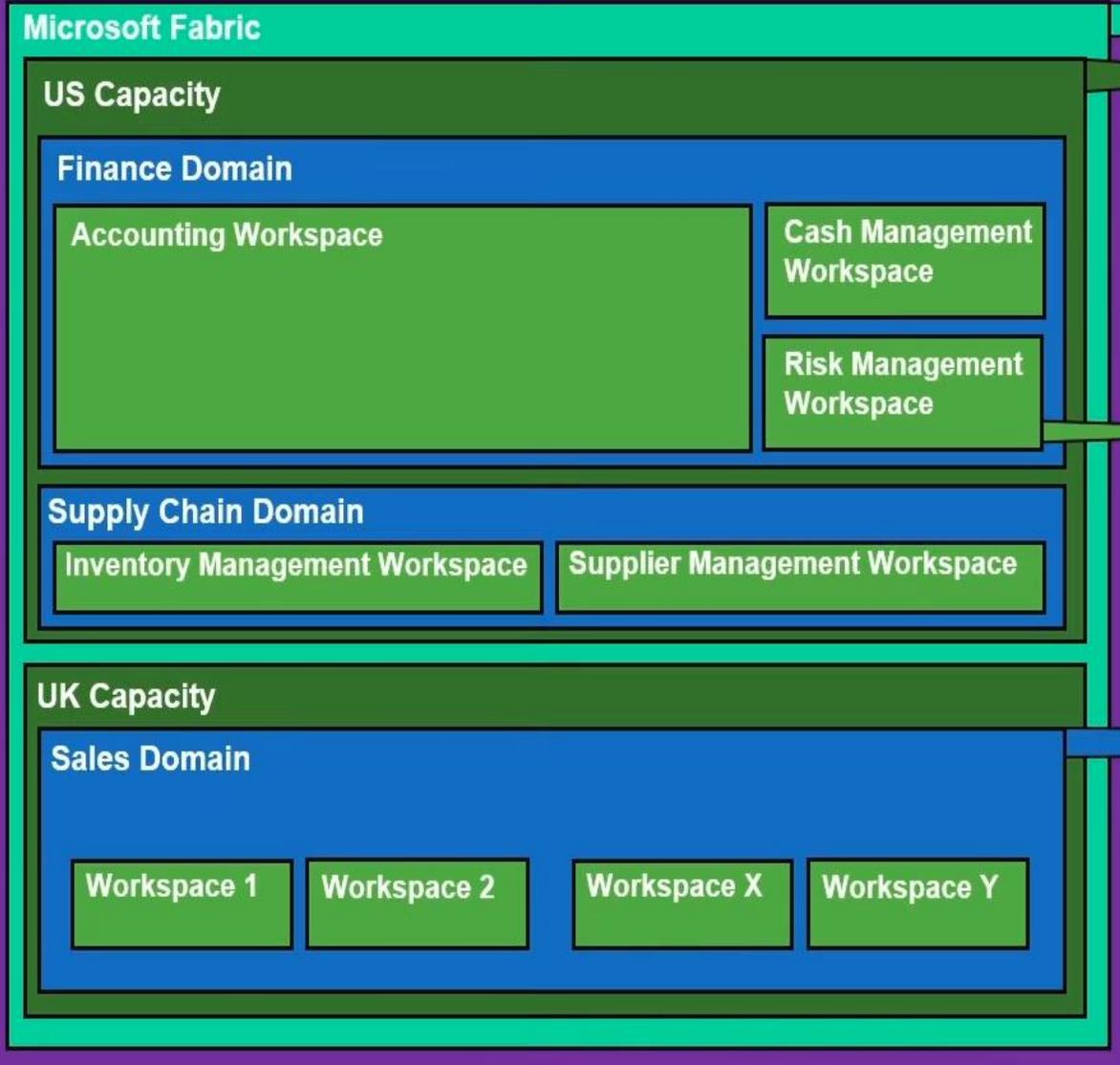
**Microsoft Entra Tenant**

- Identity & access management
- Control center for managing users and security across Microsoft Cloud services

**Workspace**

- Containers for different Fabric items
- Uses Capacity for compute

# Organizational Microsoft Entra Tenant



### Microsoft Entra Tenant

- Identity & access management
- Control center for managing users and security across Microsoft Cloud services

### Microsoft Fabric

- Deployed/enabled per Tenant
- Data platform SaaS

### Capacity

- Compute resources
- Main costs

### Workspace

- Containers for different Fabric items
- Uses Capacity for compute

### Domain

- Logical grouping of workspaces
- Makes possible for different business areas to manage data according specific regulations, restrictions, and needs

# Organizational Microsoft Entra Tenant

## Microsoft Fabric

### US Capacity

#### Finance Domain

Accounting Workspace

Cash Management Workspace

Risk Management Workspace

#### Supply Chain Domain

Inventory Management Workspace

Supplier Management Workspace

### UK Capacity

#### Sales Domain

##### International Sales Subdomain

Workspace 1    Workspace 2

##### Domestic Sales Subdomain

Workspace X    Workspace Y

### Capacity

- Compute resources
- Main costs

### Microsoft Fabric

- Deployed/enabled per Tenant
- Data platform SaaS

### Microsoft Entra Tenant

- Identity & access management
- Control center for managing users and security across Microsoft Cloud services

### Workspace

- Containers for different Fabric items
- Uses Capacity for compute

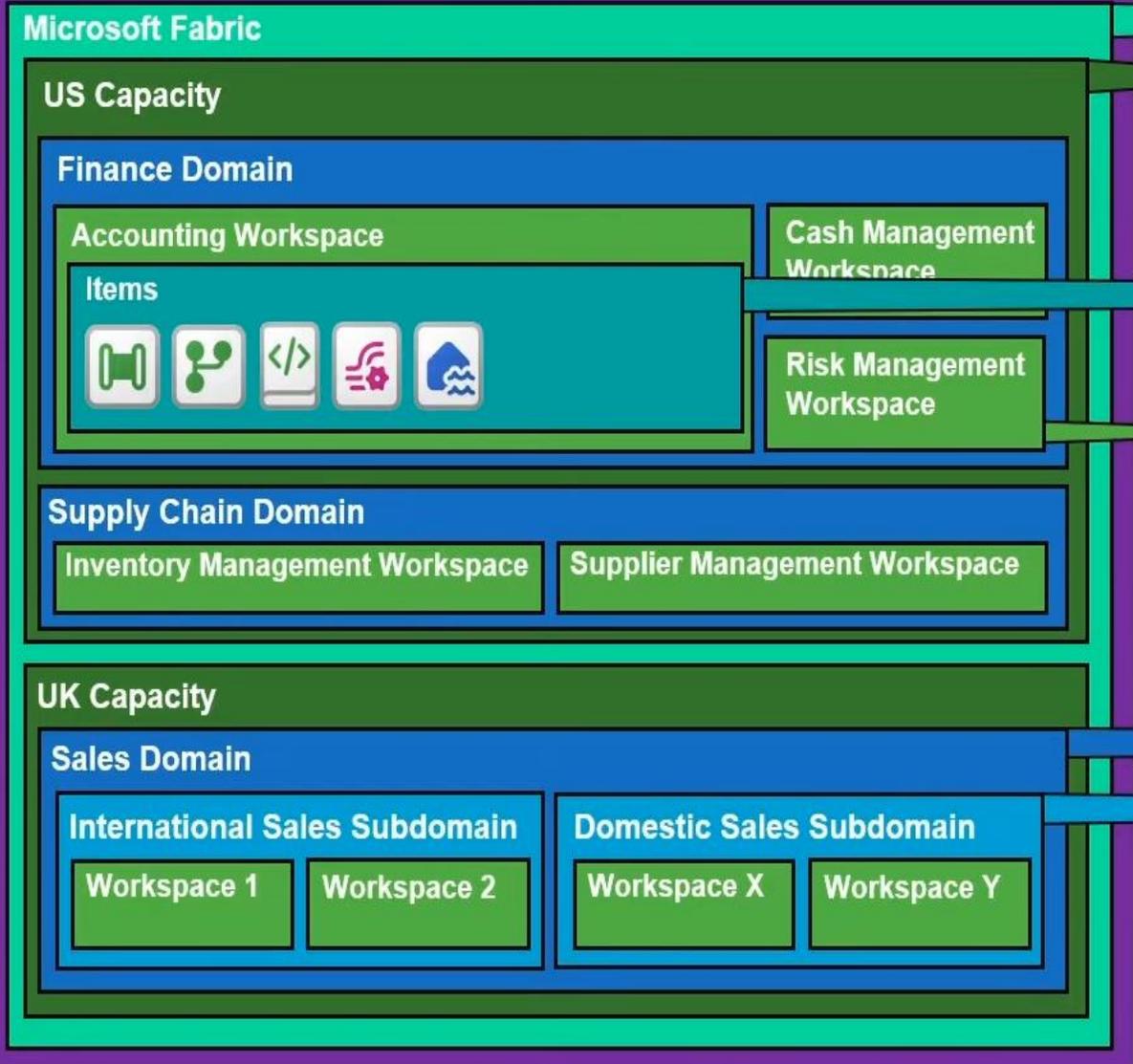
### Domain

- Logical grouping of workspaces
- Makes possible for different business areas to manage data according specific regulations, restrictions, and needs

### Subdomain

- Allows fine tuning of the logical grouping of workspaces and data

# Organizational Microsoft Entra Tenant



**Capacity**

- Compute resources
- Main costs

**Microsoft Fabric**

- Deployed/enabled per Tenant
- Data platform SaaS

**Microsoft Entra Tenant**

- Identity & access management
- Control center for managing users and security across Microsoft Cloud services

**Workspace**

- Containers for different Fabric items
- Uses Capacity for compute

**Items**

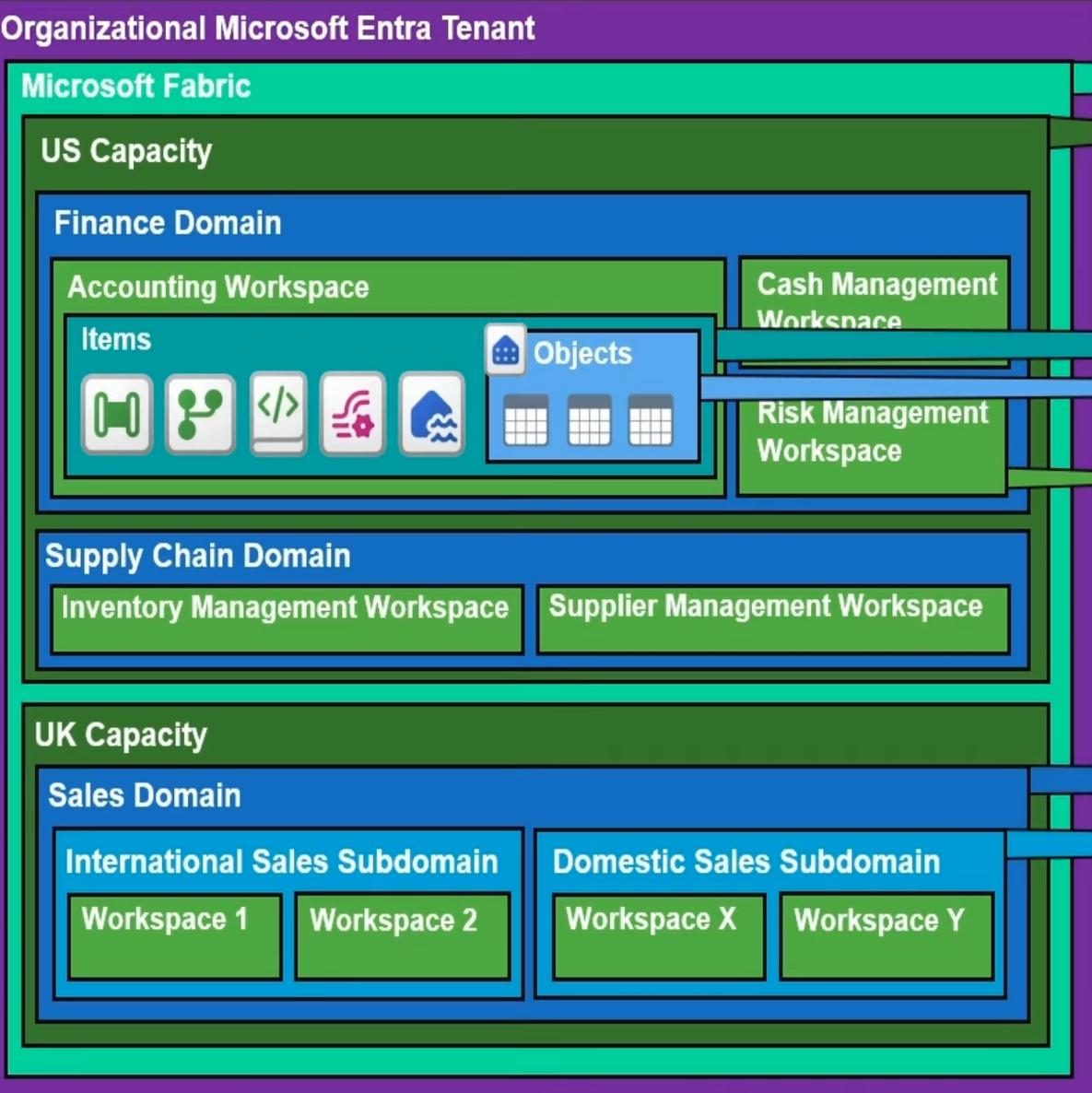
- Created within workspace
- Building blocks of the platform
- Provide different types of capabilities
  - E.g., data storing and transformation

**Subdomain**

- Allows fine tuning of the logical grouping of workspaces and data

**Domain**

- Logical grouping of workspaces
- Makes possible for different business areas to manage data according specific regulations, restrictions, and needs



**Microsoft Entra Tenant**

- Identity & access management
- Control center for managing users and security across Microsoft Cloud services

**Microsoft Fabric**

- Deployed/enabled per Tenant
- Data platform SaaS

**Capacity**

- Compute resources
- Main costs

**Items**

- Created within workspace
- Building blocks of the platform
- Provide different types of capabilities
  - E.g., data storing and transformation

**Objects**

- Created within some items like data stores
- E.g., files, tables, views & stored procedures

**Workspace**

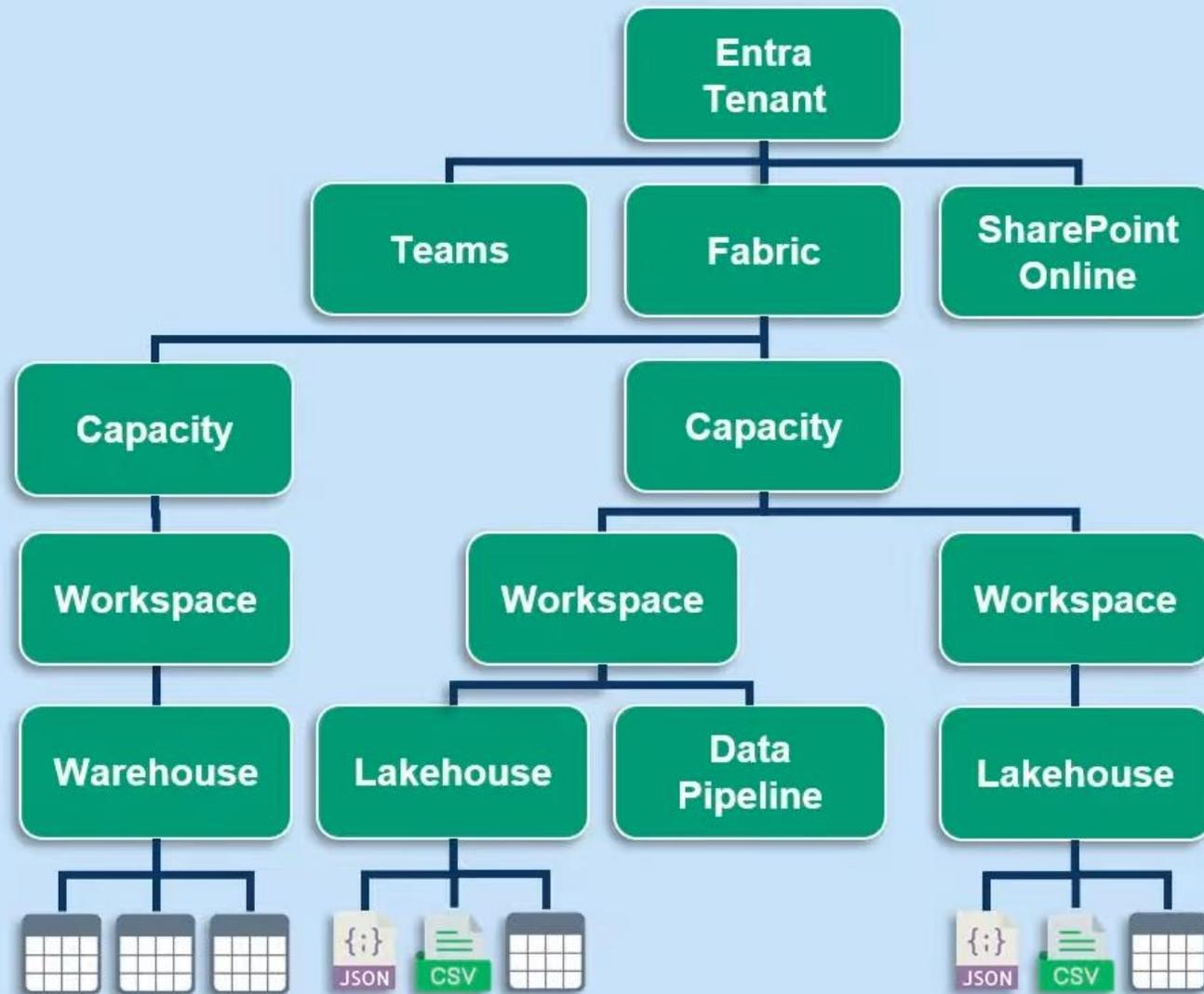
- Containers for different Fabric items
- Uses Capacity for compute

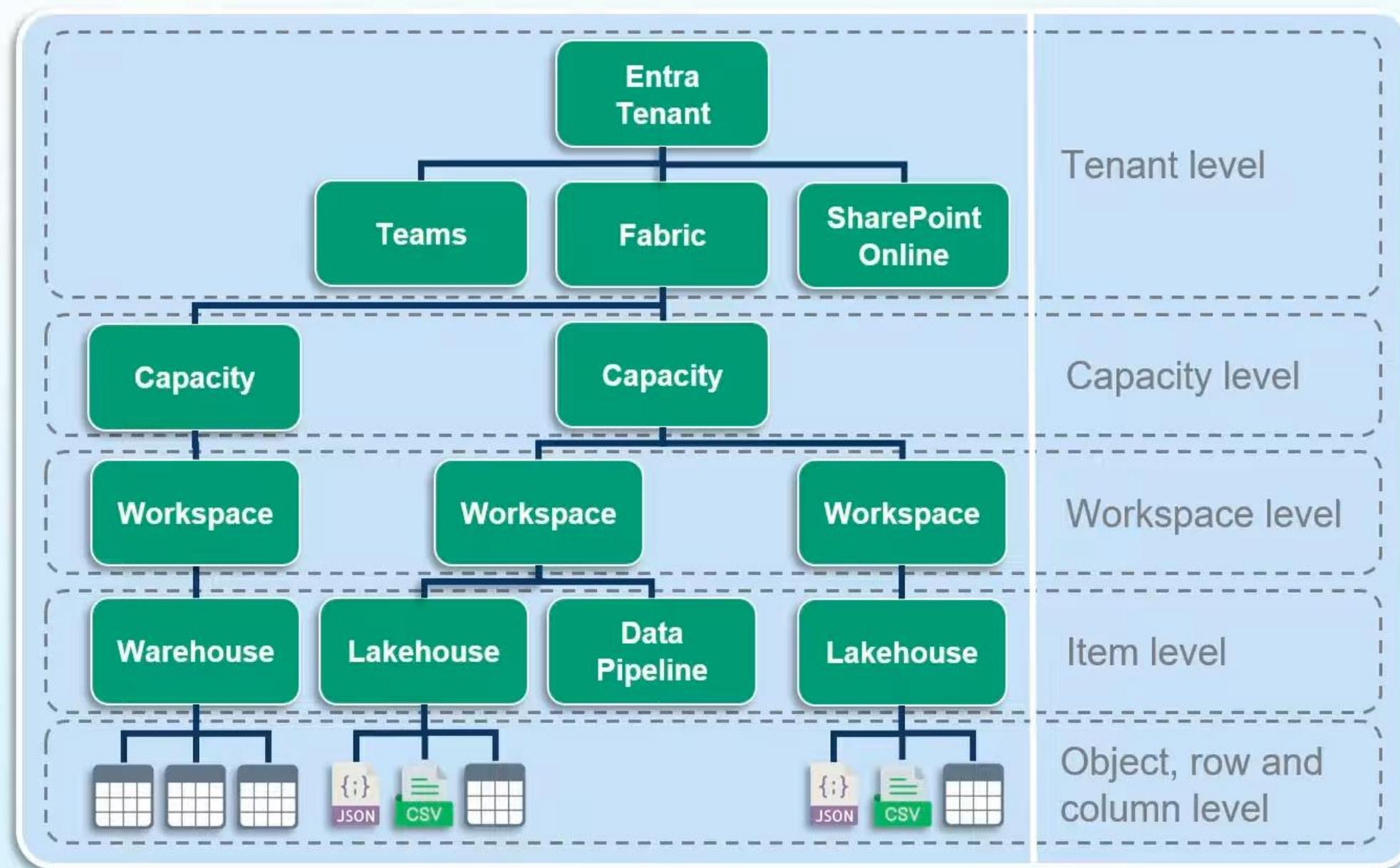
**Domain**

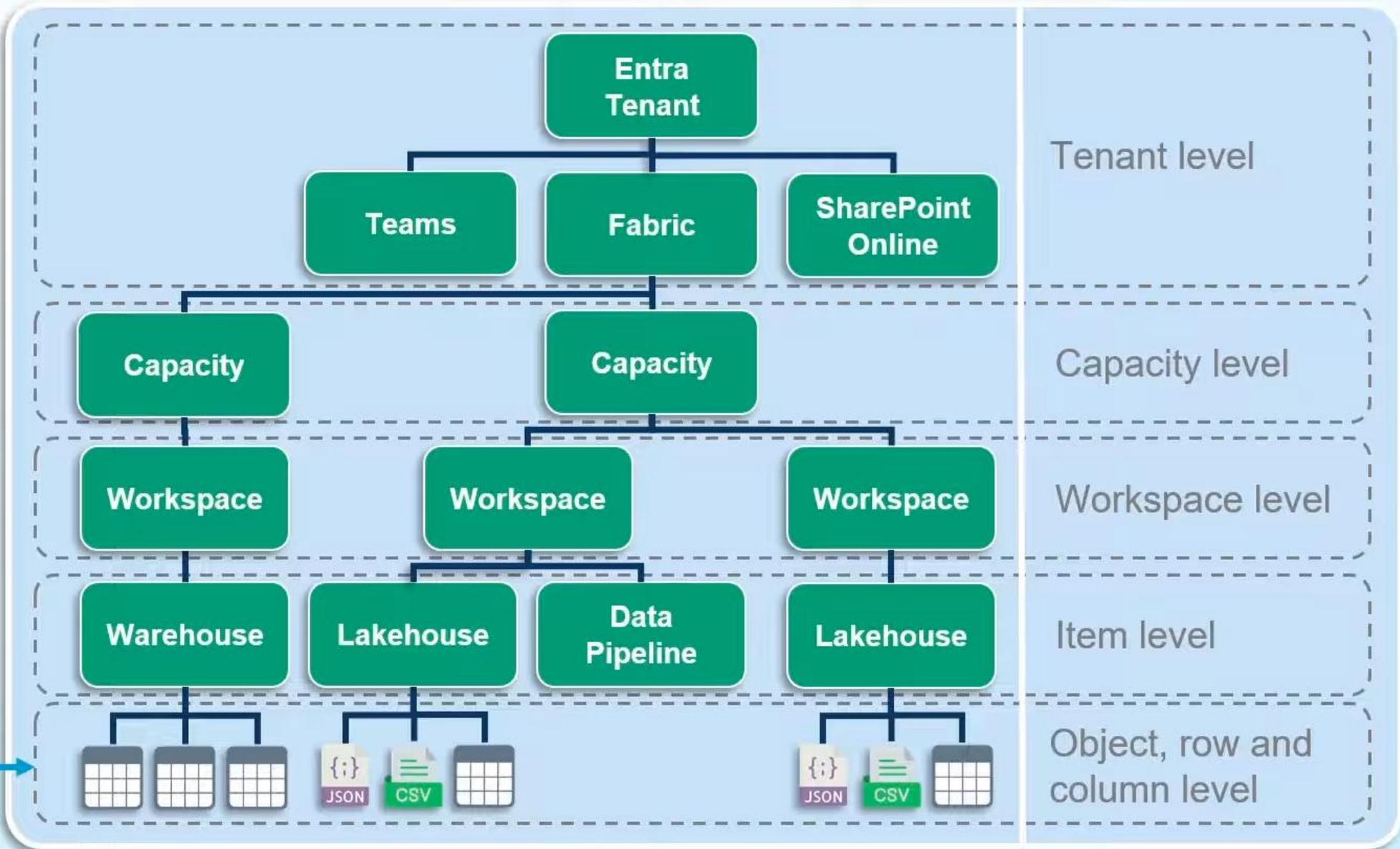
- Logical grouping of workspaces
- Makes possible for different business areas to manage data according specific regulations, restrictions, and needs

**Subdomain**

- Allows fine tuning of the logical grouping of workspaces and data







## Microsoft Fabric

### OneLake

- Unified Data Lake for entire organization
- Only one OneLake per Fabric
- SaaS version of Azure Data Lake Storage
- Can store files and tables
- Uses Delta Parquet files as the default data format
- Storing data generates costs



## Microsoft Fabric

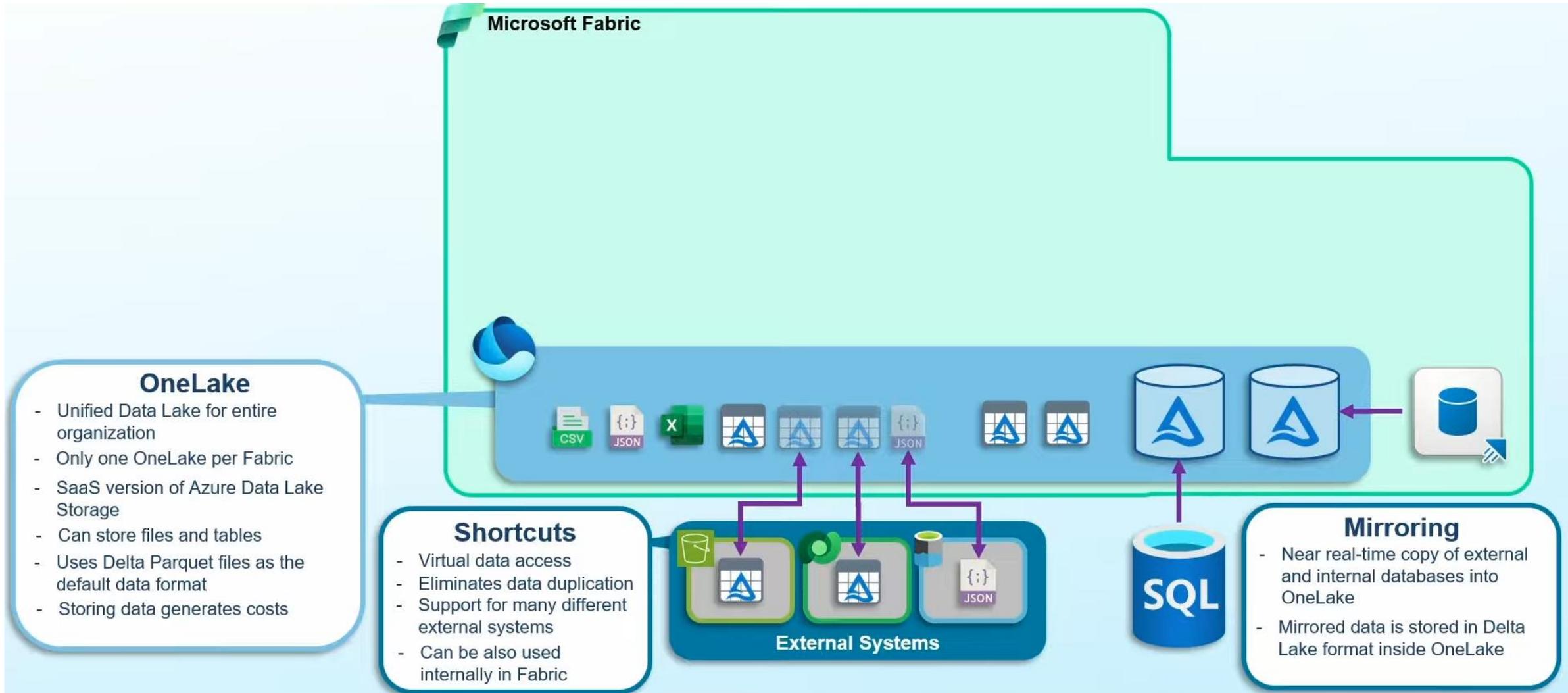
### OneLake

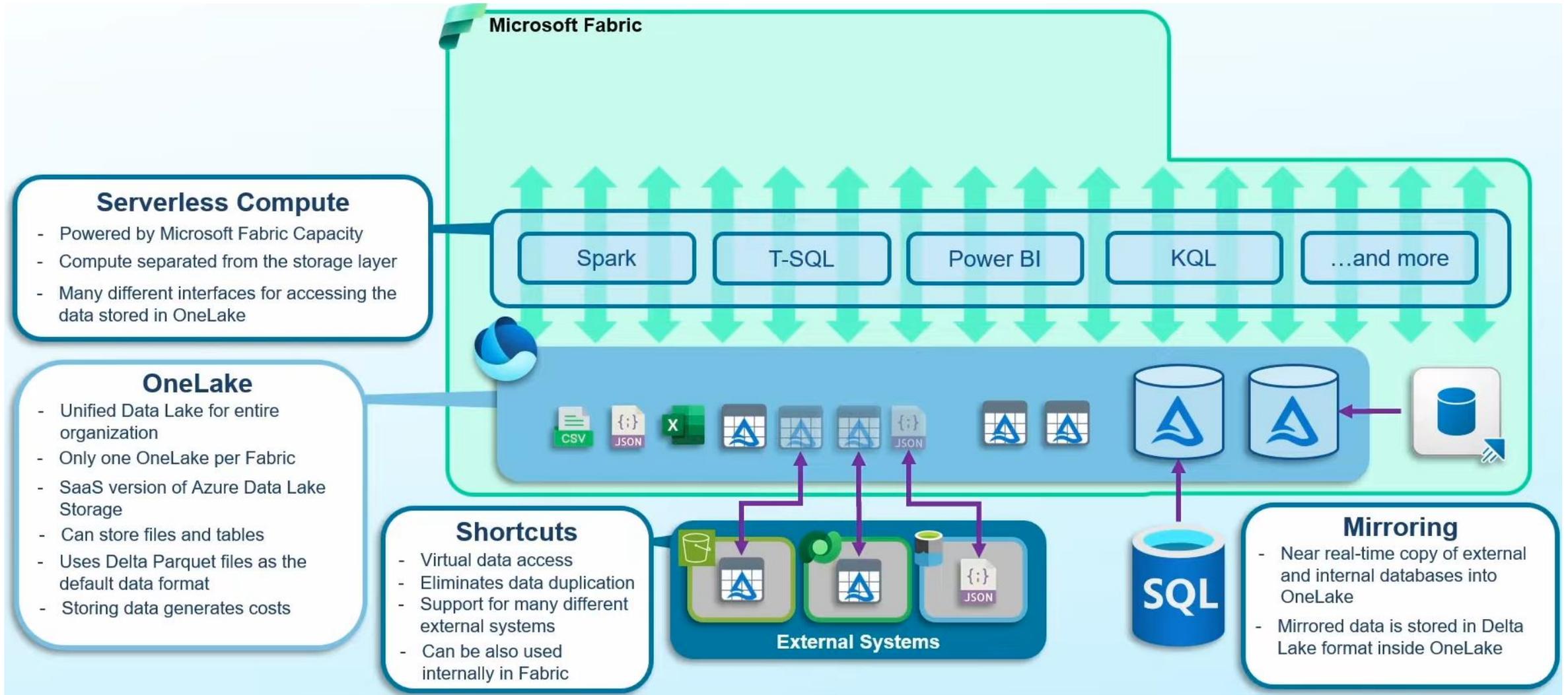
- Unified Data Lake for entire organization
- Only one OneLake per Fabric
- SaaS version of Azure Data Lake Storage
- Can store files and tables
- Uses Delta Parquet files as the default data format
- Storing data generates costs

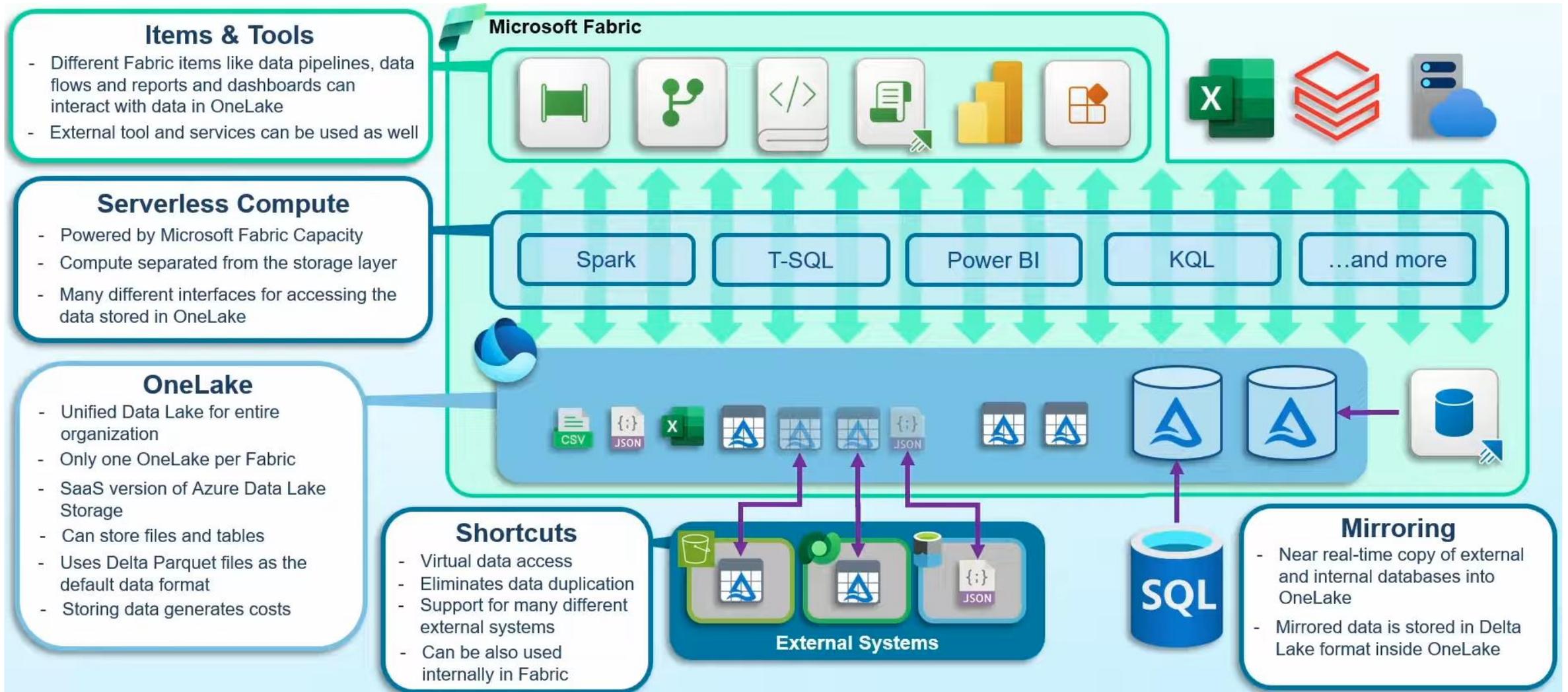
### Shortcuts

- Virtual data access
- Eliminates data duplication
- Support for many different external systems
- Can be also used internally in Fabric









## Customer Success Team



## Data Engineering Team



## IT Team

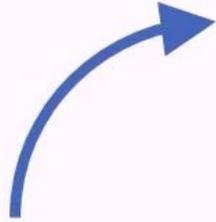
## Data Science Team



## Manufacturing Team



**Step 1:**  
Customer leaves  
a review of a  
product on the  
website



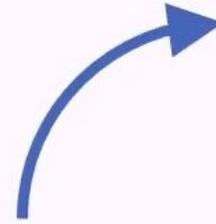
**Step 2:**  
Data is stored in  
an Azure SQL  
database,  
managed by the  
customer  
success team



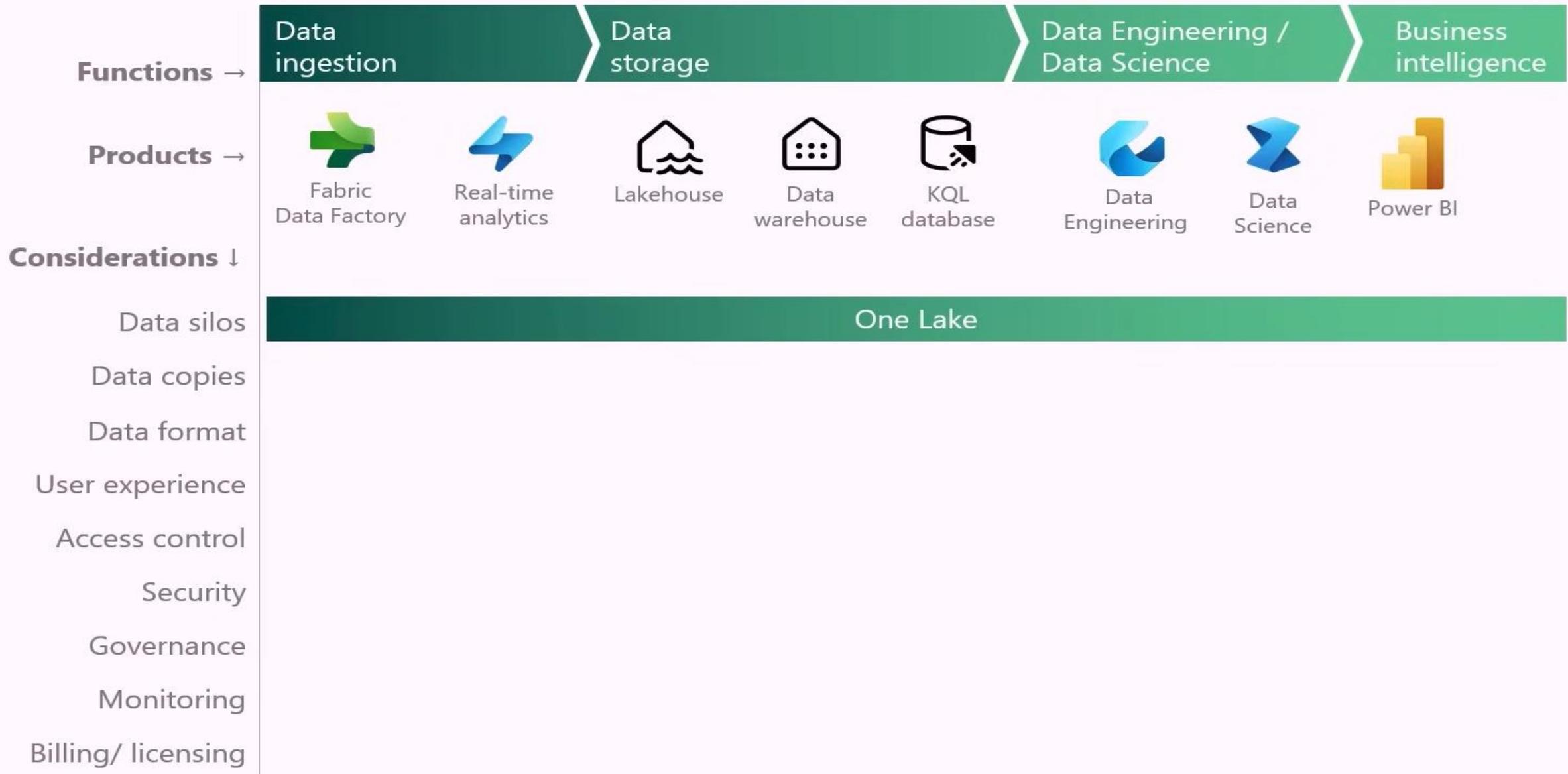
**Step 3:**  
The data engineers have  
built a data pipeline that  
copies this data every  
morning to the data lake  
for data scientists to  
analyze

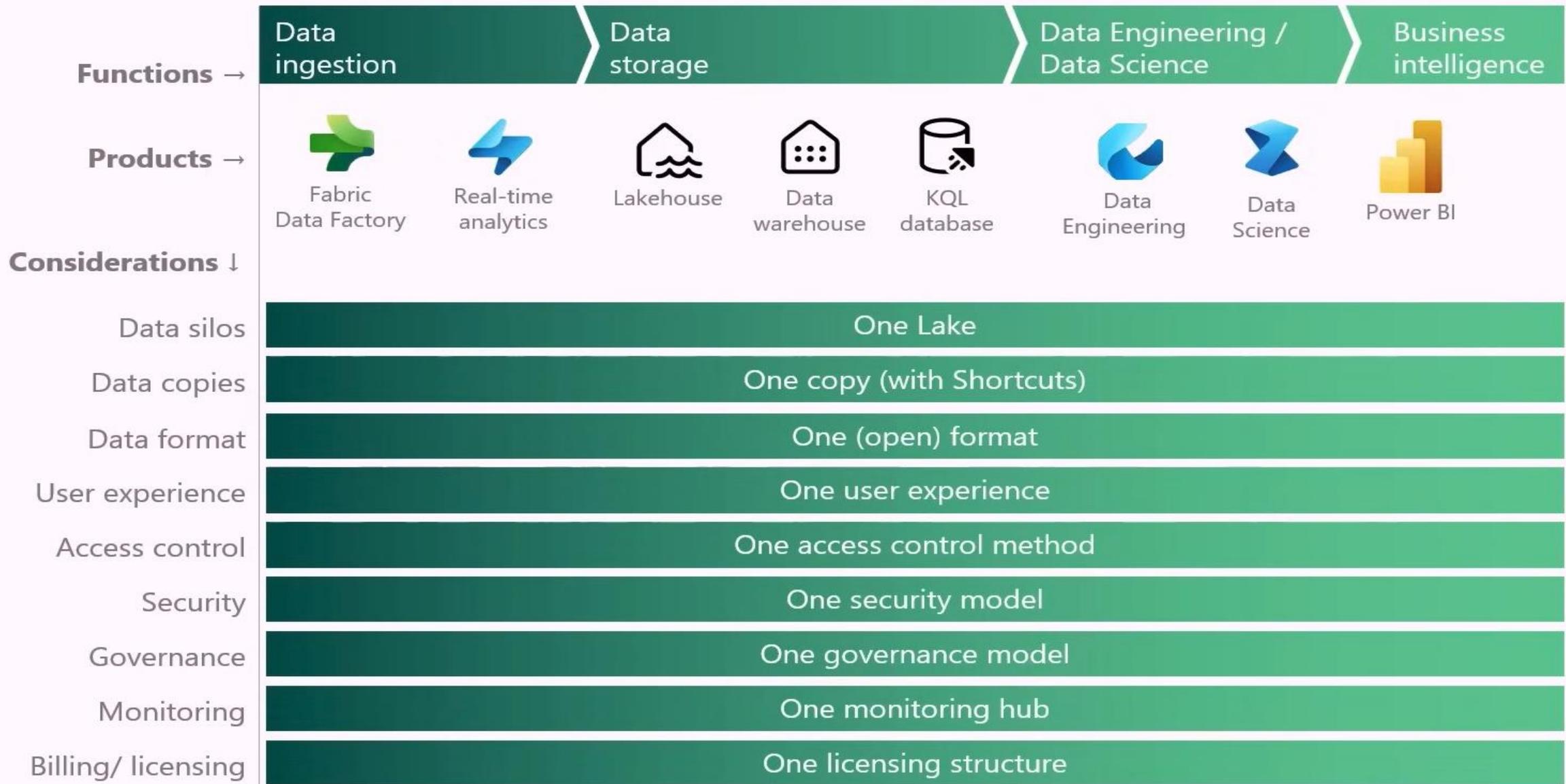


**Step 4:**  
Data scientists have built  
a model that predicts the  
sentiment of customer  
reviews and writes the  
data into a data lake  
container.



**Step 5:**  
A Power BI report  
communicates these  
findings back to product  
teams to improve  
products going forward









Data Factory



Synapse Data Warehouse



Synapse Data Engineering



Synapse Data Science



Synapse Real-time Analytics



Power BI



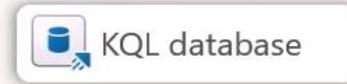
Data Activator



Data warehouse



Lakehouse



KQL database



Semantic model



Data warehouse

Lakehouse

KQL database

Semantic model

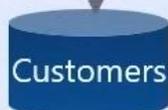


OneLake



Finance

Delta – Parquet Format



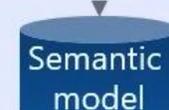
Customers

Delta – Parquet Format



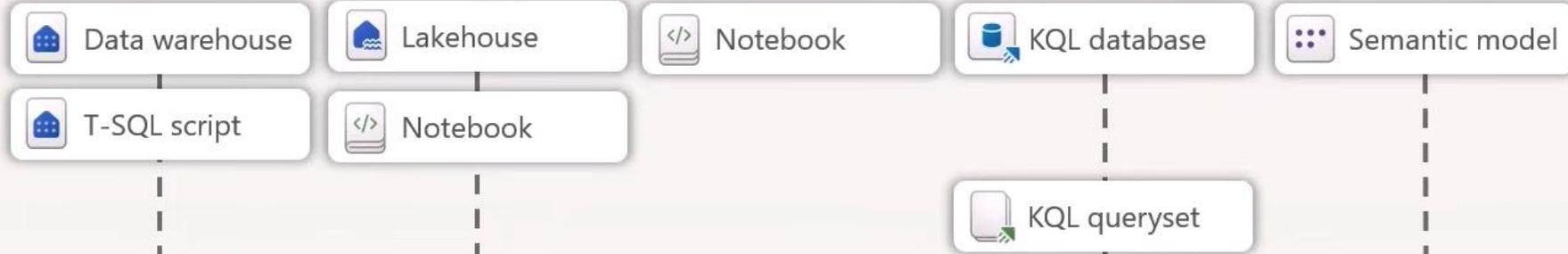
Telemetry

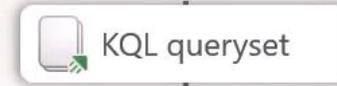
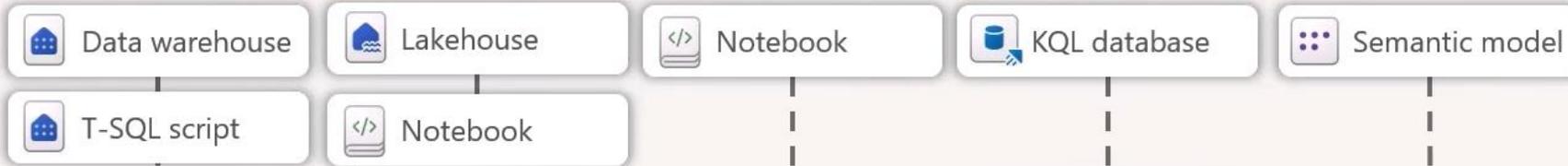
Delta – Parquet Format



Semantic model

Delta – Parquet Format





**Compute engines**



**OneLake**



Delta – Parquet Format



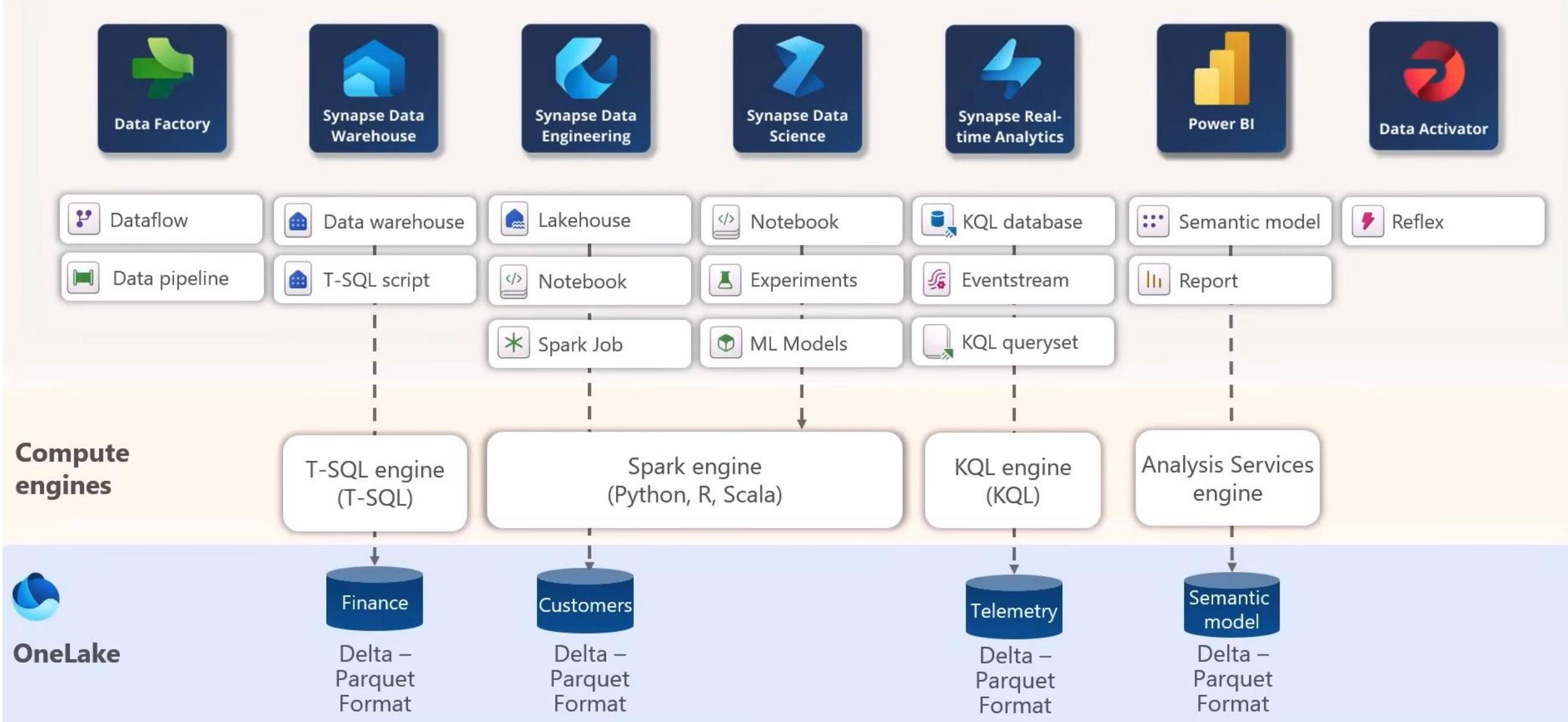
Delta – Parquet Format



Delta – Parquet Format



Delta – Parquet Format



# DATA FACTORY

## Core purpose:

Moving and transforming your data. A set of tools to help you with Extract, Transform and Load.

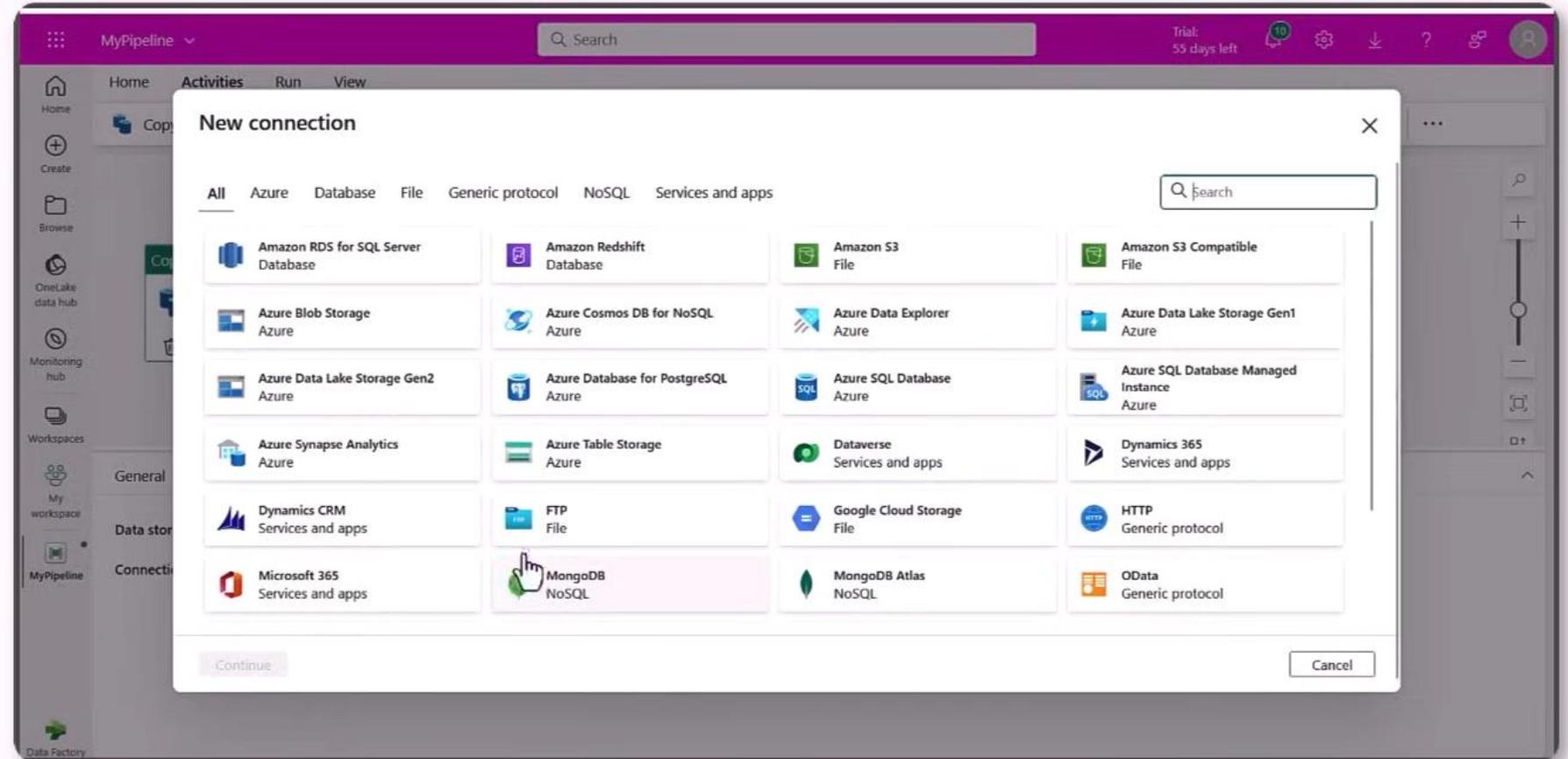
## Fabric items:

 Dataflow

 Data pipeline

## Similar to:

- Azure Data Factory
- Synapse Pipelines
- Power BI Dataflow Gen1





# SYNAPSE DATA WAREHOUSE

## Core purpose:

provides a familiar transactional data warehouse solution with tables, schemas, views, stored procedures etc. Query-able using Structured Query Language (T-SQL)

## Fabric items:



Data warehouse

## Similar to:

- SQL Server/ Azure SQL
- Synapse SQL Serverless/Dedicated
- Snowflake

# SYNAPSE DATA ENGINEERING

## Core purpose:

Enables users to design, build, and maintain **infrastructures and systems** that enable their organizations to collect, store, process, and analyze large volumes of data .

## Fabric items:

 Lakehouse

 Notebook

 Spark Job

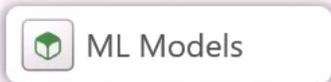
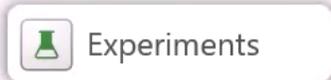
## Similar to:

- Azure Data Lake Storage\* (ADLS Gen2)
- Databricks
- Snowflake

## Core purpose:

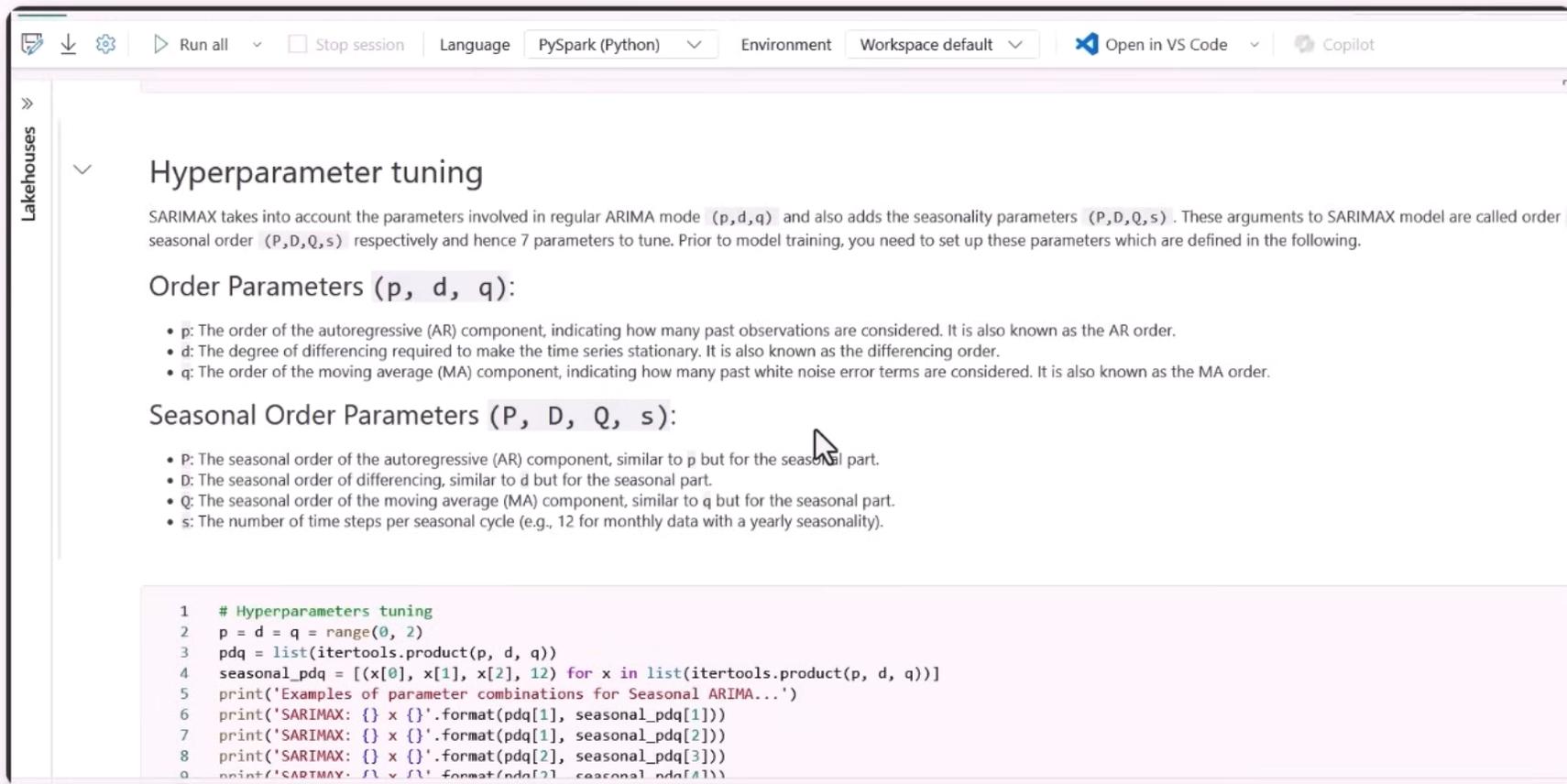
Supports the entire data science workflow in an organisation, from data exploration, preparation and cleansing to experimentation, modeling, model scoring and serving of predictive insights to BI reports.

## Fabric items:



## Similar to:

- Azure Machine Learning
- Synapse notebooks
- Databricks notebooks



The screenshot shows a Synapse notebook interface. The top navigation bar includes 'Run all', 'Stop session', 'Language' (set to PySpark (Python)), 'Environment' (Workspace default), 'Open in VS Code', and 'Copilot'. The notebook content is titled 'Hyperparameter tuning' and discusses SARIMAX parameters. It defines Order Parameters (p, d, q) and Seasonal Order Parameters (P, D, Q, s) with their respective meanings. Below the text is a code block for hyperparameter tuning.

```
1 # Hyperparameters tuning
2 p = d = q = range(0, 2)
3 pdq = list(itertools.product(p, d, q))
4 seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]
5 print('Examples of parameter combinations for Seasonal ARIMA...')
6 print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
7 print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
8 print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
9 print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

# SYNAPSE REAL-TIME ANALYTICS

## Core purpose:

Provides a set of tools to ingest, manage and analyze real-time event data

## Fabric items:

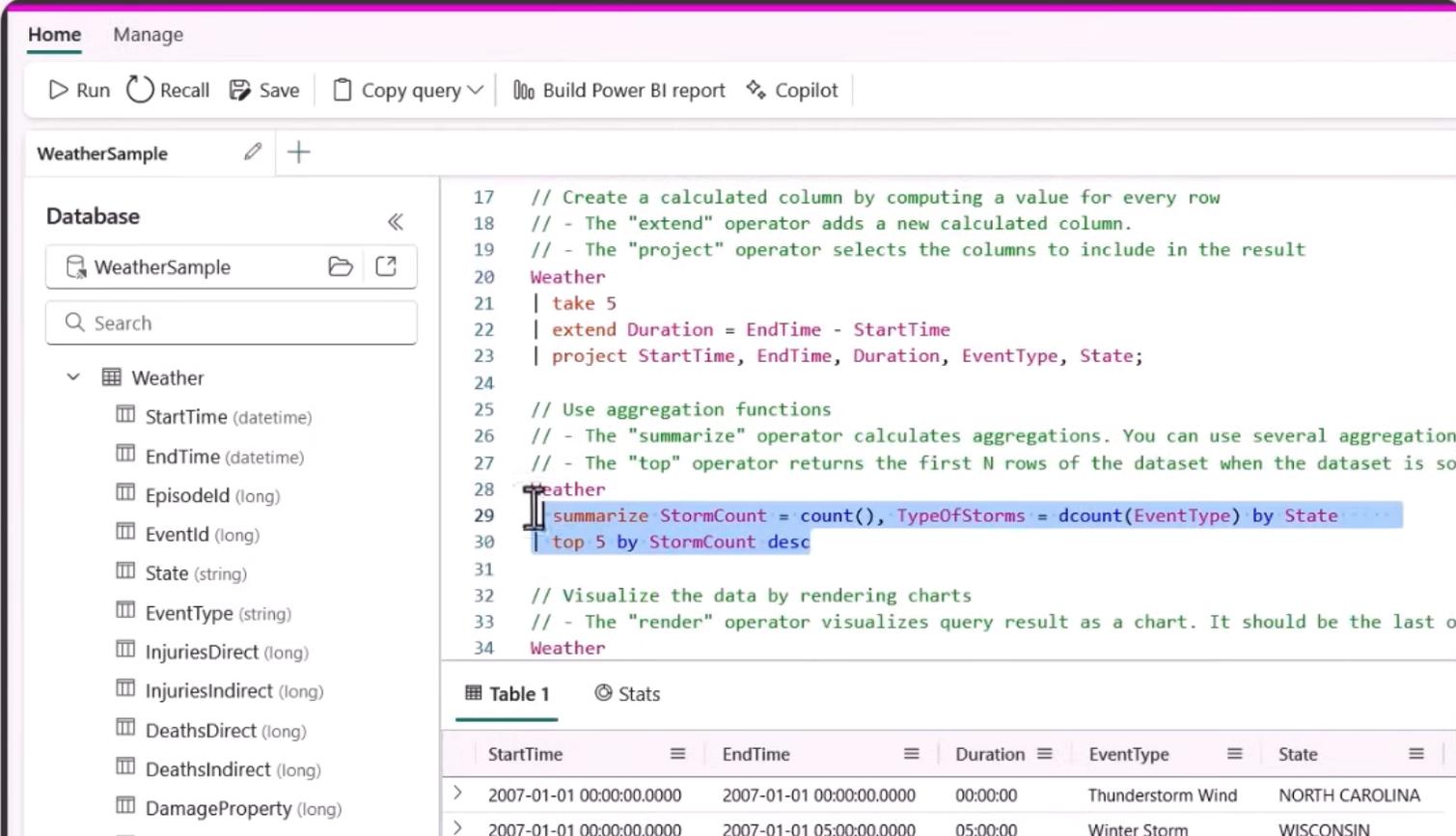
 KQL database

 Eventstream

 KQL queryset

## Similar to:

- Azure Data Explorer



The screenshot displays the Synapse Real-time Analytics interface. The top navigation bar includes 'Home' and 'Manage' tabs, along with action buttons for 'Run', 'Recall', 'Save', 'Copy query', 'Build Power BI report', and 'Copilot'. The main workspace is titled 'WeatherSample' and contains a 'Database' section with a search bar and a list of tables under the 'Weather' schema. The KQL query editor shows the following code:

```
17 // Create a calculated column by computing a value for every row
18 // - The "extend" operator adds a new calculated column.
19 // - The "project" operator selects the columns to include in the result
20 Weather
21 | take 5
22 | extend Duration = EndTime - StartTime
23 | project StartTime, EndTime, Duration, EventType, State;
24
25 // Use aggregation functions
26 // - The "summarize" operator calculates aggregations. You can use several aggregation
27 // - The "top" operator returns the first N rows of the dataset when the dataset is so
28 Weather
29 | summarize StormCount = count(), TypeOfStorms = dcount(EventType) by State
30 | top 5 by StormCount desc
31
32 // Visualize the data by rendering charts
33 // - The "render" operator visualizes query result as a chart. It should be the last o
34 Weather
```

Below the query editor, the results are displayed in a table format:

StartTime	EndTime	Duration	EventType	State
> 2007-01-01 00:00:00.0000	2007-01-01 00:00:00.0000	00:00:00	Thunderstorm Wind	NORTH CAROLINA
> 2007-01-01 00:00:00.0000	2007-01-01 05:00:00.0000	05:00:00	Winter Storm	WISCONSIN

## Core purpose:

Power BI is Microsoft's business intelligence solution that allows you to create reports to present visual insights to business users.

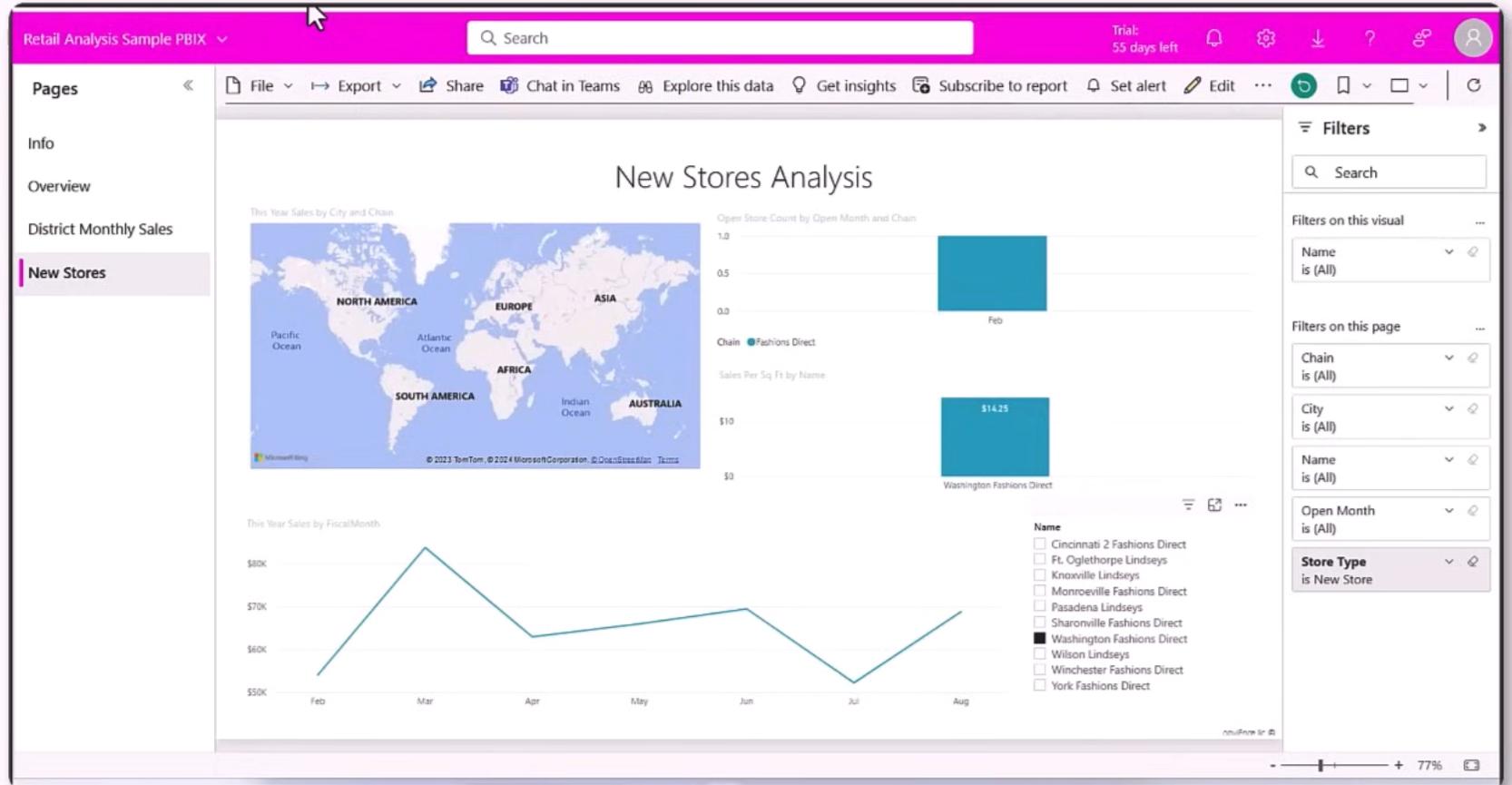
## Fabric items:

Report

Semantic model

## Similar to:

- Tableau
- Looker



## Core purpose:

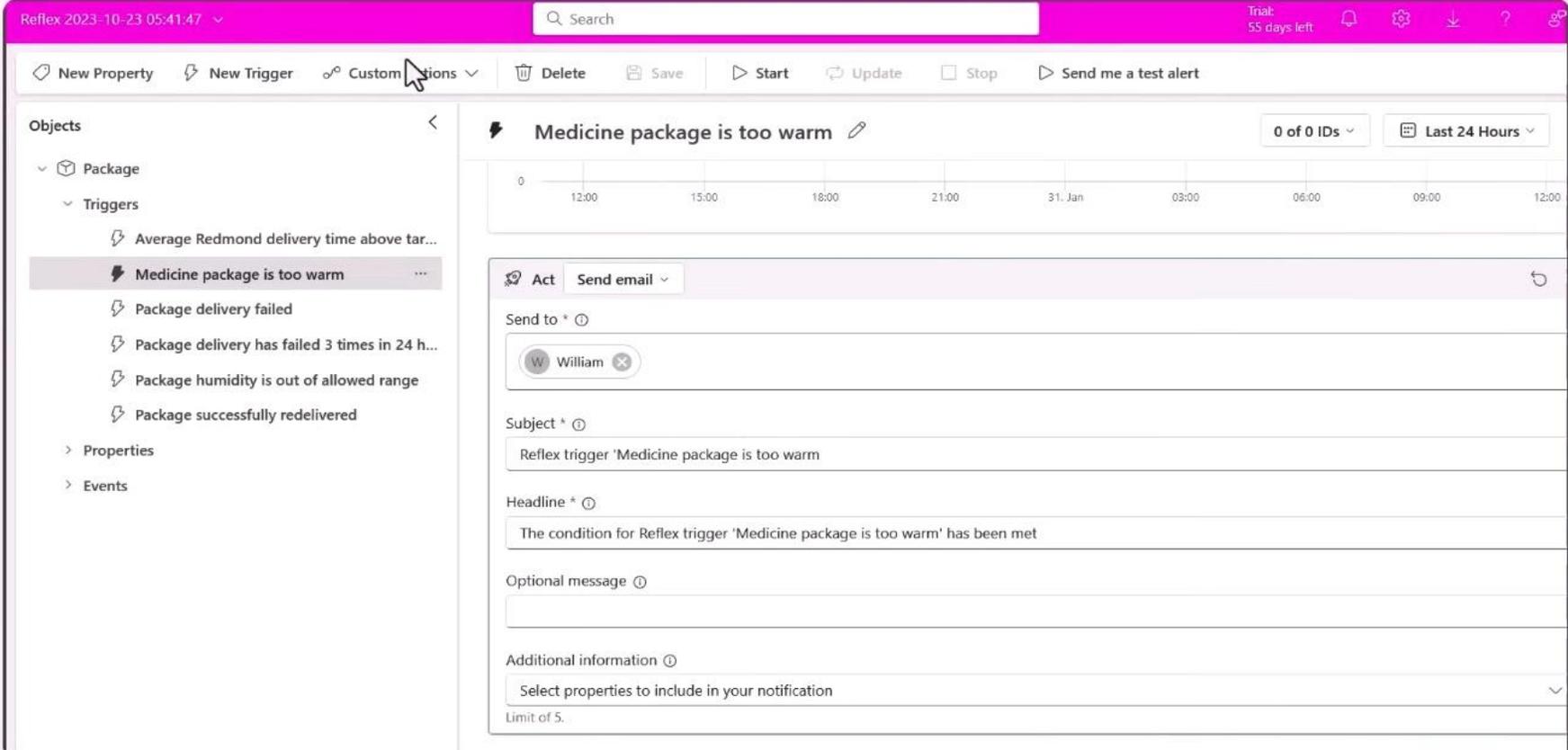
Automatically taking actions (like running a Power Automate routine) when **patterns** or **conditions** are detected in changing data, such as data in Power BI reports and Eventstreams

## Fabric items:

 Reflex

## Similar to:

- Power Automate
- Azure Functions



The screenshot displays the Data Activator interface. At the top, there's a search bar and a trial status indicator: "Trial 55 days left". Below this is a navigation bar with options: "New Property", "New Trigger", "Custom Conditions", "Delete", "Save", "Start", "Update", "Stop", and "Send me a test alert".

The main area is split into two panes. The left pane, titled "Objects", shows a tree view with "Package" expanded to show "Triggers". The trigger "Medicine package is too warm" is selected and highlighted. Other triggers listed include "Average Redmond delivery time above tar...", "Package delivery failed", "Package delivery has failed 3 times in 24 h...", "Package humidity is out of allowed range", and "Package successfully redelivered".

The right pane shows the configuration for the selected trigger. It features a timeline graph at the top with a title "Medicine package is too warm" and a filter "0 of 0 IDs" and "Last 24 Hours". Below the graph is an "Act" section with a dropdown menu set to "Send email". The configuration fields include:

- Send to \***: A field containing "William" with a close button.
- Subject \***: A field containing "Reflex trigger 'Medicine package is too warm'".
- Headline \***: A field containing "The condition for Reflex trigger 'Medicine package is too warm' has been met".
- Optional message**: An empty text area.
- Additional information**: A dropdown menu set to "Select properties to include in your notification" with a "Limit of 5" indicator.

# Microsoft Fabric



- **Data Factory:** Data integration combining Power Query with the scale of Azure Data Factory to move and transform data.
- **Data Engineering:** Data engineering with a Spark platform for data transformation at scale.
- **Data Warehouse:** Data warehousing with industry-leading SQL performance and scale to support data use.
- **Data Science:** Data science with Azure Machine Learning and Spark for model training and execution tracking in a scalable environment.
- **Real-Time Intelligence:** Real-time Intelligence to query and analyze large volumes of data in real-time.
- **Power BI:** Business intelligence for translating data to decisions.

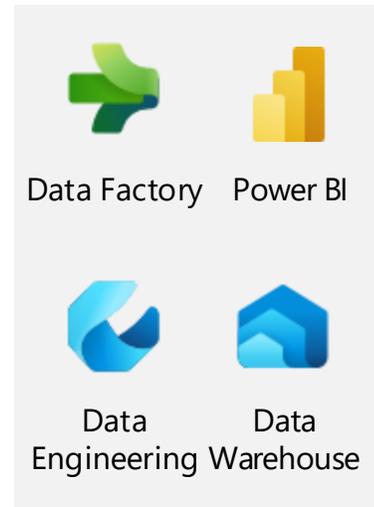
# Data teams and Fabric

Fabric's unified management and governance make it easier for data professionals to work together.

## Data Engineers



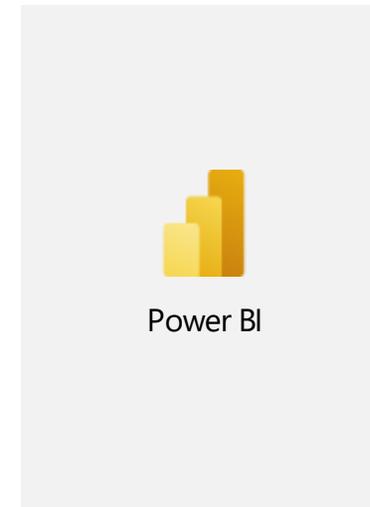
## Analytics Engineers



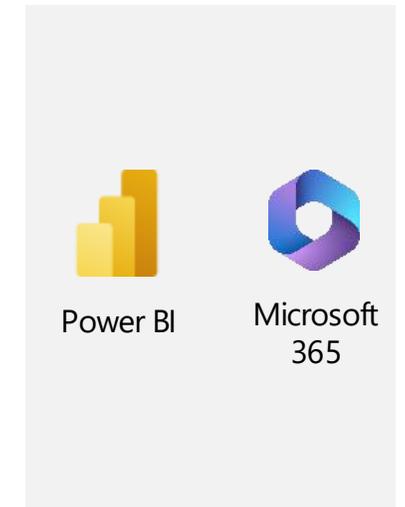
## Data Scientists



## Data Analysts



## Decision Makers



## Data Stewards

# Enable and use Microsoft Fabric

Fabric must be enabled in your tenant by either:

- Fabric admin (formerly known as Power BI admin)
- Power Platform admin
- Microsoft 365 admin

Workspaces must then be assigned to Premium per capacity or Fabric license mode.

# Knowledge check



- 1 Which of the following is a key benefit of using Microsoft Fabric in data projects?**
  - It allows data professionals to work on data projects independently, without the need for collaboration
  - It requires duplication of data across different systems and teams to ensure data availability
  - It provides a single, integrated environment for data professionals and the business to collaborate on data projects
- 2 How do compute engines in Fabric interact with data stored in OneLake?**
  - Compute engines can directly access data in OneLake without the need for duplication or movement
  - OneLake automatically converts data into csv format for seamless integration with compute engines
  - Compute engines in Fabric can only interact with a limited subset of data formats within OneLake
- 3 Which of the following Fabric experiences is used to move and transform data?**
  - Data Science
  - Data Warehousing
  - Data Factory

# Recap

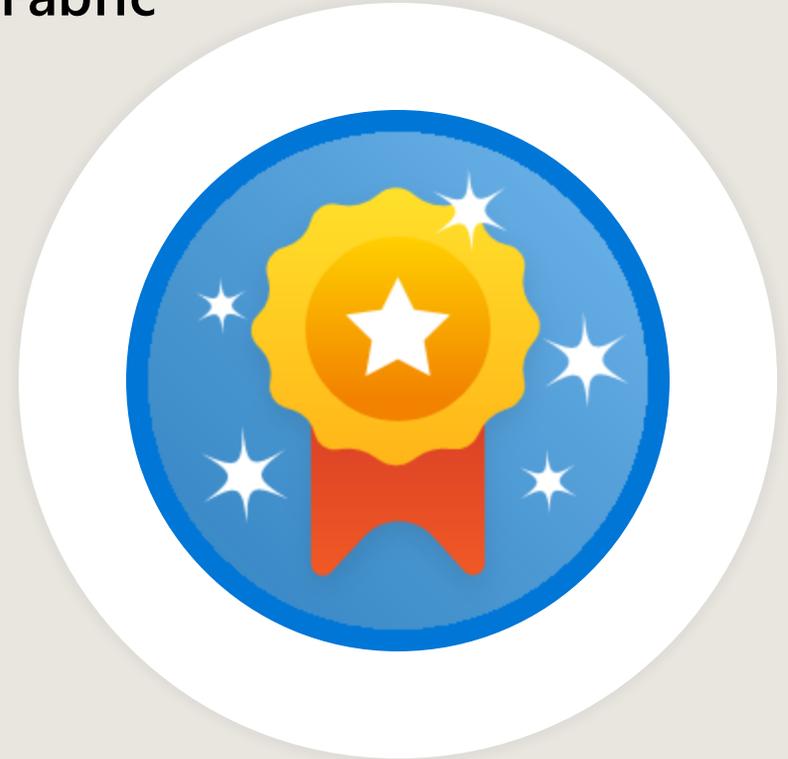
In this section, we covered:

- Different workloads in Microsoft Fabric.
- How Fabric offers flexibility for data teams to perform tasks.

# Further reading

Introduction to end-to-end analytics using Microsoft Fabric

<https://aka.ms/fabric-intro>



# Get started with lakehouses in Microsoft Fabric



# Learning objectives

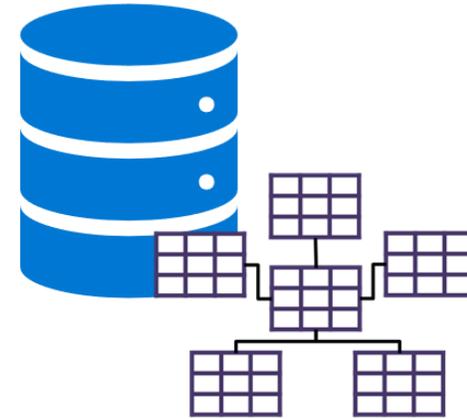
- Describe end-to-end analytics in Microsoft Fabric
- Understand data teams and roles that use Fabric
- Describe how to enable and use Fabric

# What is a lakehouse?



**Data *Lake***

- Scalable, distributed file storage
- Flexible schema-on-read semantics
- Big data technology compatibility



**Data *Warehouse***

- Relational schema modeling
- SQL-based querying
- Proven basis for reporting and analytics

# Work with a Fabric lakehouse

The screenshot displays the Microsoft Fabric interface. At the top, a table header lists columns: Name, Type, Task, Owner, Refreshed, Next refresh, Endorsement, Sensitivity, and Included in app. Below this is a navigation bar with options like Home, File, Refresh, Explore this data, Analyze in Excel, Lineage, Open data model, and Write DAX queries. The main workspace is titled 'Home Reporting' and includes a 'SQL analytics endpoint' dropdown. A notification states: 'This SQL analytics endpoint has a default Power BI semantic model. To automatically add objects, go to SQL analytics endpoint settings. To manually add objects, use Manage default semantic model. Learn more'. The 'Explorer' pane on the left shows a tree view for 'Warehouses' (DP601\_Bronze) and 'Queries' (My queries, Shared queries). The 'SQL query 1' editor shows the following SQL code:

```
1 SELECT TOP 10 *
2 FROM sales
3 WHERE OrderQty > 20
```

Below the query editor, the 'Results' tab is active, displaying a table with 8 columns: SalesOrderID, OrderDate, CustomerID, Lineltem, ProductID, OrderQty, and LineltemTotal. The table contains 3 rows of data.

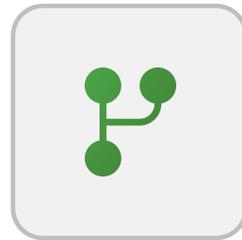
	123 SalesOrderID	OrderDate	123 CustomerID	123 Lineltem	123 ProductID	123 OrderQty	12F LineltemTotal
1	71783	2022-06-02	29957	7	976	25	19136.14
2	71784	2022-06-03	29736	39	864	23	763.11
3	71797	2022-07-01	29796	33	864	23	763.11

# Load data into a lakehouse

Ingest and connect to data sources to transform and load for analytics.



**Notebooks**



**Dataflows Gen2**



**Pipelines**



**Shortcuts**

# Explore, transform, and visualize data in the lakehouse

Tools and techniques to explore and transform data:

- ★ Apache Spark
  - 📄 Notebooks
  - 📅 Spark Job Definitions
- 🏠 SQL analytics endpoint
- 🔗 Dataflows Gen2
- 📡 Data Pipelines

Visualize lakehouse data using Power BI

# Exercise

30 minutes



## Create and ingest data with a Microsoft Fabric lakehouse

# Knowledge check



- 1 What is a Microsoft Fabric lakehouse?**
  - A relational database based on the Microsoft SQL Server database engine.
  - A hierarchy of folders and files in Azure Data Lake Store Gen2.
  - An analytical store that combines the file storage flexibility of a data lake with the SQL-based query capabilities of a data warehouse.
  
- 2 You want to include data in an external Azure Data Lake Store Gen2 location in your lakehouse, without the requirement to copy the data. What should you do?**
  - Create a Data pipeline that uses a Copy Data activity to load the external data into a file.
  - Create a shortcut.
  - Create a Dataflow Gen2 that extracts the data and loads it into a table.
  
- 3 You want to use Apache Spark to interactively explore data in a file in the lakehouse. What should you do?**
  - Create a notebook.
  - Switch to the SQL analytics endpoint mode.
  - Create a Dataflow Gen2.

# Recap

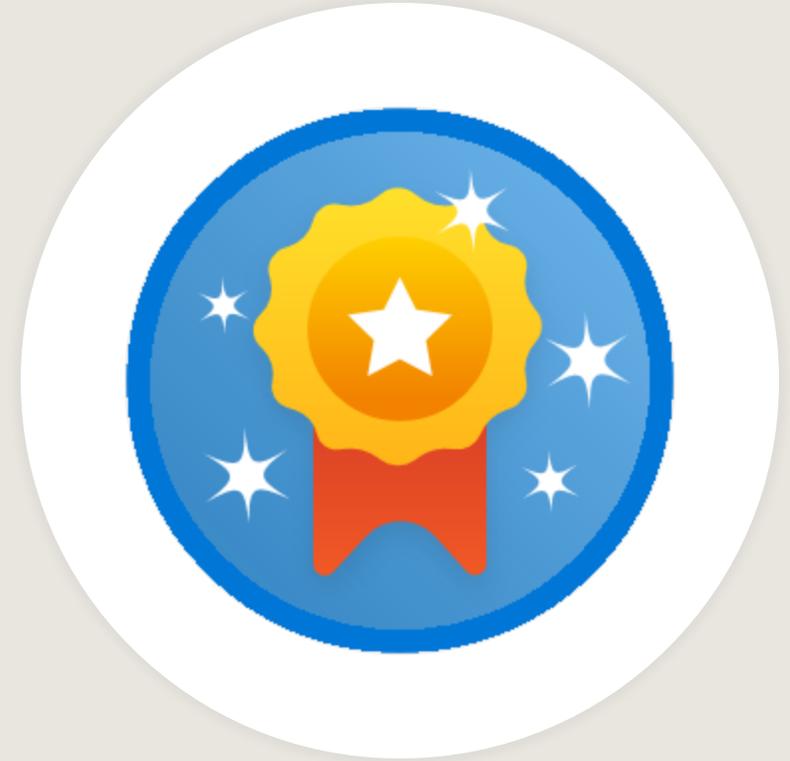
In this section, we covered:

- Lakehouse architecture.
- How to work with a lakehouse.
- Data ingestion, transformation, and exploration tools.

# Further reading

Get started with lakehouses in Microsoft Fabric

<https://aka.ms/fabric-lakehouse>



# Use Apache Spark in Microsoft Fabric



# Learning objectives

- Describe end-to-end analytics in Microsoft Fabric
- Understand data teams and roles that use Fabric
- Describe how to enable and use Fabric

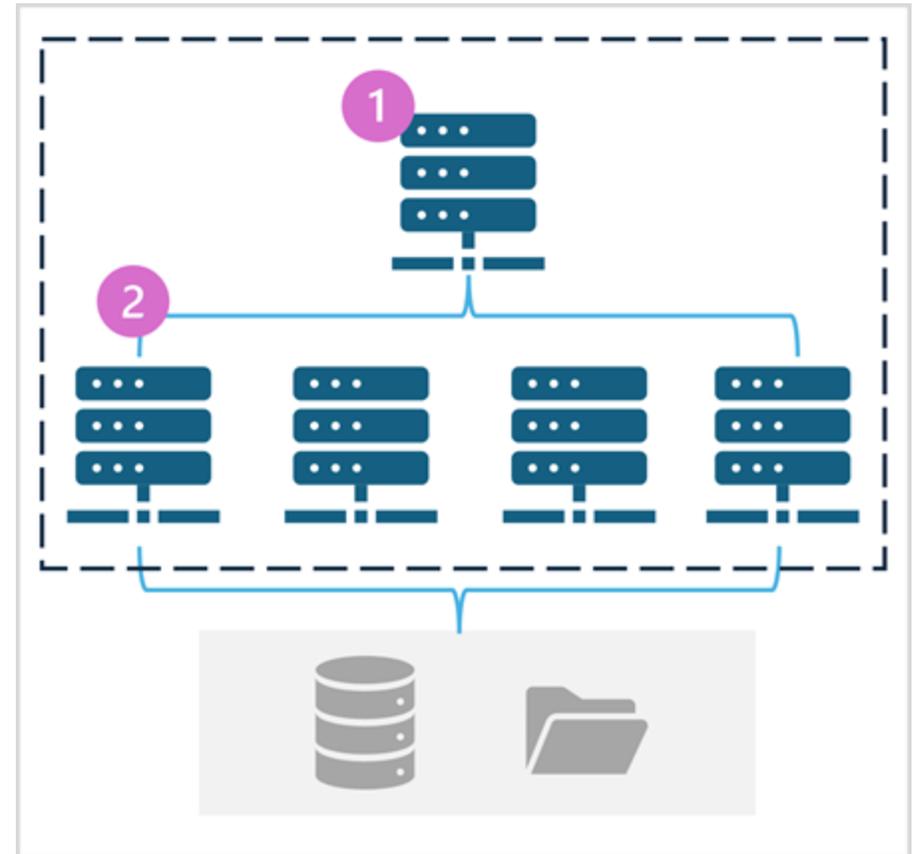
# What is Apache Spark?

Apache Spark is a core technology for large-scale data analytics.

Microsoft Fabric provides support for Spark clusters, enabling you to analyze and process data in a lakehouse at scale.

# How to use Apache Spark

- Distributed data processing framework through Spark pools.
- Use notebooks to ingest, transform, load, analyze, and visualize data.
- Notebooks in Fabric support many programming languages, including PySpark and Spark SQL.



# Prepare to use Apache Spark

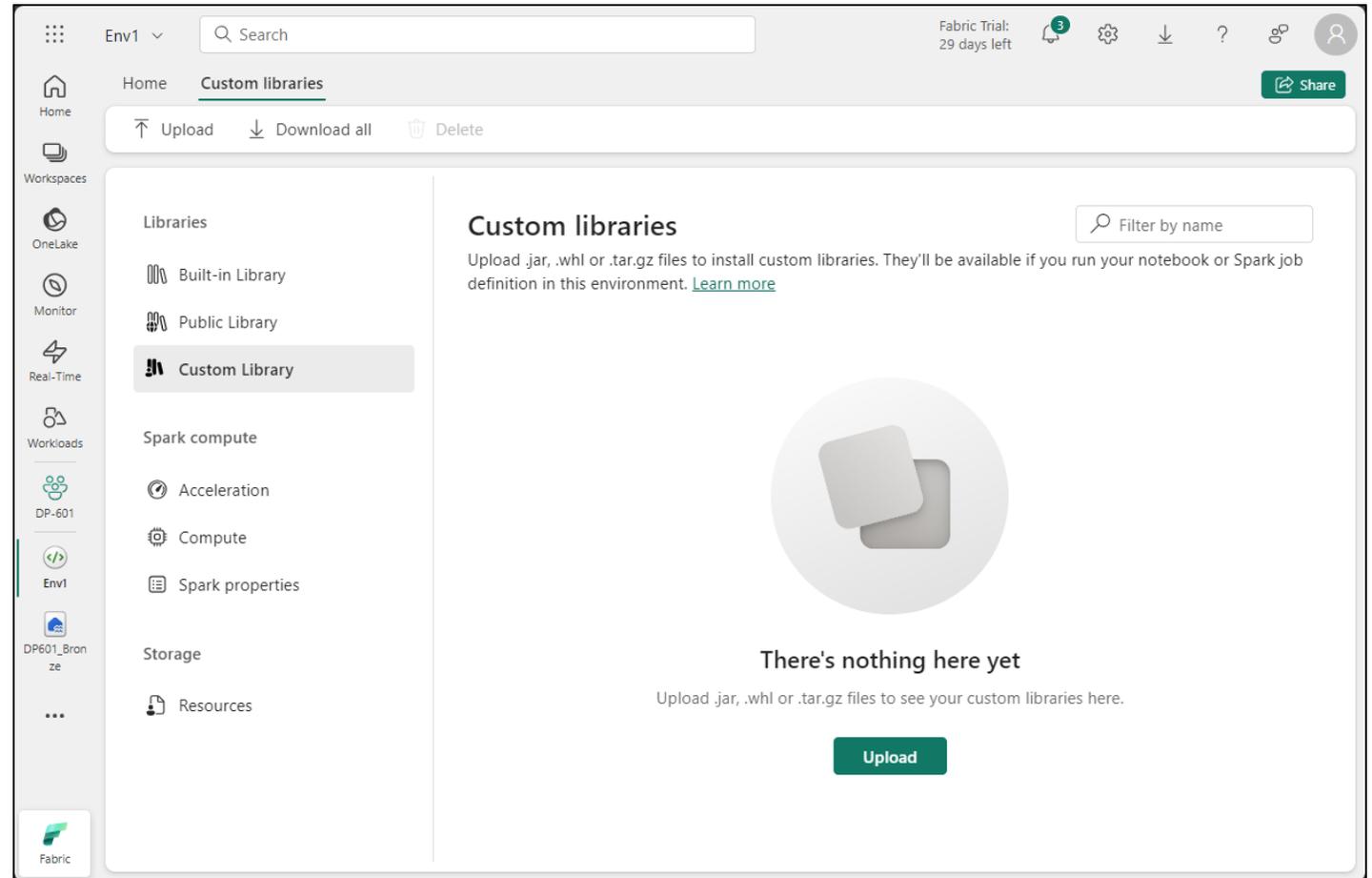
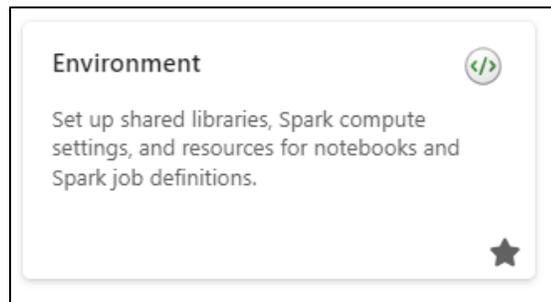
- Each workspace is assigned a Spark cluster.
- Workspace admins can manage settings for the Spark cluster in the Workspace settings.
- Specific configuration settings include:
  - Node family
  - Runtime version
  - Spark properties

The screenshot shows the 'Workspace settings' page in the Azure portal. The left sidebar contains a list of settings categories: General, License info, Azure connections, System storage, Git integration, OneLake, Workspace identity, Network security, Monitoring, Power BI, Delegated Settings, Data, Engineering/Science, and Spark settings (which is highlighted with a blue bar). The main content area is titled 'Spark settings' and includes a description: 'Configure and manage settings for Spark workloads and the default environment for the workspace.' Below this, there are tabs for 'Pool', 'Environment', 'Jobs', 'High concurrency', and 'Automatic log', with 'Pool' selected. The 'Default pool for workspace' section contains a dropdown menu set to 'StarterPool' and a table of pool details. The table has columns for 'Node family', 'Node size', and 'Number of nodes'. The 'Customize compute configurations for items' section has a toggle switch turned 'On'.

Node family	Node size	Number of nodes
Memory optimized	Medium	1 - 10

# Configure the Spark environment

- Libraries
- Spark Compute
- Storage



# Run Spark in Fabric

To edit and run Spark code in Fabric, use *notebooks* or create a *Spark Job Definition*

**Notebook** 

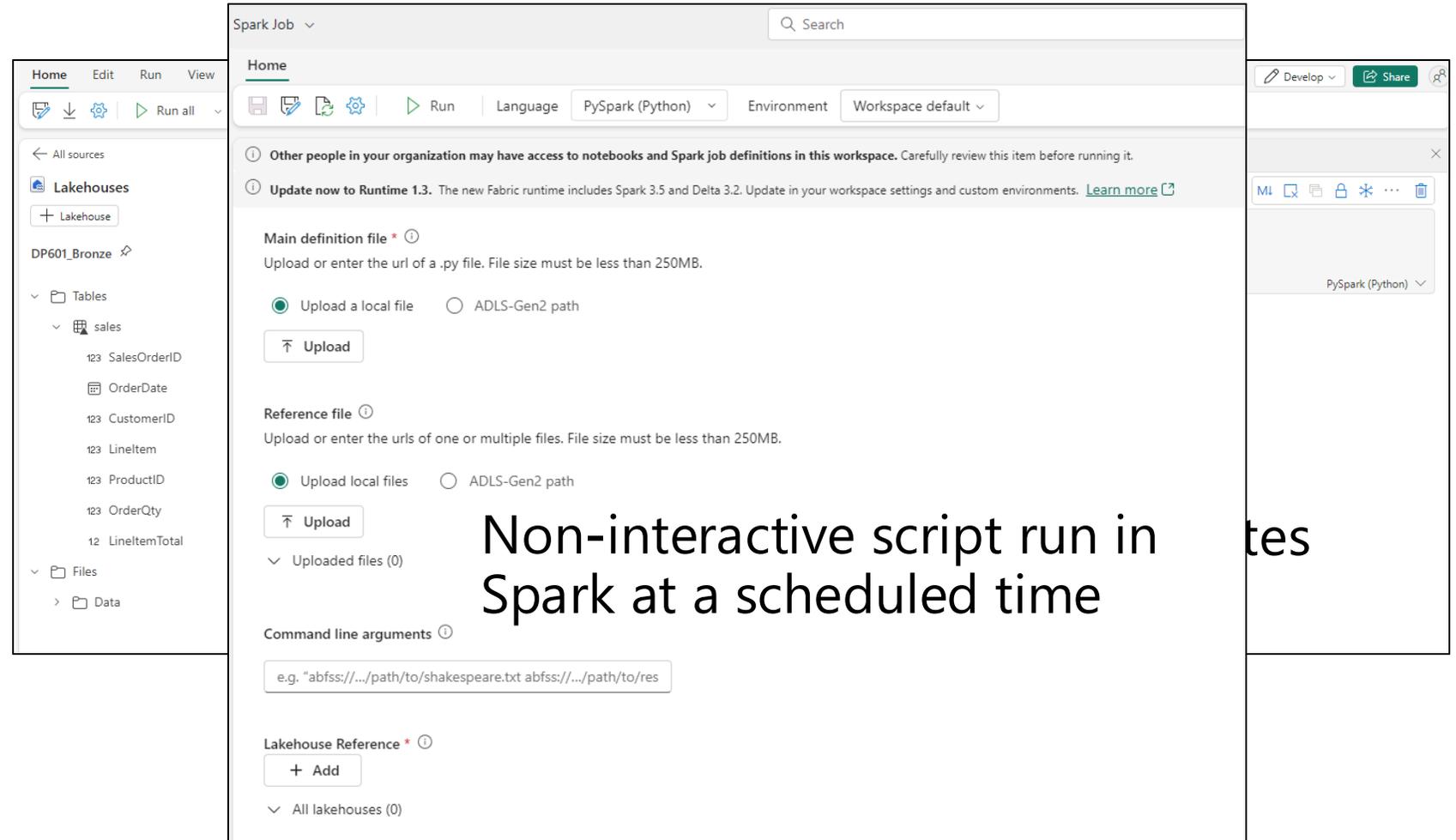
Explore, analyze, and visualize data and build ML models. Supports Apache Spark, Python, T-SQL, and more.



**Spark Job Definition** 

Define, schedule, and manage your Apache Spark jobs for big data processing.





Spark Job

Home

Language PySpark (Python) Environment Workspace default

Other people in your organization may have access to notebooks and Spark job definitions in this workspace. Carefully review this item before running it.

Update now to Runtime 1.3. The new Fabric runtime includes Spark 3.5 and Delta 3.2. Update in your workspace settings and custom environments. [Learn more](#)

Main definition file \* 

Upload or enter the url of a .py file. File size must be less than 250MB.

Upload a local file  ADLS-Gen2 path

 Upload

Reference file 

Upload or enter the urls of one or multiple files. File size must be less than 250MB.

Upload local files  ADLS-Gen2 path

 Upload

Uploaded files (0)

Command line arguments 

e.g. "abfss://.../path/to/shakespeare.txt abfss://.../path/to/res"

Lakehouse Reference \* 

 Add

All lakehouses (0)

Non-interactive script run in Spark at a scheduled time

# Ingest data with Spark

- Define data source
- Authentication type
- Credentials
- Display or load data

```
# Azure Blob Storage access info
blob_account_name = "azureopendatastorage"
blob_container_name = "nyctlc"
blob_relative_path = "yellow"

# blob_sas_token = "add your SAS token here" #
Construct the path for connection

wasbs_path =
f'wasbs://{blob_container_name}@{blob_account_name}.
blob.core.windows.net/{blob_relative_path}'

# WASBS path for connection including SAS token

# wasbs_path =
f'wasbs://{blob_container_name}@{blob_account_name}.
blob.core.windows.net/{blob_relative_path}?{blob_sas
_token}'

# Read parquet data from Azure Blob Storage path

blob_df = spark.read.parquet(wasbs_path)

# Display the Azure Blob DataFrame display(blob_df)
```

# Load data in a Spark Dataframe

Schema can be either inferred or specified

The screenshot shows a Databricks workspace interface. On the left, a sidebar displays the file structure under 'Data', including a 'sales.csv' file and a table with columns: SalesOrderID, OrderDate, CustomerID, LineItem, ProductID, OrderQty, and LineItemTotal. The main editor displays a PySpark script:

```
1 from pyspark.sql.types import *
2 from pyspark.sql.functions import *
3
4 salesSchema = StructType([
5     StructField("SalesOrderID", IntegerType(), True),
6     StructField("OrderDate", StringType(), True),
7     StructField("CustomerID", IntegerType(), True),
8     StructField("LineItemTotal", DoubleType(), True)
9 ])
10 df = spark.read.load('Files/Data/sales.csv',
11     format='csv',
12     schema=salesSchema,
13     header=False)
14 display(df.limit(3))
```

A callout box with the text "Specify a schema" and a purple arrow points to the `schema=salesSchema` parameter in the `spark.read.load` function call.

Below the code, the execution status is shown as "1 sec - Command executed in 847 ms". A "Table view" section displays the following data:

	123 SalesOrderID	ABC OrderDate	123 CustomerID	12 LineItemTotal
1	NULL	OrderDate	NULL	NULL
2	71774	2022-06-01	29847	1.0
3	71774	2022-06-01	29847	2.0

# Transform data in a Spark Dataframe

The screenshot shows a Databricks notebook interface. The top navigation bar includes 'Home', 'Edit', 'Run', and 'View'. The main toolbar shows 'Run all', 'Standard session', 'PySpark (Python)', 'Environment', 'Workspace default', 'Data Wrangler', and 'Copilot'. The left sidebar displays a file explorer with 'All sources', 'Lakehouses', and 'Files > Data'. The central code editor contains the following code cell:

```
1 bikes_df.write.mode("overwrite").parquet('Files/Data/bikes.parquet')
```

The execution output shows a successful Spark job:

[39] ✓ 3 sec - Command executed in 3 sec 545 ms

Spark jobs (1 of 1 succeeded) Resources Log

ID	Description	Status	Stages	Tasks	Duration	Processed	Data
Job 30	parquet at NativeMethodAccessorImpl.java:0	✓ Succeeded	1/1	1/1 succeeded	1 sec 937 ms	206 rows	204

A purple callout box on the right side of the notebook contains the text "Save dataframe" with a large arrow pointing to the code cell.

# Work with data using Spark SQL

Use the metastore to define tables and views

```
# Create a view in the metastore
df.createOrReplaceTempView("products_view")

# Save a dataframe as a new table
df.write.format("delta").saveAsTable("products")
```

Create external tables

```
# Create external table
df.write.format("delta").saveAsTable("myexternaltable", path="Files/myexternaltable")
```

# Query data using the Spark SQL API

Use the Spark SQL API in code written in any language to query data in the catalog

```
# Return data from the products table as a dataframe using PySpark

bikes_df = spark.sql("SELECT ProductID, ProductName, ListPrice \
                      FROM products \
                      WHERE Category IN ('Mountain Bikes', 'Road Bikes')")

display(bikes_df)
```

```
# Use %%sql magic to query objects in the catalog using native SQL

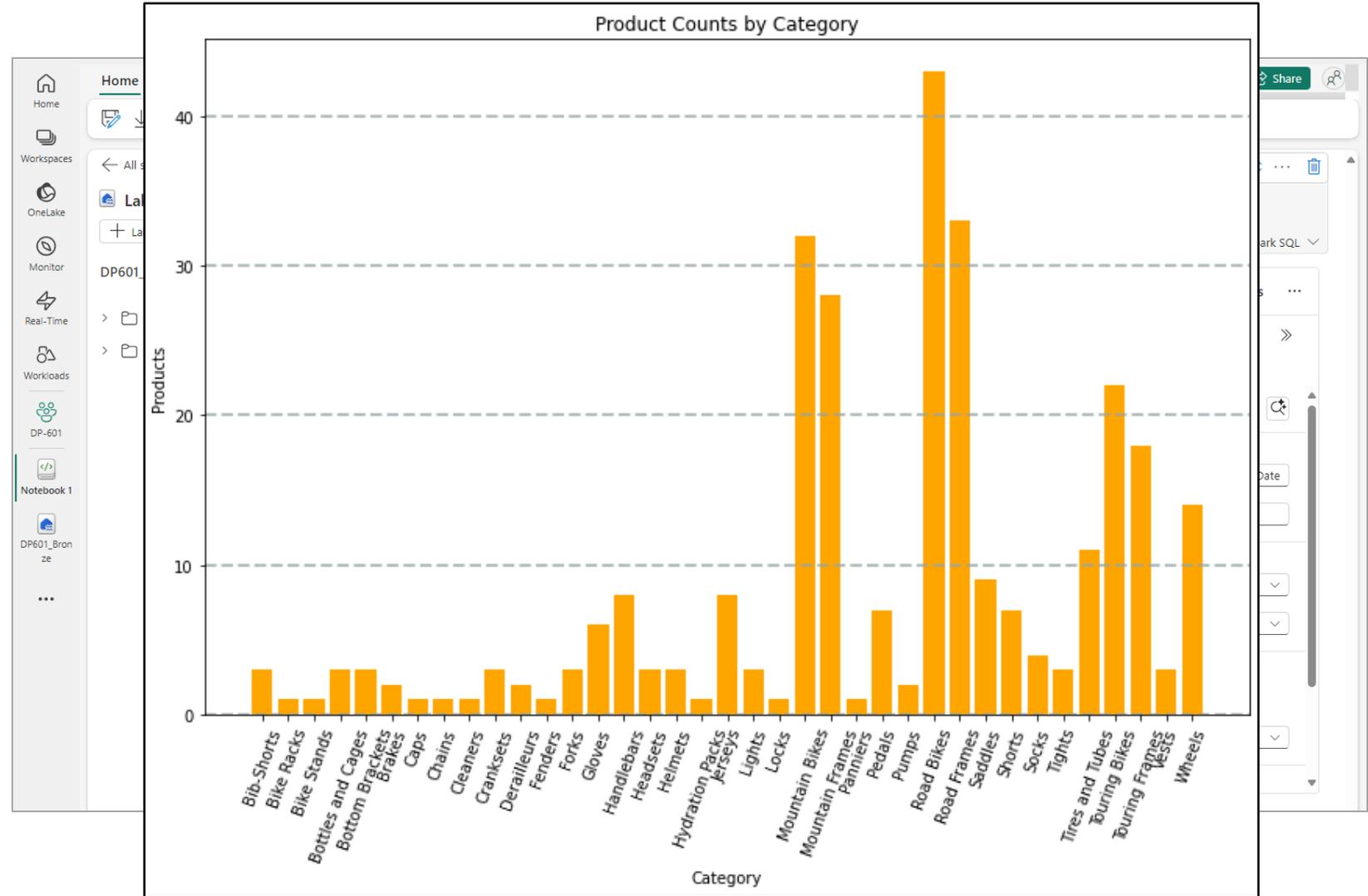
%%sql

SELECT Category, COUNT(ProductID) AS ProductCount
FROM products
GROUP BY Category
ORDER BY Category
```

# Visualize data

## Two ways to visualize data in notebooks:

1. Use built-in notebook charts
2. Use graphics packages in code
  - This example uses Matplotlib



# Exercise



45 minutes

## Analyze data with Apache Spark

# Knowledge check



**1** You want to use Apache Spark to explore data interactively in Microsoft Fabric. What should you create?

- A Spark job definition.
- A Data Factory pipeline.
- A notebook.

**2** You need to use Spark to analyze data in a CSV file. What's the simplest way to accomplish this goal?

- Load the file into a dataframe.
- Import the data into a table in a warehouse.
- Convert the data to Parquet format.

**3** Which method is used to split the data across folders when saving a dataframe?

- splitBy
- distributeBy
- partitionBy

# Recap

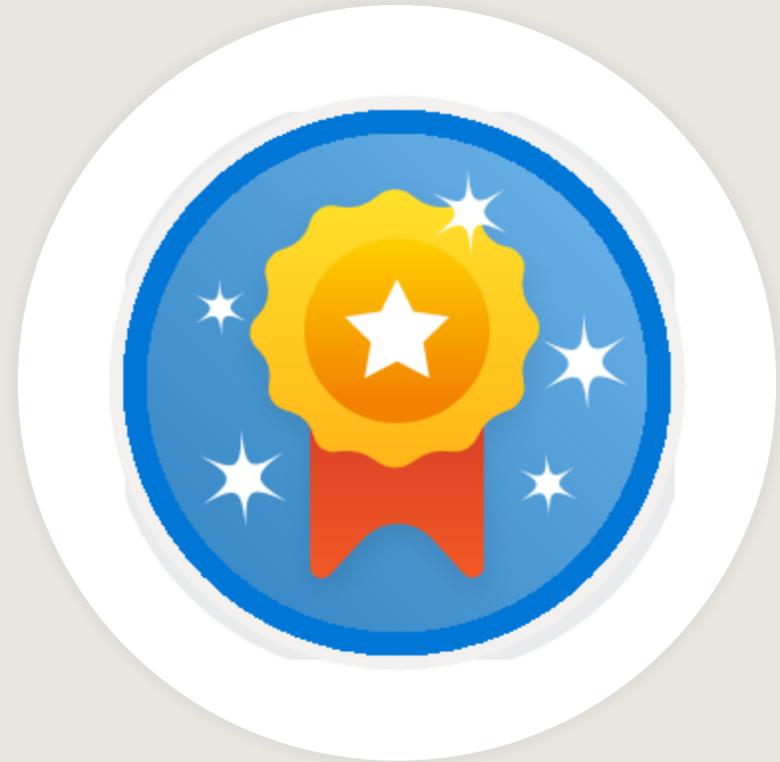
In this section, we covered:

- How Apache Spark is integrated in Fabric.
- Ingest, transform, and load data into a lakehouse using Spark with notebooks.
- Worked with and visualized data using PySpark and Spark SQL.

# Further reading

## Use Apache Spark in Microsoft Fabric

<https://aka.ms/fabric-spark>



# Work with Delta Lake tables in Microsoft Fabric



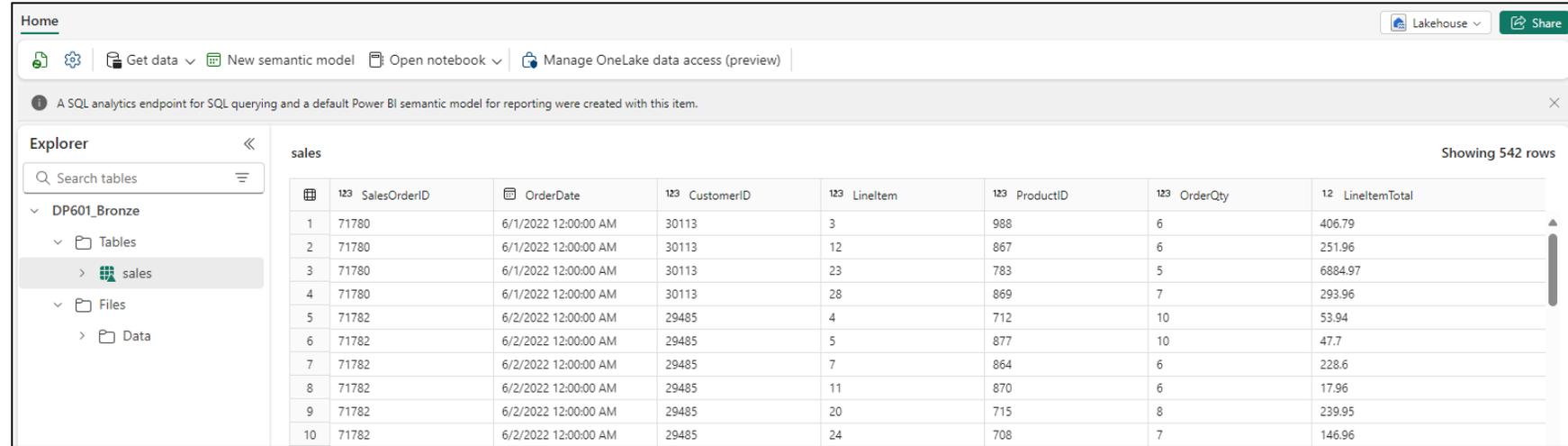
# Learning objectives



- Understand Delta Lake and delta tables in Microsoft Fabric
- Create and manage delta tables using Spark
- Optimize delta tables
- Use delta tables with Spark structured streaming

# Understand Delta Lake

- Relational tables that support querying and data modification
- Support for ACID transactions
- Data versioning and time travel
- Standard formats and interoperability



Home Lakehouse Share

Get data New semantic model Open notebook Manage OneLake data access (preview)

A SQL analytics endpoint for SQL querying and a default Power BI semantic model for reporting were created with this item.

Explorer

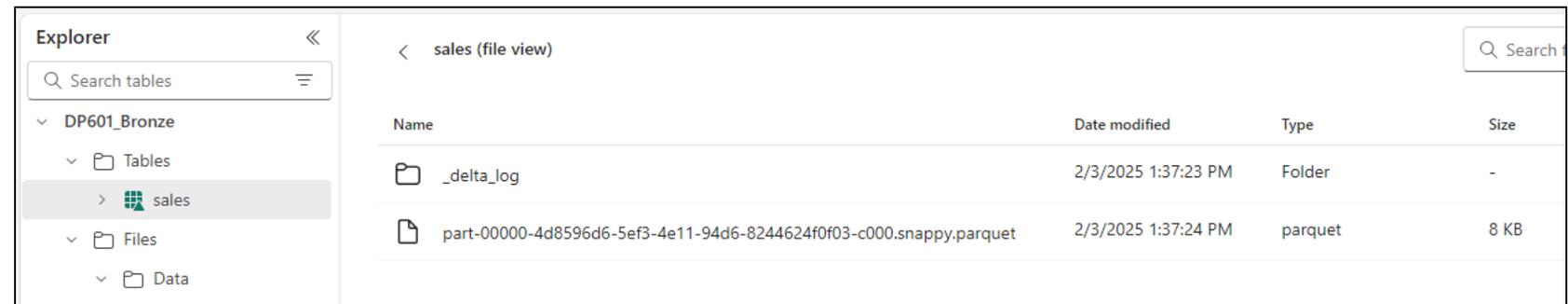
Search tables

DP601\_Bronze

- Tables
- sales
- Files
- Data

sales Showing 542 rows

	123 SalesOrderID	OrderDate	123 CustomerID	123 Lineltem	123 ProductID	123 OrderQty	12 LineltemTotal
1	71780	6/1/2022 12:00:00 AM	30113	3	988	6	406.79
2	71780	6/1/2022 12:00:00 AM	30113	12	867	6	251.96
3	71780	6/1/2022 12:00:00 AM	30113	23	783	5	6884.97
4	71780	6/1/2022 12:00:00 AM	30113	28	869	7	293.96
5	71782	6/2/2022 12:00:00 AM	29485	4	712	10	53.94
6	71782	6/2/2022 12:00:00 AM	29485	5	877	10	47.7
7	71782	6/2/2022 12:00:00 AM	29485	7	864	6	228.6
8	71782	6/2/2022 12:00:00 AM	29485	11	870	6	17.96
9	71782	6/2/2022 12:00:00 AM	29485	20	715	8	239.95
10	71782	6/2/2022 12:00:00 AM	29485	24	708	7	146.96



Explorer

Search tables

DP601\_Bronze

- Tables
- sales
- Files
- Data

sales (file view)

Search

Name	Date modified	Type	Size
_delta_log	2/3/2025 1:37:23 PM	Folder	-
part-00000-4d8596d6-5ef3-4e11-94d6-8244624f0f03-c000.snappy.parquet	2/3/2025 1:37:24 PM	parquet	8 KB

# Create Delta tables using code in Spark

## 1. Save a dataframe as a managed table

```
# Load a file into a dataframe
df = spark.read.load('Files/mydata.csv', format='csv', header=True)

# Save the dataframe as a delta table
df.write.format("delta").saveAsTable("mytable")
```

## 2. Use Spark SQL

```
%%sql

CREATE TABLE salesorders
(
  Orderid INT NOT NULL,
  OrderDate TIMESTAMP NOT NULL,
  CustomerName STRING,
  SalesTotal FLOAT NOT NULL
)
USING DELTA
```

## 3. Save a dataframe in delta format in an explicit path

```
delta_path = "Files/mydatatable"
df.write.format("delta").save(delta_path)
```

# Managed vs external tables

## Managed tables

- Defined without a specific location – Files are created in the default metastore folder (**Tables/...**)
- Dropping the table deletes the files

```
# Save a dataframe as a delta table  
df.write.format("delta").saveAsTable("mytable")
```

## External tables

- Defined with an explicit file location outside of the default metastore folder
- Dropping the table does not delete the files

```
df.write.format("delta").saveAsTable("myexternaltable", path="Files/myexternaltable")
```

# Work with Delta tables in Spark

1. Use Spark SQL to embed a SQL statement in PySpark
  - Embed SQL statements in other languages using the **spark.sql** library.

```
spark.sql("INSERT INTO products VALUES (1, 'Widget', 'Accessories', 2.99)")
```

2. Native Spark SQL using %%sql magic
  - Use %%sql magic in a notebook to run SQL statements.

```
%%sql  
  
UPDATE products  
SET Price = 2.49 WHERE ProductId = 1;
```

3. Use the Delta API
  - Create an instance of a DeltaTable from a folder location containing files in delta format, and then use the API to modify the data in the table.

```
from delta.tables import *  
from pyspark.sql.functions import *  
  
# Create a DeltaTable object  
delta_path = "Files/mytable"  
deltaTable = DeltaTable.forPath(spark, delta_path)  
  
# Update the table (reduce price of accessories by 10%)  
deltaTable.update(  
    condition = "Category == 'Accessories'",  
    set = { "Price": "Price * 0.9" })
```

# Spark Structured Streaming

Read data from many different kinds of streaming source, including:

- Network ports
- Real time message brokering services such as Azure Event Hubs or Kafka
- File system locations

# Use Delta lake as a streaming source

## Work with streaming data in a dataframe

```
%%sql # Read and display the input table
INSERT IN df = spark.read.format("delta").table("orders_in")
Price) display(df)
VALUES
(3001 # Load a streaming DataFrame from the Delta table
(3002 stream_df = spark.readStream.format("delta") \
(3003     .option("ignoreChanges", "true") \
(3004     .table("orders_in")
(3005

# Verify that the stream is streaming
stream_df.isStreaming
```

# Use Delta lake as a streaming sink

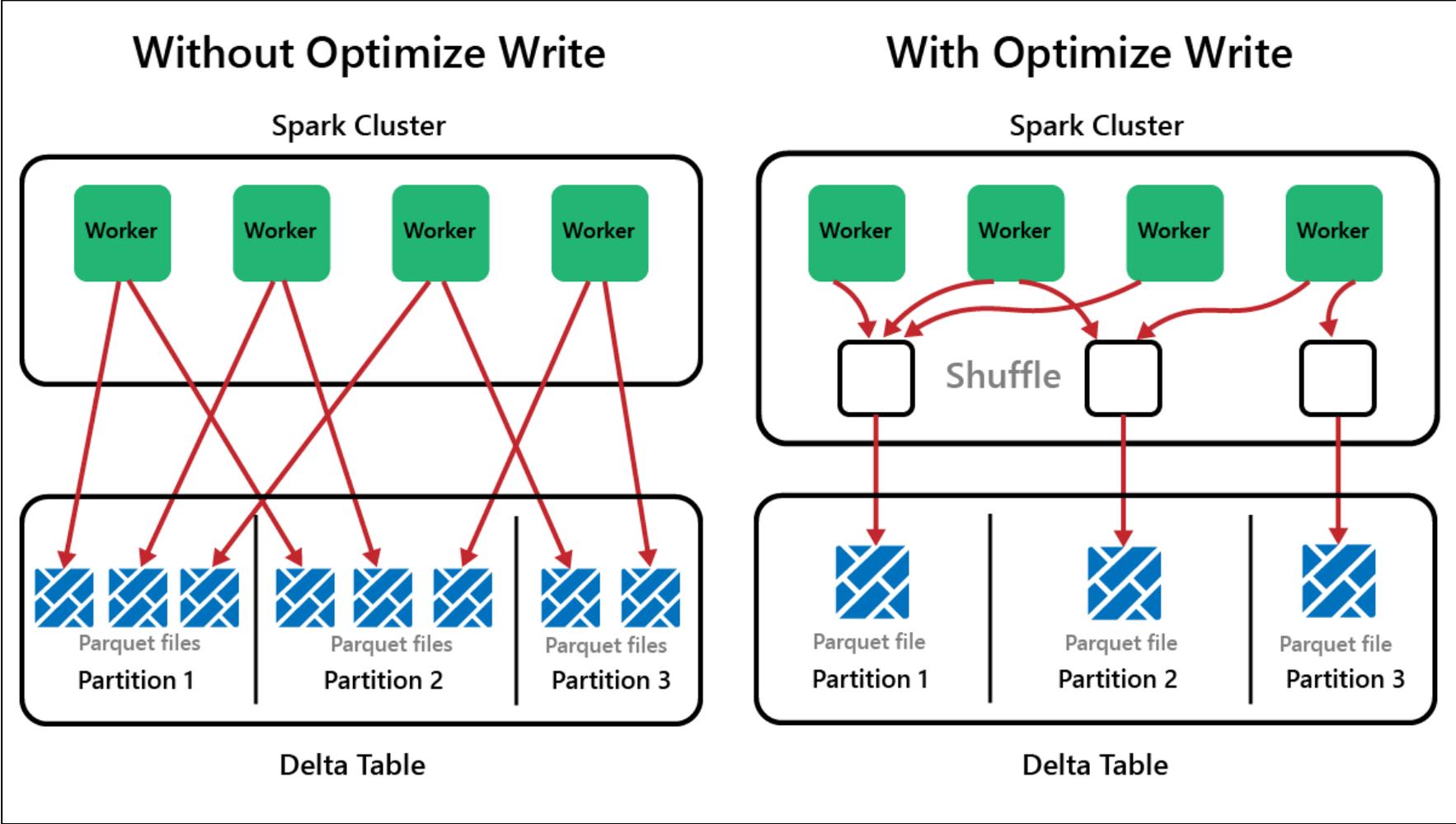
Write to a Delta table, query table, and stop stream.

```
# Write the stream to a delta table
output_table_path = 'Tables/orders_processed'
checkpointpath = 'Files/delta/checkpoint'
deltastream =
transformed_df.writeStream.format("delta").option("checkpointLocation",
checkpointpath).start(output_table_path)

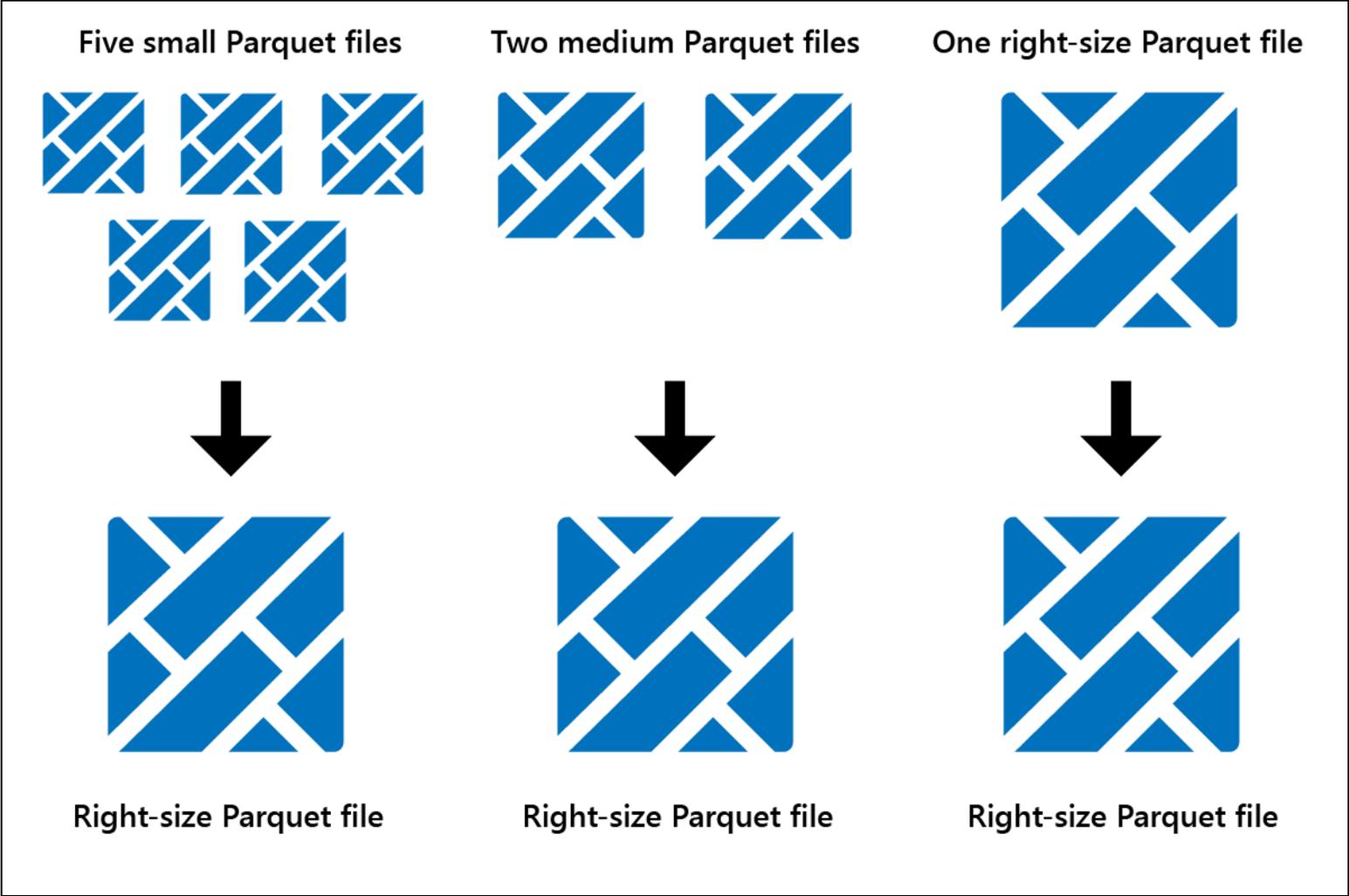
print("Streaming to orders_processed...")

# Stop the streaming data to avoid excessive processing costs
delta_stream.stop()
```

# OptimizeWrite function



# Optimize command



# V-Order function

Home

Get data ▾ New

A SQL analytics endpoint for SQL quer

Explorer

lakehouse2

Tables

products

Files

## Run maintenance commands

### Optimize file size

Run the Delta Lake OPTIMIZE command in Spark to compact and rewrite Delta Parquet files. [Learn more](#)

- Run OPTIMIZE command
- Apply V-order to maximize reading speeds in Fabric ⓘ

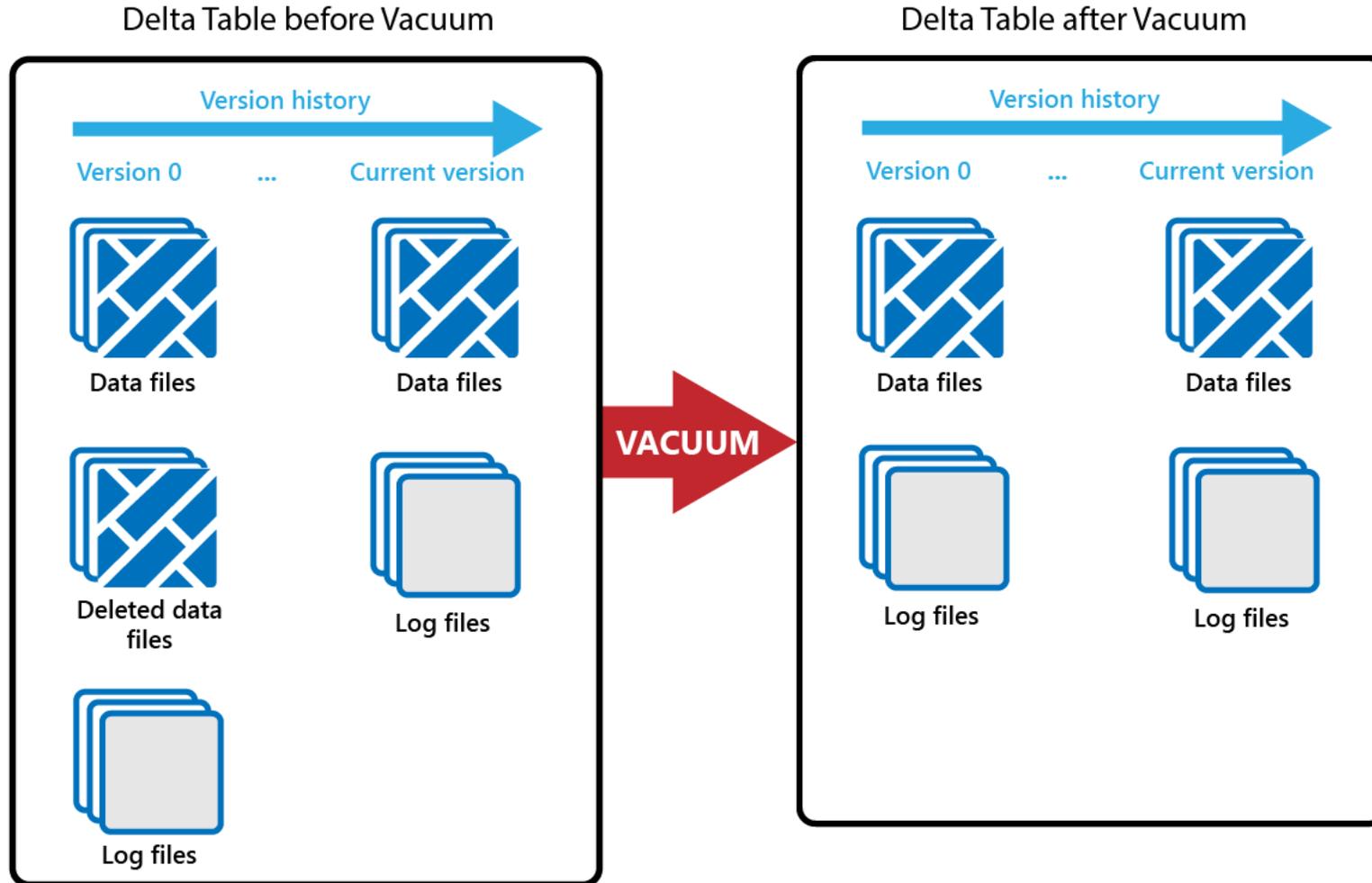
### Remove unreferenced files

Run the Delta Lake VACUUM command in Spark to remove files that are no longer referenced and that are older than the retention threshold you set below. [Learn more](#)

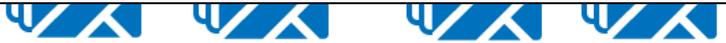
Run VACUUM command using retention threshold

**Run now** Cancel

# Vacuum command



# Partition files

Non-partitioned Delta table	Partitioned Delta table using single column	Partitioned Delta table using multiple columns								
<p>Explorer</p> <ul style="list-style-type: none"><li>lakehouse1<ul style="list-style-type: none"><li>Tables</li><li>Files<ul style="list-style-type: none"><li>partitioned_products<ul style="list-style-type: none"><li>Category=Bib-Shorts</li><li><b>Category=Bike Racks</b></li><li>Category=Bike Stands</li><li>Category=Bottles and Cages</li><li>Category=Bottom Brackets</li><li>Category=Brakes</li></ul></li></ul></li></ul></li></ul>	<p>Files &gt; partitioned_products &gt; Category=Bike Racks</p> <table border="1"><thead><tr><th>Name</th><th>Date modified</th><th>Type</th><th>Size</th></tr></thead><tbody><tr><td>part-00022-0ae3e0aa-093e-4470-807c-c77ed1d...</td><td>8/21/2024 2:2...</td><td>parquet</td><td>1 KB</td></tr></tbody></table>	Name	Date modified	Type	Size	part-00022-0ae3e0aa-093e-4470-807c-c77ed1d...	8/21/2024 2:2...	parquet	1 KB	 <p>Parquet files</p>
Name	Date modified	Type	Size							
part-00022-0ae3e0aa-093e-4470-807c-c77ed1d...	8/21/2024 2:2...	parquet	1 KB							

# Data versioning and time travel

Delta tables have transaction logs to “time travel” to previous versions of data.

```
# Use SQL to see the history of a table
```

version	timestamp	operation	operationParameters
2	2023-04-04T21:46:43Z	UPDATE	{"predicate":"(ProductId = 1)"}
1	2023-04-04T21:42:48Z	WRITE	{"mode":"Append","partitionBy":[]}
0	2023-04-04T20:04:23Z	CREATE TABLE	{"isManaged":"true","description":null,"partitionBy":[],"properties":{}}

# Exercise



45 minutes

## Use delta tables in Apache Spark

# Knowledge check



**1** Which of the following descriptions best fits Delta Lake?

- A Spark API for exporting data from a relational database into CSV files
- A relational storage layer for Spark that supports tables based on Parquet files
- A synchronization solution that replicates data between SQL Server and SPark

**2** You've loaded a Spark dataframe with data, that you now want to use in a delta table. What format should you use to write the dataframe to storage?

- CSV
- PARQUET
- DELTA

**3** You have a managed table based on a folder that contains data files in delta format. If you drop the table, what happens?

- The table metadata and data files are deleted
- The table definition is removed from the metastore, but the data files remain intact
- The table definition remains in the metastore, but the data files are deleted

# Recap

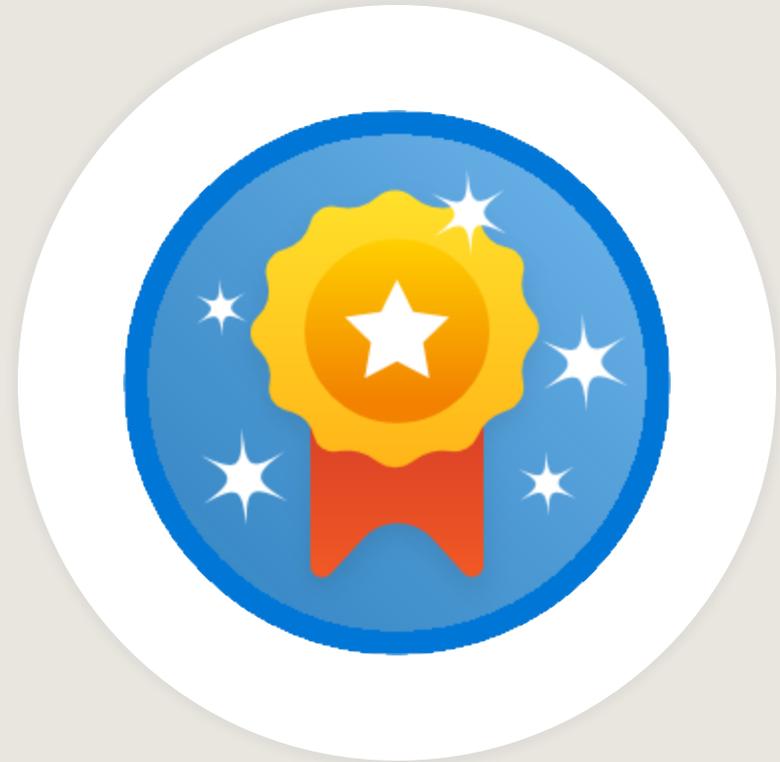
In this section, we covered:

- Delta Lake adds relational database semantics to Apache Spark.
- Microsoft Fabric lakehouse tables are based on Delta Lake.
- Utilize advanced features and techniques through the Delta Lake API.

# Further Reading

Work with Delta Lake tables in Microsoft Fabric

<https://aka.ms/fabric-delta>



# Ingest data with Dataflow Gen2 in Microsoft Fabric



# Learning objectives

- Describe Dataflow Gen2 capabilities in Microsoft Fabric
- Create Dataflow solutions to ingest and transform data
- Include a Dataflow in a pipeline

# Understand Dataflows Gen2

- Low-code graphical environment for defining ETL solutions
- Extract data from multiple sources, transform it, and load it into a destination
- Run dataflows independently or as an activity in a Pipeline

The screenshot displays the Microsoft Fabric Dataflows Gen2 interface. The main workspace shows a dataflow named 'sales' with three steps: 'Source', 'Promoted headers', and 'Changed column type'. The 'Changed column type' step is currently selected, and its settings are visible in the right-hand panel. The panel includes sections for 'Query settings', 'Properties', 'Applied steps', and 'Data destination'. A dropdown menu is open under 'Data destination', showing options like 'Lakehouse', 'Warehouse', 'SQL database', 'Azure SQL database', and 'Azure Data Explorer (Kusto)'. The bottom of the interface shows a table of data with columns: SalesOrderNumber, SalesOrderLineNumber, OrderDate, CustomerName, and En. The table contains 7 rows of data. The status bar at the bottom indicates 'Completed (3:20 s) Columns: 9 Rows: 99+'. A 'Publish' button is visible in the bottom right corner.

	SalesOrderNumber	SalesOrderLineNumber	OrderDate	CustomerName	En
1	SO43701	1	7/1/2019	Christy Zhu	chr
2	SO43704	1	7/1/2019	Julio Ruiz	juli
3	SO43705	1	7/1/2019	Curtis Lu	cur
4	SO43700	1	7/1/2019	Ruben Prasad	rub
5	SO43703	1	7/1/2019	Albert Alvarez	alb
6	SO43697	1	7/1/2019	Cole Watson	col
7	SO43699	1	7/1/2019	Eden Watson	ede

# Dataflow Gen2 benefits and limitations

## Benefits:

- Self-service access to data.
- Optimize performance with dataflows.
- Simplify data source complexity.
- Ensure data consistency and quality.

## Limitations:

- Not a data warehouse replacement.
- No row-level security.
- Requires Fabric capacity workspace.

# Explore Dataflow Gen2

1. Power Query ribbon
2. Queries pane
3. Diagram view
4. Data preview
5. Query settings pane

The screenshot displays the Microsoft Power Query interface with five numbered callouts highlighting key features:

- 1. Power Query ribbon:** The top ribbon includes tabs for Home, Transform, Add column, View, and Help. The Home tab contains various actions like 'Get data', 'Enter data', 'Options', 'Manage parameters', 'Refresh', 'Advanced editor', 'Add data destination', 'Choose columns', 'Remove columns', 'Keep rows', 'Remove rows', 'Filter rows', 'Sort', 'Transform', 'Combine', 'Map to entity', and 'Export temp'.
- 2. Queries pane:** Located on the left, it shows a list of queries including 'orders' and 'orders-2'.
- 3. Diagram view:** The central area shows a dependency diagram with nodes for 'orders-2' (3 steps) and 'orders' (9 steps).
- 4. Data preview:** A table view showing columns: SalesOrderID, OrderDate, MonthNo, CustomerID, LinelItem, and ProductID. The data includes 15 rows of sample information.
- 5. Query settings pane:** On the right, it shows the 'Query settings' for the 'orders' query, including Properties (Name: orders), Entity type (Custom), Applied steps (Source, Promoted h..., Changed co..., Appended ..., Filtered rows, Added cust..., Reordered c..., Changed co..., Changed co...), and Data destination (Lakehouse).

At the bottom of the interface, it indicates 'Columns: 9 Rows: 99+' and a 'Publish' button.

# Integrate Dataflow Gen2 and pipelines

Common activities include:

- Copy data
- Incorporate Dataflow
- Add Notebook
- Get metadata
- Execute a script or stored procedure

The screenshot displays the configuration page for a pipeline named 'pipeline2' in the Azure Data Factory portal. The interface is divided into several sections:

- Navigation:** 'Home', 'Activities', 'Run', and 'View' tabs are visible at the top.
- Toolbar:** Includes icons for saving, copying, adding activities, and settings.
- Activity List:** A 'Dataflow' activity is shown with an external link icon.
- General Settings:**
  - Name:** Dataflow
  - Description:** (Empty text area)
  - Activity state:** Active (indicated by a green dot)
  - Timeout:** 0.12:00
  - Retry:** 0
  - Advanced:** (Expandable section)
- Right Panel:**
  - About:** Shows 'pipeline2' as a 'Data pipeline'.
  - Endorsement:** (Empty section)
  - Schedule:**
    - History: 'No previous history'.
    - Refresh: 'The scheduled refresh is turned off'.
    - Buttons: 'Run' (refresh icon) and 'Schedule'.
    - Schedule Configuration:**
      - Schedule:** (Clock icon)
      - Scheduled run:** Off (radio button selected)
      - Repeat:** By the minute (dropdown menu)
      - Every:** 15 minute(s)
      - Start date and time:** mm/dd/yyyy --:-- --
      - Time zone:** (Label)

# Exercise

30 minutes



## Create and use a Dataflow Gen2 in Microsoft Fabric

# Knowledge check



## 1 What is a Dataflow Gen2?

- A hybrid database that supports ACID transactions
- A way to export data to Power BI Desktop
- A way to import and transform data with Power Query Online

## 2 Which Fabric experience lets you create a Dataflow Gen2?

- Real-time analytics
- Data warehouse
- Data Factory

## 3 How do Dataflow Gen2 in Microsoft Fabric facilitate collaboration?

- By providing a shared and collaborative environment for data transformation and integration
- By enabling real-time data streaming and event processing
- By integrating with popular business intelligence tools for data visualization

# Recap

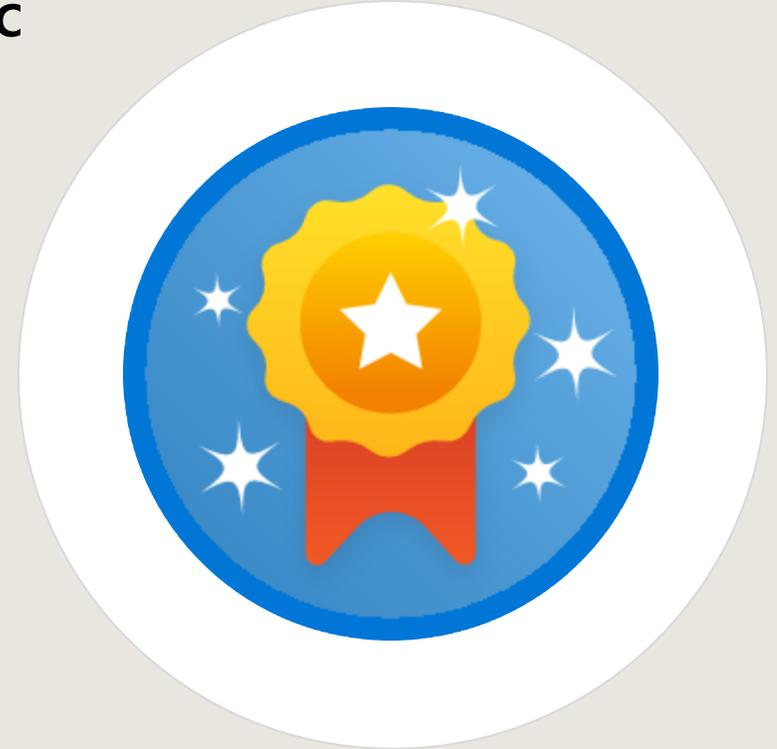
In this section, we covered:

- Dataflow Gen2 capabilities in Microsoft Fabric.
- Dataflow Gen2 solutions to ingest and transform data.
- Using a Dataflow Gen2 in a pipeline.

# Further reading

Ingest Data with Dataflow Gen2 in Microsoft Fabric

<https://aka.ms/fabric-dataflow>



# Orchestrate processes and data movement with Microsoft Fabric



# Learning objectives

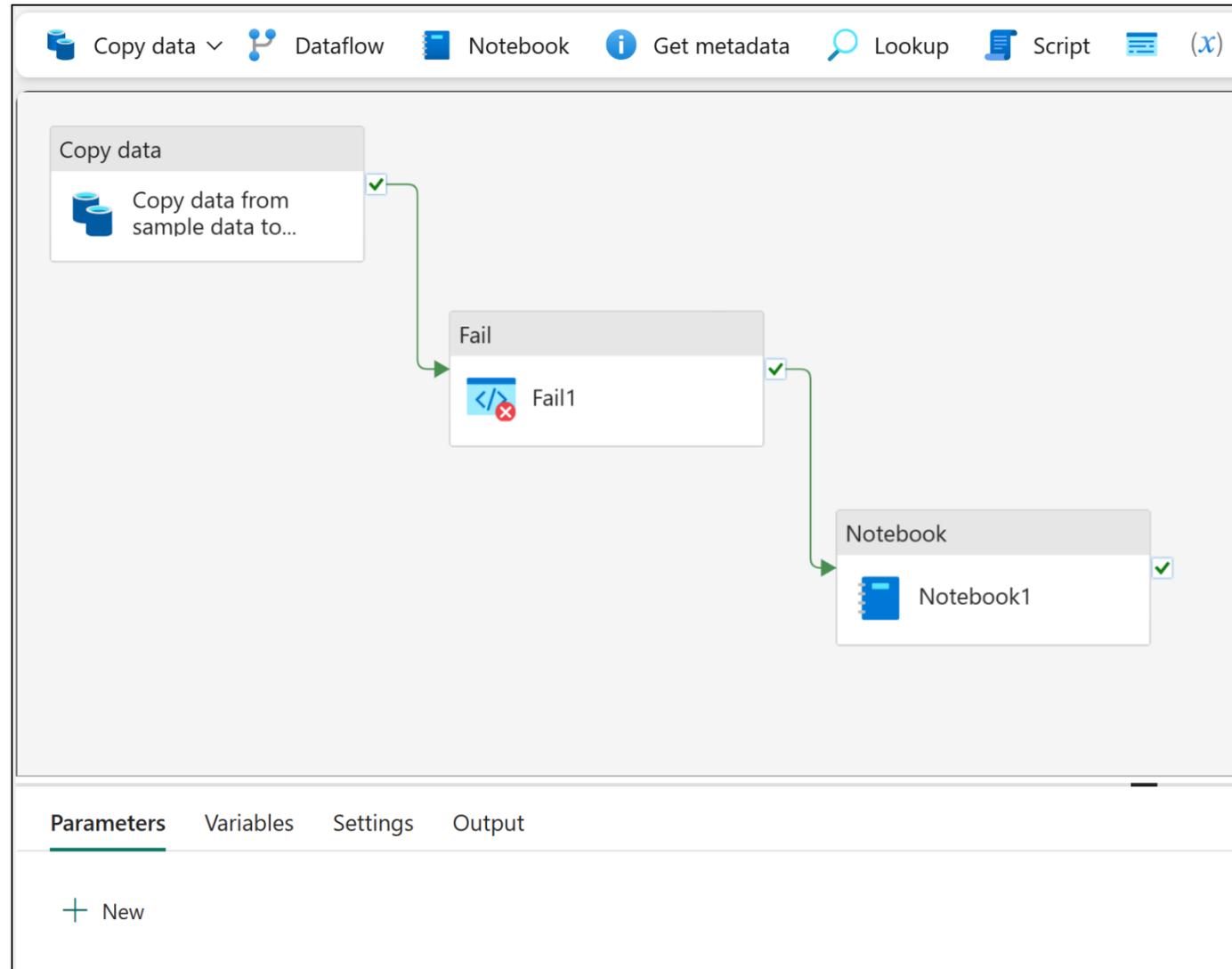


- Describe pipeline capabilities in Microsoft Fabric.
- Use the Copy Data activity in a pipeline.
- Create pipelines based on predefined templates.
- Run and monitor pipelines.

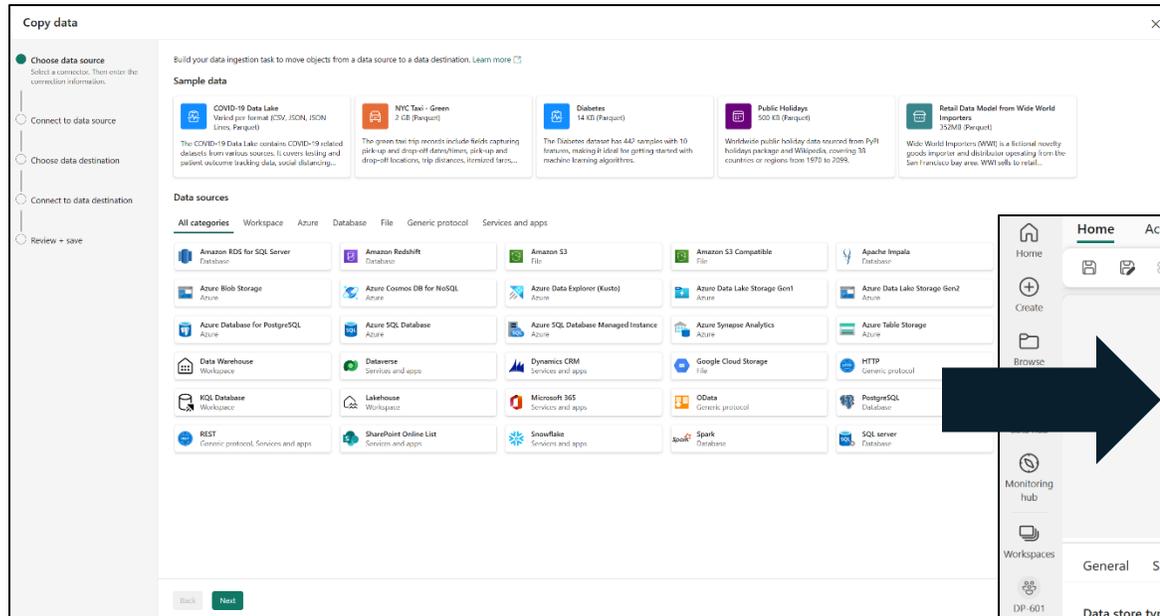
# Pipelines in Microsoft Fabric

## Pipeline concepts:

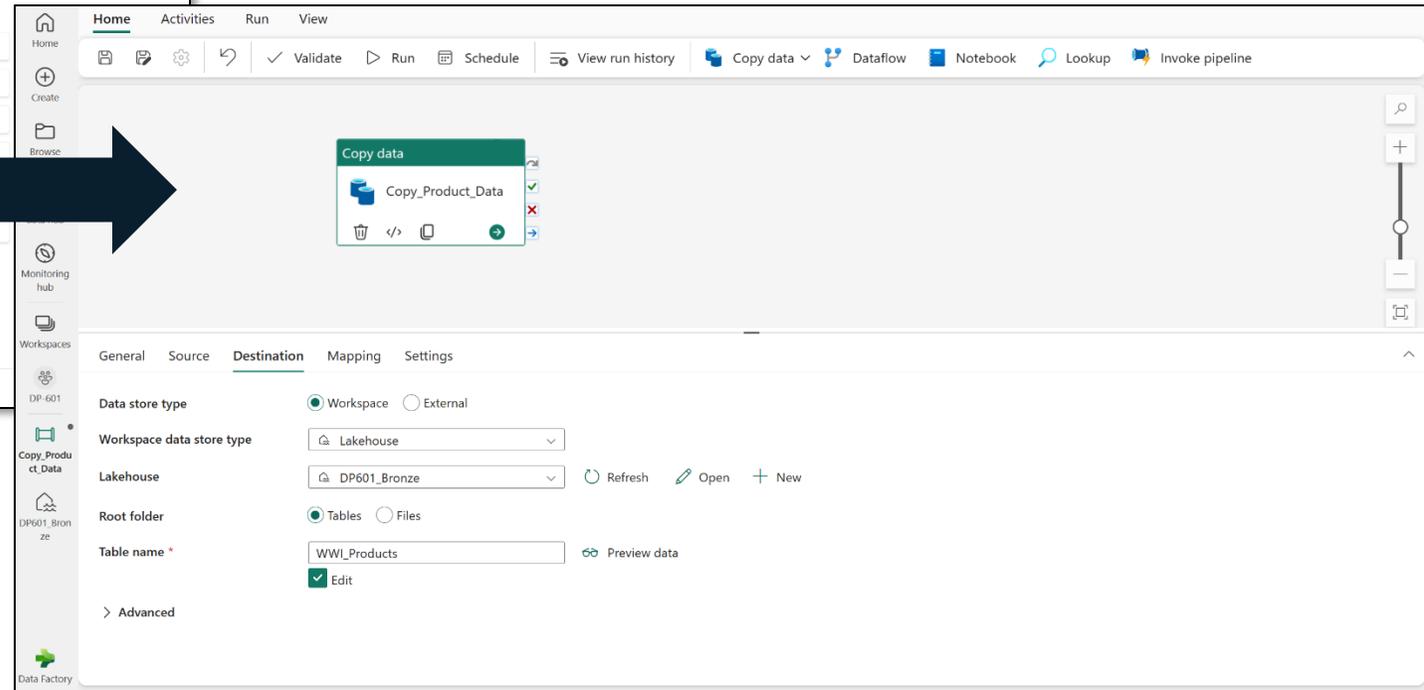
- Activities
  - Data transformation
  - Control flow
- Parameters
- Pipeline runs



# Common Activities – copy data



1. Use the copy data tool.



2. Edit the settings below the pipeline canvas.

# Use templates for common activities



## 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...

# Run and monitor pipelines

1. Validate
2. Run
3. Schedule
4. View run history

The screenshot displays the Azure Data Factory (ADF) interface. The top navigation bar includes 'Home', 'Activities', 'Run', and 'View'. A red box highlights the 'Run' section, which contains 'Validate', 'Run', 'Schedule', and 'View run history' buttons. The main workspace shows a 'Copy data' activity with the name 'Copy from To process to In...'. Below the workspace, the 'General' tab is active, showing fields for 'Name', 'Description', 'Timeout', and 'Retry'. The 'Run history' table on the right lists the execution status of the 'Process rides pipeline' activity.

Name	Status	Start time
Process rides pipeline	Completed	3:17 PM, 6/21/23
Process rides pipeline	Failed	2:31 PM, 6/28/23
Process rides pipeline	Completed	2:35 PM, 6/28/23
Process rides pipeline	Completed	2:44 PM, 6/28/23
Process rides pipeline	Completed	2:50 PM, 6/28/23
Process rides pipeline	Completed	3:41 PM, 6/28/23
Process rides pipeline	Completed	3:50 PM, 6/28/23
Process rides pipeline	Completed	3:33 PM, 6/30/23
Process rides pipeline	Completed	3:55 PM, 6/30/23

# Exercise



30 minutes

## Ingest data with a pipeline

# Knowledge check



## 1 What is a data pipeline?

- A special folder in OneLake storage where data can be exported from a lakehouse.
- A sequence of activities to orchestrate a data ingestion or transformation process.
- A saved Power Query.

## 2 You want to use a pipeline to copy data to a folder with a specified name for each run. What should you do?

- Create multiple pipelines – One for each folder name.
- Use a Dataflow Gen2.
- Add a parameter to the pipeline and use it to specify the folder name for each run.

## 3 You have previously run a pipeline containing multiple activities. What's the best way to check how long each individual activity took to complete?

- Rerun the pipeline and observe the output, timing each activity.
- View the run details in the run history.
- View the refreshed value for your lakehouse's default semantic model.

# Recap

In this section, we covered:

- How pipelines can be used to orchestrate processes and data movement.
- Common activities and ready to use templates.
- Running and monitoring pipelines.

# Further Reading

Orchestrate processes and data movement with Microsoft Fabric

<https://aka.ms/fabric-pipeline>

