O'REILLY

Microsoft Fabric Data Engineer Bootcamp (DP-700)

Learn in-demand skills and become a certified Fabric Data Engineer

Schedule Day 1

- Introduction
- Implementing and managing an analytics solution
  - Workspace settings
  - Lifecycle management
  - Security and governance
- Ingesting and transforming data
  - Orchestrating processes
  - Design and implement loading patterns

Schedule Day 2

- Ingesting and transforming data
  - Batch data
  - Streaming data
- Monitoring Fabric items
- Identifying and resolving errors
- Optimizing performance
- Practical exam tips and quiz

Learning Objectives

- Transform raw data into actionable insights by developing robust data engineering solutions in Microsoft Fabric
- Process and enrich data to enhance the analytics value
- Configure and secure Microsoft Fabric assets
- Monitor and optimize data engineering solutions

What Best Describes Your Data Profile? (POLL)

1. Data engineer
2. Analytics engineer/BI developer
3. Data analyst
4. Data/Solution architect
5. Business user
6. Non-technical user

What's Your Experience With Microsoft Fabric? (POLL)

1. What is Microsoft Fabric?
2. Heard some people talking about it, but I never tried it myself
3. Already built a few solutions using Microsoft Fabric
4. I'm an experienced data engineer, but I've used other tools (Databricks, Airflow, Kafka, dbt…)
5. Can't start my day without coffee and Microsoft Fabric


O'REILLY®
Day 1
Microsoft Fabric Data Engineer
Bootcamp (DP-700)

Introduction and Overview

What is DP-700?

1. One exam: Implementing Data Engineering Solutions Using MS Fabric
2. Data Engineering
3. Understand features and services in MS Fabric

Candidate Profile

Expertise in designing, creating, and deploying enterprise-scale data analytics solutions

Responsibilities include

- Ingesting and transforming data
- Securing and managing an analytics solution
- Monitoring and optimizing an analytics solution

Candidates should have

- SQL
- PySpark
- KQL
- Data modeling

Skills Measured Breakdown

Implement and manage an analytics solution (30-35%)

Ingest and transform data (30-35%)

Monitor and optimize an analytics solution (30-35%)

Microsoft Fabric Core Components

Microsoft Fabric Core Components

Data Factory

Real-Time Intelligence

Databases

Analytics

Industry Solutions

Power BI

Partner solutions

Copilot in Fabric

OneLake

Microsoft Purview

Players in Microsoft Fabric

Data Factory

Real-Time Intelligence

Databases

Analytics

Industry Solutions

Power BI

Partner solutions

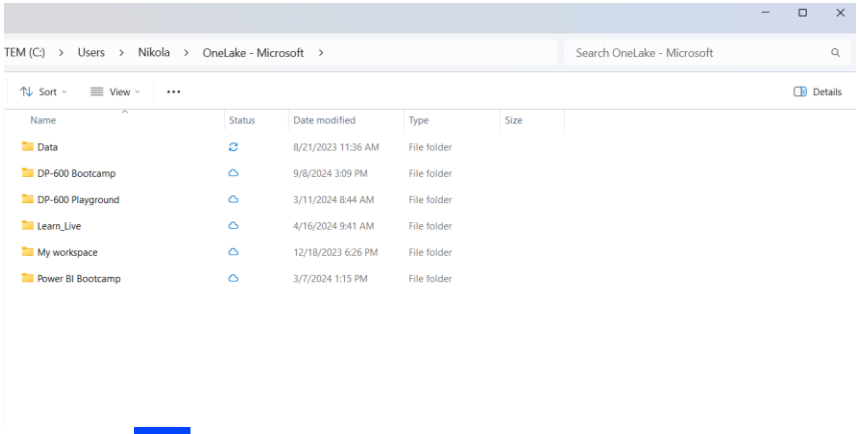
- Data Factory: Data integration combining Power Query with the scale of Azure Data Factory to move and transform data
- Analytics - Data Engineering: Data engineering with a Spark platform for data transformation at scale
- Analytics - Data Warehouse: Data warehousing with SQL performance and scale to support data use
- Analytics - Data Science: Data Science with Azure Machine Learning and Spark for model training and execution tracking
- Real-Time Intelligence: Real-time analytics to query and analyze large volumes of data in real-time
- Power BI: Business intelligence for translating data into decisions
- Databases: Handling OLTP scenarios using SaaS transactional SQL database

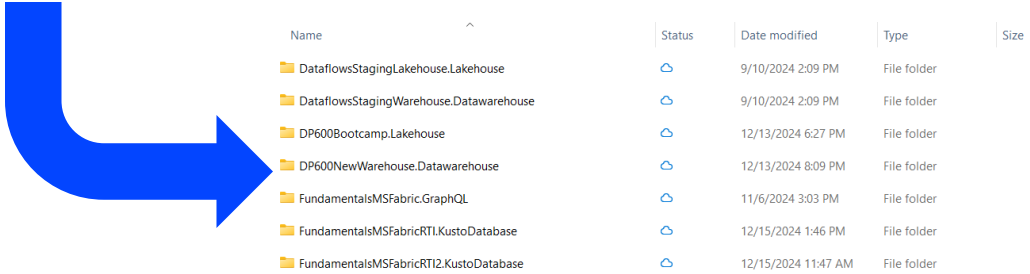
Microsoft Fabric Core Components

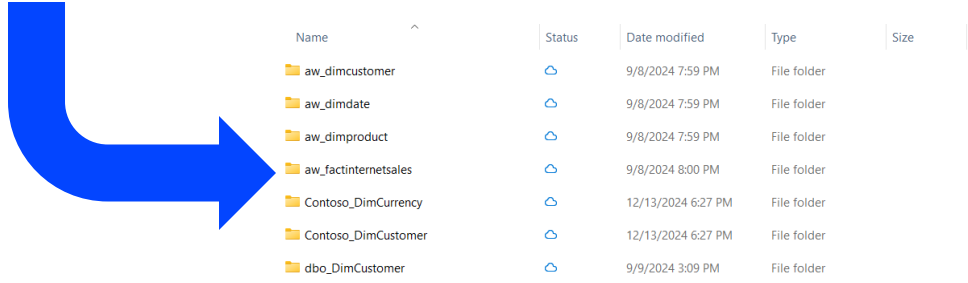
OneLake

Central storage repository for the entire organization!

1 Fabric tenant = 1 OneLake

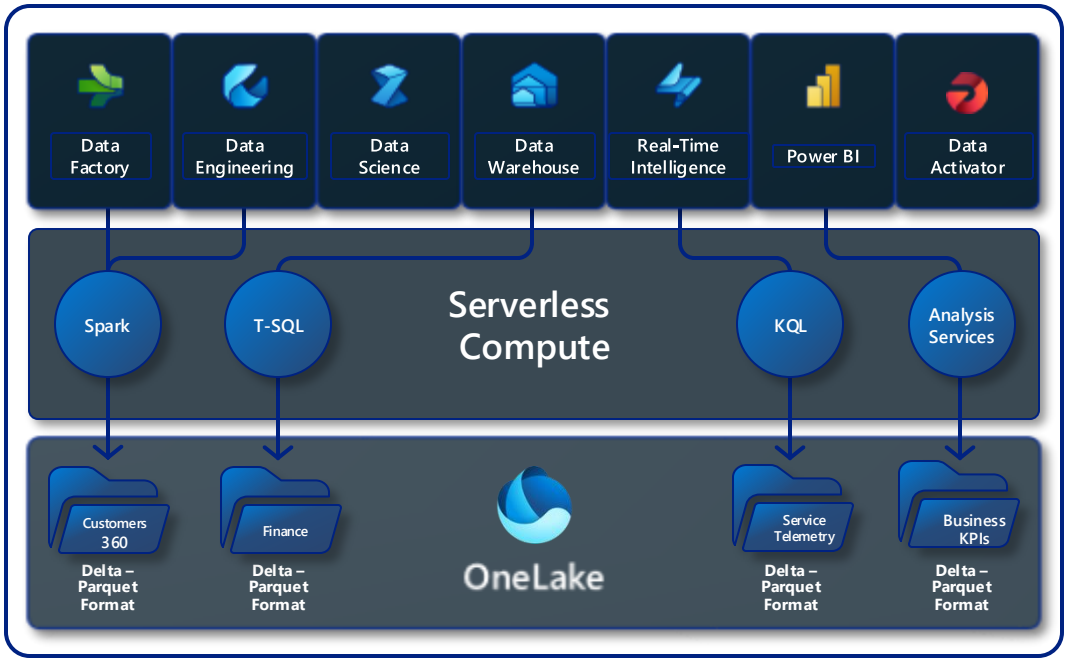

TEM (C:) › Users › Nikola › OneLake - Microsoft ›
Search OneLake - Microsoft
Sort
View
Details
Name
Status
Date modified
Type
Size
Data
8/21/2023 11:36 AM
File folder
DP-600 Bootcamp
9/8/2024 3:09 PM
File folder
DP-600 Playground
3/11/2024 8:44 AM
File folder
Learn_Live
4/16/2024 9:41 AM
File folder
My workspace
12/18/2023 6:26 PM
File folder
Power BI Bootcamp
3/7/2024 1:15 PM
File folder


Name
Status
Date modified
Type
Size
DataflowsStagingLakehouse.Lakehouse
9/10/2024 2:09 PM
File folder
DataflowsStagingWarehouse.Datawarehouse
9/10/2024 2:09 PM
File folder
DP600Bootcamp.Lakehouse
12/13/2024 6:27 PM
File folder
DP600NewWarehouse.Datawarehouse
12/13/2024 8:09 PM
File folder
FundamentalsMSFabric.GraphQL
11/6/2024 3:03 PM
File folder
FundamentalsMSFabricRTI.KustoDatabase
12/15/2024 1:46 PM
File folder
FundamentalsMSFabricRTI2.KustoDatabase
12/15/2024 11:47 AM
File folder

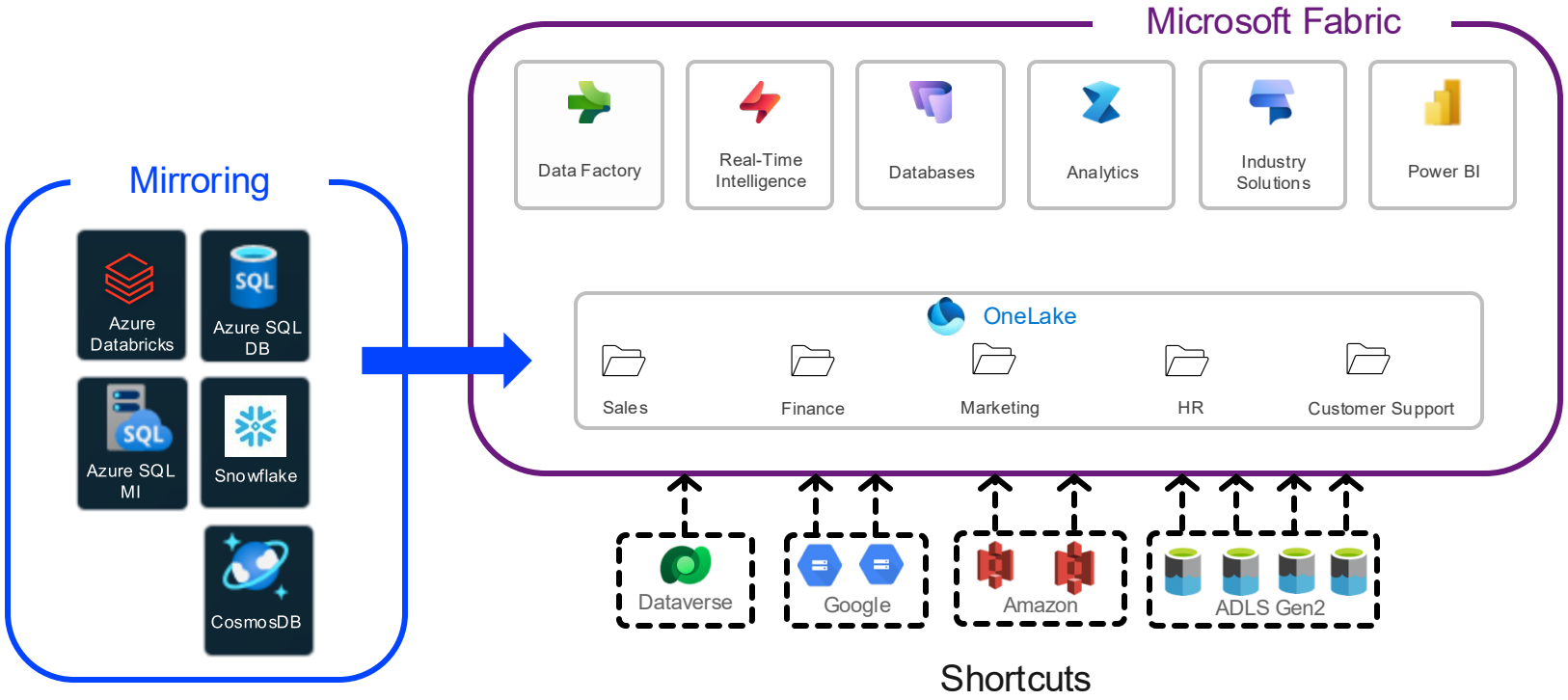

Name
Status
Date modified
Type
Size
aw_dimcustomer
9/8/2024 7:59 PM
File folder
aw_dimdate
9/8/2024 7:59 PM
File folder
aw_dimproduct
9/8/2024 7:59 PM
File folder
aw_factinternetsales
9/8/2024 8:00 PM
File folder
Contoso_DimCurrency
12/13/2024 6:27 PM
File folder
Contoso_DimCustomer
12/13/2024 6:27 PM
File folder
dbo_DimCustomer
9/9/2024 3:09 PM
File folder

One Copy for all computes

Real separation of compute and storage



All Roads Lead to… ~~Rome~~ OneLake



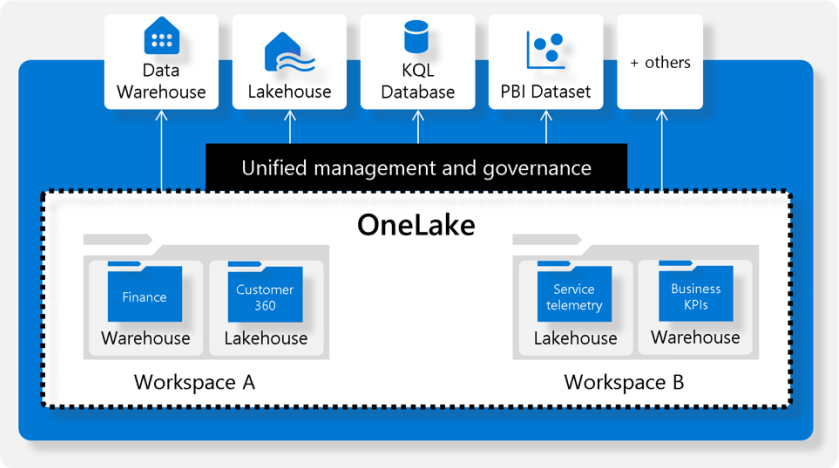
Lakehouse

- ✓ Files -> Unmanaged area -> store any format
- ✓ Tables -> Managed by Spark -> preferably Delta

- ✓ Read-only for T-SQL engine
- ✓ Only delta tables are exposed
- ✓ Needed for DirectQuery fallback

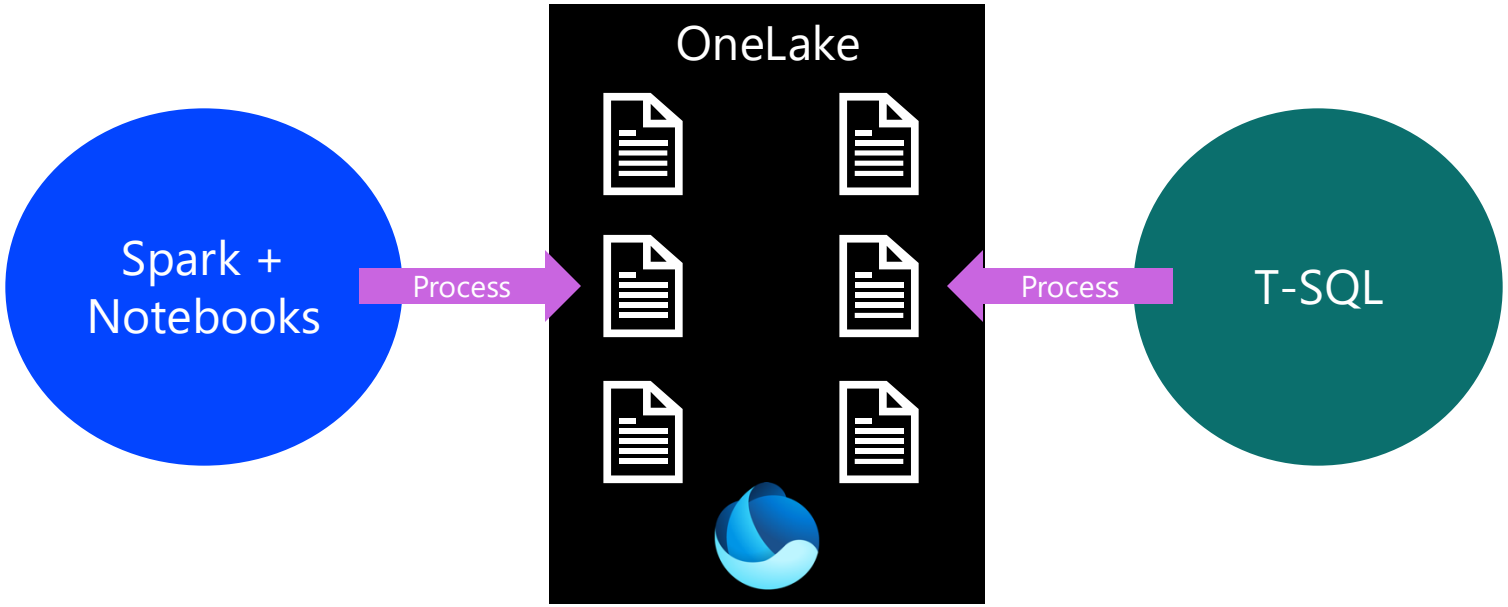
Warehouse

- Centered on the single data lake (OneLake)
- Powered by Synapse Analytics
- (Almost) Fully supports T-SQL
- Parquet file format




Lakehouse


Warehouse


OneLake
Spark + Notebooks
Process
Process
T-SQL

Q&A

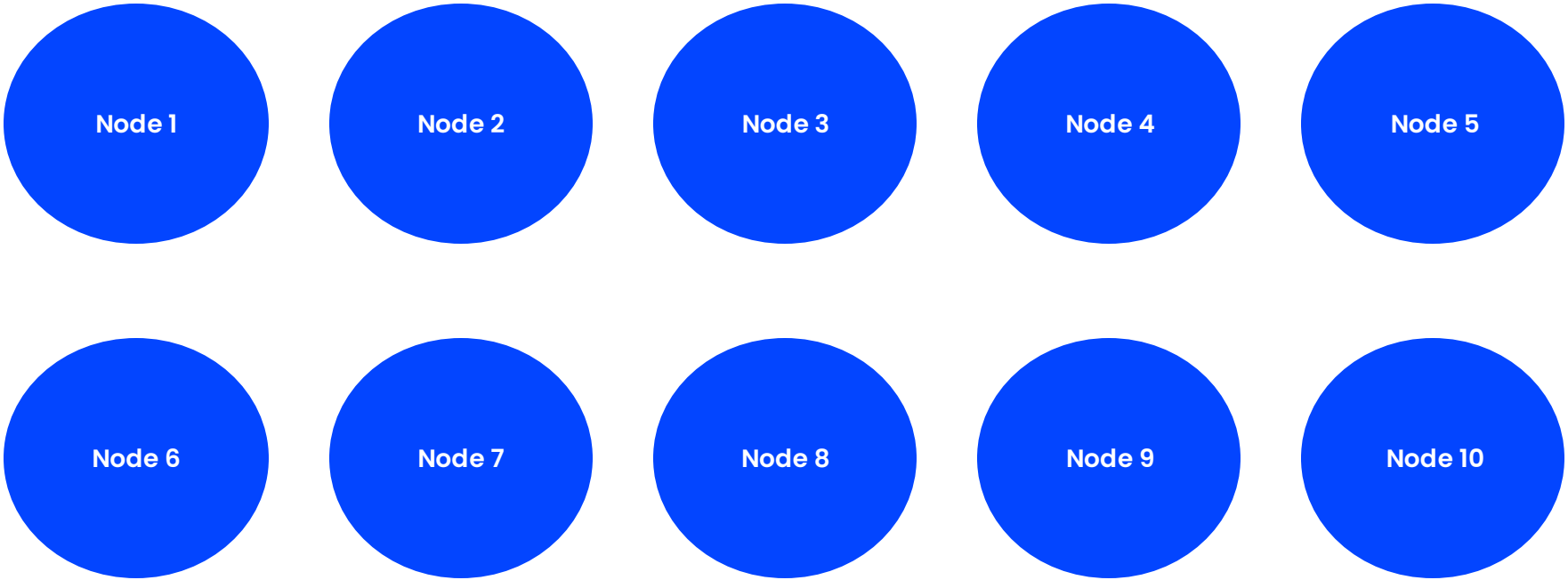
Implement and Manage an Analytics Solution

Configure Workspace Settings

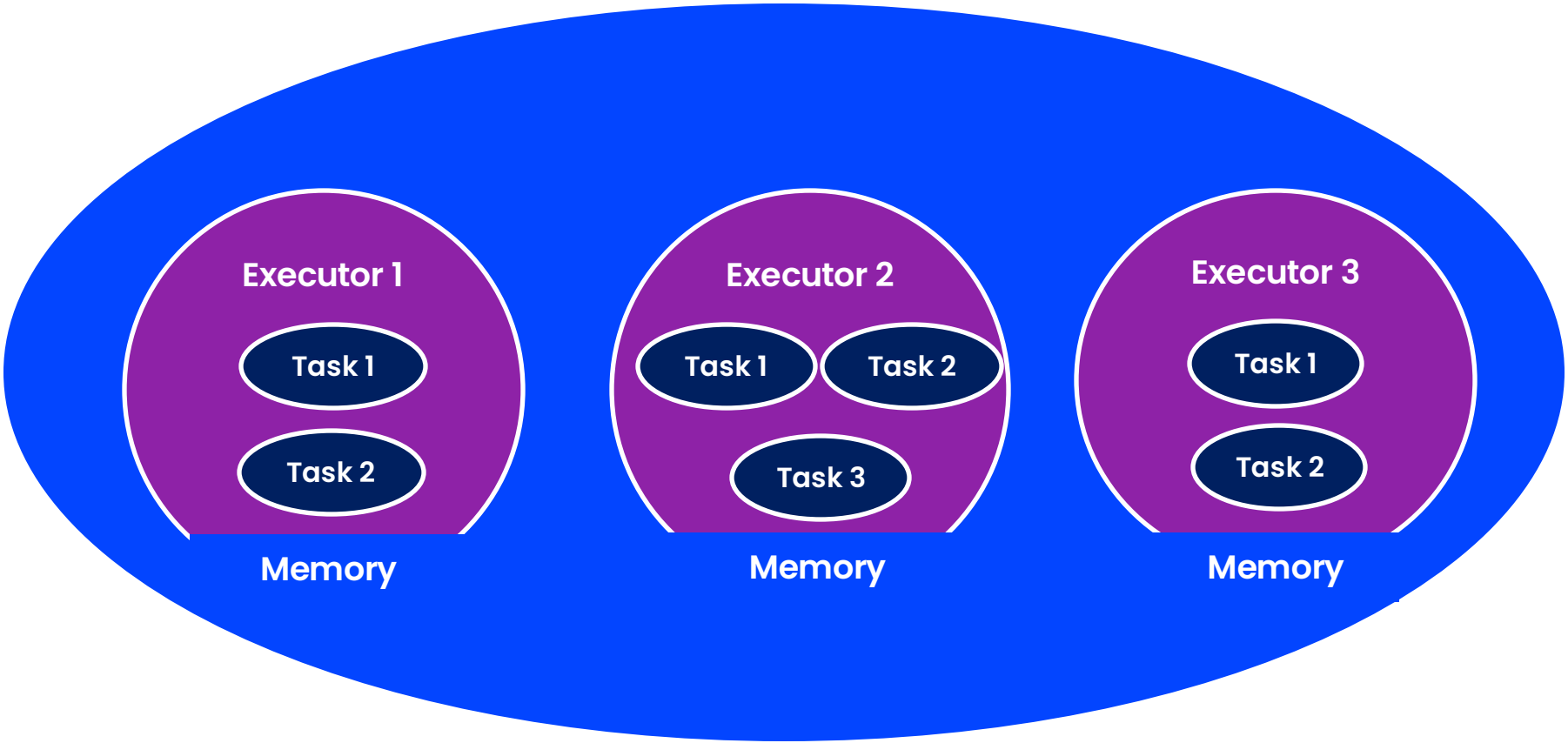
Configure workspace settings

- Configure Spark workspace settings
- Configure domain workspace settings
- Configure OneLake workspace settings
- Configure data workflow workspace settings

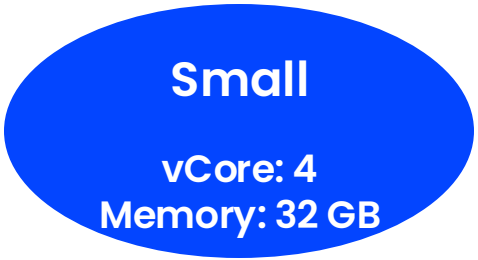
How Spark Works?

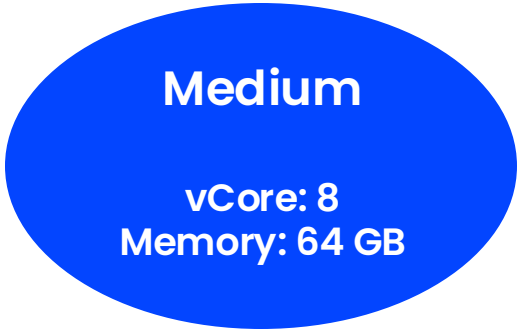


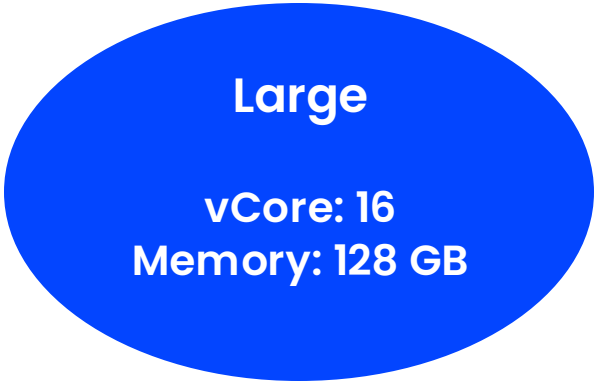
How Spark Works?

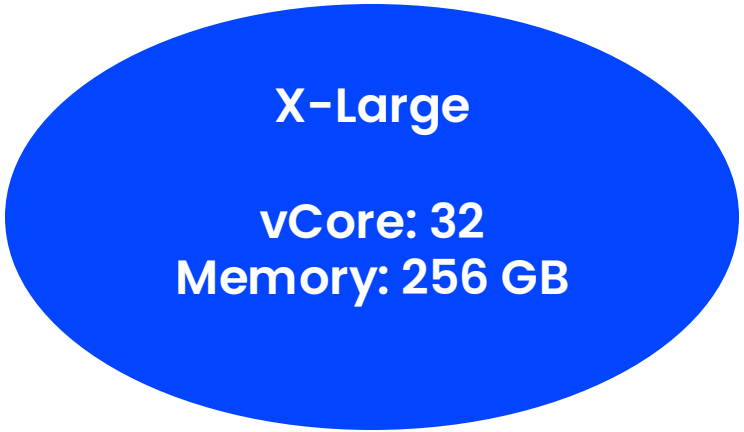


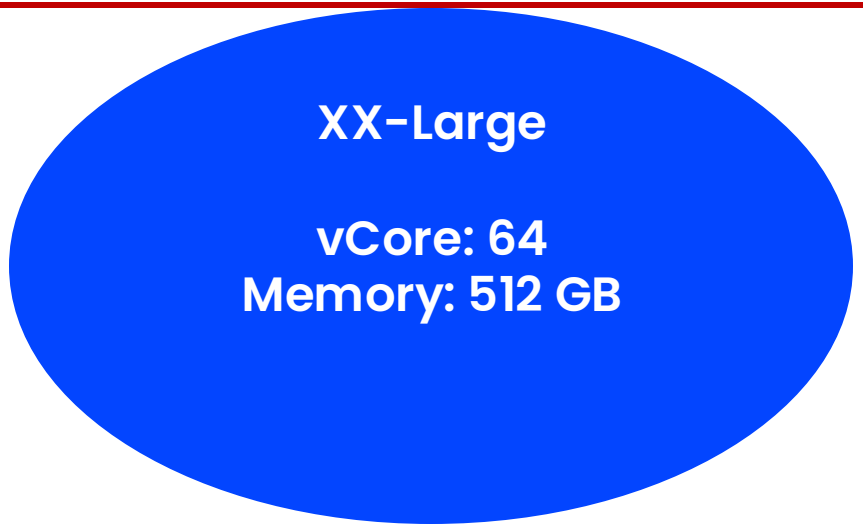
Node Sizes











Non-trial SKUs

How Spark Works?

Autoscale

- Automatic scale up and down of compute based on the amount of activity
- You define the minimum and maximum number of nodes

Dynamic allocation

- Spark may request more executors if the tasks exceed the load
- Hard to configure on your own
- You set the maximum limit for executors

Spark Workspace Settings – Pool

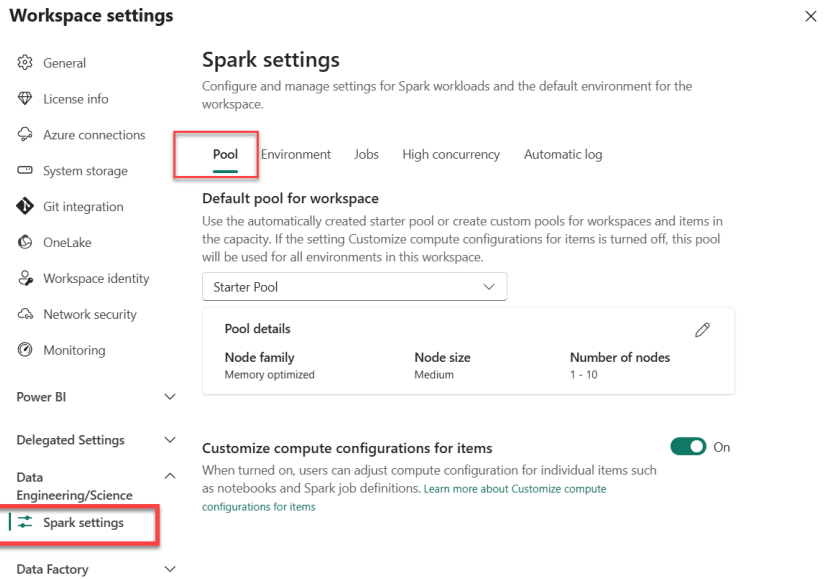
Admin role in the workspace

Starter pool

- Automatically created
- Faster Spark session launch (~ 10 seconds)

Custom pool

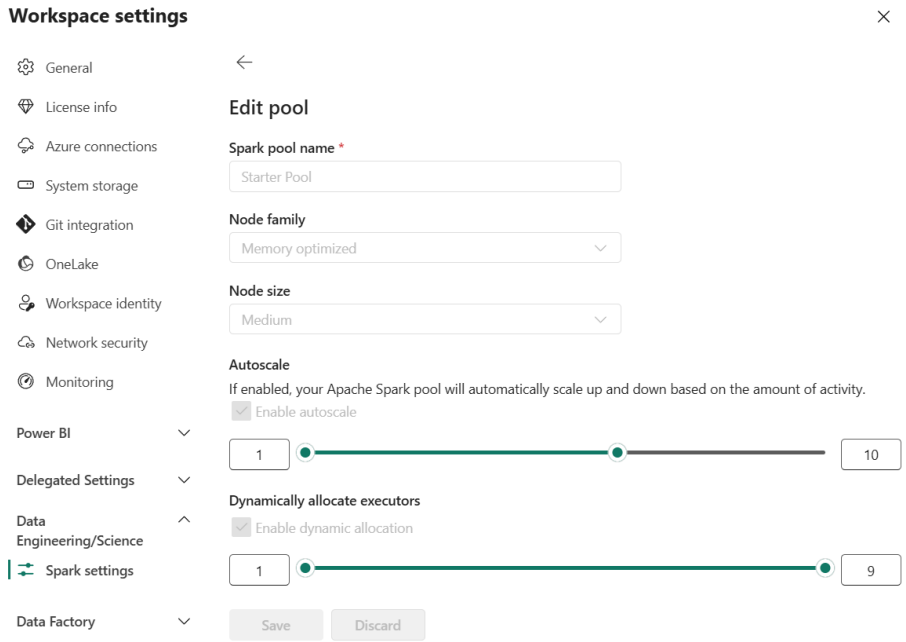
- Adjust node size
- Autoscale
- Dynamically allocate executors



Spark Workspace Settings – Starter Pool

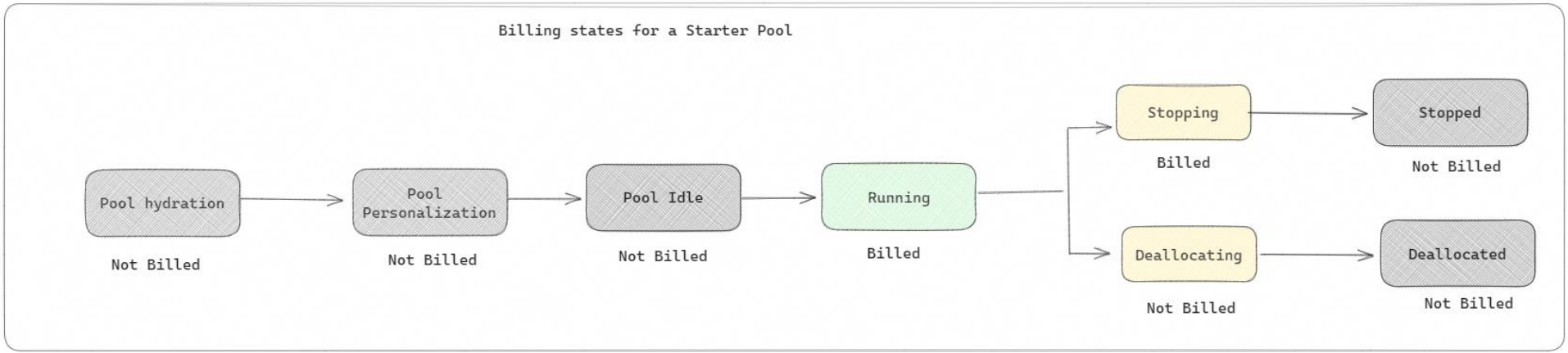
Things to consider

- The size of the data you need to process
- Concurrency level
- Spikes required for certain tasks
- Diversity of workloads – data engineers vs. data scientists



Spark Workspace Settings – Starter Pool

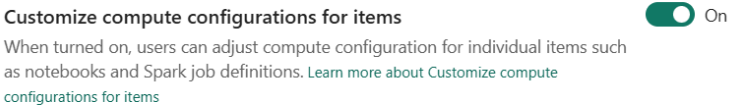
Things to consider - Billing

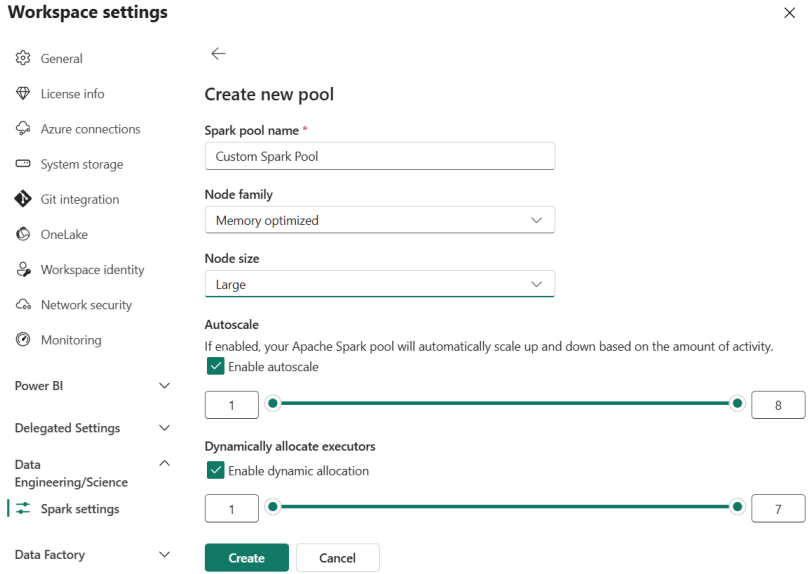


Spark Workspace Settings – Custom Pool

Things to consider

- Longer time to spin up (~ 3 minutes)
- All notebooks in the workspace will use that pool!



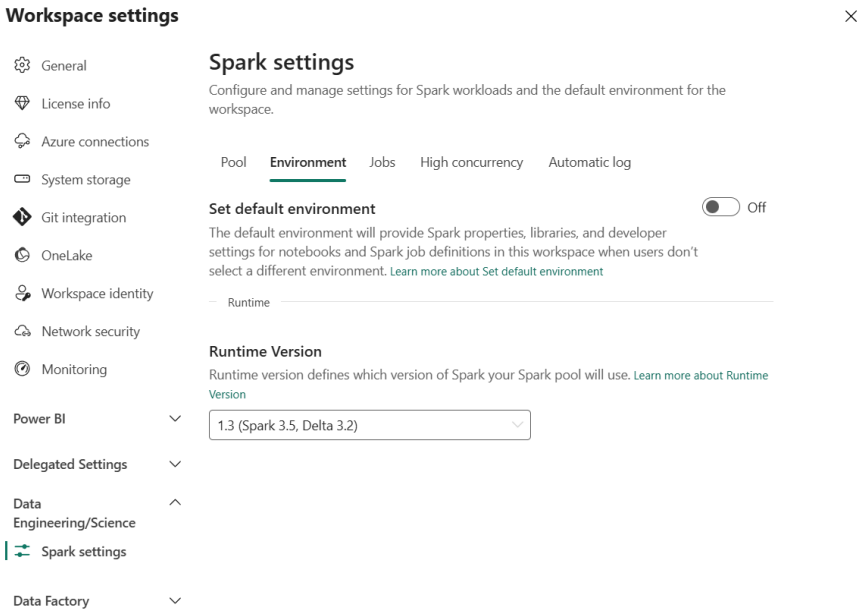


- 2 Spark VCores per Fabric CU: F2 = 4 VCores, F64 = 128 VCores…
- …+ 3x Burst Multiplier (F64 = 384 VCores)

Spark Workspace Settings – Environment

What is an environment?

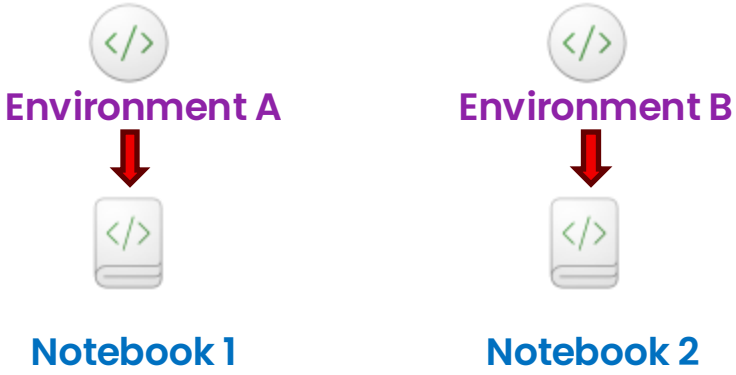
- A special Fabric item
- Contains a collection of configuration settings for executing Spark tasks (notebooks, jobs)
- Compute properties, runtime, library packages…
- Can be part of the Git integration

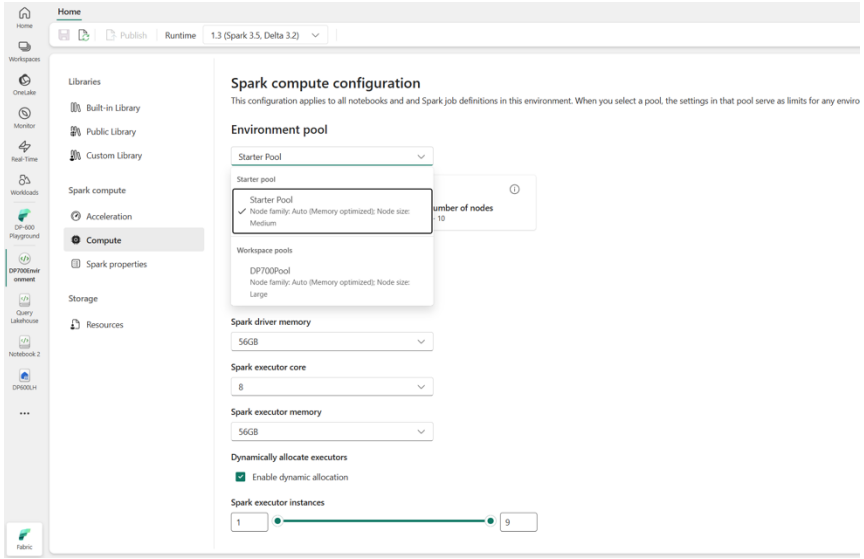


Spark Workspace Settings – Environment

Spark compute

- Pick from the existing pools
- Configure settings within the specific environment
- Granular control

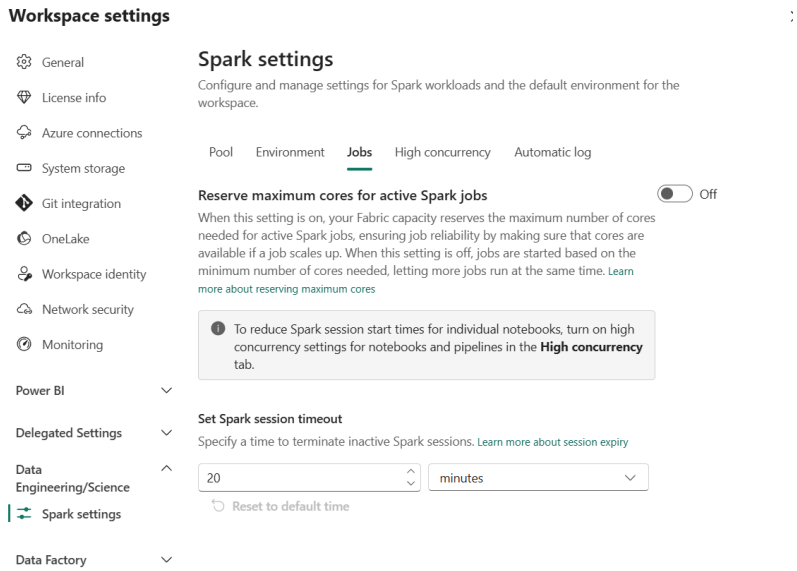




Spark Workspace Settings – Jobs

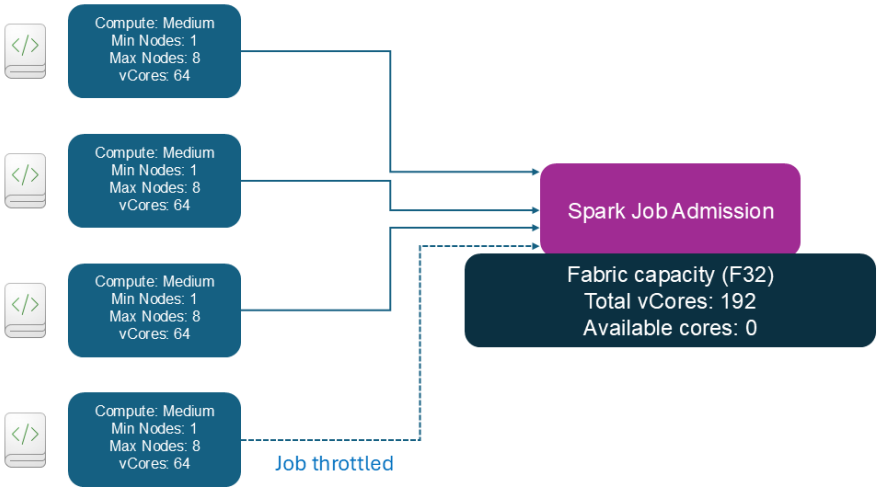
Optimistic job admission

- Relies on the minimum node setting of the Spark pool
- Evaluates job submissions based on available cores and runs them with minimum cores
- Scale up requests allowed if total Spark cores are within the capacity limits



Spark Workspace Settings – Jobs

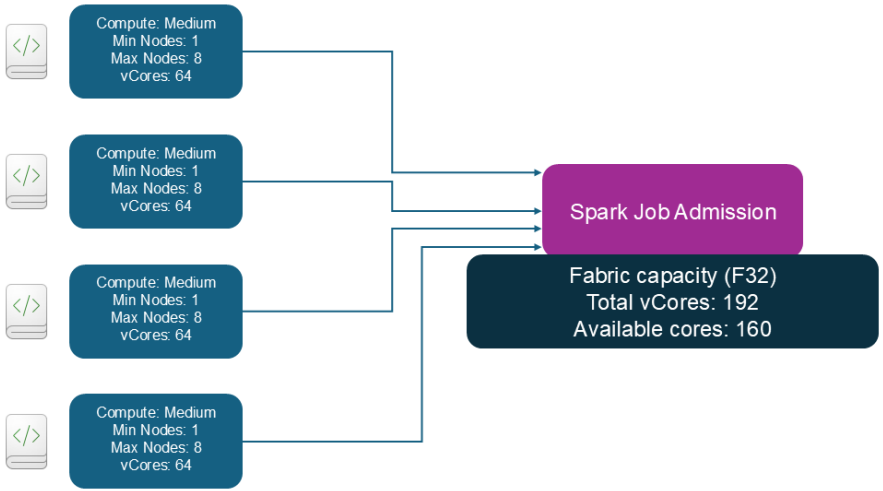
Job concurrency – NO optimistic job admission



- Total cores required for a job are reserved during job admission
- Jobs exceeding the available cores are throttled

Spark Workspace Settings – Jobs

Job concurrency – Optimistic job admission



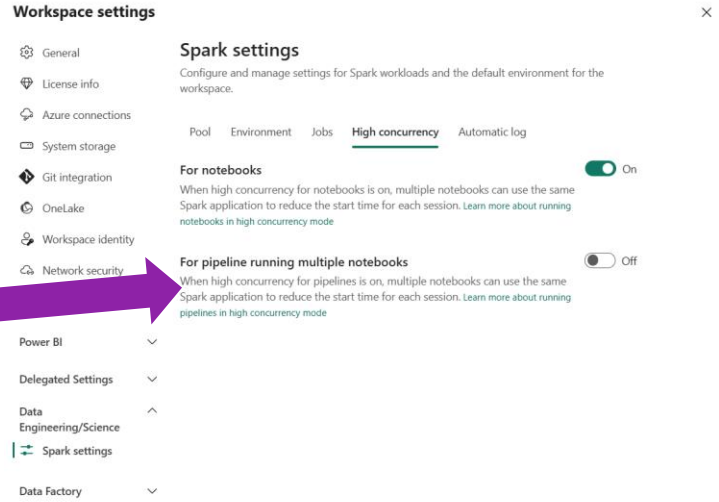
- Jobs admitted with 1 node, scale up requests approved/rejected based on available vCores
- With Autoscale, F32 can support 24 concurrent jobs (8 vCores/job)
- New jobs queued/throttled when exceeding the available vCores

Spark Workspace Settings – High concurrency

- Allows multiple users to share the same Spark session

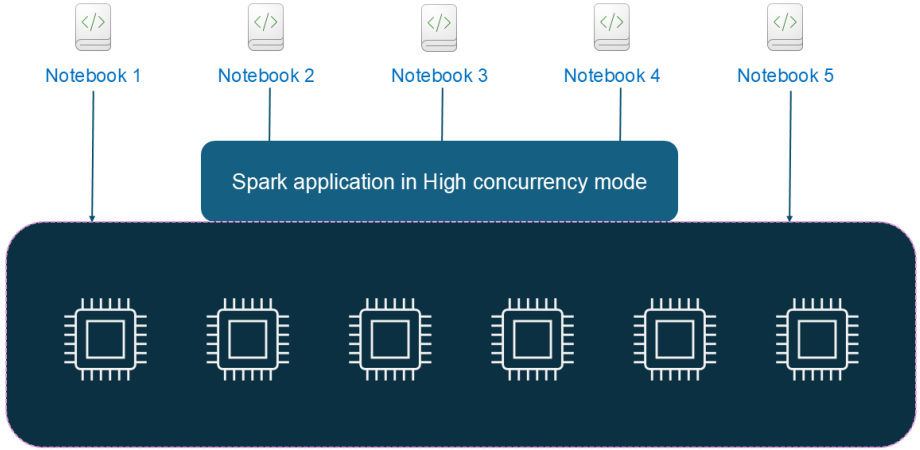
Enable a pipeline to run multiple notebooks in parallel within the same session

Configure the session tag in the Pipeline!



Spark Workspace Settings – High concurrency

- Session sharing within a single user boundary
- Users may switch between the notebooks without delays
- Cost-effective

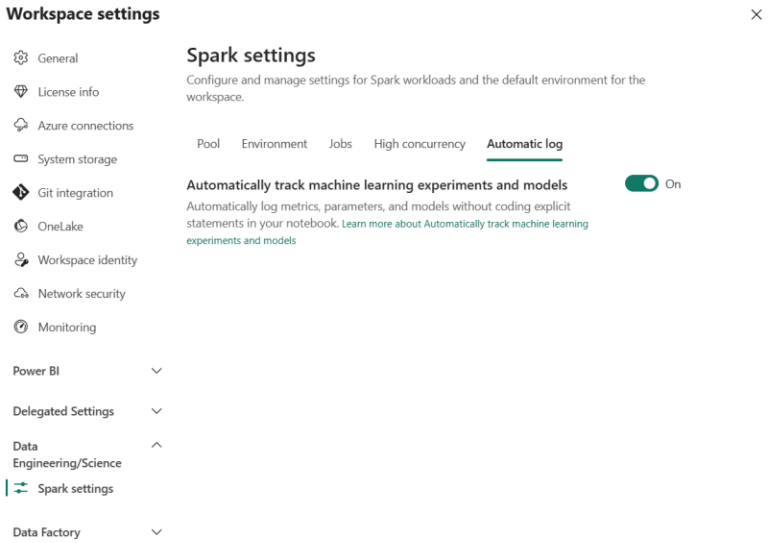


Session sharing prerequisites

- Within a single user boundary
- The same default lakehouse configuration
- The same Spark compute properties

Spark Workspace Settings – Automatic log

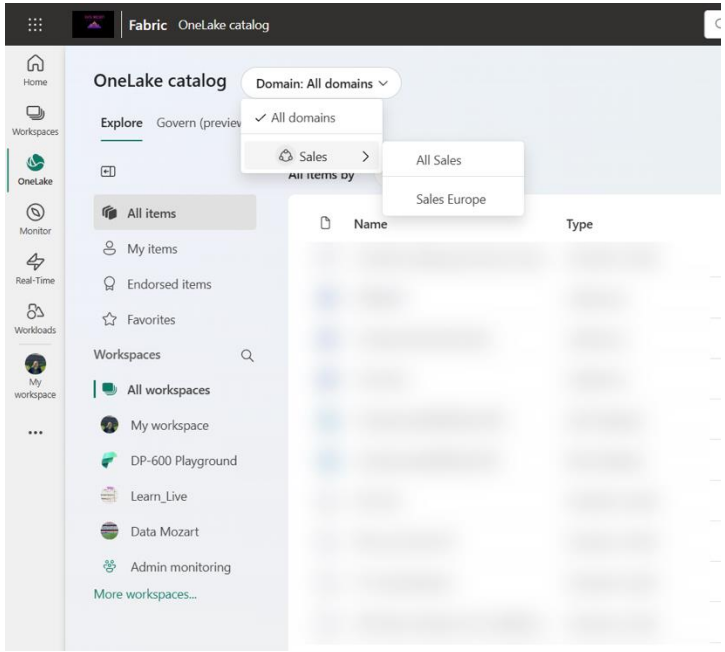
- Automatically capture the values of input parameters, output metrics, and output items of a ML model



Domain Workspace Settings

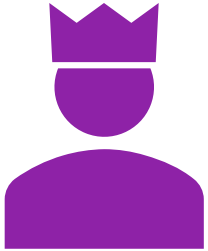
What is a Domain?

- Logically group organizational data
  - Business department (sales, finance, marketing…)
  - Enables data mesh implementation
- Workspaces are assigned to domains
- Filter content by domain in OneLake Catalog
- Data governance on the domain level



Domain Workspace Settings

Domain roles



Fabric Admin

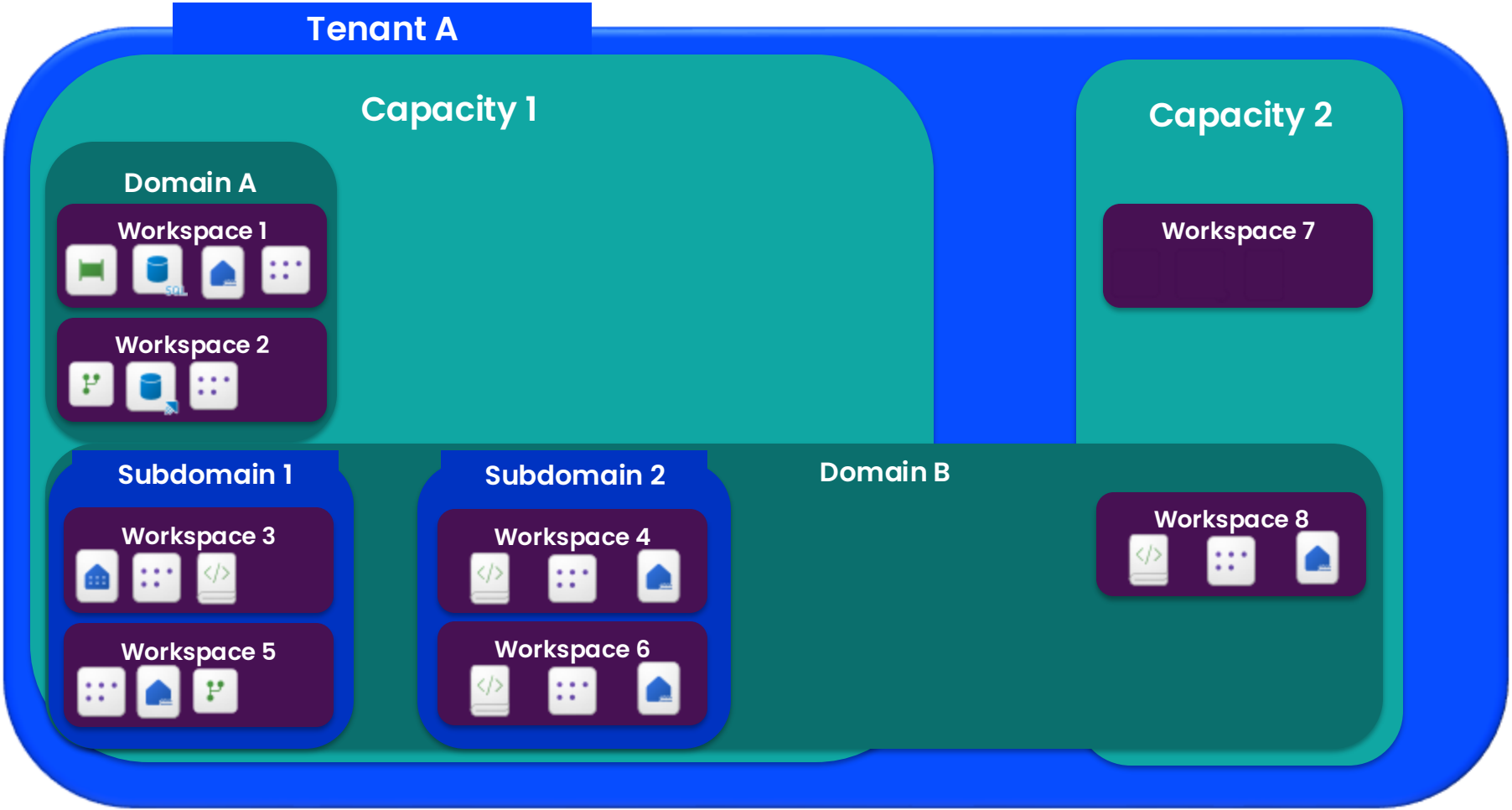
- Create/Edit domains
- Specify domain admins/contributors
- Associate workspaces with domains

Domain Admin

- Ideally business owners/experts
- Edit domains they are admins of
- Override tenant settings delegated to the domain level

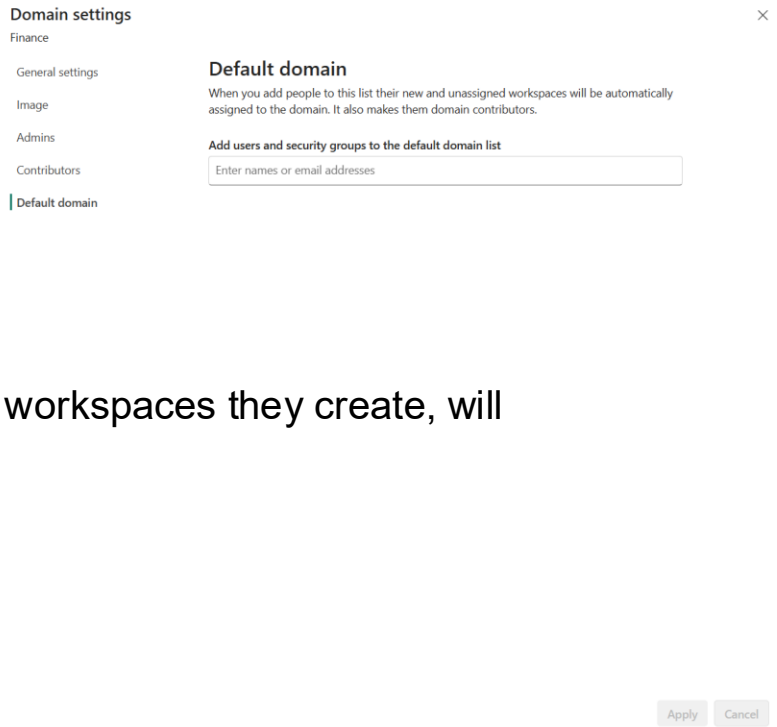
Domain Contributor

- Workspace admins authorized to assign “their” workspaces to a domain
- No access to the Domains tab in the Admin portal


Tenant A
Capacity 1
Capacity 2
Domain A
Workspace 1
Workspace 2
Workspace 7
Subdomain 1
Workspace 3
Workspace 5
Subdomain 2
Workspace 4
Workspace 6
Domain B
Workspace 8

Domain Workspace Settings

Default domain



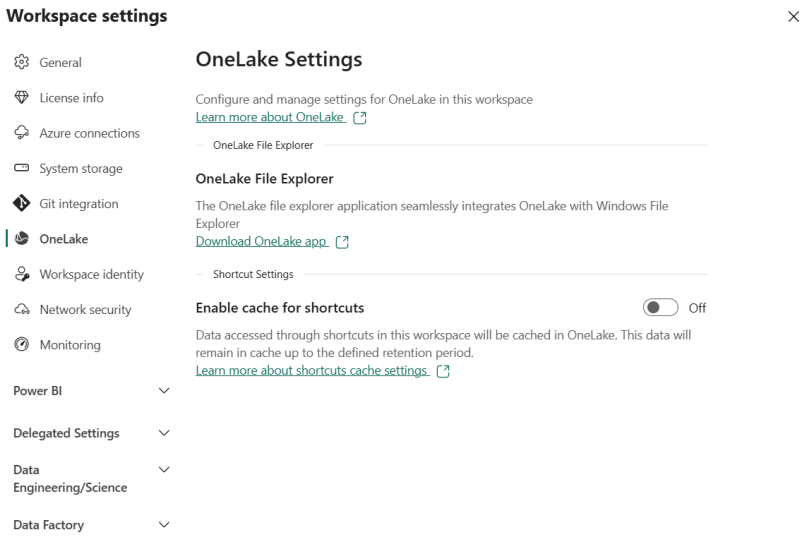
- Unassigned workspaces where the user is admin/new workspaces they create, will be automatically assigned to that specific domain

OneLake Workspace Settings

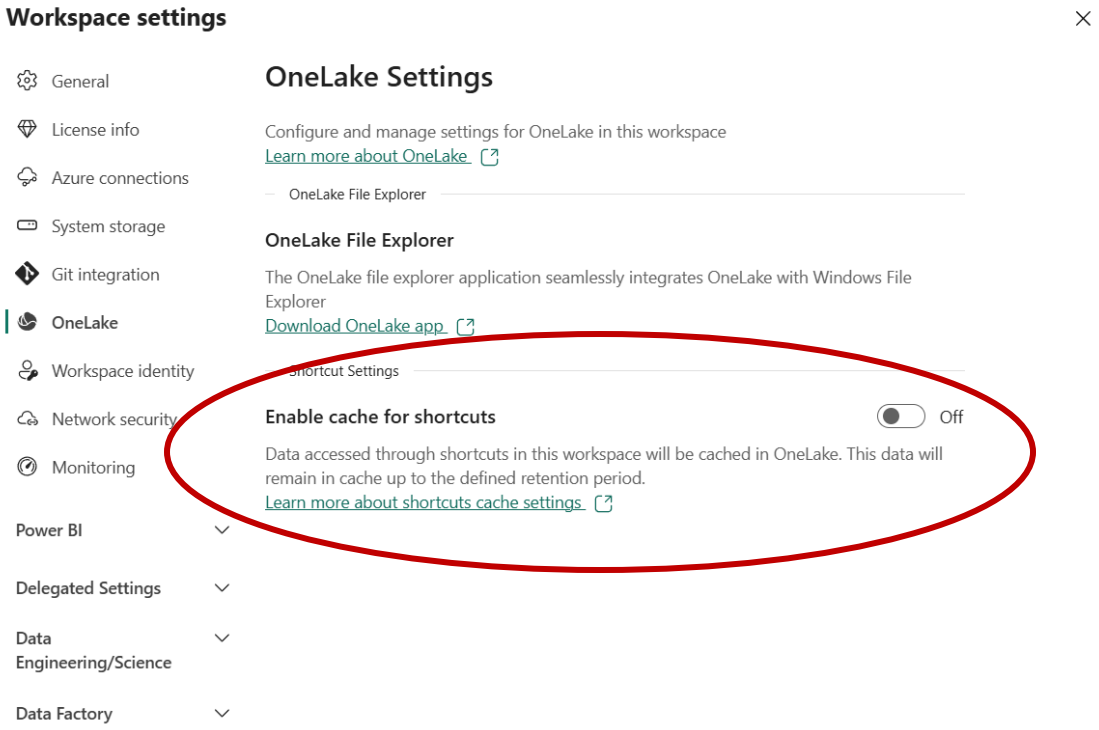
OneLake File Explorer

- Access OneLake data from Windows File Explorer
- Sync data between OneLake and Windows
- Similar to OneDrive
- Must be enabled in the Admin portal
- KQL databases/Semantic models are NOT available by default!*

**Must enable OneLake Integration feature*



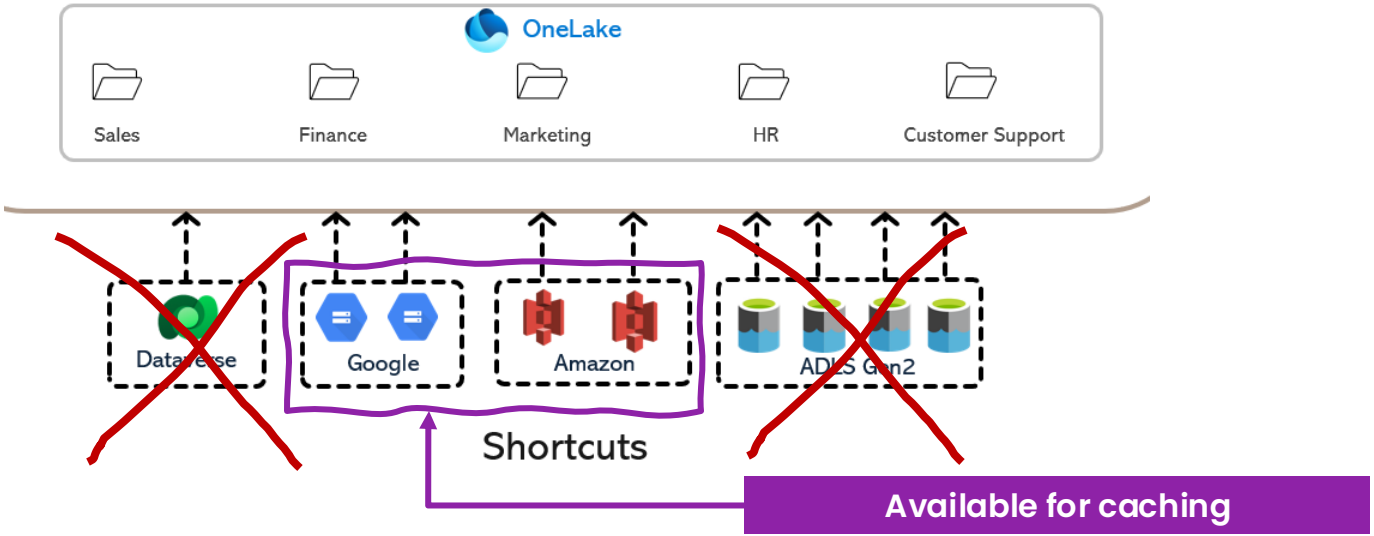
OneLake Workspace Settings



OneLake Workspace Settings

Shortcuts caching

- Reduce egress costs (applies to external shortcuts only)



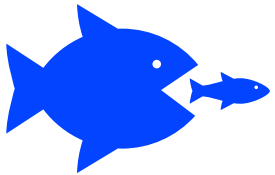
OneLake Workspace Settings

Shortcuts caching | **Limitations**

24h

If the file hasn’t been accessed for

24h+ -> kicked out from the cache



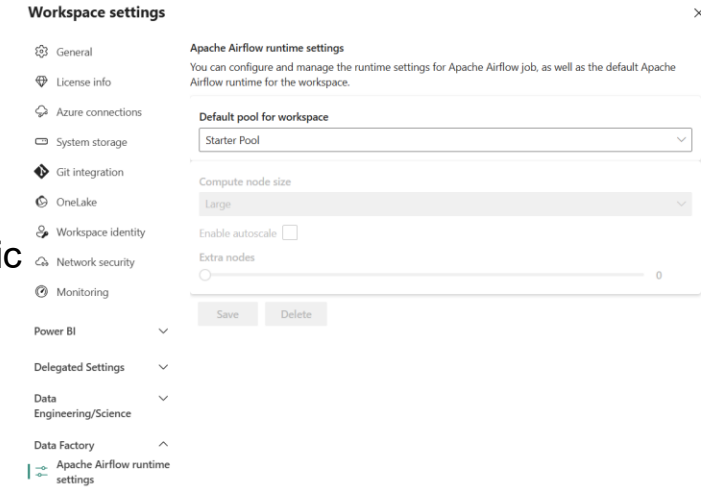
1 GB

No caching for individual files > 1GB

Data Workflow Workspace Settings

AKA Apache Airflow runtime settings ☺

- Orchestration tool
- Configure and run DAGs (directed acyclic graphs) inside Fabric
- Similar to Spark job settings: Starter and Custom pools
- Starter pool -> Instant Apache Airflow runtime
- Custom pool -> Always on until manually paused



DEMO

- Configure Spark workspace settings

Q&A

Break

Implement Lifecycle Management in Microsoft Fabric

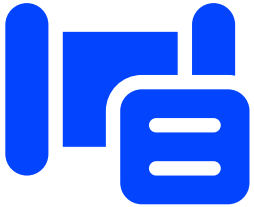
Implement lifecycle management in Fabric

- Configure version control
- Implement database projects
- Create and configure Deployment pipelines

3 Main Components of Lifecycle Management

Git integration

- Continuous Integration (CI)



Deployment pipelines

- Continuous Deployment (CD)

Fabric APIs

- Automation and programmatic management of CI/CD processes

Enable Git Integration in the Admin Portal

Must be enabled

If Azure DevOps region != Fabric capacity

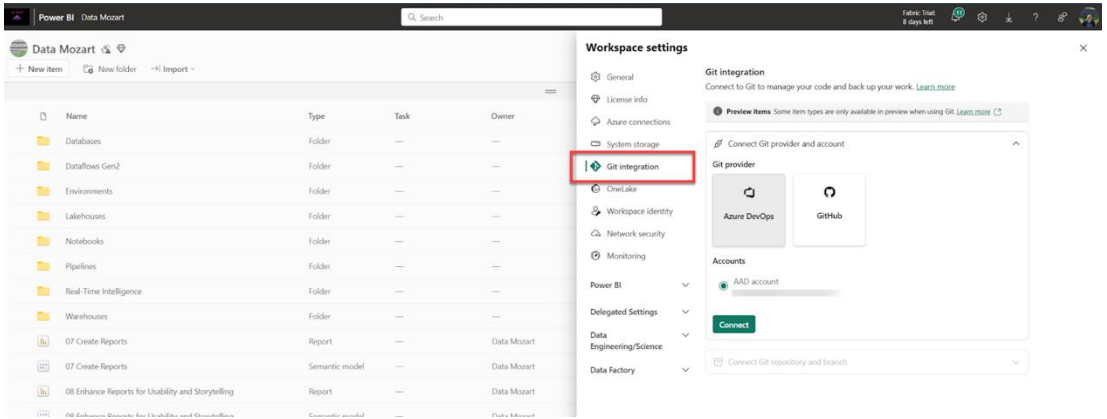
Must be enabled if you use GitHub

Git integration

- Users can synchronize workspace items with their **Git** repositories
  Enabled for the entire organization
- Users can export items to **Git** repositories in other geographical locations
  Disabled for the entire organization
- Users can export workspace items with applied sensitivity labels to **Git** repositories
  Enabled for the entire organization
- Users can sync workspace items with **Git**Hub repositories
  Enabled for the entire organization

Configure Version Control

- Git in Azure Repos
- GitHub
- GitHub Enterprise



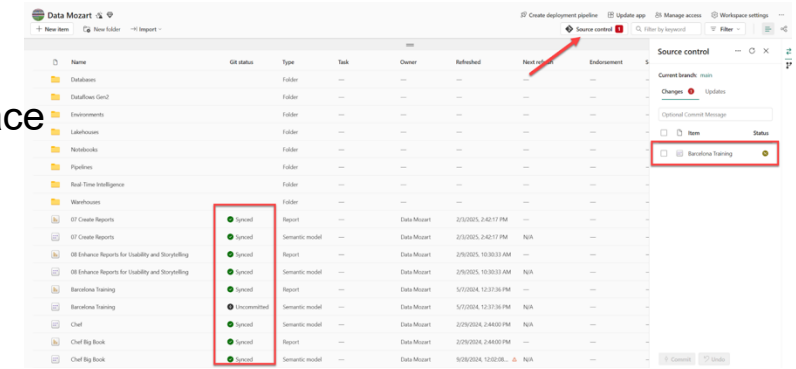
✓ **Set up Azure DevOps or GitHub account**

✓ **Connect to the Fabric workspace**

Configure Version Control

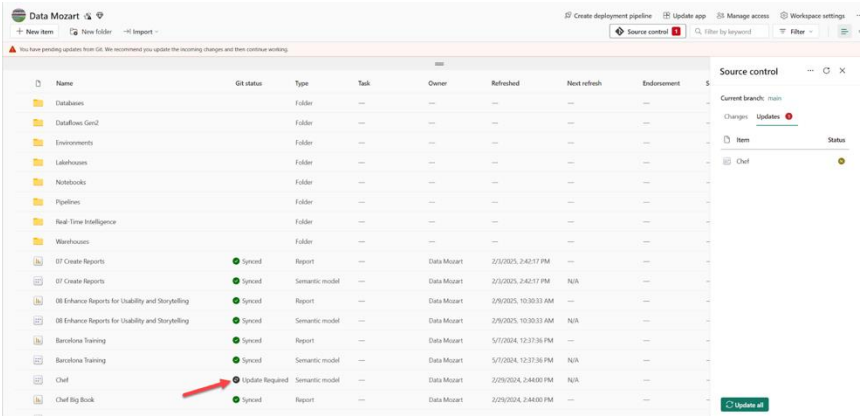
Item status

- Synced
- Conflict – the item was changed in both the workspace AND Git repository
- Unsupported item
- Uncommitted changes in the workspace
- Update required from Git
- Item identical in both places but needs to be updated to the last commit



Configure Version Control

Change vs. Update



- Changes in the Fabric workspace -> sync by using *Changes* in the Source control window. You **CAN** pick individual items to commit
- New commits in the Git repo -> sync by using the *Updates* in the Source control window. Always updates the entire branch (you **CAN NOT** pick individual items)

Configure Version Control

Recommended practices

- Use an isolated environment (separate workspace)
- Leverage client tools for development (PBI Desktop, VS Code)
- Development is done in a separate branch (not in the Main branch)
- If you develop in Fabric Web UI -> ***Branch out to new workspace***

Configure Version Control

Common actions and workspace roles

Action	Admin	Member	Contributor
Connect/Disconnect workspace from Git repo	✅		
Sync workspace with Git repo	✅		
Switch branch in the workspace	✅		
View Git connection details	✅	✅	✅
See workspace Git status information	✅	✅	✅
Update from Git	✅	✅	✅ • Write permission on items • Item owner • Build permission on external dependencies (if any)
Commit workspace changes to Git	✅	✅	✅ • Write permission on items • Item owner • Build permission on external dependencies (if any)
Create a new Git branch from Fabric	✅		
Branch out to another workspace	✅	✅	✅

Deployment Pipelines

Automated movement of Fabric items through different development lifecycle stages

Deployment Pipelines

Development

- Design, review, and play around
- Start small with minimal data amounts

Test

- Content meets criteria
- Test on more realistic data volumes
- Test items you plan to share (Power BI App)

Production

- Highest possible level of quality and accuracy

Deployment Pipelines

Deployment options

- Full deployment
- Selective deployment
- Backward deployment

Deployment Pipelines

Item pairing

- Deployment – when an item is copied from Environment A to Environment B
- Assigning a workspace to the deployment stage – the pipeline will try to pair the items by the item name and item type (the folder location also matters!)

Deployment Pipelines

Item pairing

Development	Test	Item pairing
Item name: Report1 Item type: Power BI report	Item name: Report1 Item type: Power BI report	✅ Items are paired
Item name: Report1 Item type: Power BI report	Item name: Report1 Item type: Power BI report Item name: Report1 Item type: Power BI report	⛔ No pairing (duplicates identified); Deployment fails
Item name: Report1 Item type: Power BI report Folder location: Folder1	Item name: Report1 Item type: Power BI report Folder location: Folder2 Item name: Report1 Item type: Power BI report Folder location: Folder1 Item name: Report1 Item type: Power BI report	⛔ No pairing, but deployment succeeds ✅ Items are paired ⛔ No pairing, but deployment succeeds

Deployment Pipelines

Deployment rules

- Different stages may have different configuration settings
  - Different database or query parameter
  - Dev for querying sample data, Test/Prod for querying the entire database
- Semantic model in prod to point to prod database -> Define the rule in the Production stage for that semantic model

Deployment Pipelines

Copy item properties between the stages

✅	⛔
Data sources	Data
Parameters	Permissions for a workspace or specific item
Report visuals	Workspace settings
Report pages	Personal bookmarks
Dashboard tiles	Role assignment for semantic models
Model metadata	Refresh schedule
Item relationships	Data source credentials
Sensitivity labels*	Endorsement settings

* Either when a new item is deployed, or an existing item is deployed to an empty stage; OR when the source item contains a sensitivity label and the target item doesn't

Deployment Pipelines

Permission model

Role	Permission
Pipeline Admin	• View the pipeline • Share the pipeline with others • Edit and delete the pipeline • Unassign a workspace from the stage • **NO permissions on the workspace content**
Workspace Viewer (and Pipeline Admin)	• Consume content • Unassign a workspace from the stage
Workspace Contributor (and Pipeline Admin)	**The same as Viewer +** • Compare stages • View semantic models • Deploy items (must be at least Contributor in both source and target workspaces)
Workspace Member (and Pipeline Admin)	**The same as Contributor +** • View workspace content • Update semantic models • Configure semantic model deployment rules
Workspace Admin (and Pipeline Admin)	**The same as Member+** • Assign workspaces to a stage

Lifecycle Management in Fabric

Recommended practices

1 Content preparation

- Separate development between teams
- Planning a permission model
- Connect different stages to different databases
- Use parameters to change configuration settings between stages

2 Development

- Back up the work in Git repository
- Rolling back changes
- Use an isolated environment (workspace)

3 Test

- Simulate the production environment
- Configure deployment rules
- Check related items to avoid breaking changes
- Update data items
- Test the Power BI App if needed

4 Production

- Define who can deploy to production
- Set deployment rules
- Update the production Power BI App if needed
- Deploy using Git integration

Implement Database Projects

Deploy changes to Fabric Warehouse/SQL database in Fabric

Implement Database Projects

What?

- Structured representation of DWH schema and objects (tables, views, sprocs…)
- Versioning, collaboration, continuous deployment
- Similar to SSDT projects
- Created and managed from Fabric Web UI or VS Code

Implement Database Projects

Why?

1 Version control and collaboration

- Git integration to track all changes to database objects
- Teams collaborate on schema design
- Enables pull requests, code reviews, change history

2 Automation and CI/CD

- Can be automatically built and deployed with deployment pipelines

3 Change management

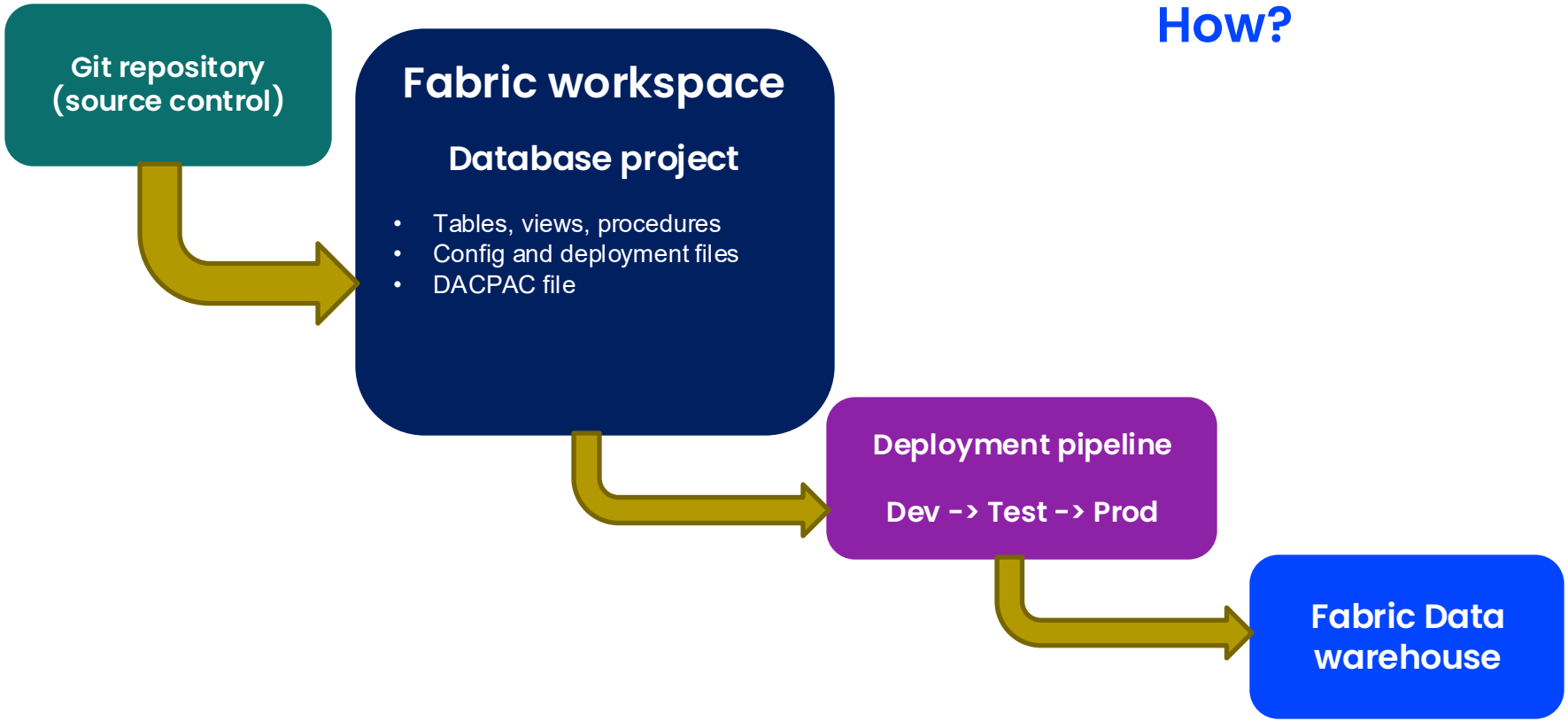
- No live changes to the database -> You define the desired state in the project and deploy changes incrementally
- Compare current vs. desired state and safely apply updates

4 Compliance and governance

- Audit-ready versioning of the database schema
- Tracking changes

Implement Database Projects

How?



DEMO

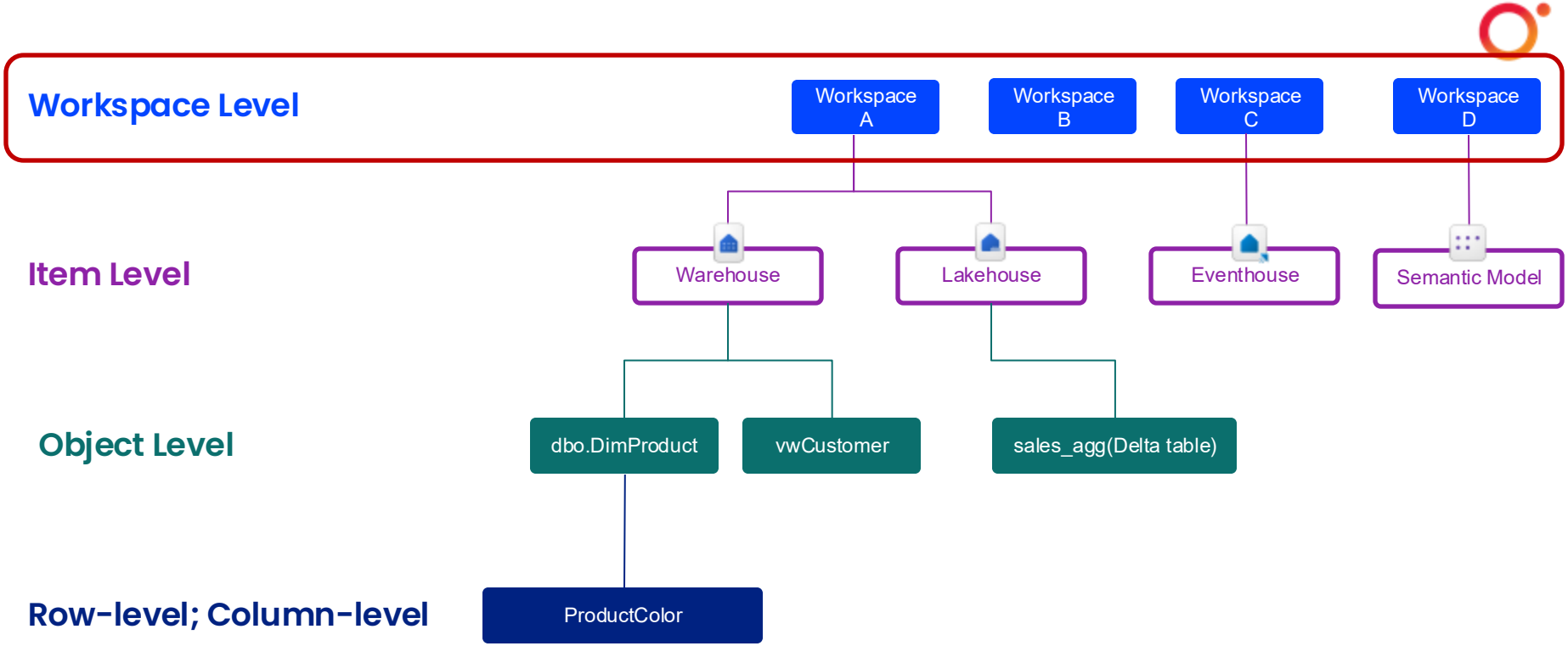
- Configure source control
- Create and manage deployment pipeline

Q&A

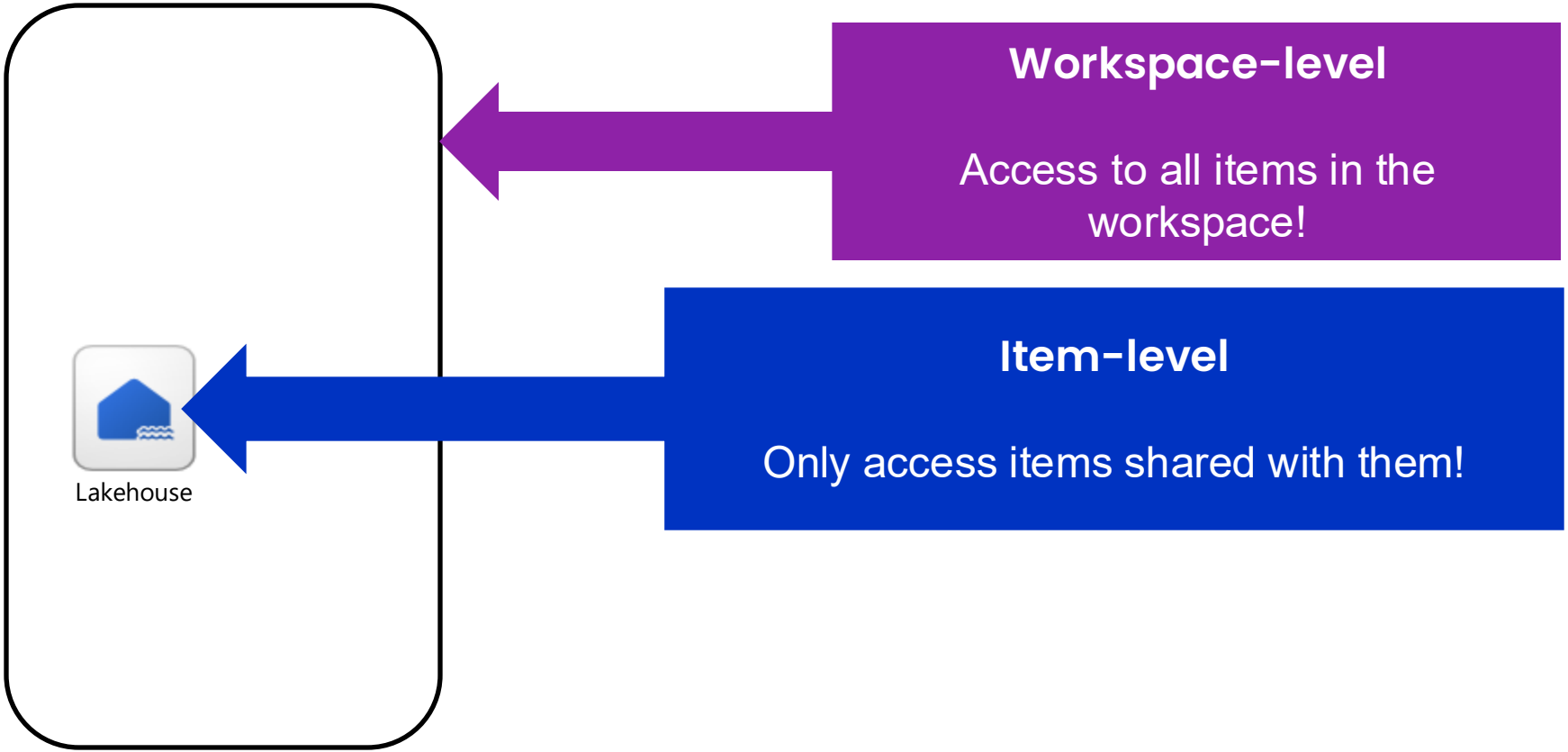
Configure Security and Governance

Configure security and governance

- Implement workspace-level access controls
- Implement item-level access controls
- Implement row-level, column-level, object-level, and folder/file-level access controls
- Implement dynamic data masking
- Apply sensitivity labels to items
- Endorse items
- Implement and use workspace logging


Workspace Level
Workspace A
Workspace B
Workspace C
Workspace D
Item Level
Warehouse
Lakehouse
Eventhouse
Semantic Model
Object Level
dbo.DimProduct
vwCustomer
sales_agg(Delta table)
Row-level; Column-level
ProductColor

Access Control in Fabric



Workspace Access Control

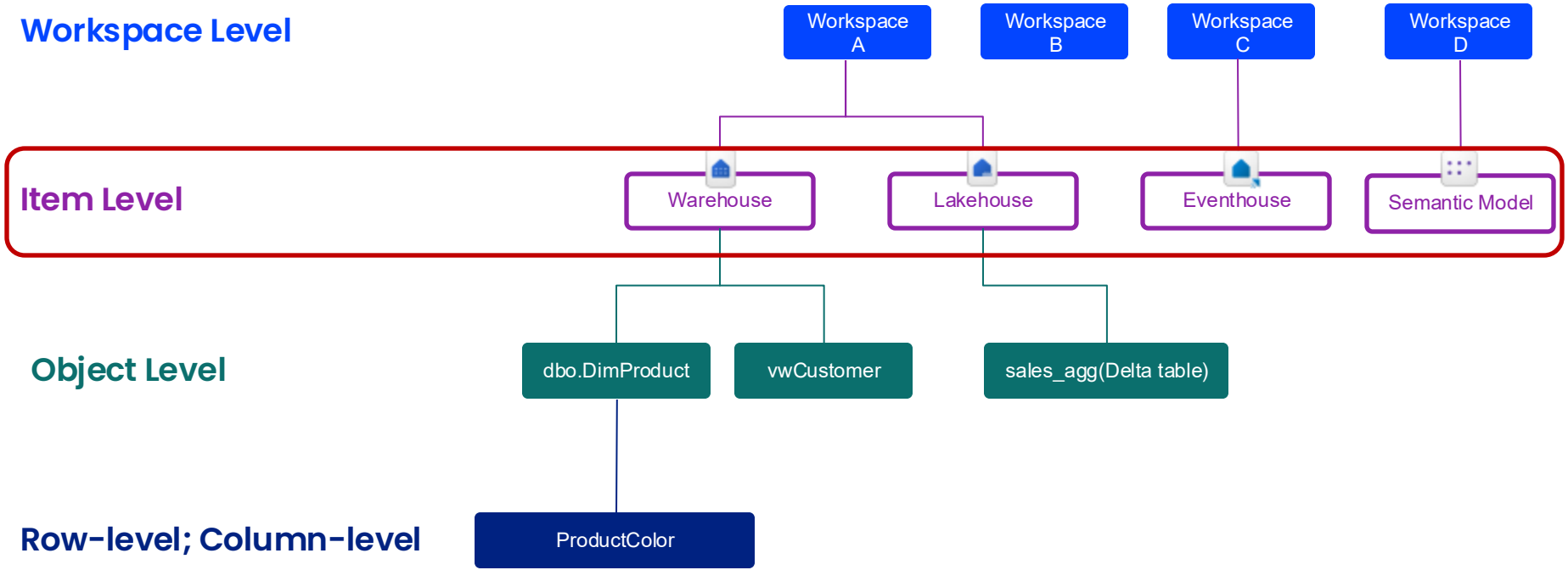
- 4 roles
  - Admin
  - Member
  - Contributor
  - Viewer
- Applies to ALL the items in the workspace!
- Individual user vs. Entra ID/M365 group

Capability	Admin	Member	Contributor	Viewer
Update and delete the workspace.	✓			
Add or remove people, including other admins.	✓			
Add members or others with lower permissions.	✓	✓		
Allow others to reshare items.[1]	✓	✓		
View and read content of data pipelines, notebooks, Spark job definitions, ML models and experiments, and Event streams.	✓	✓	✓	✓
View and read content of KQL databases, KQL query-sets, and real-time dashboards.	✓	✓	✓	✓
Connect to SQL analytics endpoint of Lakehouse or the Warehouse	✓	✓	✓	✓
Read Lakehouse and Data warehouse data and shortcuts[2] with T-SQL through TDS endpoint.	✓	✓	✓	✓
Read Lakehouse and Data warehouse data and shortcuts[2] through OneLake APIs and Spark.	✓	✓	✓	
Read Lakehouse data through Lakehouse explorer.	✓	✓	✓	
Write or delete data pipelines, notebooks, Spark job definitions, ML models and experiments, and Event streams.	✓	✓	✓	
Write or delete KQL query-sets, real-time dashboards, and schema and data of KQL databases, Lakehouses, data warehouses, and shortcuts.	✓	✓	✓	
Execute or cancel execution of notebooks, Spark job definitions, ML models and experiments.	✓	✓	✓	
Execute or cancel execution of data pipelines.	✓	✓	✓	
View execution output of data pipelines, notebooks, ML models and experiments.	✓	✓	✓	✓
Schedule data refreshes via the on-premises gateway.[3]	✓	✓	✓	
Modify gateway connection settings.[3]	✓	✓	✓	

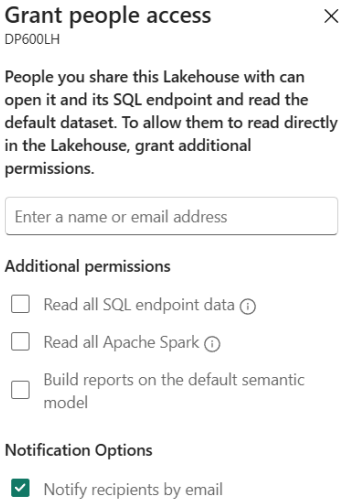
[1] Contributors and Viewers can also share items in a workspace, if they have Reshare permissions.

[2] Additional permissions are needed to read data from shortcut destination. Learn more about shortcut security model.

[3] Keep in mind that you also need permissions on the gateway. Those permissions are managed elsewhere, independent of workspace roles and permissions.


Workspace Level
Workspace A
Workspace B
Workspace C
Workspace D
Item Level
Warehouse
Lakehouse
Eventhouse
Semantic Model
Object Level
dbo.DimProduct
vwCustomer
sales_agg(Delta table)
Row-level; Column-level
ProductColor

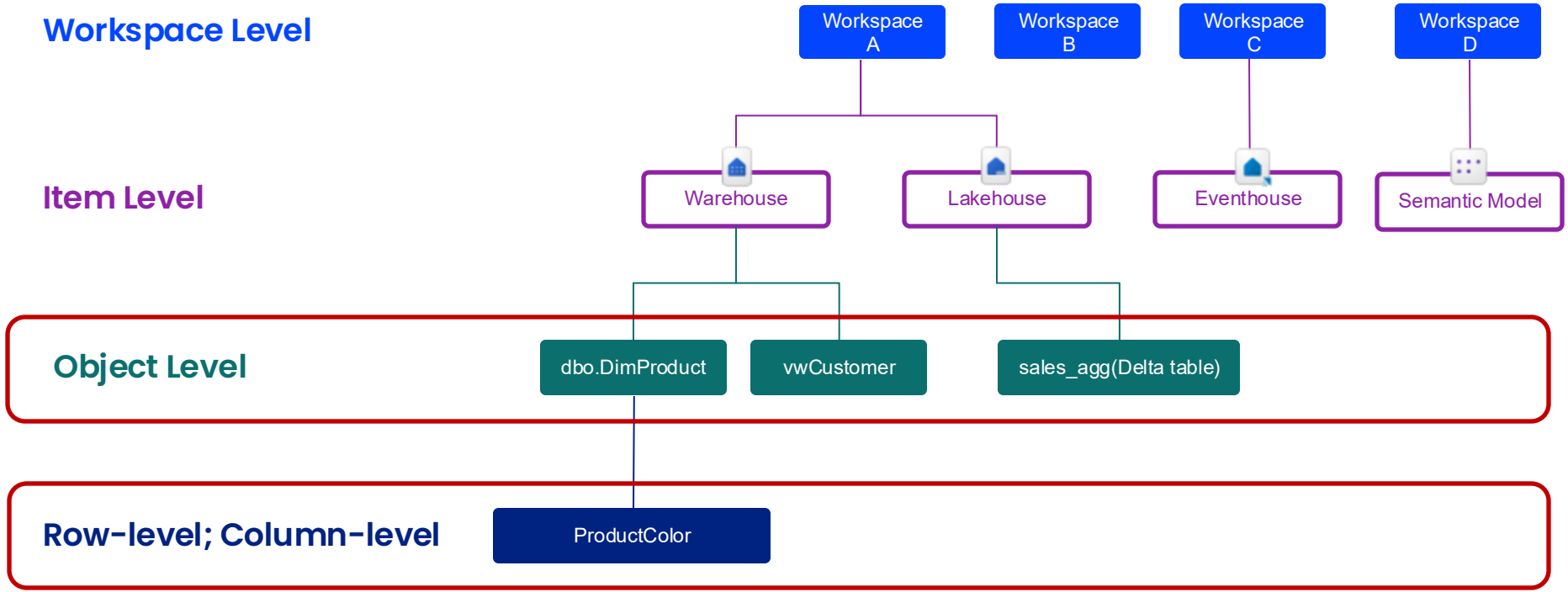
Item-level Access Control



Permission granted while sharing	Effect
Read	Recipient can discover the item in the data hub and open it. Connect to the Warehouse or SQL analytics endpoint of the Lakehouse.
Edit	Recipient can edit the item or its content.
Share	Recipient can share the item and grant permissions up to the permissions that they have. For example, if the original recipient has *Share*, *Edit*, and *Read* permissions, they can at most grant *Share*, *Edit*, and *Read* permissions to the next recipient.
Read All with SQL analytics endpoint	Read data from the SQL analytics endpoint of the Lakehouse or Warehouse data through TDS endpoints.
Read all with Apache Spark	Read Lakehouse or Data warehouse data through OneLake APIs and Spark. Read Lakehouse data through Lakehouse explorer.
Build	Build new content on the semantic model.
Execute	Execute or cancel execution of the item.

[Item permission model]

- **Nothing checked –** access LH from OneLake hub, but none of the tables (suitable for OLS/CLS)
- **Read all SQL endpoint data**
- **Read all Apache Spark**
- **Build reports on the default semantic model**


Workspace Level
Item Level
Object Level
Row-level; Column-level
Workspace A
Workspace B
Workspace C
Workspace D
Warehouse
Lakehouse
Eventhouse
Semantic Model
dbo.DimProduct
vwCustomer
sales_agg(Delta table)
ProductColor

Row-level Security

```
-- Creating schema for Security
CREATE SCHEMA Security;
GO
```

```
-- Creating a function for the SalesRep evaluation
CREATE FUNCTION Security.tvf_securitypredicate(@UserName AS varchar(50))
    RETURNS TABLE
WITH SCHEMABINDING
AS
    RETURN SELECT 1 AS tvf_securitypredicate_result
WHERE @UserName = USER_NAME();
GO
```

```
-- Using the function to create a Security Policy
CREATE SECURITY POLICY YourSecurityPolicy
ADD FILTER PREDICATE Security.tvf_securitypredicate(UserName_column)
ON sampleschema.sampletable
WITH (STATE = ON);
GO
```

User must have access to the table

- Workspace Viewer role
- Item-level shared with ReadData permission
- OLS

Object-level and Column-level Security

Object-level security

```
GRANT SELECT ON Sales.FactResellerSales TO [SalesReps];
```

Column-level security

```
GRANT SELECT ON dbo.Customer (CustomerName, Email, PhoneNumber) TO [SalesReps];
```

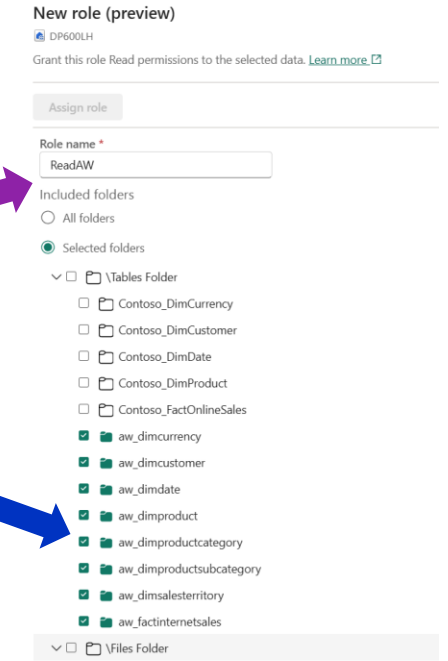
- No workspace role
- Item-level sharing with NO additional permissions

Folder-level Access Control

Read access to specific folders (OLS for Lakehouse)

- Data access role security ONLY applies to users accessing OneLake directly!
- Currently supported only for the Lakehouse item

1. Create a role
2. Choose tables/files from Lakehouse
3. Assign role to user/group



User 1 requests to read the data from Folder A in Lakehouse B

1. Check workspace permissions (Admin, Member, Contributor, Viewer, none)
2. Check Lakehouse B permissions (item-level)
3. Check Folder A permissions (folder-level)

OneLake Security

Will replace the existing OneLake data access roles feature

Security rules enforced directly on the table/file in OneLake, then all Fabric engines will respect these rules!

Security “lives” with the data and it’s not dependent on the Fabric engine/workload

OneLake Security

Data

- Tables/folders users can access

Permission

- Set of permissions users have on the data

Members

- Users that are assigned to the role

Constraints

- Specific data components (rows, columns) excluded from the role access

Dynamic Data Masking

Impacts how the data is displayed to the end user

It's NOT a security feature!

Dynamic Data Masking

- Applies to the Warehouse and SQL analytics endpoint of the Lakehouse
- The underlying data is not changed
- Complements OLS/RLS!
- Can be applied for new tables (CREATE TABLE) and existing tables (ALTER TABLE)

Dynamic Data Masking Functions

1 Default

- Based on the column data type

```
PhoneNumber varchar(12) MASKED WITH (FUNCTION = 'default()')
```

2 Email

- 1st letter of the email and the constant suffix ".com"
- aXXX@XXXX.com

```
Email varchar(100) MASKED WITH (FUNCTION = 'email()')
```

3 Random

- For numeric columns
- Random value from the specified range

```
Salary bigint MASKED WITH (FUNCTION = 'random(1, 1000000)')
```

4 Custom string (partial)

- Exposes the first and last letters, adding a custom number of masked characters in the middle

```
FirstName varchar(100) MASKED WITH (FUNCTION = 'partial(2, "XXXXXX", 0)')
```

John Doe -> JoXXXXXX

Dynamic Data Masking

Bypassing masking

```
SELECT ID, Name, Salary FROM Employees
WHERE Salary > 99999 and Salary < 100001;
```

ID	Name	Salary
1	John Doe	100000
2	Jane Doe	100000

```
SELECT c.name
       , tbl.name AS table_name
       , c.is_masked
       , c.masking_function
FROM sys.masked_columns AS c
JOIN sys.tables AS tbl
ON c.[object_id] = tbl.[object_id]
WHERE is_masked = 1;
```

Check masked columns

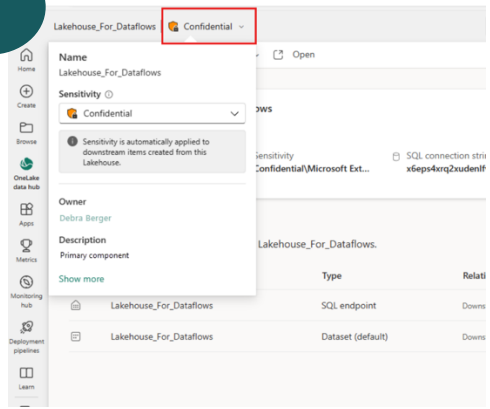
Sensitivity Labels

First created in Purview

Microsoft Purview Information Protection

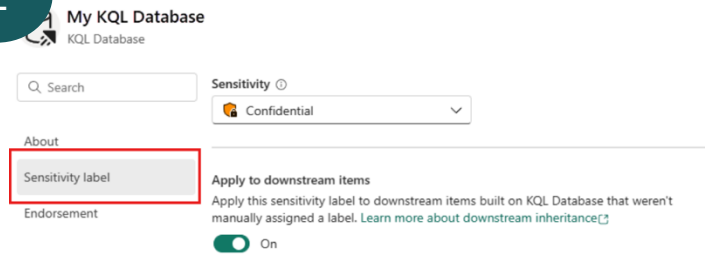
- Meet governance and compliance requirements
- Only authorized people can access the data
- 2 ways to apply

1



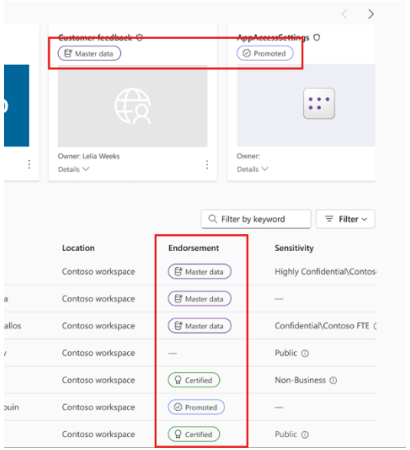
From the flyout menu

2



In item settings

Endorsement



- Makes it easier for users to find high-quality, trustworthy data
- Labeled with a badge in the UI
- 3 endorsement badges

1

Promoted

- ✓ Item ready for sharing and reuse
- ✓ All items except dashboards
- ✓ Any user with write permission on item can promote it

2

Certified

- ✓ Authorized reviewer
- ✓ Item meets org's quality standards
- ✓ All items except dashboards
- ✓ Any user can ***request***, only users specified by Fabric Admin can certify items

3

Master data

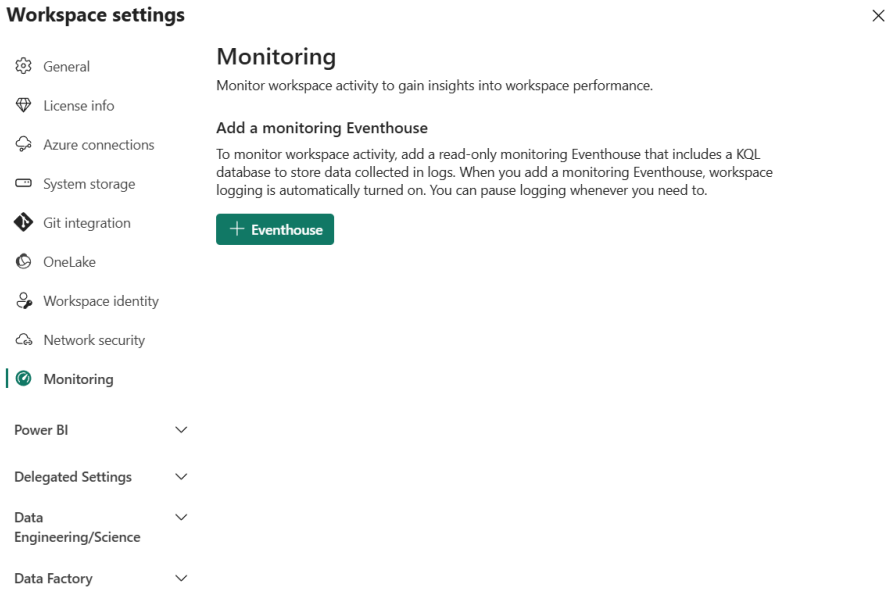
- ✓ Core source of org data
- ✓ Single source of truth
- ✓ Items that contain data (lakehouses, semantic models)
- ✓ Only users specified by Fabric Admin can label Master data

Workspace Logging

A special database (Eventhouse) that collects and organizes logs and metrics from Fabric items in the workspace

Workspace Logging

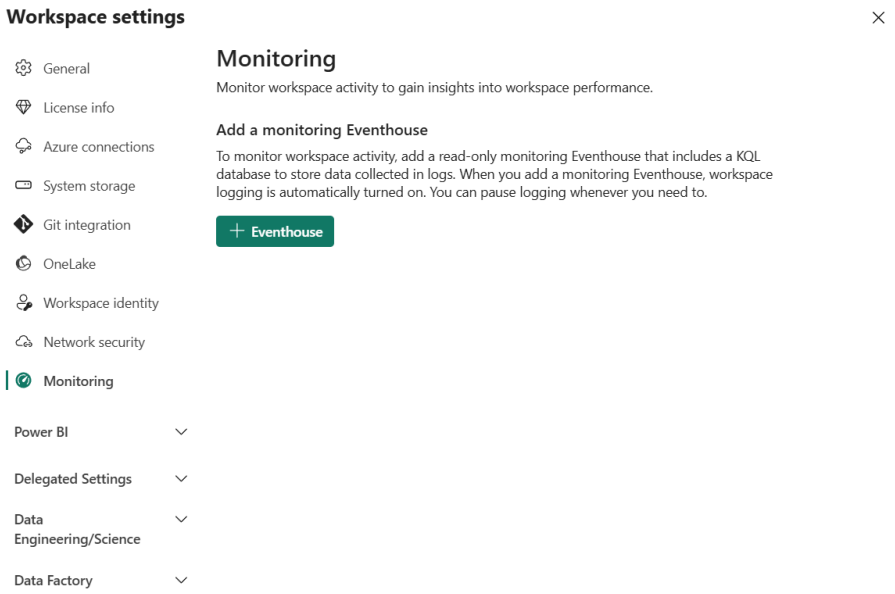
- Tenant settings in the Admin portal
- Workspace Admin -> Workspace settings
- + Eventhouse



Workspace Logging

What's logged?

- Data engineering (GraphQL operations)
- Eventhouse in RTI
  - Command logs
  - Data operation logs
  - Ingestion results logs
  - Metrics
  - Query logs
- Mirrored database
- Power BI semantic models



Use templates and sample queries

DEMO

- Configure and manage workspace access
- Configure and manage item-level permissions
- Implement OneLake Data Access (folder-level permissions)

Q&A

Break

Orchestrate Processes

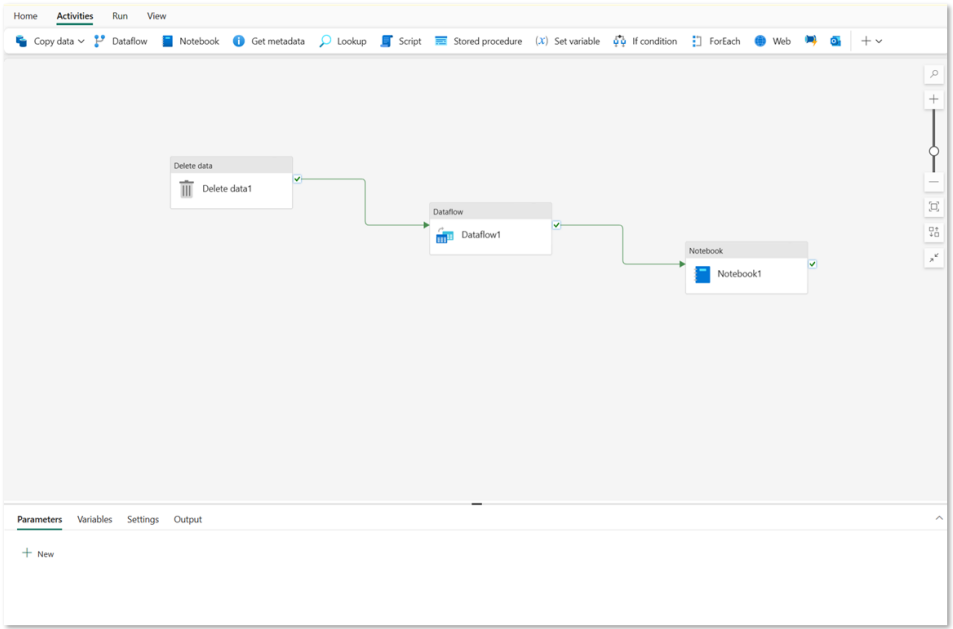
Orchestrate processes

- Choose between a pipeline and a notebook
- Design and implement schedules and event-based triggers
- Implement orchestration patterns with notebooks and pipelines

Choose Your Team? (POLL)

1. Low-code/No-code
2. Code-first

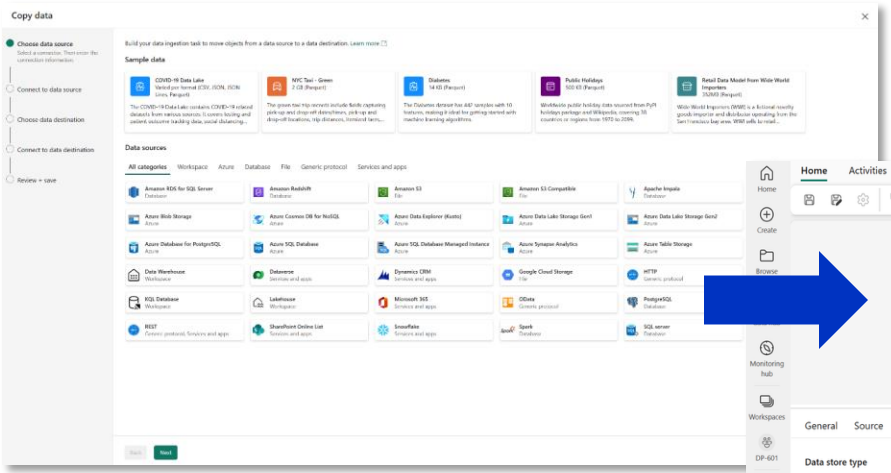
Fabric Pipelines



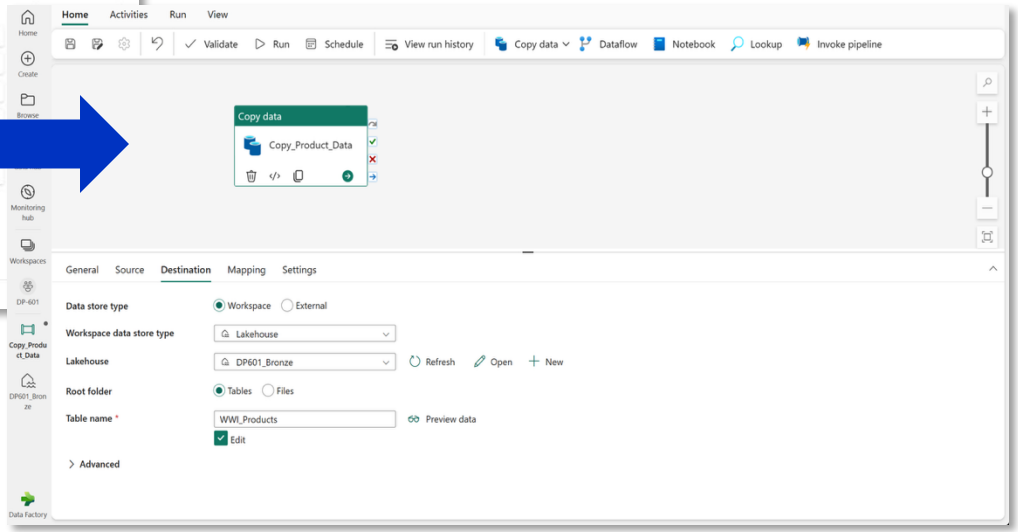
Pipeline concepts:

- **Activities**
  - Data transformation
  - Control flow
- **Parameters**
- **Schedule runs**

Common Activities – Copy Data



1. Use the copy data tool



2. Edit the settings below the pipeline canvas

Pipeline Activities

Move and transform

- Copy data
- Dataflow
- Delete data

Transform

- Spark Job Definition
- Notebook
- Script
- Stored procedure
- KQL

Orchestrate

- Invoke Pipeline (Legacy)
- Invoke Pipeline (Preview)
- Web
- WebHook
- Semantic model refresh (Preview)
- Azure Databricks

Notifications

- Office 365 Outlook (Preview)
- Teams (Preview)

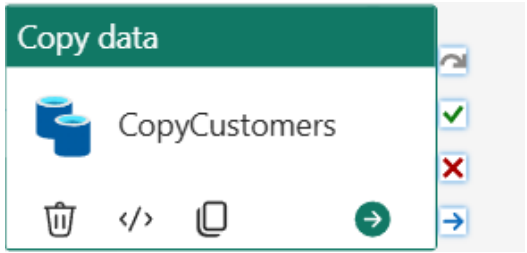
Control flow

- If conditions
- Switch
- Filter
- Wait
- ForEach
- Until
- Set variable
- Append variable
- Fail

Metadata and validation

- Lookup
- Get metadata

Pipeline Workflow



On Skip

The next activity runs only if the previous activity is skipped

On Success

The next activity runs only if the previous activity is **completed successfully**

On Fail

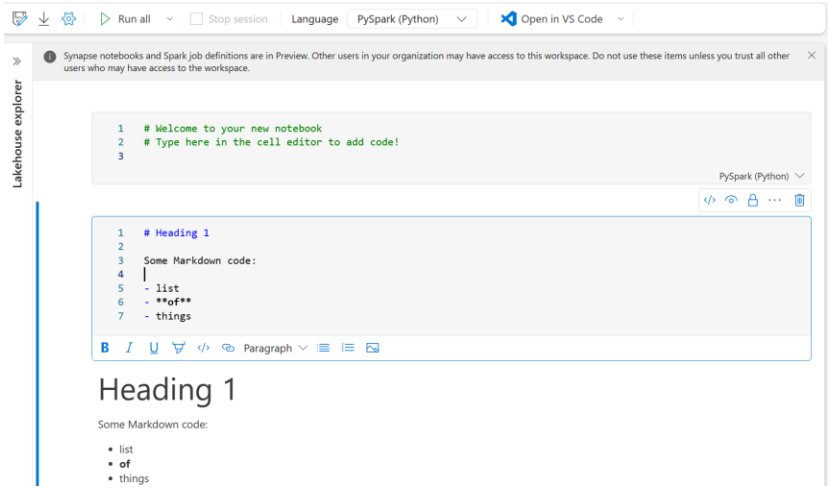
The next activity runs only if the previous activity failed

On Completion

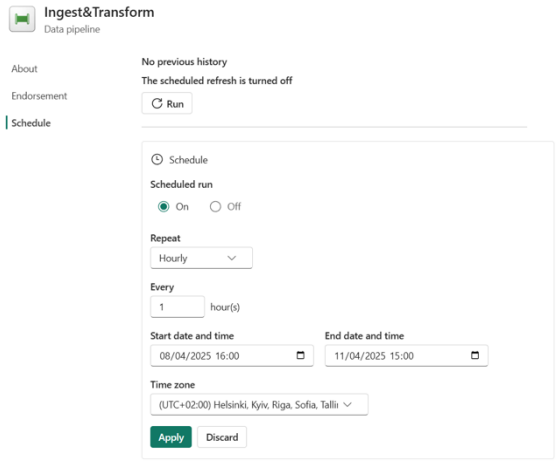
The next activity runs when the previous activity is completed (**regardless if it failed or succeeded**)

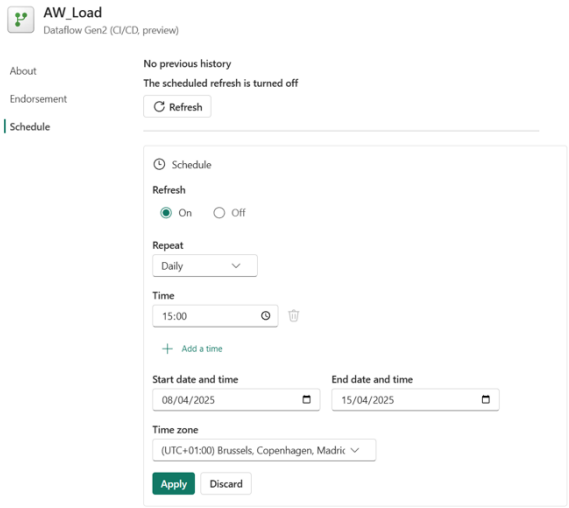
Fabric Notebooks

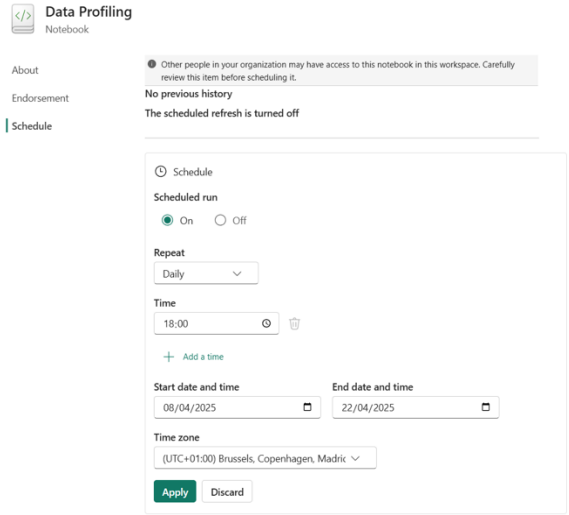
- Code (PySpark, Scala, R, Spark SQL, Python)
- Markdown (comments)
- Run or freeze individual or multiple cells
- Ingest and transform
- Support automation



Scheduling

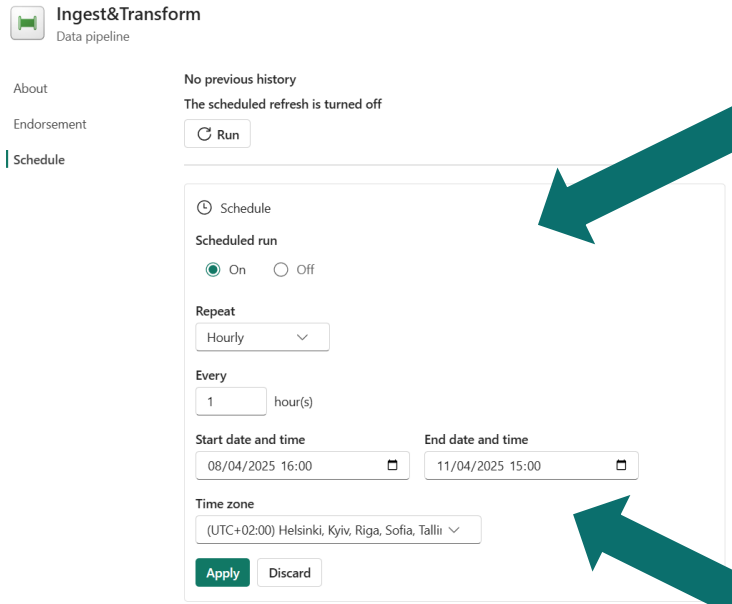




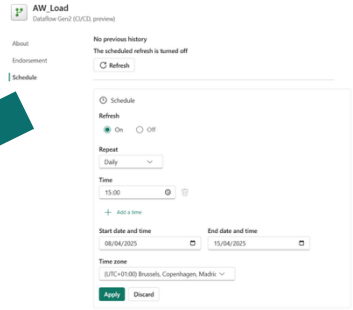


Pipeline **Dataflow Gen2** **Notebook**

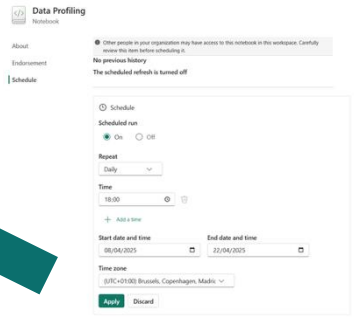
Scheduling



Pipeline



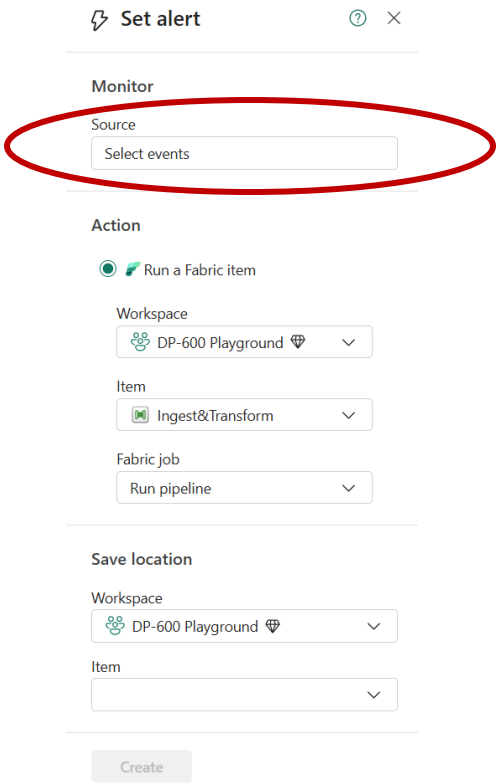
Dataflow Gen2



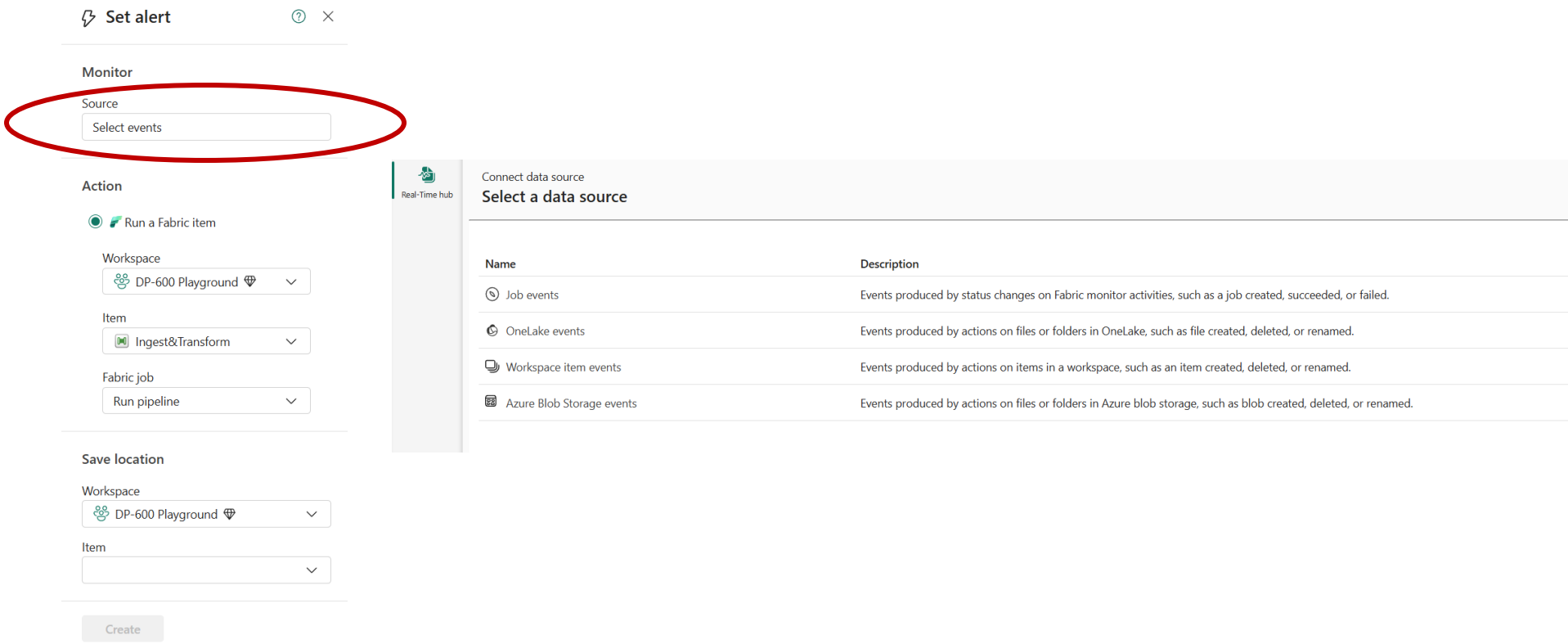
Notebook

Use Pipelines for scheduling tasks

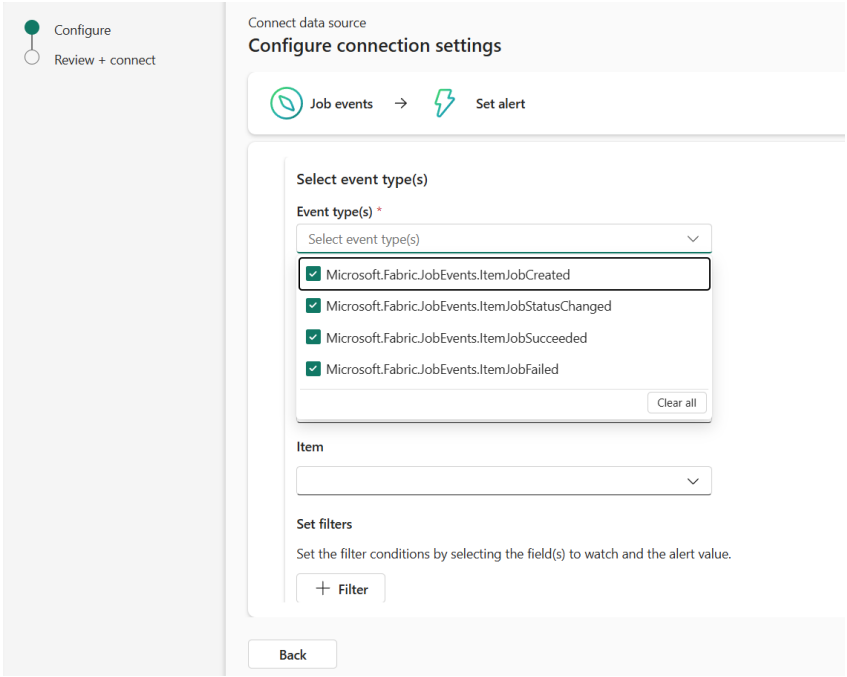
Event-based Triggers

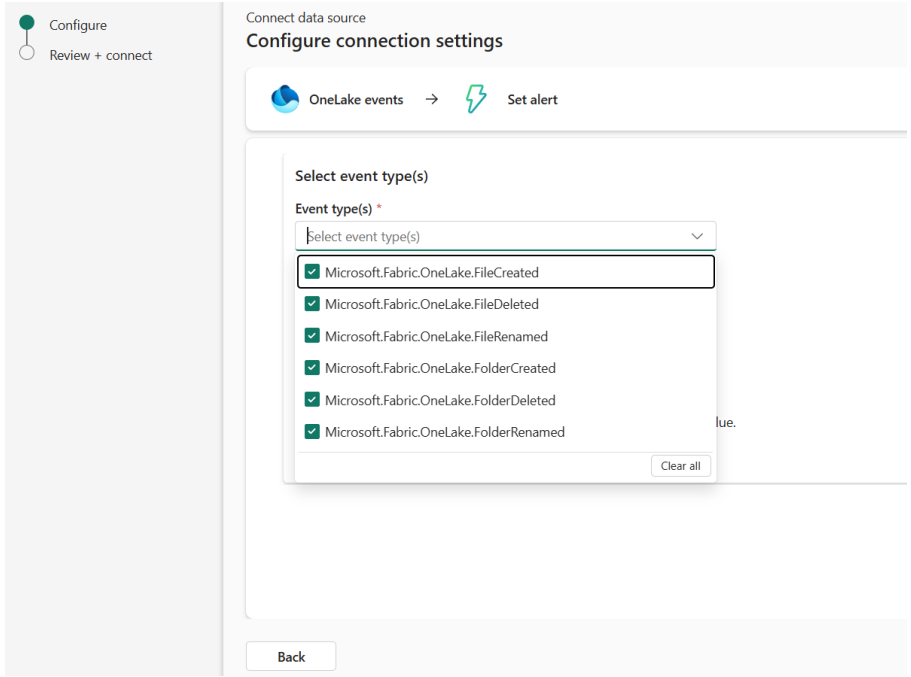


Event-based Triggers

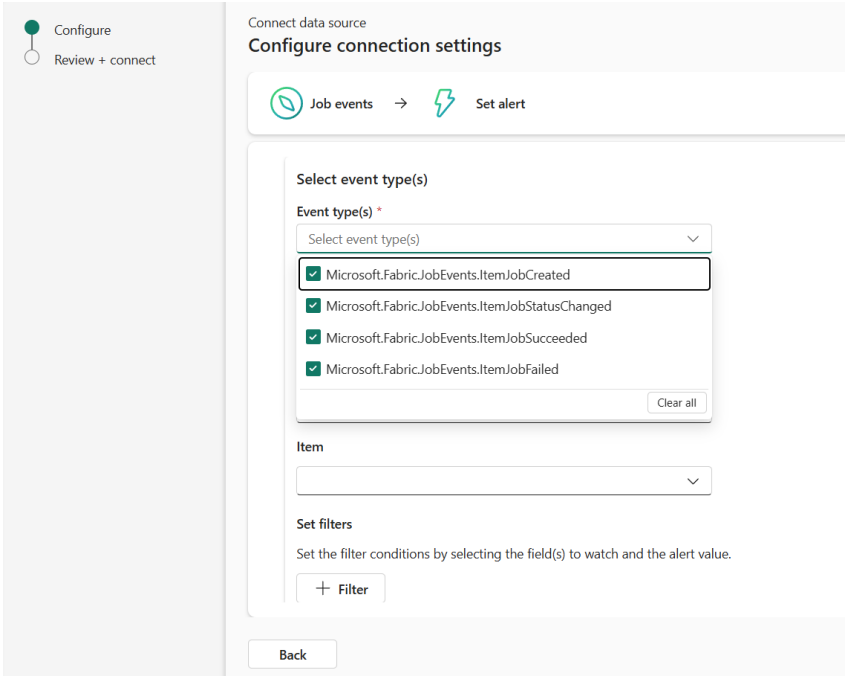


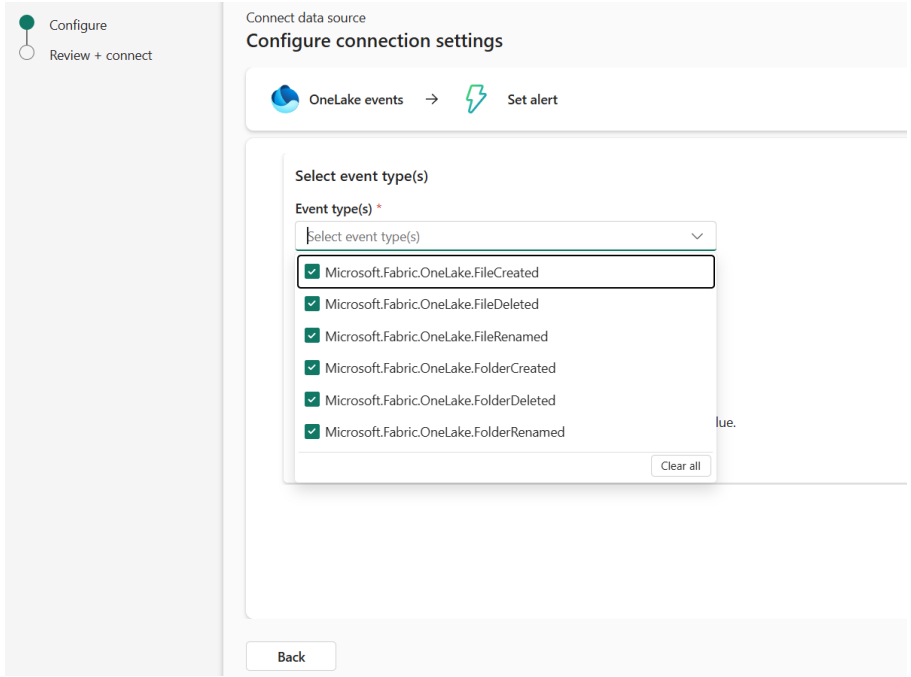
Event-based Triggers



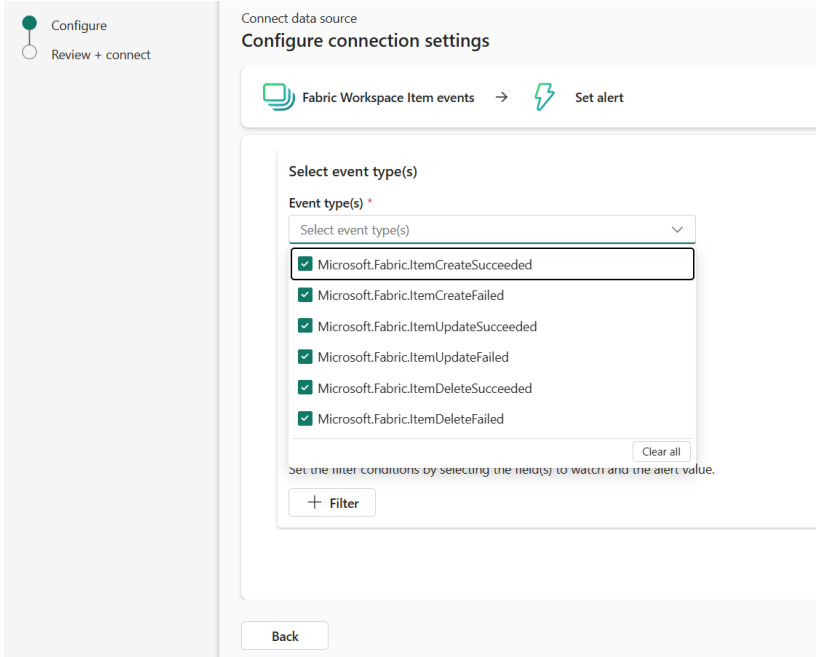


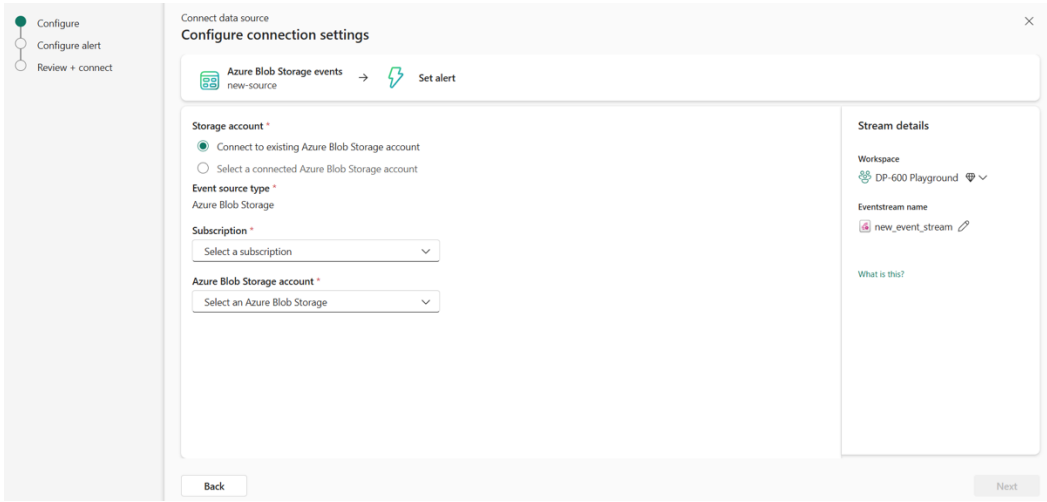
Event-based Triggers





Event-based Triggers

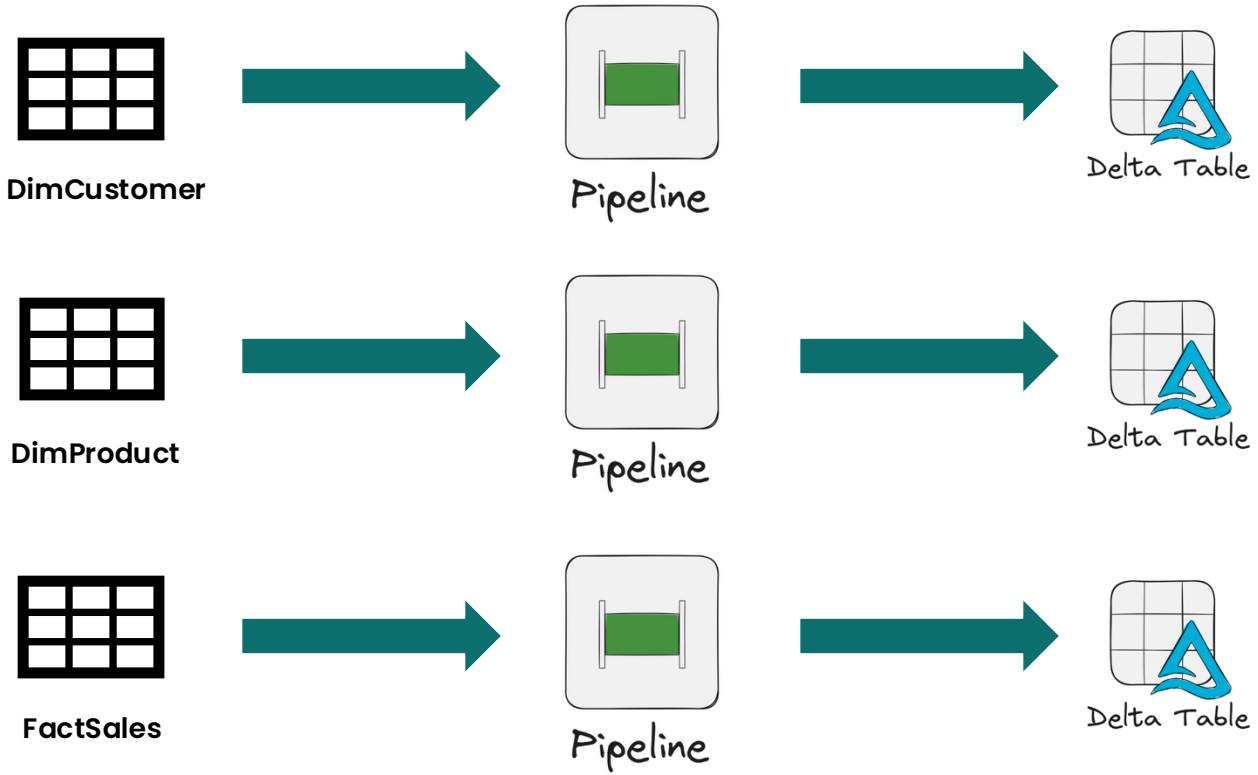




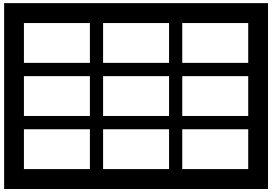
Orchestration Patterns

- Metadata-driven pipelines
- Parent/Child pipeline pattern
- Notebook orchestration

Metadata-driven Pipelines



Metadata-driven Pipelines



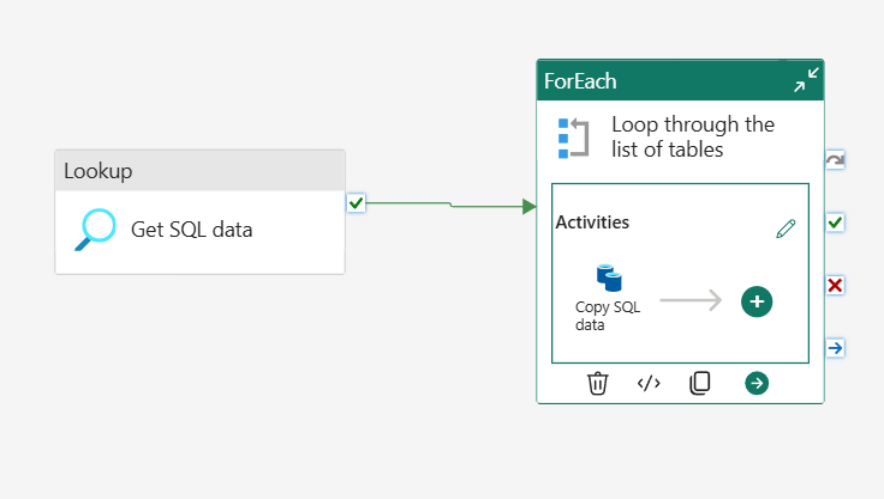
Control table

- Parameterize everything
  - Server name

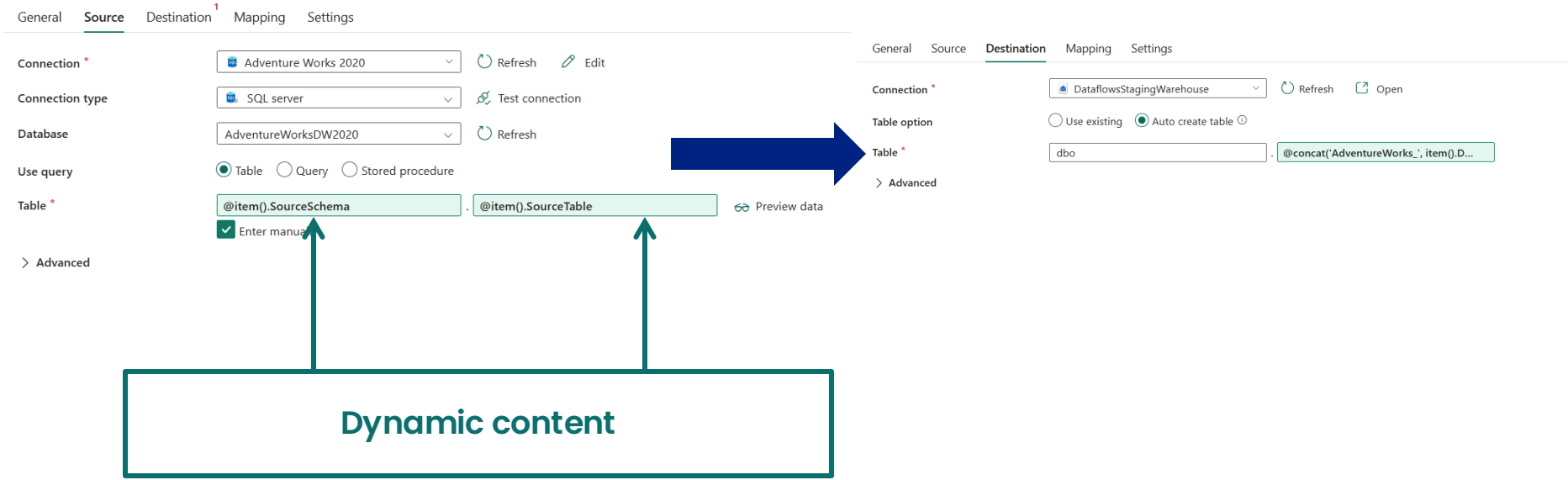
SourceTable	TargetTable	SourcePath	TransformationScript	LoadType
Customers	DimCustomer	/bronze/customers	clean_customer.sql	Full
Sales	FactSales	/bronze/sales	clean_sales.sql	Incremental

- Easily extend with a new line of data -> no need for a new pipeline
- Bulk Copy from Database template

Metadata-driven Pipelines

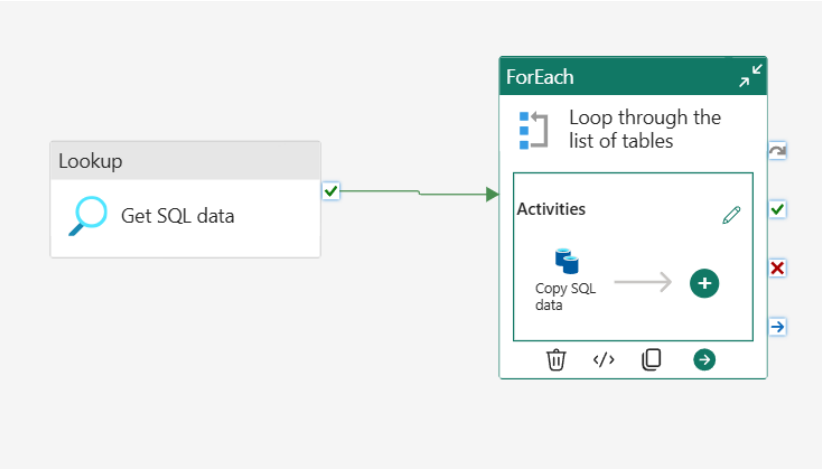


Metadata-driven Pipelines



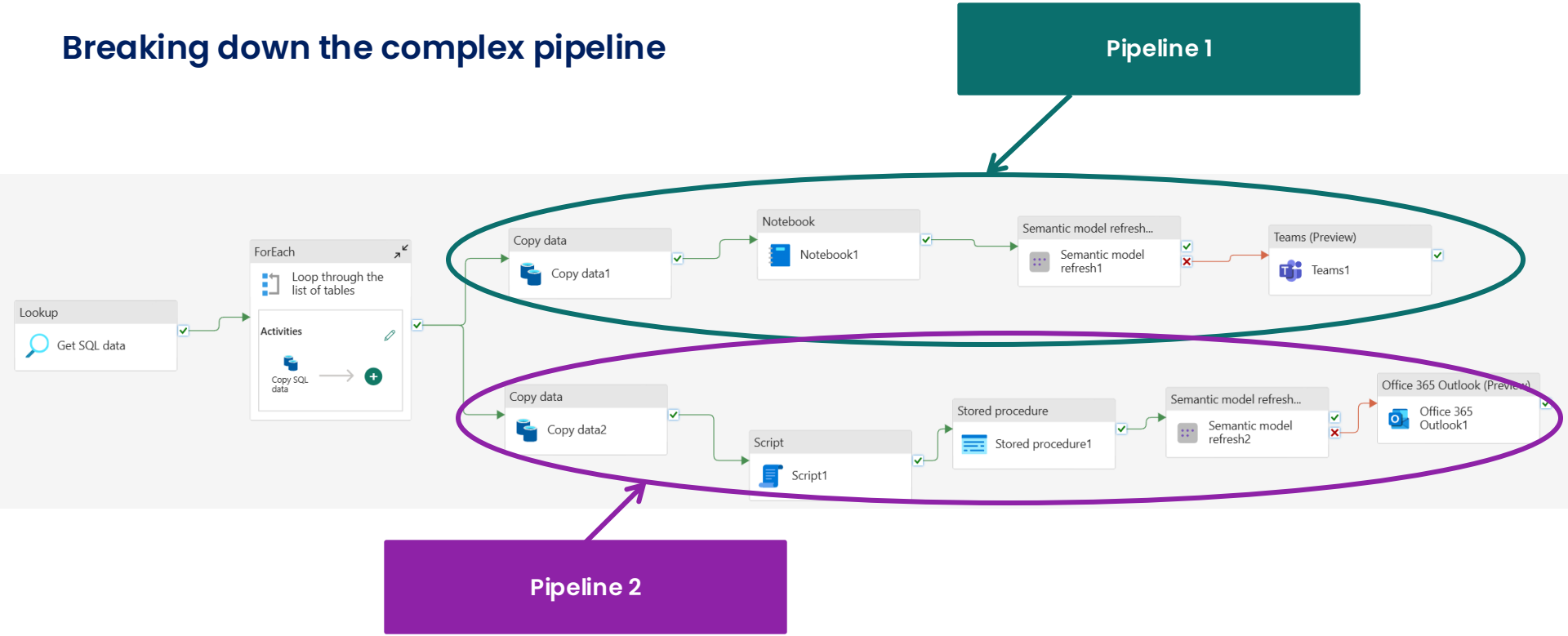
Parent-Child Pipelines

Breaking down the complex pipeline



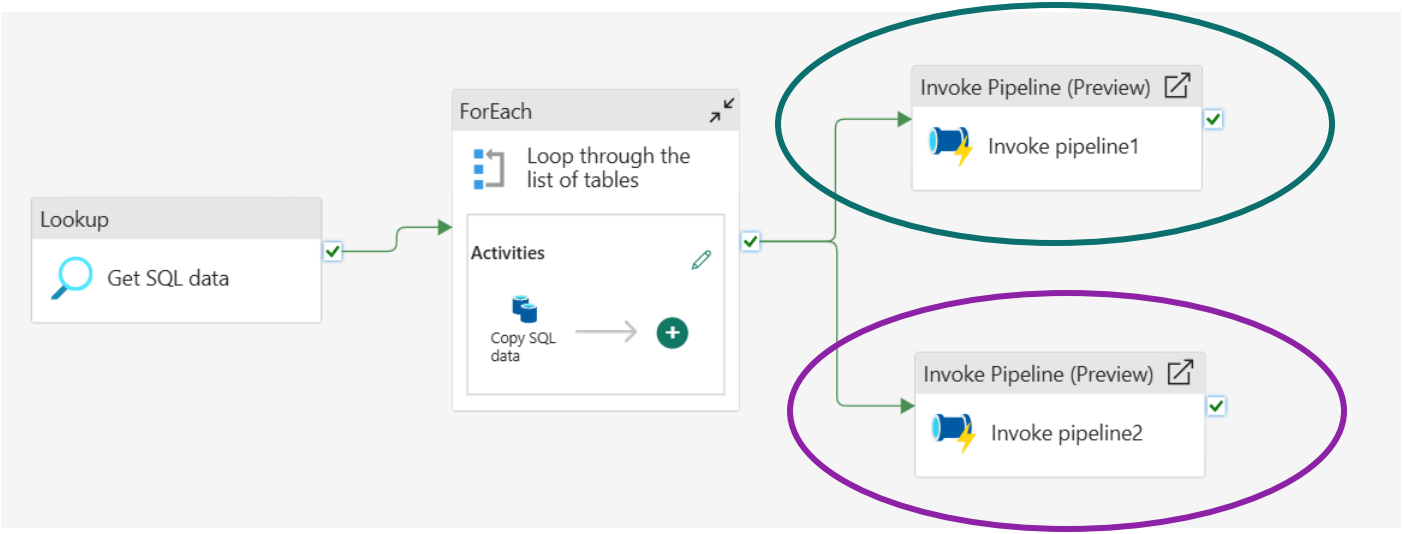
Parent-Child Pipelines

Breaking down the complex pipeline



Parent-Child Pipelines

Breaking down the complex pipeline



- **Pass parameters from parent to child**

Notebook Orchestration

- Use *notebookutils* package

```
1  from notebookutils import notebook as notebook
2
3  notebook.runMultiple (["Notebook1", "Notebook2", "Notebook3"])
```

- The simplest method
- Arbitrary order of execution

Notebook Orchestration

- Use *notebookutils* package and *DAG object* (Directed Acyclic Graph)

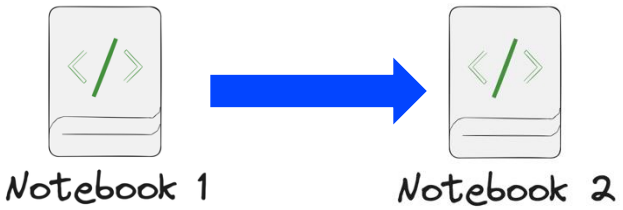
```python
from notebookutils import notebook as notebook

DAG = {
    "activities": [
        {
            "name": "Notebook1",
            "path": "Notebook1",
            "timeoutPerCellInSeconds": 60,

        },
        {
            "name": "Notebook2",
            "path": "Notebook2",
            "timeoutPerCellInSeconds": 30,
            "retry": 2,
            "retryIntervalSeconds": 20,
            "dependencies": ["Notebook1"]
        }
    ],
    "timeoutInSeconds": 1800,
}

notebook.runMultiple(DAG)
```

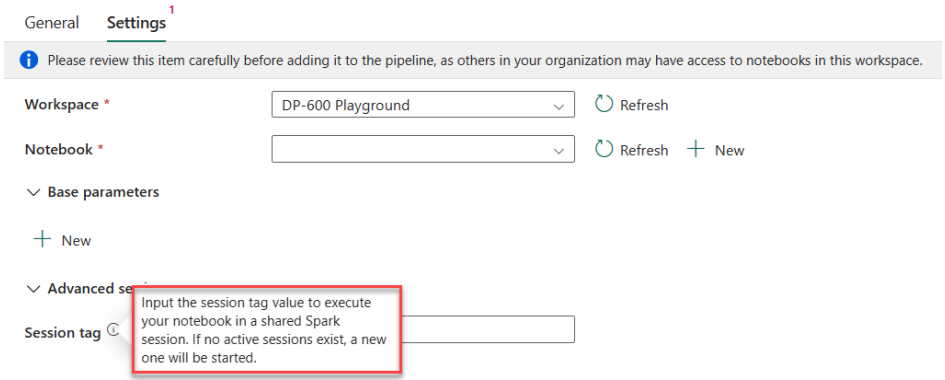
- Similar to a pipeline
- "Activities" – a list of notebooks to be executed
- "Dependencies" – defines the order of execution



Notebook Orchestration

DAG vs. Pipeline

- **DAG** – all notebooks run within the scope of the same Spark session
- **Pipeline** – Set the ***Session tag*** property -> configured notebooks run in the scope of the same Spark session



DEMO

- Create metadata-driven data pipeline

Q&A

Ingest and Transform Data

Design and Implement Loading Patterns

Design and implement loading patterns

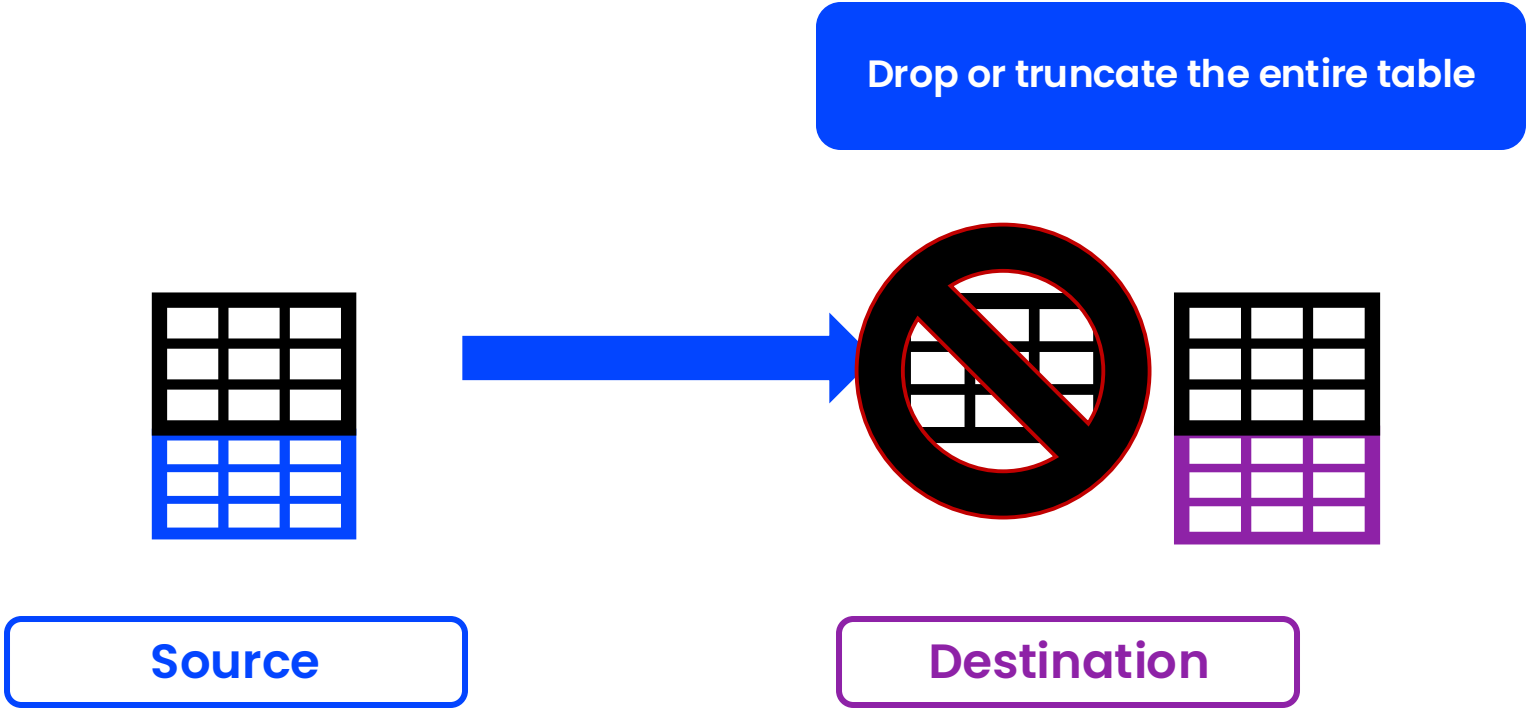
- Design and implement full and incremental data loads
- Prepare data for loading into a dimensional model
- Design and implement a loading pattern for streaming data

Full Data Loading Workflow – Initial Load

Source

Destination

Full Data Loading Workflow



Full Data Loading Workflow

Product ID	Product Name
1	T-Shirt
2	Ball
3	Socks
4	Mug

Source

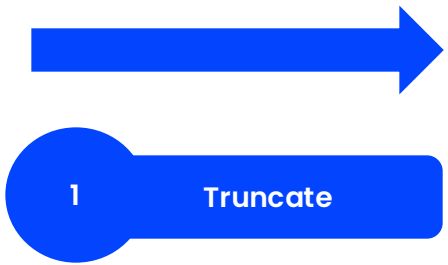
Product ID	Product Name
1	T-Shirt
2	Ball
3	Socks
4	Mug

Destination

Full Data Loading Workflow - Truncate

Product ID	Product Name
1	T-Shirt
2	Ball
3	Socks
4	Mug

Source



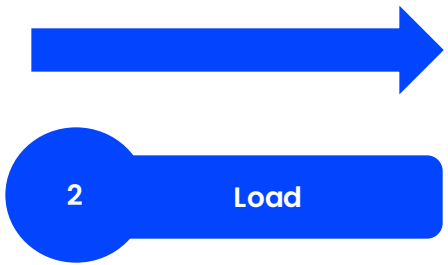
Product ID	Product Name

Destination

Full Data Loading Workflow - Load

Product ID	Product Name
1	T-Shirt
2	Ball
3	Socks
4	Mug

Source



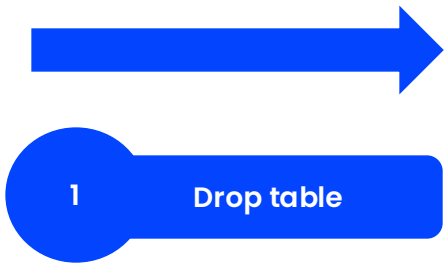
Product ID	Product Name
1	T-Shirt
2	Ball
3	Socks
4	Mug

Destination

Full Data Loading Workflow - Drop

Product ID	Product Name
1	T-Shirt
2	Ball
3	Socks
4	Mug

Source

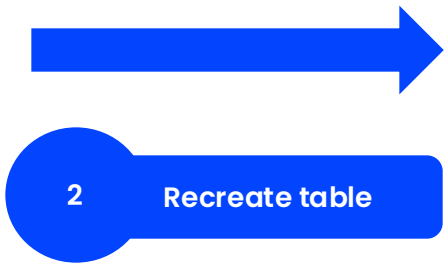


Product ID	Product Name
1	T-Shirt
2	Ball
3	Socks
4	Mug

Destination

Full Data Loading Workflow - Recreate

Product ID	Product Name
1	T-Shirt
2	Ball
3	Socks
4	Mug

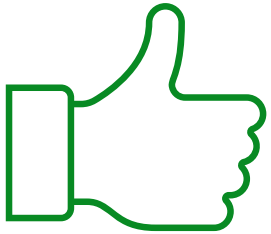


Product ID	Product Name
1	T-Shirt
2	Ball
3	Socks
4	Mug

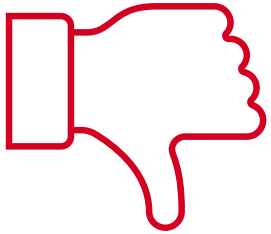
Source

Destination

Full Data Loading – Pros and Cons



- Simple to implement
- Guarantees complete data consistency



- Inefficient with large datasets
- Higher processing time and cost
- Potential data unavailability during the load

Incremental load

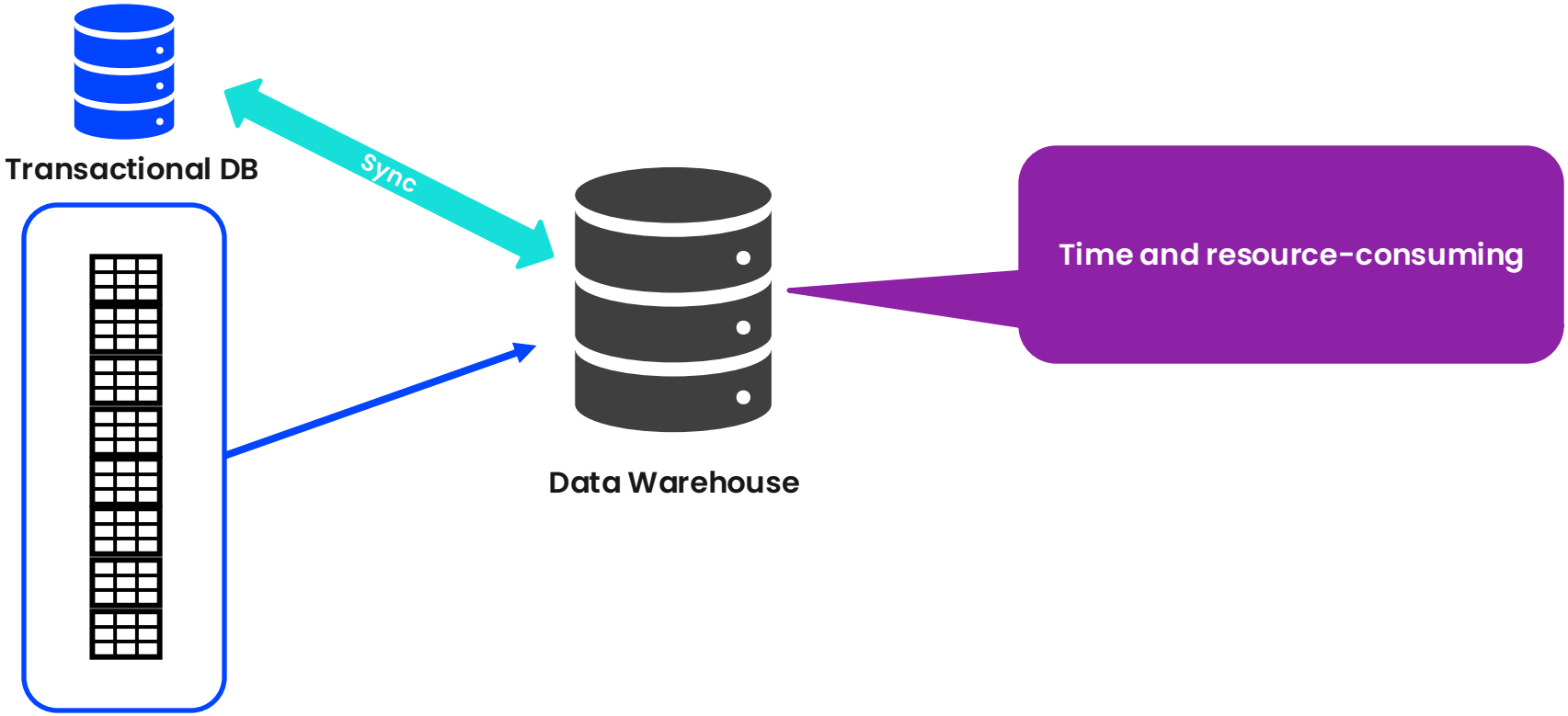
=

Process of loading

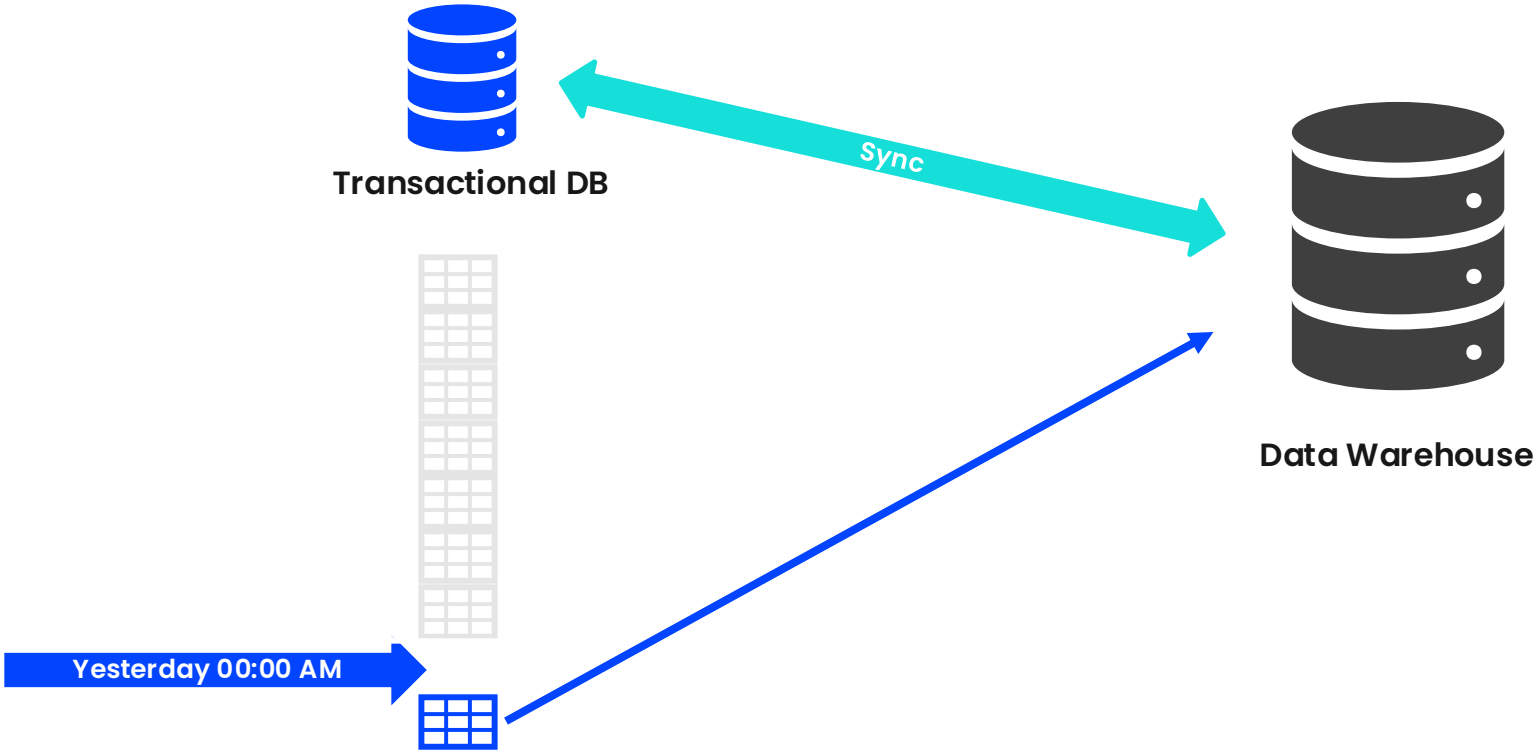
new or updated data,

after the previous loading cycle.

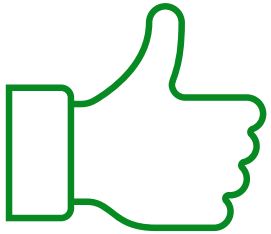
Why Incremental Loading?



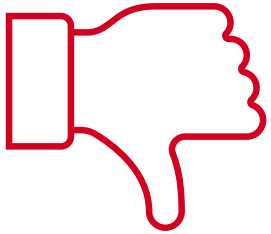
Why Incremental Loading?



Incremental Data Loading – Pros and Cons



- Faster and more efficient
- Reduced network and processing overhead
- Suitable for frequent updates



- More complex to implement
- Potential data inconsistency (if timestamps are not reliable)

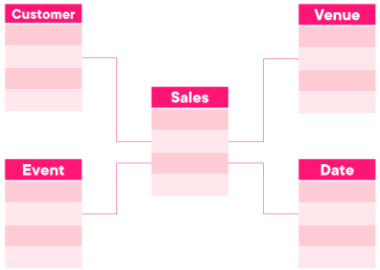
Full vs. Incremental Load – Which to Choose?

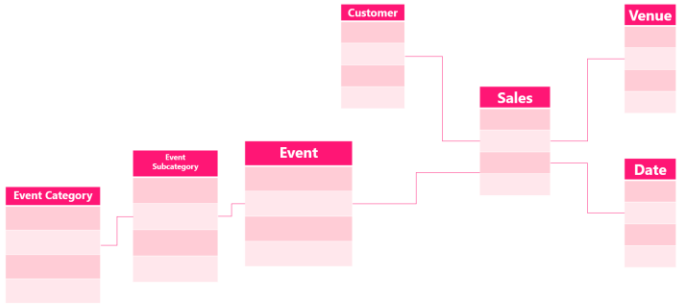
Full loading	vs.	Incremental loading
Small dataset		Large dataset
Quick and easy one-time sync		Frequent updates
The data source doesn’t support change tracking		Performance and cost optimization

Disclaimer

I assume you know what dimension and fact tables are!

I assume you know what star and snowflake schema are!





Dimension Tables - Extended

Slowly changing dimension (SCD)

- Changes to attributes
- Analyze changes over time
- Changes may include:
  - Customer address, Product price
- SCD Type 2 is most commonly used

Adam

Born: California, in 1995

Customer Name	Customer Location	Valid From	Valid To
Jerry	California	1995-05-05	NULL

Dimension Tables – SCD Type 2

Adam

Born: California, in 1995

Moved to: New York, in 2020

Customer Name	Customer Location	Valid From	Valid To
Jerry	California	1995-05-05	2019-12-31
Jerry	New York	2020-01-01	NULL

Dimension Tables – SCD Type 2

Adam

Born: California, in 1995

Moved to: New York, in 2020

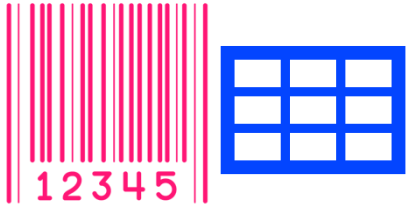
Moved to: California, in 2025

Customer Name	Customer Location	Valid From	Valid To
Jerry	California	1995-05-05	2019-12-31
Jerry	New York	2020-01-01	2024-12-31
Jerry	California	2025-01-01	NULL

Surrogate Key vs. Natural Key

Surrogate key

- Artificially generated unique identifier (usually an integer)
- No business meaning
- 1, 2, 3… or GUIDs



Natural key

- Existing data attribute(s)
- Has business meaning
- Customer key, Product code…

Surrogate Key vs. Natural Key

Surrogate key

The Secret Code Sticker

- It's like a library card number for each toy
- Helps the computer keep track of things easily
- No one else gets this exact number
- Doesn't change, even if the toy's details change
- Always unique

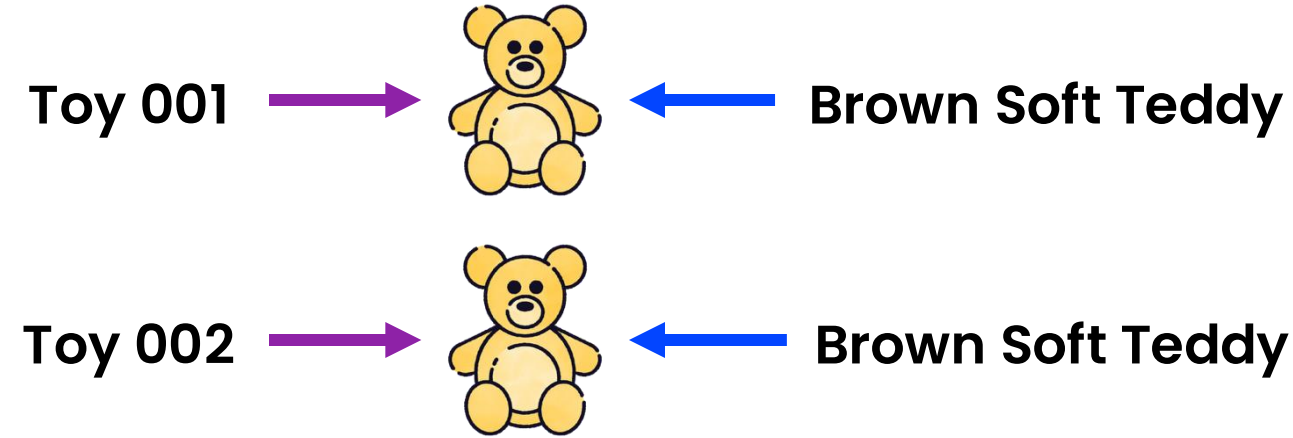
Natural key

The Original Name Tag

- For a teddy bear, it might be "Brown Soft Teddy"
- What humans naturally use to identify something
- Might change or be similar to other items
- Can sometimes be the same for multiple toys

Surrogate Key vs. Natural Key

Surrogate key	Natural key
The Secret Code Sticker	The Original Name Tag



Stream Processing in a Nutshell

Stream processing

Source

One or more high-frequent data flow

Transformation

Data processing as soon as it becomes available

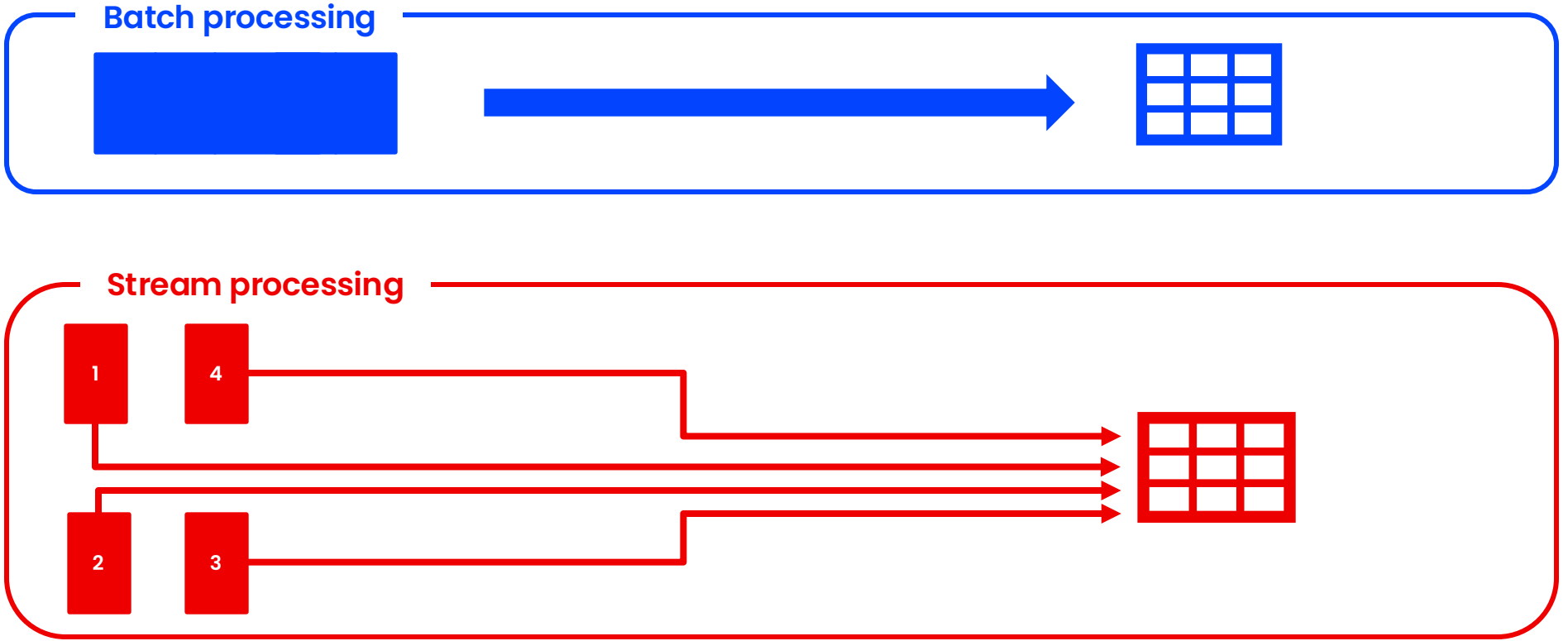
Destination

A receiver that acts on the data. It may react immediately or store data for later use

Stream vs. Batch Processing

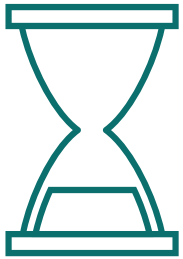
Stream vs. Batch Processing

Stream vs. Batch Processing



Stream Processing Key Considerations

Data velocity and volume



Latency requirements

Data quality and schema evolution

Fault tolerance

Stream Processing in Microsoft Fabric

Eventhouse

Real-Time Intelligence

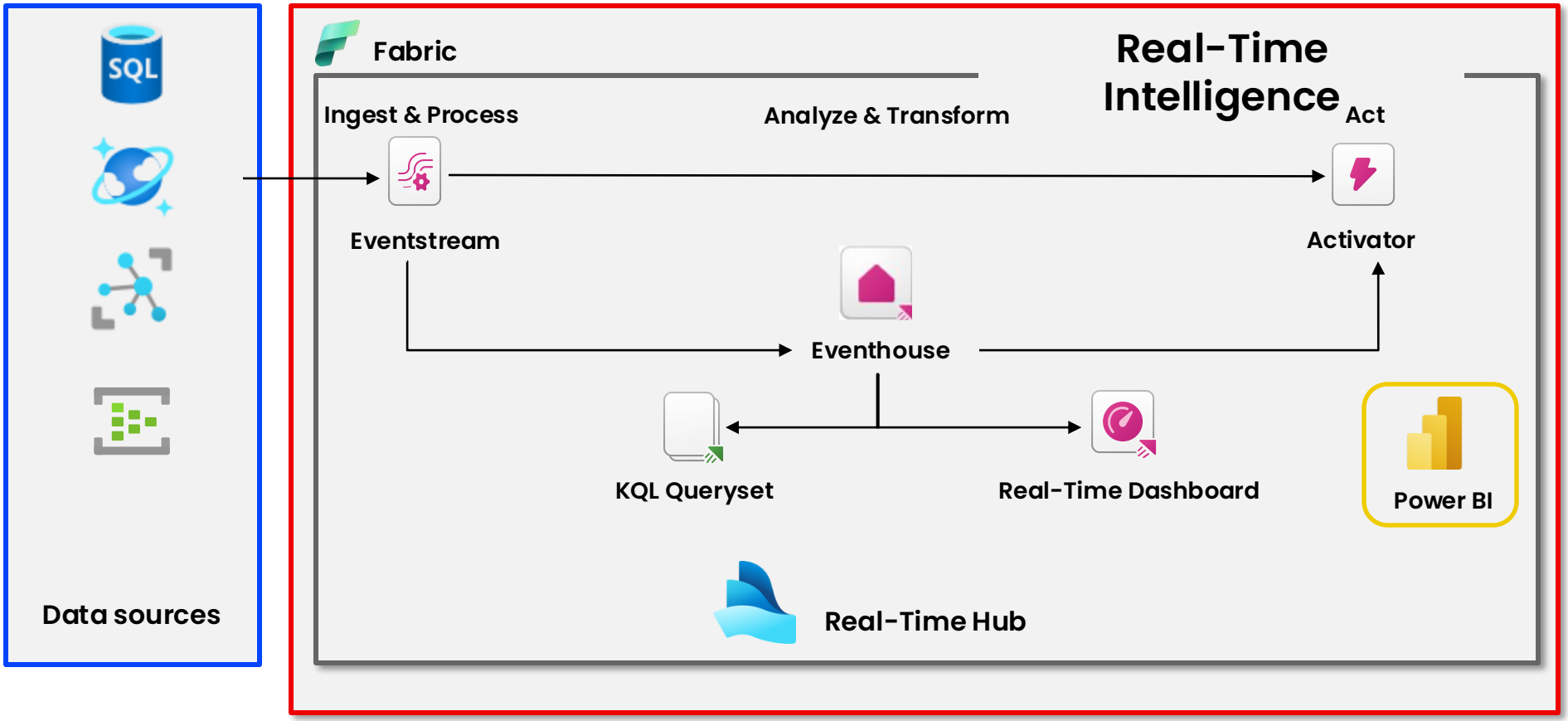
Eventstream

KQL database

Activator

KQL queryset

Stream Processing in Microsoft Fabric



DEMO

- Implement an incremental data loading process
  - Pipelines
  - Notebooks
  - Dataflows

Q&A


O'REILLY®
Day 2
Microsoft Fabric Data Engineer
Bootcamp (DP-700)

Introduction and Recap

Ingest and Transform Batch Data

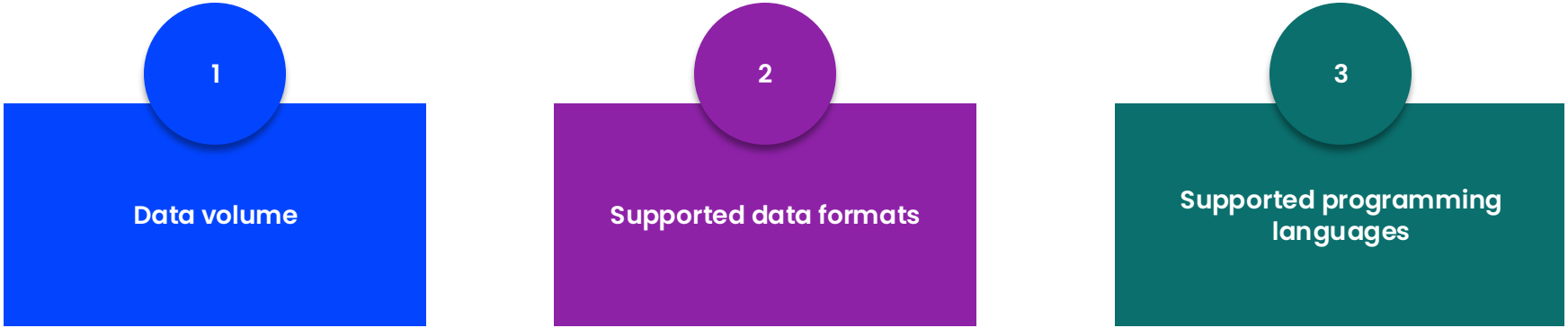
Ingest and transform batch data

- Choose an appropriate data store
- Choose between dataflows, notebooks, KQL/T-SQL for data transformation
- Implement mirroring
- Ingest data by using pipelines
- Transform data with PySpark/SQL/KQL
- Denormalize data
- Group and aggregate data
- Handle duplicate, missing, and late-arriving data

Choose Your Team? (POLL)

1. Python/PySpark
2. SQL/T-SQL
3. Other (please specify in the chat)

Choosing an Analytical Engine



Choosing an Analytical Engine

1

Data volume

- Big data: > 5 TB of stored data, and 100+ GB of ingested data/day
- No such thing as a minimum data amount suitable for WH vs. LH
- MPP makes the difference only with large amounts of data

NOT a determining factor when choosing between the lakehouse vs. warehouse vs. eventhouse

Choosing an Analytical Engine

2

Supported data formats

Data Format	Lakehouse	Warehouse	Eventhouse
Structured	Yes	Yes	Yes
Semi-structured	Yes	Limited (JSON)	Yes
Unstructured	Yes	No	Yes

It's more nuanced...

- Using data downstream
- Optimizing storage cost
- Processing streaming data

Streaming or event-based data (telemetry, logs, time-series...)

Choosing an Analytical Engine

3

Supported programming languages

Operation type	Lakehouse	Warehouse	Eventhouse

Read	• Spark
	  • PySpark
	  • Spark SQL
	  • Scala
	  • R
	• T-SQL
	• Python
		• T-SQL	• KQL
			• T-SQL

Write	• Spark
	  • PySpark
	  • Spark SQL
	  • Scala
	  • R
	• Python
		• T-SQL
		• Python (using pyodbc library)
			• KQL

Choosing an Analytical Engine

What if I choose the lakehouse for my silver layer, and the warehouse for the gold layer? Can I combine the data from both?

OneLake Interoperability

Interoperability feature	Lakehouse	Warehouse	Eventhouse
Data stored in OneLake	Yes	Yes	Yes, when the OneLake integration property enabled
Data stored in delta format	Yes	Yes	Yes, when the OneLake integration property enabled
Source for shortcuts	Yes	Yes	Yes, when the OneLake integration property enabled
Target for shorctuts	Yes	Yes, via cross-database queries	Yes
Cross lakehouse/warehouse/eventhouse queries	Yes	Yes	Yes, when the OneLake integration property enabled

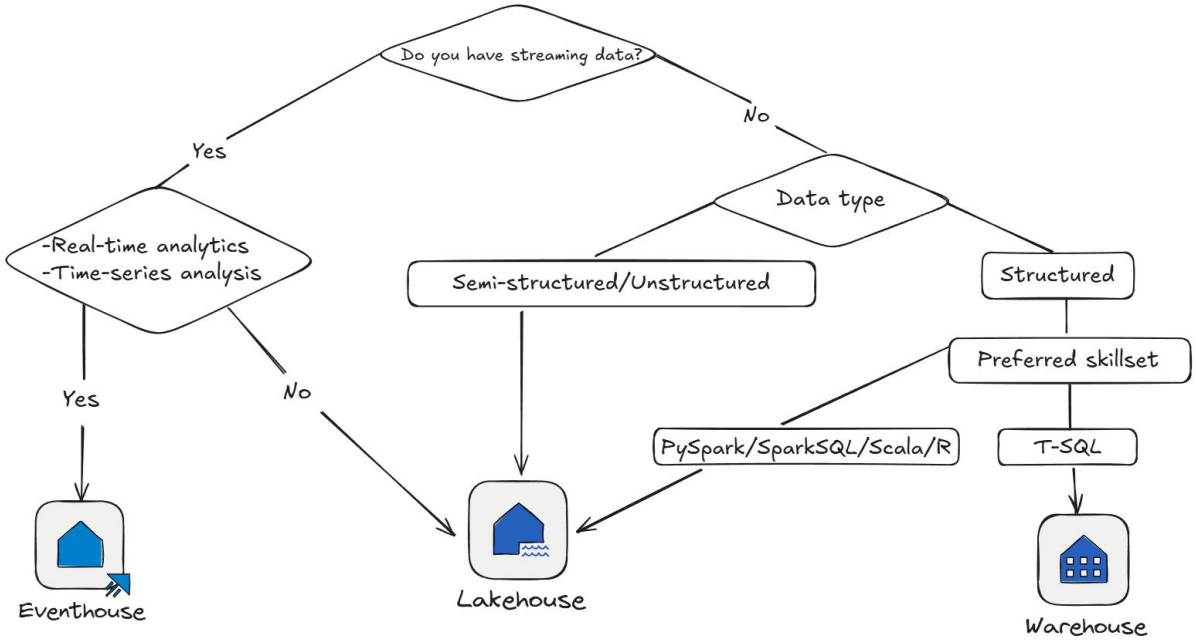
Choosing an Analytical Engine

Scenario-based decision guide

Scenario	Lakehouse	Warehouse	Eventhouse
Operational reports with low data latency	★★★	★★★	★★★★★
Enterprise data warehousing	★★★★★	★★★★★	★★★
Implement a medallion design pattern*	★★★★★	★★★	★★★★★
Implement data marts	★★★★★	★★★★★	★
Real-time intelligence	★★★	★	★★★★★
Handling arbitrary unstructured data	★★★★★	★	★★★

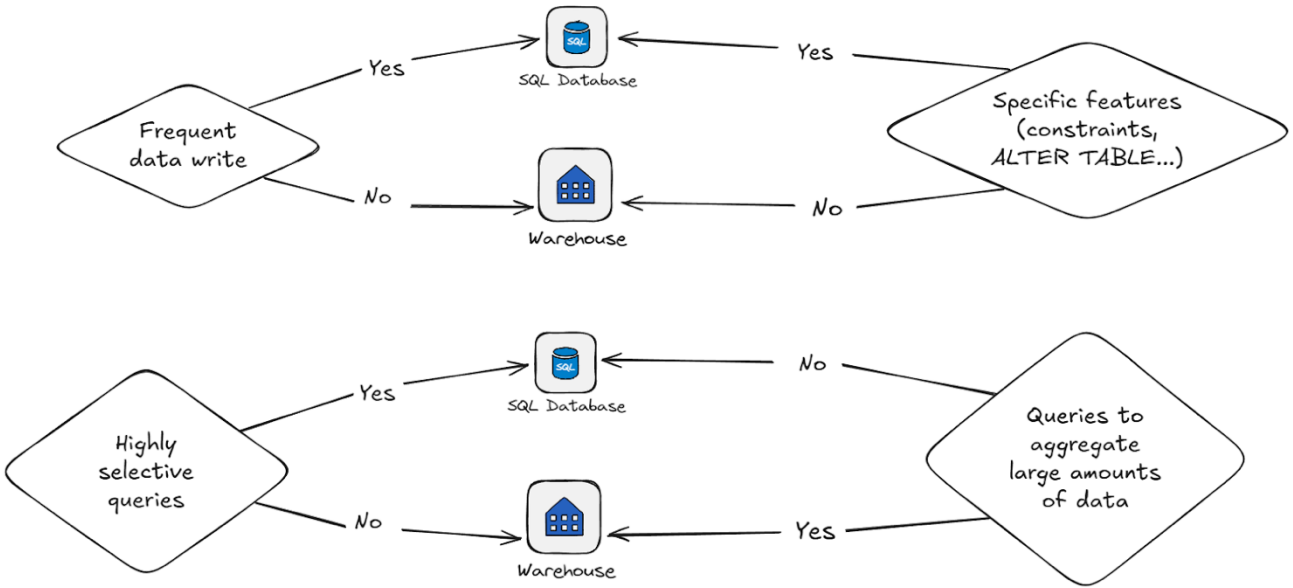
*When a single analytical engine is used across all medallion layers

Choosing an Analytical Engine



Choosing the Appropriate Storage

SQL database in Fabric vs. Warehouse



Choosing the Appropriate Data Transformation Tool

Dataflows | Notebooks | KQL | T-SQL

Low-code approach

Code-first approach

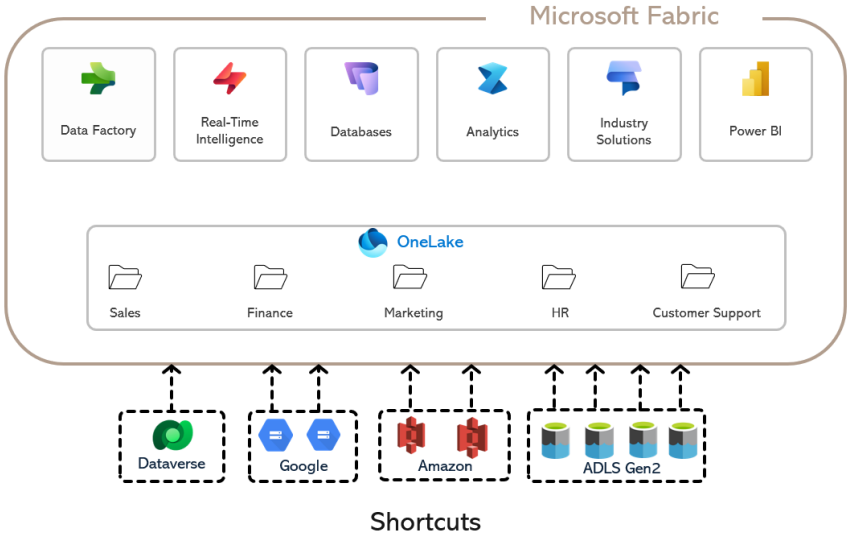
Dataflows Gen2

Notebook

KQL Database

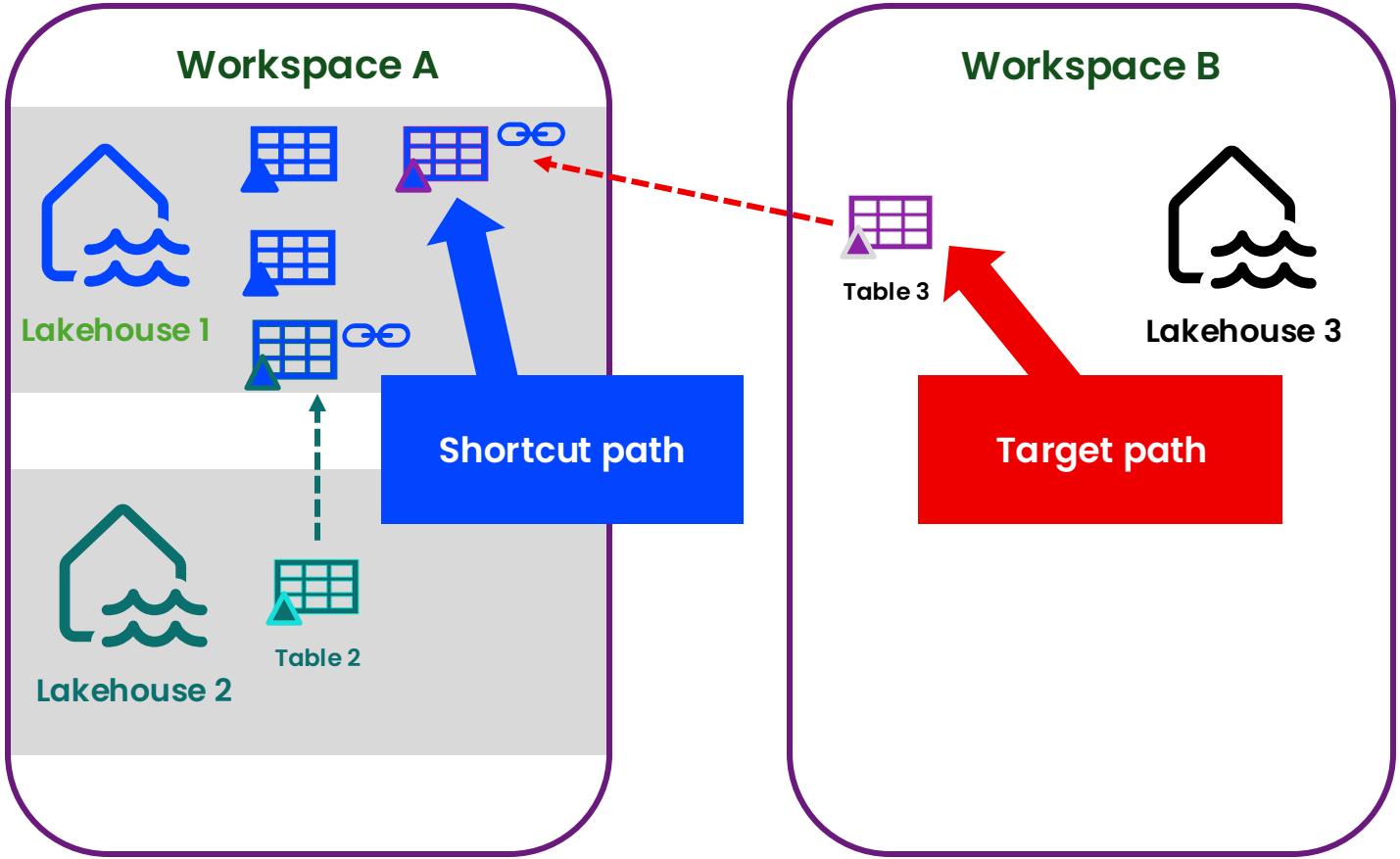
SQL Database

Understanding Shortcuts

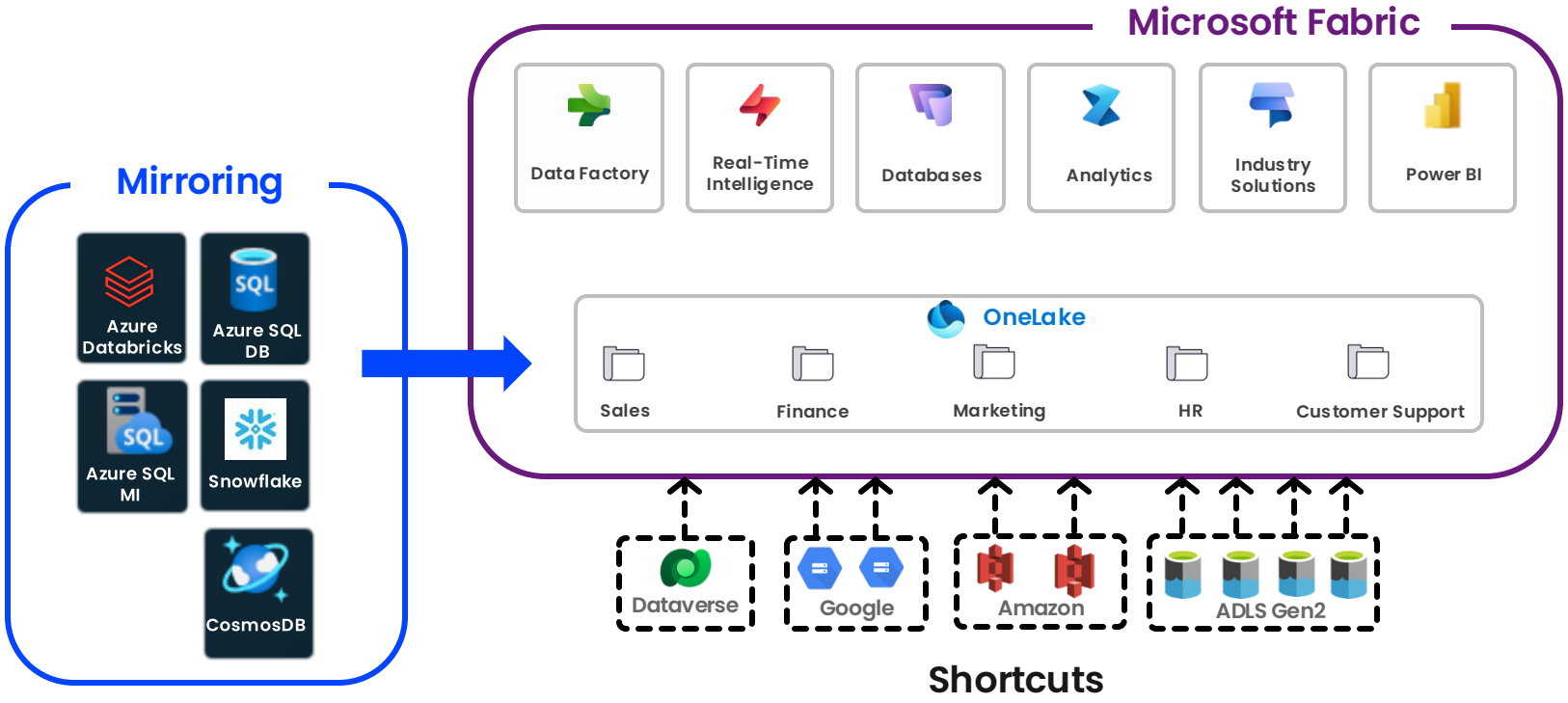


OneLake objects that point to other storage locations

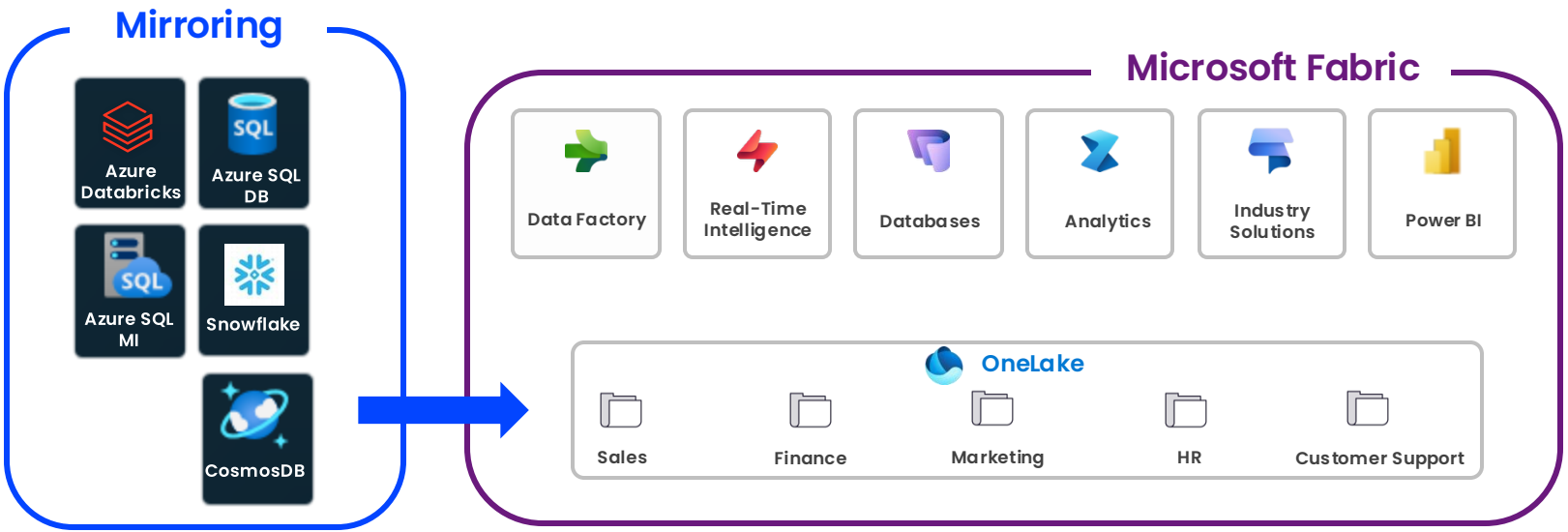
Understanding Shortcuts



All Roads Lead to… ~~Rome~~ OneLake

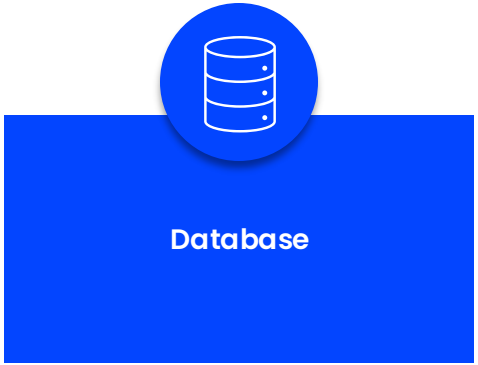


Understanding Mirroring

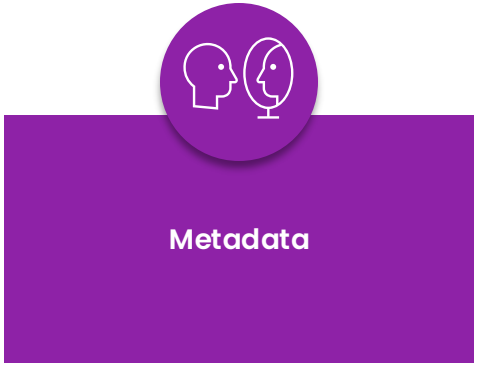


- Near real-time replica
- No complex ETL
- Combine data with other Fabric workloads

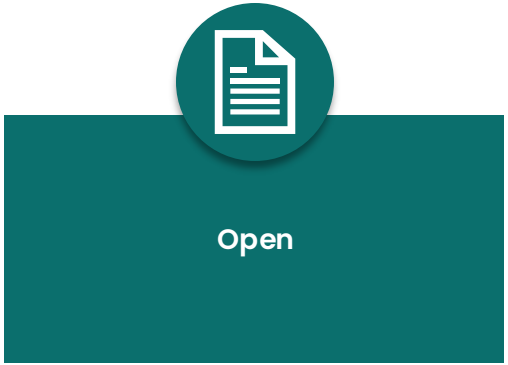
Mirroring Types in Fabric



Entire database or individual tables



Catalog names, schemas…instead of physically moving the data (Azure Databricks)



Extends to other data sources based on the Delta format

Transforming Data in Microsoft Fabric

Views | Functions | Stored procedures

Lakehouse

Using PySpark or Spark SQL

Warehouse

The same as in the traditional T-SQL workloads

Eventhouse

Supports materialized views

Views don’t work with Direct Lake mode!

Transforming Data in Microsoft Fabric

Enrich Data by Adding New Columns or Tables

Lakehouse

Warehouse

Eventhouse

Enrich Data by Adding New Columns or Tables

Lakehouse

- Use notebooks or Spark jobs
- Operating on a dataframe
- Use withColumn() and select() functions

```
from pyspark.sql.functions import col, split

# Create customer dataframe

dfdimCustomer = df.dropDuplicates(["CustomerName","Email"]).select(col("CustomerName"),col("Email")) \
    .withColumn("First",split(col("CustomerName"), " ").getItem(0)) \
    .withColumn("Last",split(col("CustomerName"), " ").getItem(1))
```

Enrich Data by Adding New Columns or Tables

Warehouse

- ALTER TABLE…ADD COLUMN
- ~~ALTER TABLE…ALTER COLUMN~~
- ~~ALTER TABLE…DROP COLUMN~~

```
1   ALTER TABLE dbo.DimProduct
2   ADD ProductSubCategory VARCHAR(255);
```

Enrich Data by Adding New Columns or Tables

Eventhouse

- .alter table command
- Existing non-specified columns will be dropped
- Use .show table [myTable] cslschema to get the existing table schema before you alter it
- Adds a nullable column to the end of the schema

```
.alter table MyTable (ColumnX:string, ColumnY:int)
.alter table MyTable (ColumnX:string, ColumnY:int) with (docstring = "Some documentation", folder = "Folder1
```

Enrich Data by Adding New Columns or Tables

Eventhouse

- *extend* operator
- Creates a calculated column and appends to the end of the result set

```
StormEvents
| project EndTime, StartTime
| extend Duration = EndTime - StartTime
```

Normalization vs. Denormalization

Normalization

Process of organizing the data in a database

- In most cases, 3rd normal form is optimal
- Data writing speed

Denormalization

Creates redundant data in the table

- Data reading speed

Denormalization

Fact Sales

SalesKey	ProductKey	SalesAmount
123	1	100
456	2	200
789	1	300
357	1	400

Dim Product

ProductKey	ProductSubcategoryKey	ProductName
1	11	Shirt
2	11	Hoodie
3	12	Mug

Dim ProductSubcategory

ProductSubcategoryKey	ProductCategoryKey	SubcategoryName
11	111	Women
12	122	Kitchen

Dim ProductCategory

ProductCategoryKey	CategoryName
111	Clothes
122	House

Denormalization

Fact Sales

SalesKey	ProductKey	SalesAmount
123	1	100
456	2	200
789	1	300
357	1	400

Dim Product

ProductKey	SubcategoryName	CategoryName	ProductName
1	Women	Clothes	Shirt
2	Women	Clothes	Hoodie
3	Kitchen	House	Mug

Aggregations

Large Fact Table

Date	Customer ID	Product ID	Sales Amount
2021-10-12	123	11	10
2021-10-12	456	12	20
2021-10-12	789	12	50
2021-10-13	123	13	30
2021-10-13	456	11	10

Create Aggregated Tables

Date	Sales Amount
2021-10-12	80
2021-10-13	40

Product ID	Sales Amount
11	20
12	70
13	30

Customer ID	Sales Amount
123	40
456	30
789	50

Date	Customer ID	Product ID	Sales Amount
2021-10-12	123	11	10
2021-10-12	456	12	20
2021-10-12	789	12	50
2021-10-13	123	13	30
2021-10-13	456	11	10

Aggregations in Microsoft Fabric

Lakehouse

PySpark/Spark SQL

Warehouse

T-SQL

Eventhouse

KQL

Identify and Resolve Duplicate Data

Lakehouse

Warehouse

Eventhouse

PySpark/Spark SQL

```
from
# Cre
dfdimCustomer = df.dropDuplica
```

T-SQL

```
WITH CTE AS (
    SELECT
    DELETE FROM CTE
    WHERE RowNum > 1;
```

KQL

```
DeviceEventsAll
| where EventDateTime > ago(90d)
| summarize hint.strategy=shuffle arg_max(EventDateTime, *) by DeviceId, EventId, StationId
```

```
ne"),col("Email")) \
```

Identify and Resolve Missing Data

Lakehouse

PySpark/Spark SQL

Warehouse

T-SQL

Eventhouse

KQL

Identify and Resolve Missing Data

Lakehouse

PySpark/Spark SQL

- ✓ *fillna()* and *fill()*
- ✓ Return the same results

```
#Replace 0 for null for all integer columns
df.na.fill(value=0).show()

#Replace 0 for null on only population column
df.na.fill(value=0,subset=["population"]).show()


df.na.fill("").show(false)


df.na.fill("unknown",["city"]) \
    .na.fill("",["type"]).show()
```

Identify and Resolve Missing Data

Warehouse

✓ *COALESCE()* and *ISNULL()*

First non-null expression

Replace NULL with the specified value

T-SQL

```
SELECT Name, Class, Color, ProductNumber,
COALESCE(Class, Color, ProductNumber) AS FirstNotNull
FROM Production.Product;
```

```
SELECT Description, DiscountPct, MinQty, ISNULL(MaxQty, 0.00) AS 'Max Quantity'
FROM Sales.SpecialOffer;
```

Identify and Resolve Missing Data

Eventhouse

KQL

- ✓ *isnull()* – returns a Boolean result for non-string columns
- ✓ *isempty()* – returns a Boolean result for string columns

a	b	isnull_a	isempty_a	strlen_a	isnull_b
		false	true	0	true
		false	false	1	true
a	1	false	false	1	false

- ✓ *series_fill_const()*– replaces missing values in a series with a specified constant value
- ✓ *coalesce()*

DEMO

- Create OneLake shortcut
- Implement common data transformations
  - PySpark
  - SQL
  - KQL
- Implement denormalization and aggregation
- Handling duplicates and missing values

Q&A

Break

Ingest and Transform Streaming Data

Ingest and transform streaming data

- Choose an appropriate streaming engine
- Choose between native storage, followed storage, or shortcuts in Real-Time Intelligence
- Process data by using Eventstreams
- Process data by using Spark structured streaming
- Process data by using KQL
- Create windowing functions

Choose the Appropriate Streaming Engine

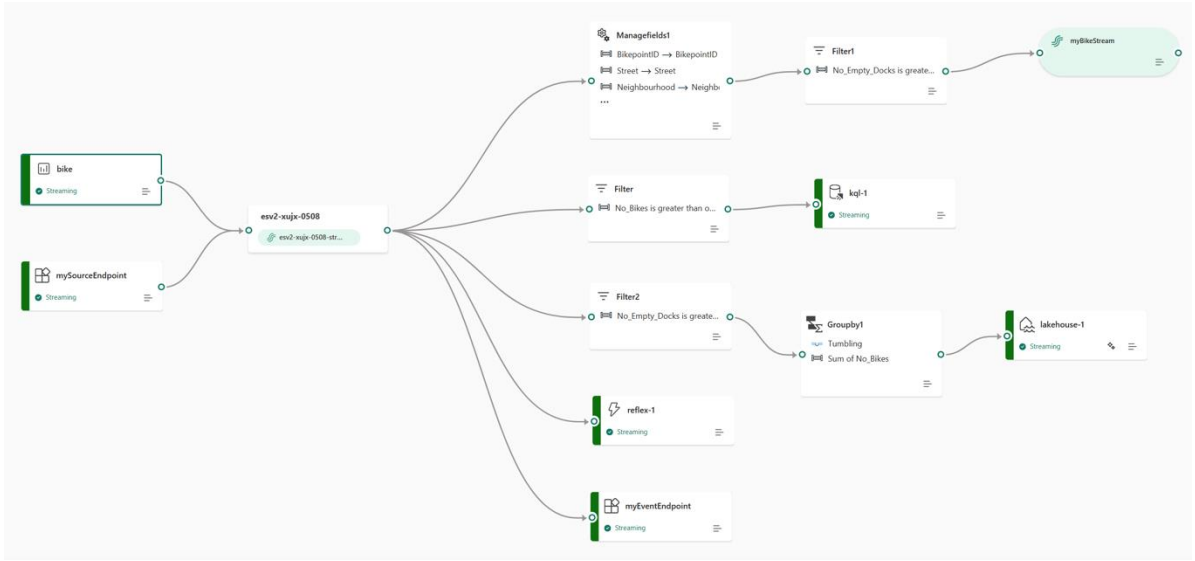
Eventstream

Spark Structured Streaming

Choose the Appropriate Streaming Engine

Eventstream

- Bring real-time events to Fabric
- No-code transformations (filter, manage fields, aggregate, group by, join…)



Choose the Appropriate Streaming Engine

Sources

- Azure Event Hubs
- Azure IoT Hub
- CDC for most DBs
- Apache Kafka
- Azure Blob Storage events
- Fabric Workspace Item events (creating, updating, deleting Fabric item)
- Fabric OneLake events (changes in files and folders)
- Fabric Job events

Operations

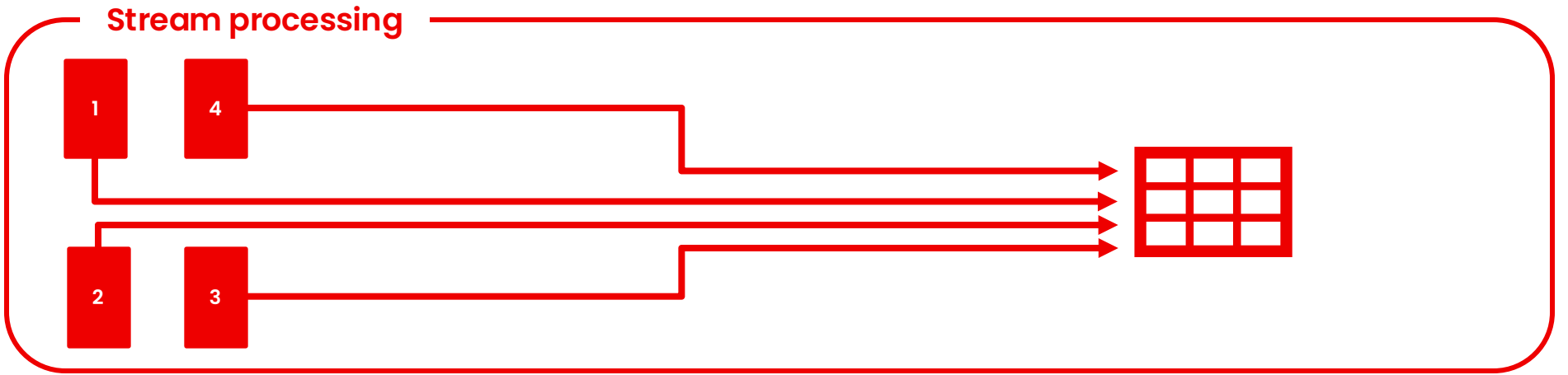
- Aggregate
- Expand
- Filter
- Group by
- Join
- Manage fields
- Union

Destinations

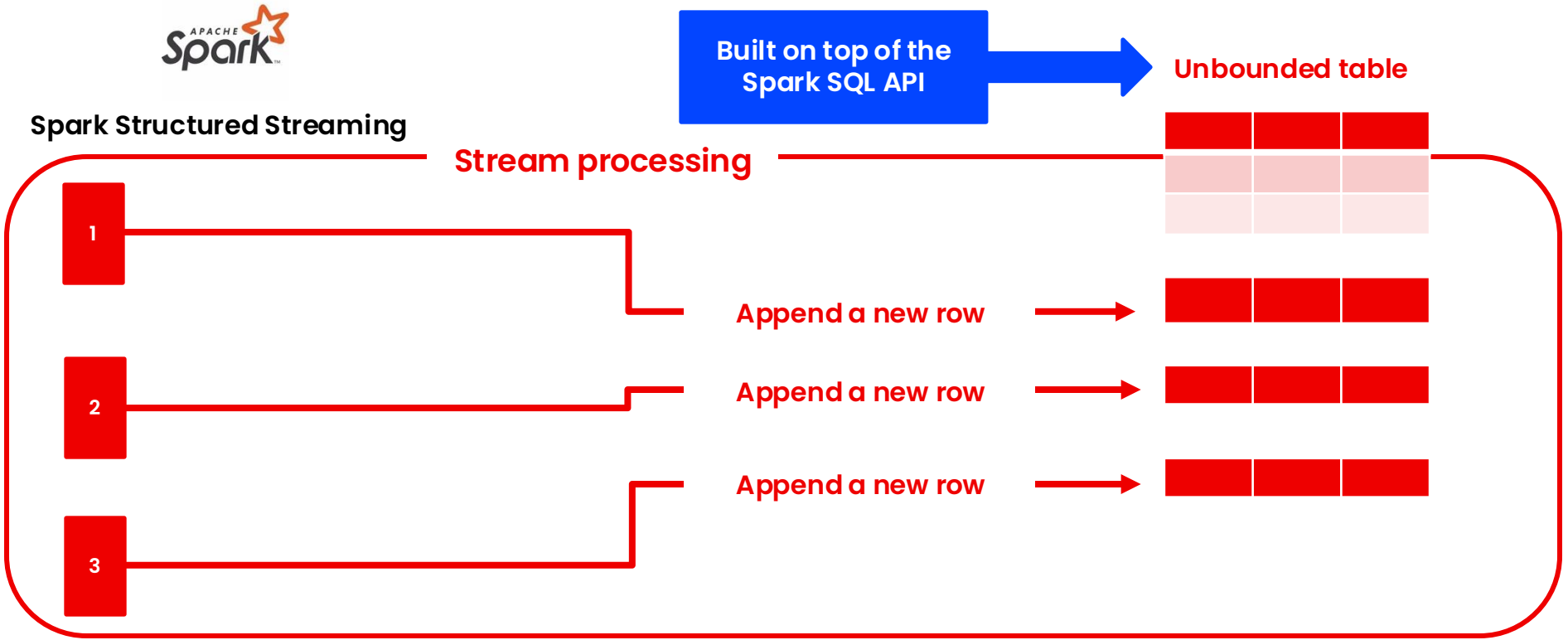
- Custom endpoint
- Eventhouse
- Lakehouse
- Derived stream
- Fabric Activator

Choose the Appropriate Streaming Engine

Spark Structured Streaming



Choose the Appropriate Streaming Engine



Choose the Appropriate Streaming EngineC

Spark Structured Streaming

Azure Event Hubs example

```python
import pyspark.sql.functions as f
from pyspark.sql.types import *

# Define the schema for incoming data
schema = StructType([
    StructField("column1", StringType(), True),
    StructField("column2", IntegerType(), True),
    # Add additional fields as needed
])

# Read streaming data from Event Hubs
df = spark.readStream \
    .format("eventhubs") \
    .options(**event_hub_config) \
    .load()

# Parse the JSON messages
parsed_df = df.withColumn("data", f.from_json(f.col("body").cast("string"), schema)).select("

# Write the streaming data to a Delta table
parsed_df.writeStream \
    .format("delta") \
    .option("checkpointLocation", "/path/to/checkpoint") \
    .outputMode("append") \
    .toTable("your_delta_table")
```

Eventstream vs. Spark Structured Streaming

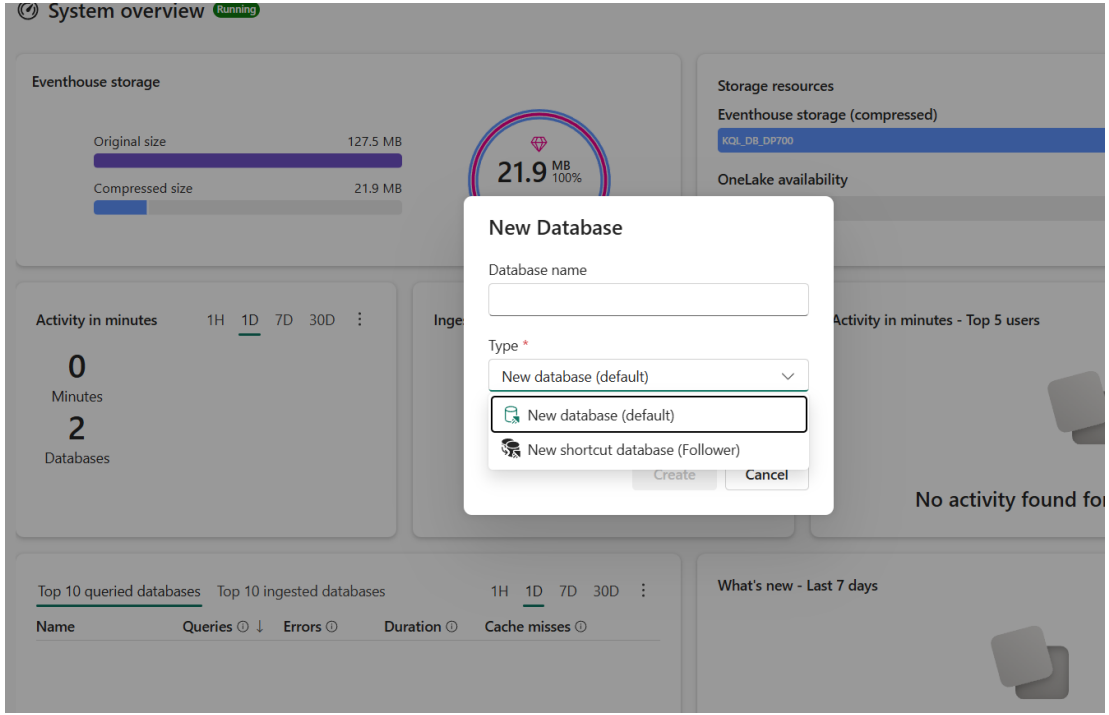
Eventstream

- No code solution
- Out-of-the-box supported sources
- Simple and easy data transformations
- Support multiple destinations
  - Lakehouse
  - KQL database
  - Derived stream
  - Activator

Spark Structured Streaming

- Code-first approach
- Already using Spark for data engineering
- Custom complex transformations
- Supports only a lakehouse destination

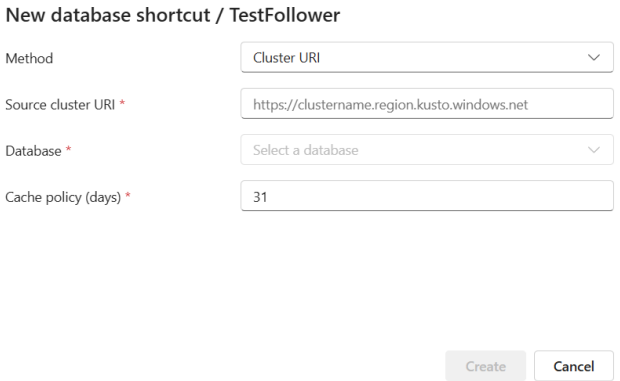
Choose the Storage – Native vs. Followed



Follower Database

- Attach a database located in a different cluster
- Read-only mode
- Changes synchronized in the leader database -> lags are possible
- The follower database “sees” the data without needing to ingest it
- The attached database can’t be deleted

Azure Data Explorer



Follower Database

Database shortcut

- Behaves as a follower database
- Use ADX data in Fabric RTI
- Separate resources to protect the production environment

KQL database

Azure Data Explorer

Follower Database

Database shortcut

KQL database

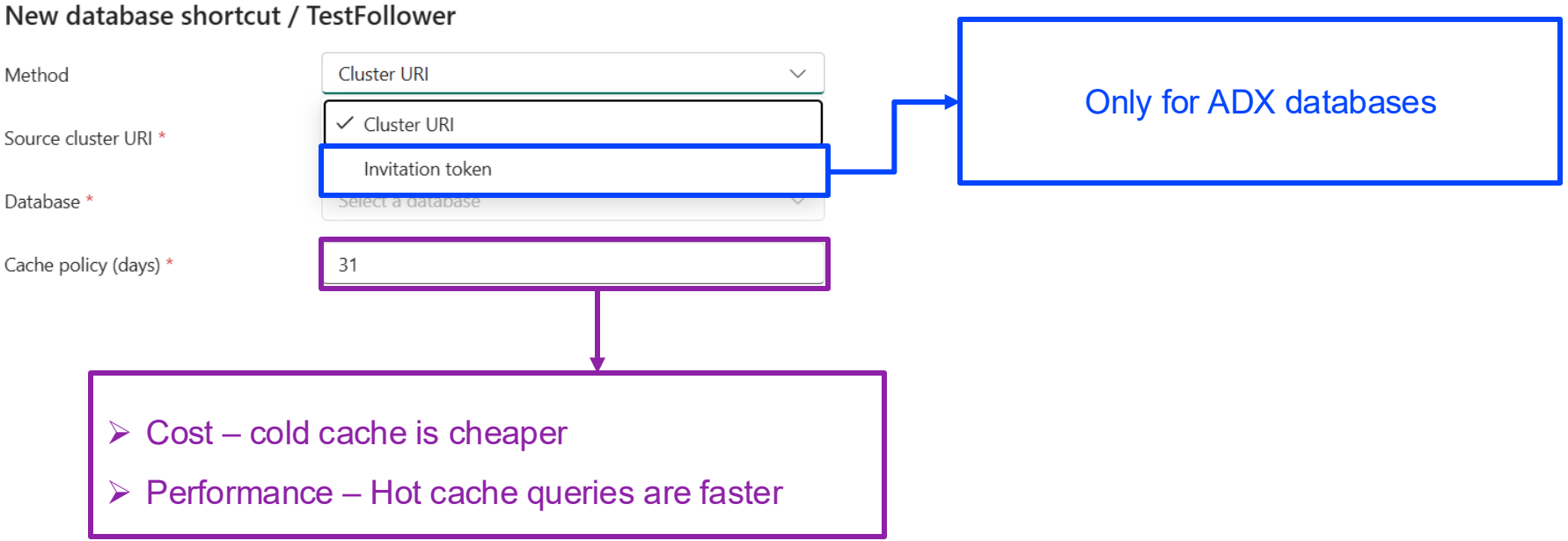
- Data consumer (creator of the shortcut in RTI)

Azure Data Explorer

- Data provider (owner of the source database)

Follower Database

Database shortcut



Processing Data with KQL

Ingest from query

Command	If table exists	If table doesn't exist
.set	The command fails	The table is created and data is ingested
.append	Data is appended to the table	The command fails
.set-or-append	Data is appended to the table	The table is created and data is ingested
.set-or-replace	Data replaces the data in the table	The table is created and data is ingested

Processing Data with KQL

Ingest from query

Create and update table from query source

```
.set RecentErrors <|
   LogsTable
   | where Level == "Error" and Timestamp > now() - time(1h)
```

Append data to a table

```
.append OldExtents with(tags='["TagA","TagB"]') <|
   MyExtents
   | where CreatedOn < now() - time(30d)
   | project ExtentId
```

Processing Data with KQL

Ingest from query

Create or append a table with possibly existing tagged data

```
.set-or-append async OldExtents with(tags='["ingest-by:myTag"]', ingestIfNotExists='["myTag"]') <|
   MyExtents
   | where CreatedOn < now() - time(30d)
   | project ExtentId
```

Create table or replace data with associated data

```
.set-or-replace async OldExtents with(tags='["ingest-by:myTag"]', ingestIfNotExists='["myTag"]') <|
   MyExtents
   | where CreatedOn < now() - time(30d)
   | project ExtentId
```

Processing Data with KQL

.show data operations command

- Returns a table with data operations that reached a final state
- Available for 30 days
- Database Admin or Database Monitor permission

Timestamp	Database	Table	ClientActivityId	OperationKind	OriginalSize	ExtentSize	RowCount	ExtentCount	TotalCpu	Duration	Principal	Properties
2024-07-18 15:21:10.5432134	TestLogs	UTResults	DM.IngestionExecutor;abcd1234-1234-1234-abcd-1234abcdce;1	UpdatePolicy	100,829	75,578	279	1	00:00:00.2656250	00:00:28.9101535	aadapp=xxx	{"SourceTable": "UTLogs"}
2024-07-18 15:21:12.9481819	TestLogs	UTLogs	DM.IngestionExecutor;abcd1234-1234-1234-abcd-1234abcdce;1	BatchIngest	1,045,027,298	123,067,947	1,688,705	2	00:00:22.9843750	00:00:29.9745733	aadapp=xxx	{"Format": "Csv","NumberOfInputStreams":2}
2024-07-18 15:21:16.1095441	KustoAuto	IncidentKustoGPTSummary	cdef12345-6789-ghij-0123-klmn45678	SetOrAppend	1,420	3,190	1	1	00:00:00.0156250	00:00:00.0638211	aaduser=xxx	

Processing Data with KQL

.export to table command

- Exporting data to an external table
- Table Admin permission

```
.export to table ExternalBlob <| T
```

Non-partitioned external table

```
.create external table PartitionedExternalBlob (Timestamp:datetime, CustomerName:string)
kind=blob
partition by (CustomerName:string=CustomerName, Date:datetime=startofday(Timestamp))
pathformat = ("CustomerName=" CustomerName "/" datetime_pattern("yyyy/MM/dd", Date))
dataformat=csv
(
   h@'http://storageaccount.blob.core.windows.net/container1;secretKey'
)
```

```
.export to table PartitionedExternalBlob <| T
```

Partitioned external table

Processing Data with KQL

Materialized views

- *Aggregation* query over a source table or another materialized view
- More performant than running the aggregation directly over the source table

Materialized part

- Already processed records from the source table
- Single record per *group by*

+

Delta

- Newly ingested records from the source table that are yet to be processed

Processing Data with KQL

Materialized views

Query the entire view

- Most up-to-date results
- Performance may be suboptimal

```
ViewName
```

Query the materialized part only

- materialized_view() function
- Specify max latency

```
materialized_view("ViewName")
```

Freshness! vs. **Performance!**

Windowing Functions in KQL

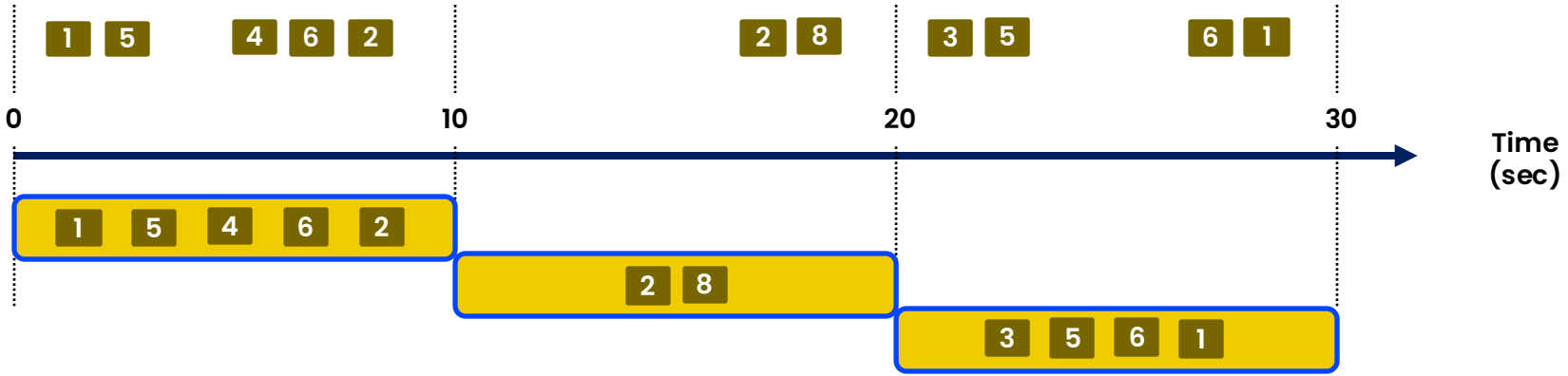
- Set-based operations over a subset of events

Time is the fundamental requirement when working with streaming data

Windowing Functions in KQL

Tumbling window

Fixed-sized, non-overlapping continuous intervals

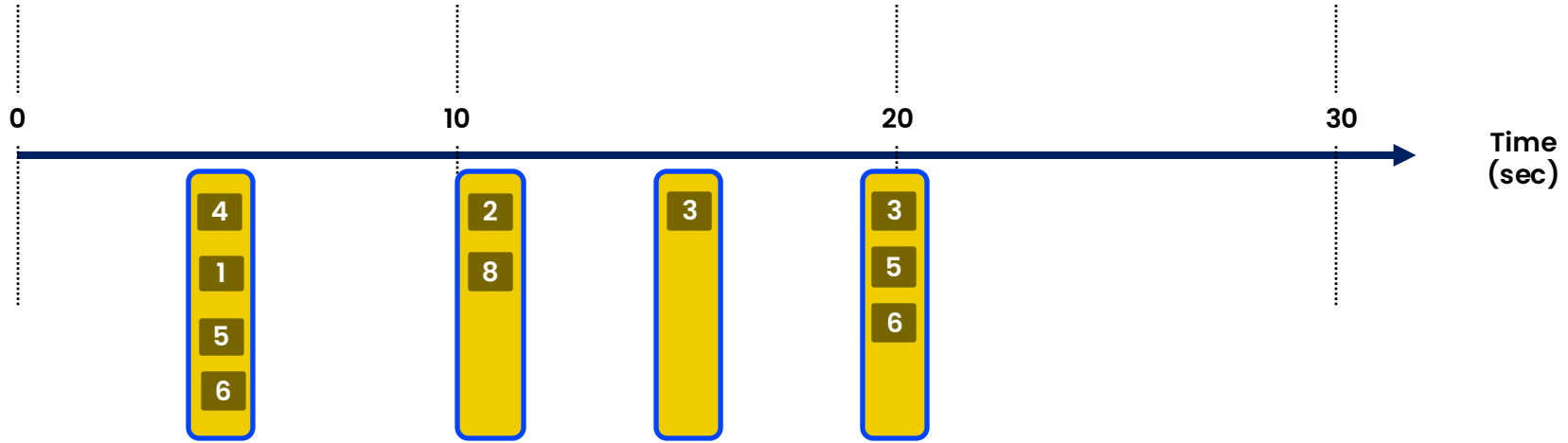


Count of events every 10 seconds

Windowing Functions in KQL

Snapshot window

Group events with the same timestamp

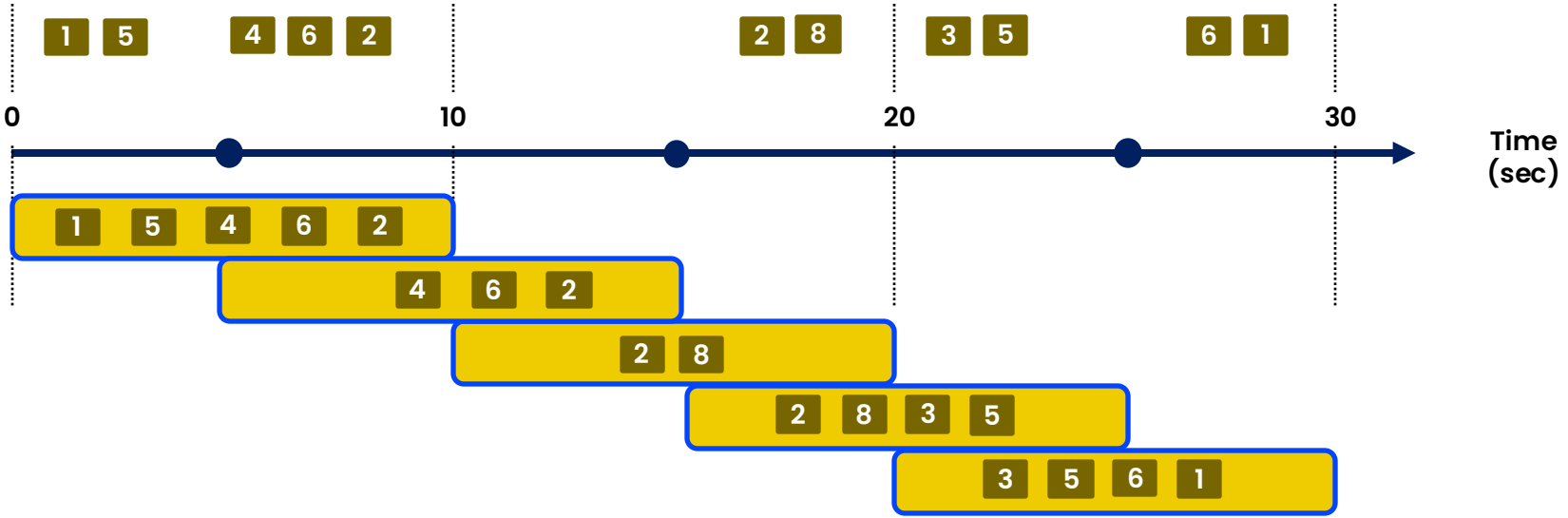


Count of events that occur at exactly the same time

Windowing Functions in KQL

Hopping window

Fixed-sized, overlapping continuous intervals

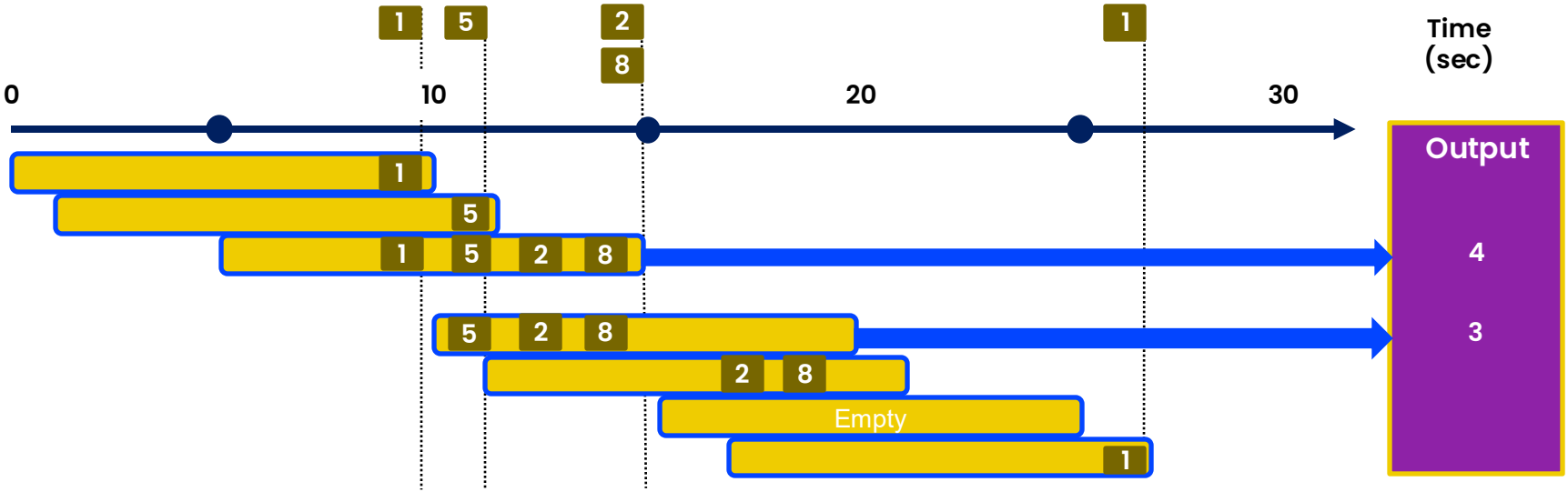


Every 5 seconds, give me the count of events over the last 10 seconds

Windowing Functions in KQL

Sliding window

Logically consider all possible windows of a given length

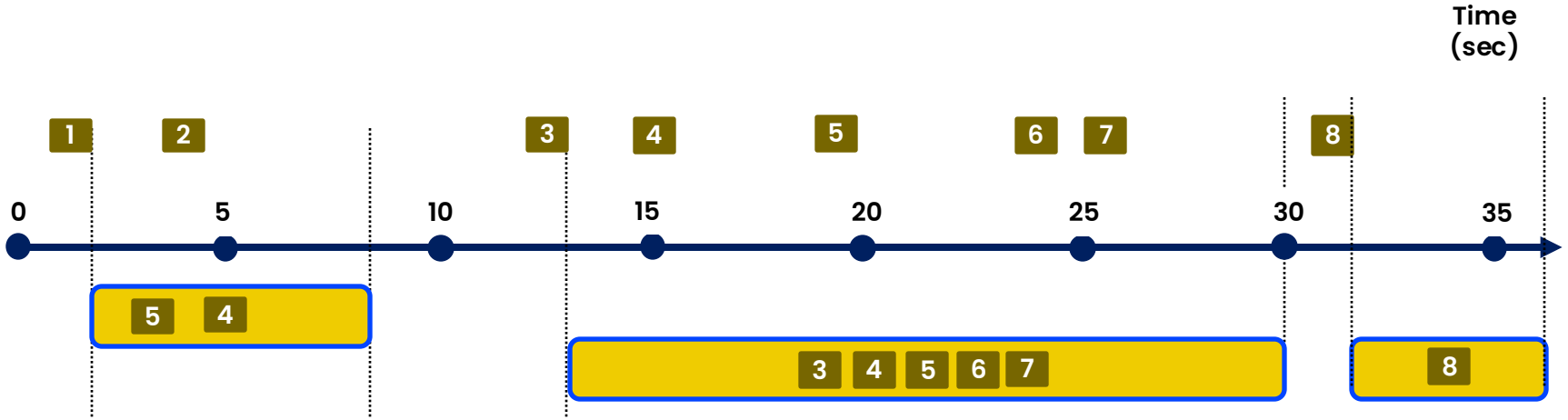


Alert me whenever an event occurs more than 2 times in under 10 seconds

Windowing Functions in KQL

Session window

Events arriving at similar times, filtering out periods with no data



Count of events that occur within 5 seconds to each other

Windowing Functions in KQL

Window Type	Fixed	Overlapping	Scenario
Tumbling	✅	❌	Simple time bins (e.g. 5 sec)
Snapshot	✅	❌	State capture at regular intervals
Hopping	✅	✅	Rolling summaries with overlap
Sliding	✅	✅	Real-time rolling metrics
Session	❌	❌	Grouping by inactivity gaps

Windowing Functions in KQL

Window Type	Example	Description
Tumbling	SensorEvents | where Timestamp > ago(1d) | summarize AvgTemp = avg(Temperature) by bin(Timestamp, 1h)	Counts or aggregates in distinct 1-hour intervals
Snapshot	let snapshot_times = range t from ago(1h) to now() step 15m; snapshot_times | join kind=inner (SensorEvents | summarize LatestReading = arg_max(Timestamp, *) by DeviceId) on $left.t between (Timestamp - 1m) .. (Timestamp + 1m)	Captures a view of most recent values at fixed points
Hopping	SensorEvents | where Timestamp > ago(6h) | summarize AvgTemp = avg(Temperature) by bin(Timestamp, 15m), hop = 1h	Averages over 1-hour windows, calculated every 15 minutes
Sliding	SensorEvents | where Timestamp > ago(1d) | summarize count() by sliding_window(1h)	Always shows the count within the last hour, continuously moving
Session	SensorEvents | sort by DeviceId, Timestamp | extend SessionId = row_window_session(Timestamp, 30m, DeviceId) | summarize SessionStart = min(Timestamp), SessionEnd = max(Timestamp), EventCount = count() by DeviceId, SessionId	Groups all events from the same device into “sessions” where events are no more than 30 mins apart

Window Functions in KQL

Window functions != Windowing functions

- Operate on multiple rows in a row set at a time
- Require serialization of rows! (a specific order)
- Serialization
  - *Serialize operator – "freezes" the order of rows in an arbitrary way*
  - *Sort operator – forces a particular order*
- Serialization is expensive!

Window Functions in KQL

Window functions

- next()
- prev()
- row_cumsum()
- row_number()
- row_rank_dense()
- row_rank_min()
- row_window_session()

DEMO

- Process data with Eventstream
- Process data with KQL
- KQL window functions

Q&A

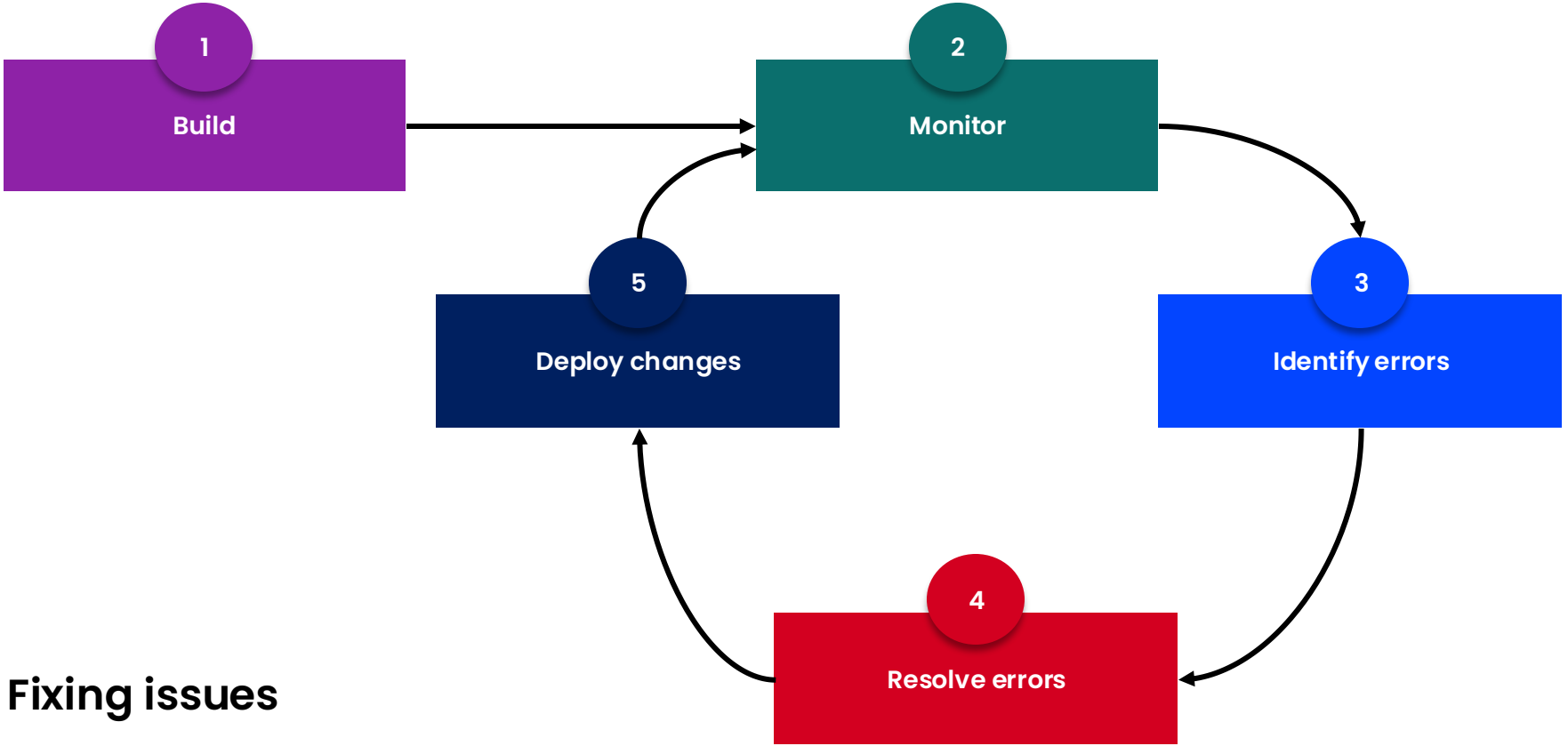
Monitor and Optimize an Analytics Solution

Monitor Fabric Items

Monitor Fabric items

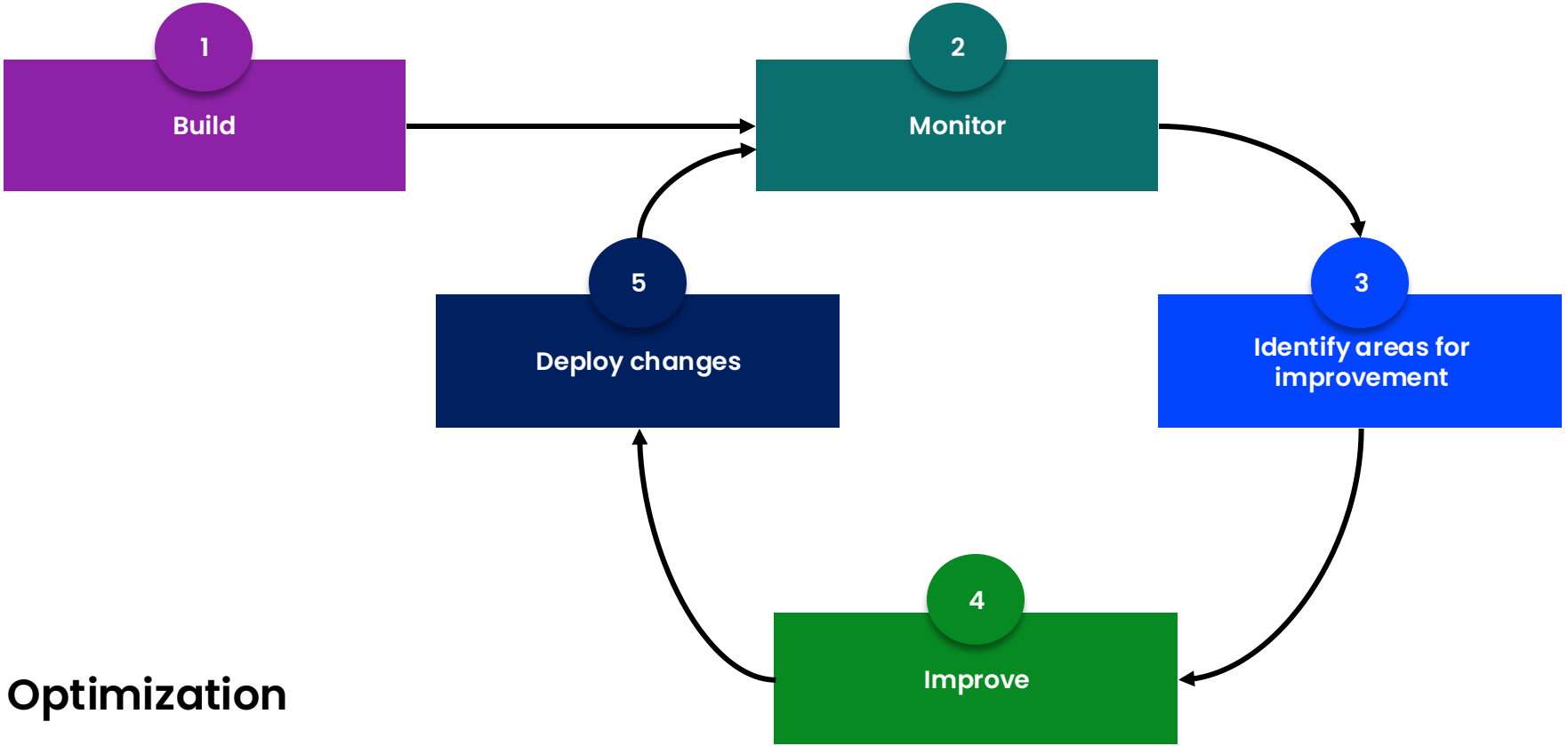
- Data ingestion
- Data transformation
- Semantic model refresh
- Configure alerts

Why Does it Matter?



Fixing issues

Why Does it Matter?



Optimization

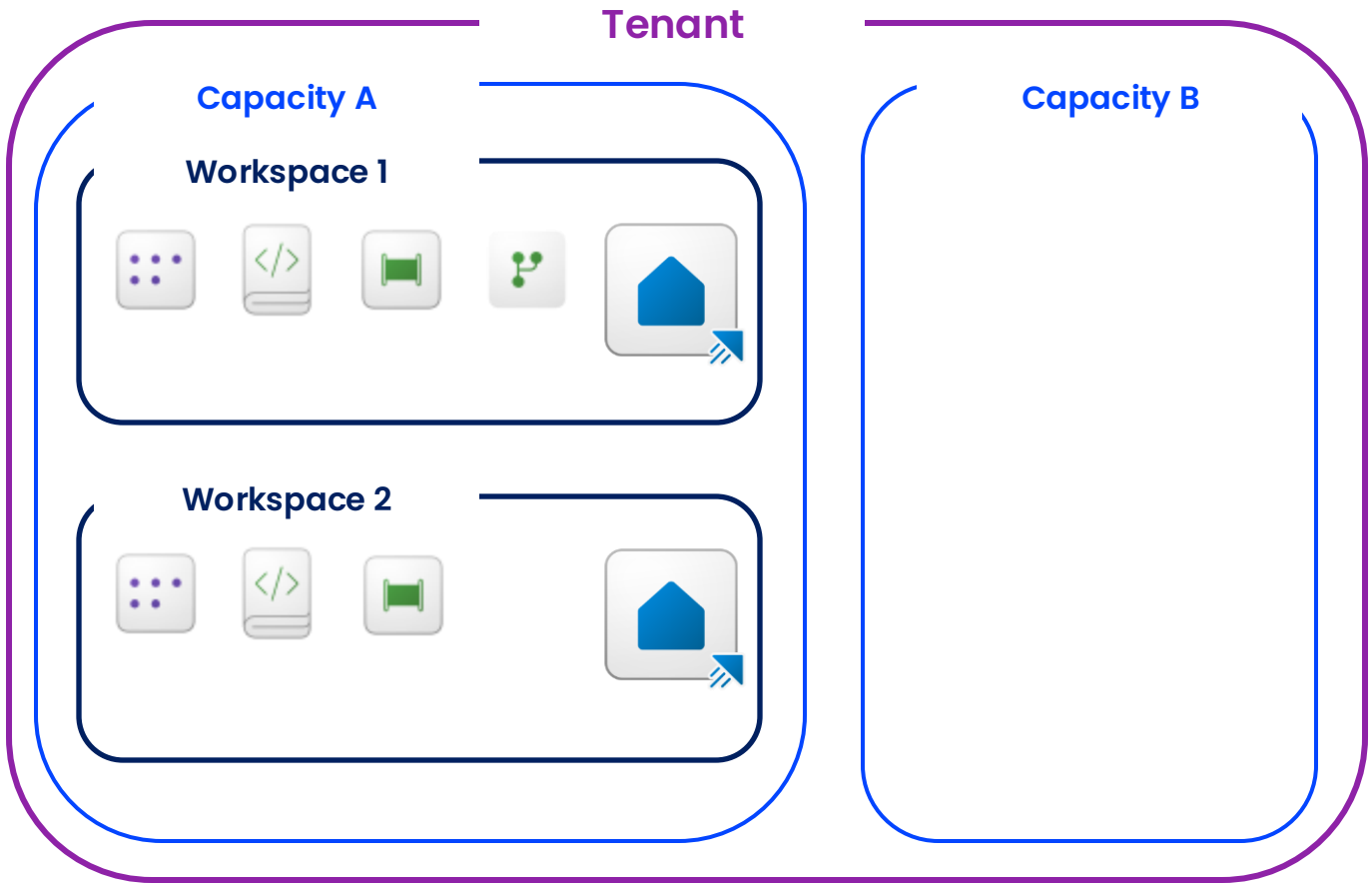
Different Scopes of Monitoring

Monitor Hub

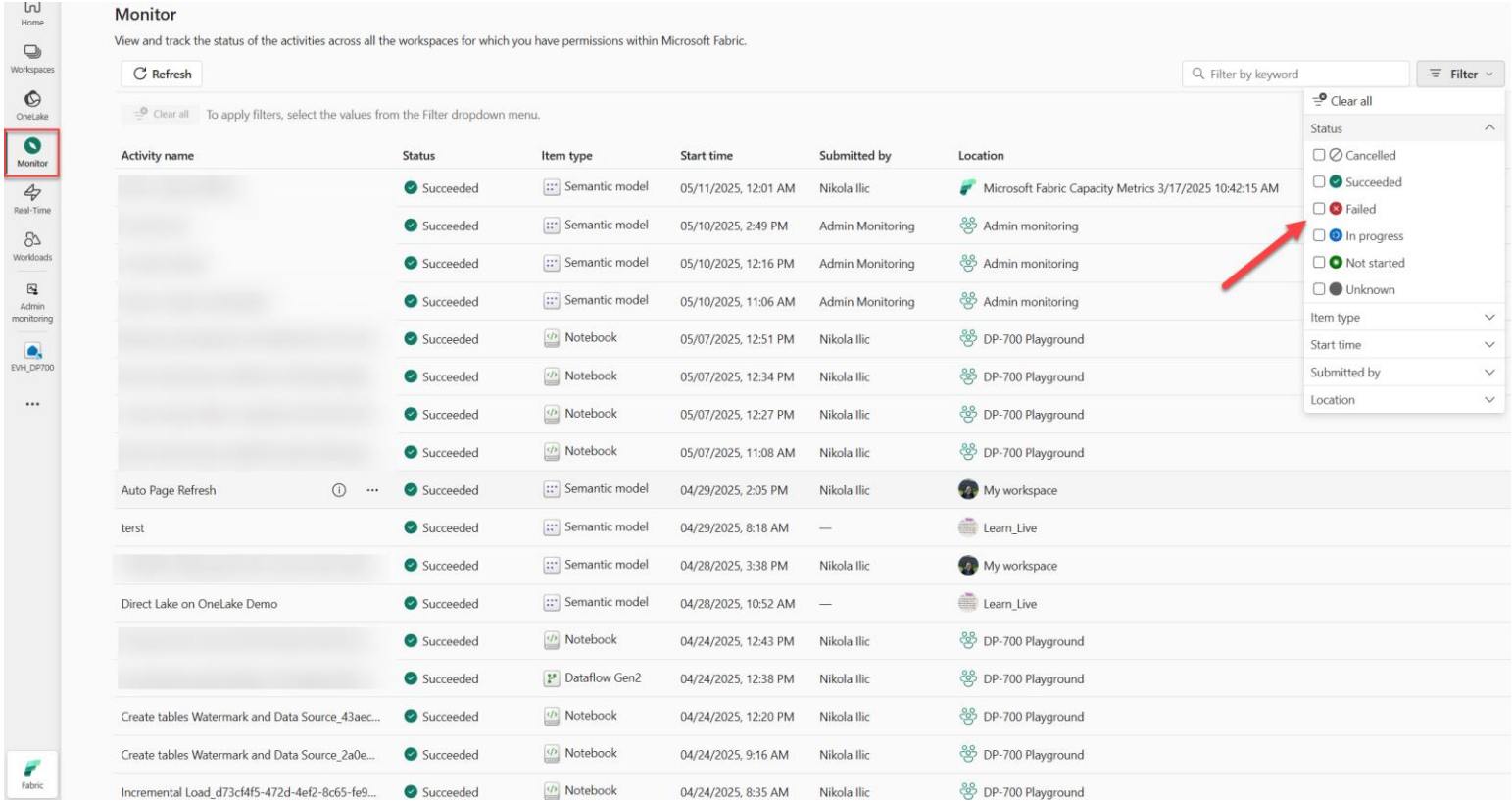
Capacity Metrics App

Workspace Monitoring Eventhouse

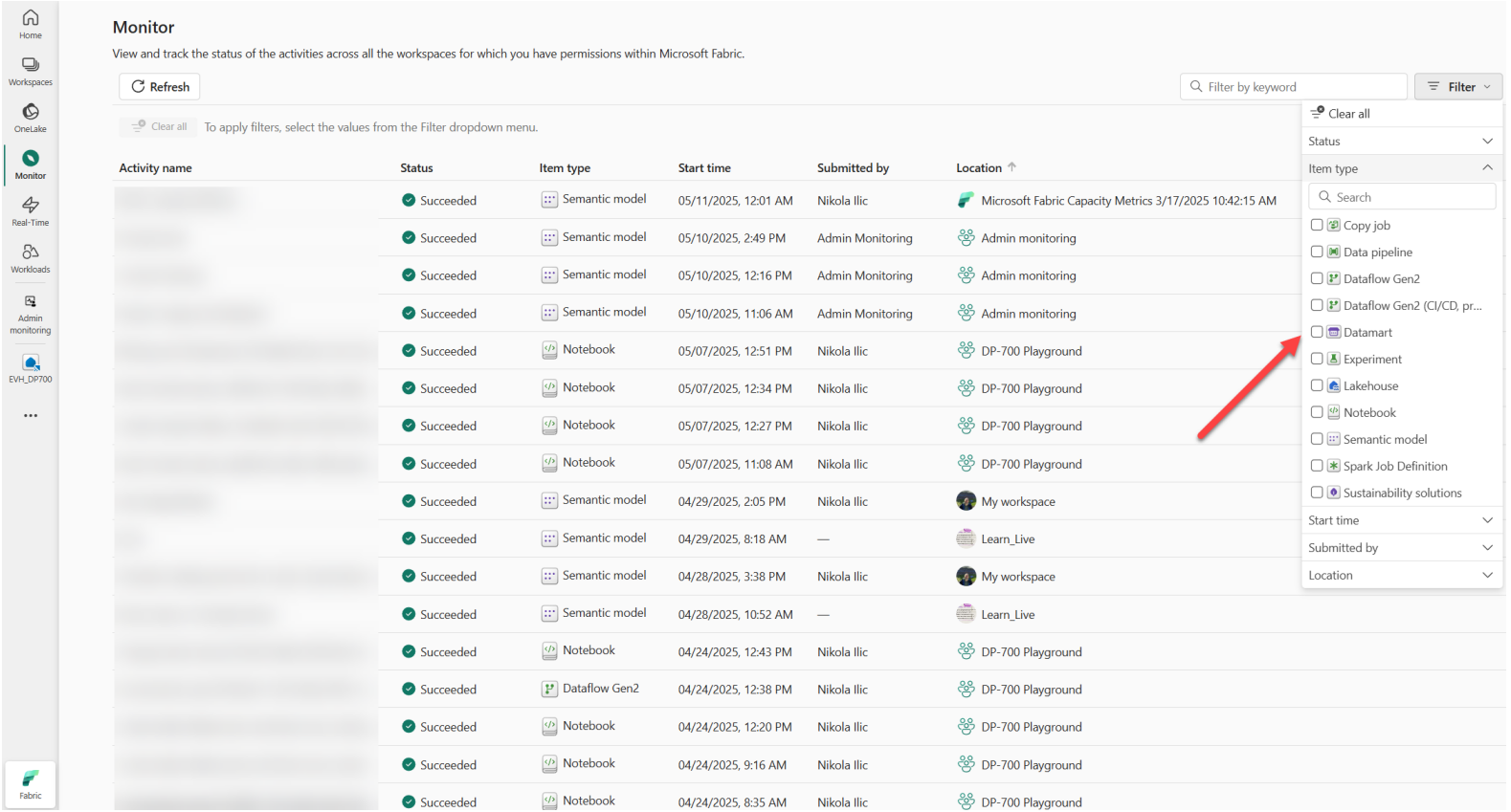
Items' previous runs details

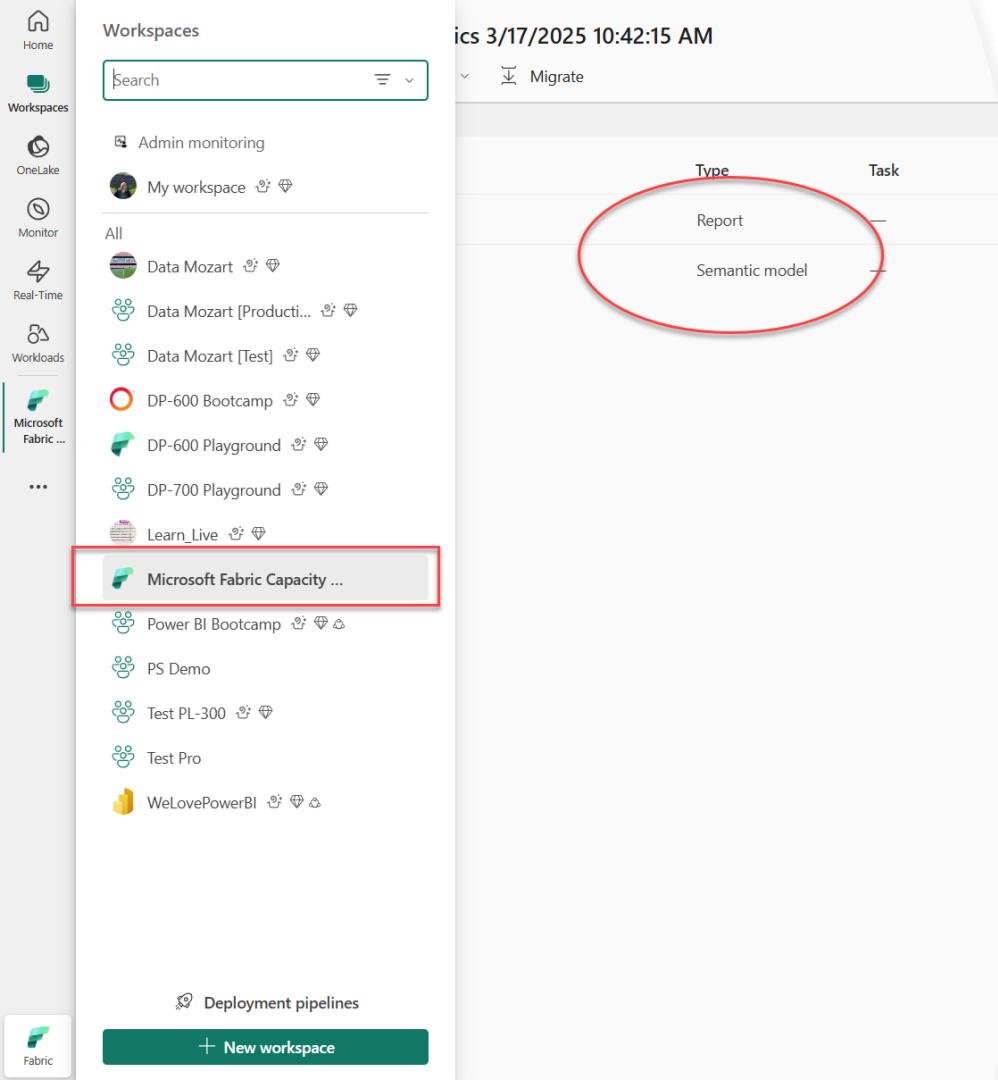


Monitor Hub


Home
Workspaces
OneLake
Monitor
Real-Time
Workloads
Admin monitoring
EVH_DP700
Fabric
Monitor
View and track the status of the activities across all the workspaces for which you have permissions within Microsoft Fabric.
Refresh
Filter by keyword
Filter
Clear all
To apply filters, select the values from the Filter dropdown menu.
Activity name
Status
Item type
Start time
Submitted by
Location
Succeeded Semantic model 05/11/2025, 12:01 AM Nikola Ilic Microsoft Fabric Capacity Metrics 3/17/2025 10:42:15 AM
Succeeded Semantic model 05/10/2025, 2:49 PM Admin Monitoring Admin monitoring
Succeeded Semantic model 05/10/2025, 12:16 PM Admin Monitoring Admin monitoring
Succeeded Semantic model 05/10/2025, 11:06 AM Admin Monitoring Admin monitoring
Succeeded Notebook 05/07/2025, 12:51 PM Nikola Ilic DP-700 Playground
Succeeded Notebook 05/07/2025, 12:34 PM Nikola Ilic DP-700 Playground
Succeeded Notebook 05/07/2025, 12:27 PM Nikola Ilic DP-700 Playground
Succeeded Notebook 05/07/2025, 11:08 AM Nikola Ilic DP-700 Playground
Auto Page Refresh Succeeded Semantic model 04/29/2025, 2:05 PM Nikola Ilic My workspace
terst Succeeded Semantic model 04/29/2025, 8:18 AM — Learn_Live
Succeeded Semantic model 04/28/2025, 3:38 PM Nikola Ilic My workspace
Direct Lake on OneLake Demo Succeeded Semantic model 04/28/2025, 10:52 AM — Learn_Live
Succeeded Notebook 04/24/2025, 12:43 PM Nikola Ilic DP-700 Playground
Succeeded Dataflow Gen2 04/24/2025, 12:38 PM Nikola Ilic DP-700 Playground
Create tables Watermark and Data Source_43aec... Succeeded Notebook 04/24/2025, 12:20 PM Nikola Ilic DP-700 Playground
Create tables Watermark and Data Source_2a0e... Succeeded Notebook 04/24/2025, 9:16 AM Nikola Ilic DP-700 Playground
Incremental Load_d73cf4f5-472d-4ef2-8c65-fe9... Succeeded Notebook 04/24/2025, 8:35 AM Nikola Ilic DP-700 Playground
Clear all
Status
Cancelled
Succeeded
Failed
In progress
Not started
Unknown
Item type
Start time
Submitted by
Location

Monitor Hub

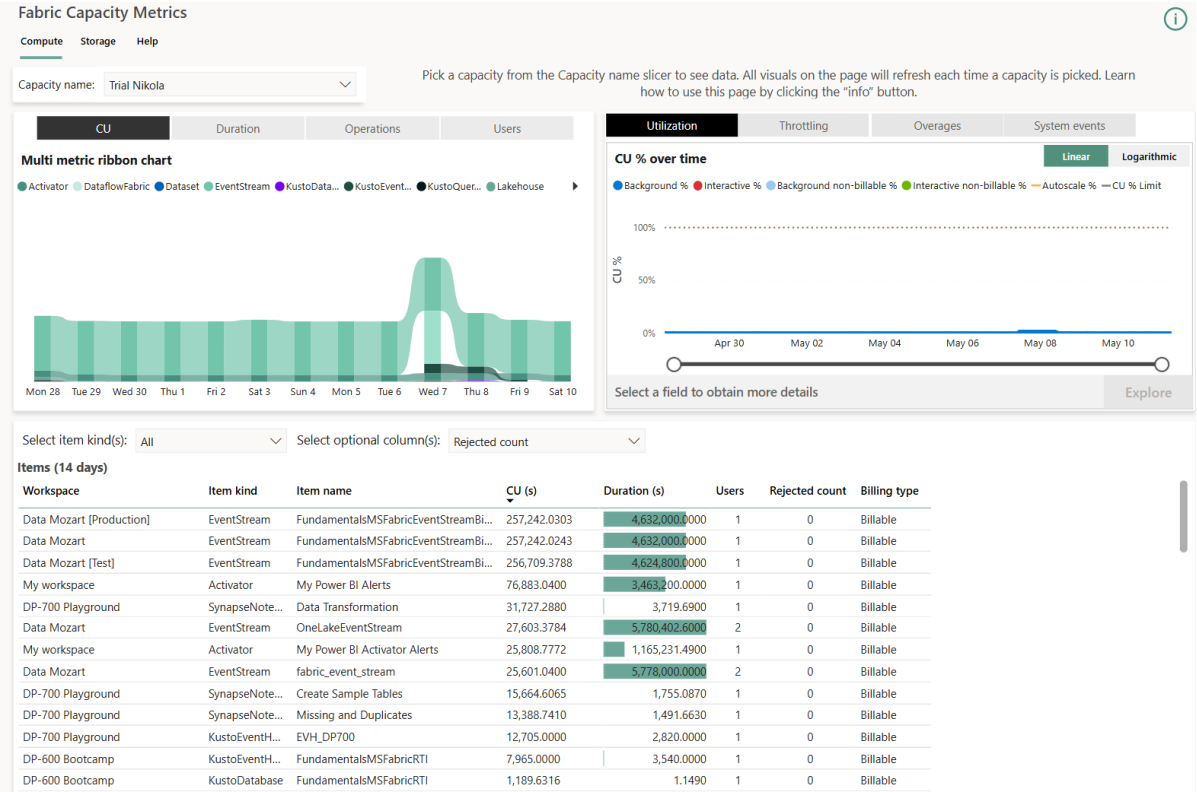

Home
Workspaces
OneLake
Monitor
Real-Time
Workloads
Admin monitoring
EVH_DP700
Fabric
Monitor
View and track the status of the activities across all the workspaces for which you have permissions within Microsoft Fabric.
Refresh
Filter by keyword
Filter
Clear all
To apply filters, select the values from the Filter dropdown menu.
Activity name
Status
Item type
Start time
Submitted by
Location
Succeeded
Semantic model
05/11/2025, 12:01 AM
Nikola Ilic
Microsoft Fabric Capacity Metrics 3/17/2025 10:42:15 AM
Succeeded
Semantic model
05/10/2025, 2:49 PM
Admin Monitoring
Admin monitoring
Succeeded
Semantic model
05/10/2025, 12:16 PM
Admin Monitoring
Admin monitoring
Succeeded
Semantic model
05/10/2025, 11:06 AM
Admin Monitoring
Admin monitoring
Succeeded
Notebook
05/07/2025, 12:51 PM
Nikola Ilic
DP-700 Playground
Succeeded
Notebook
05/07/2025, 12:34 PM
Nikola Ilic
DP-700 Playground
Succeeded
Notebook
05/07/2025, 12:27 PM
Nikola Ilic
DP-700 Playground
Succeeded
Notebook
05/07/2025, 11:08 AM
Nikola Ilic
DP-700 Playground
Succeeded
Semantic model
04/29/2025, 2:05 PM
Nikola Ilic
My workspace
Succeeded
Semantic model
04/29/2025, 8:18 AM
—
Learn_Live
Succeeded
Semantic model
04/28/2025, 3:38 PM
Nikola Ilic
My workspace
Succeeded
Semantic model
04/28/2025, 10:52 AM
—
Learn_Live
Succeeded
Notebook
04/24/2025, 12:43 PM
Nikola Ilic
DP-700 Playground
Succeeded
Dataflow Gen2
04/24/2025, 12:38 PM
Nikola Ilic
DP-700 Playground
Succeeded
Notebook
04/24/2025, 12:20 PM
Nikola Ilic
DP-700 Playground
Succeeded
Notebook
04/24/2025, 9:16 AM
Nikola Ilic
DP-700 Playground
Succeeded
Notebook
04/24/2025, 8:35 AM
Nikola Ilic
DP-700 Playground
Clear all
Status
Item type
Search
Copy job
Data pipeline
Dataflow Gen2
Dataflow Gen2 (CI/CD, pr...
Datamart
Experiment
Lakehouse
Notebook
Semantic model
Spark Job Definition
Sustainability solutions
Start time
Submitted by
Location



Capacity Metrics App

- **Download the app from the Marketplace and connect it to the capacity**
- **A special Power BI report and semantic model**

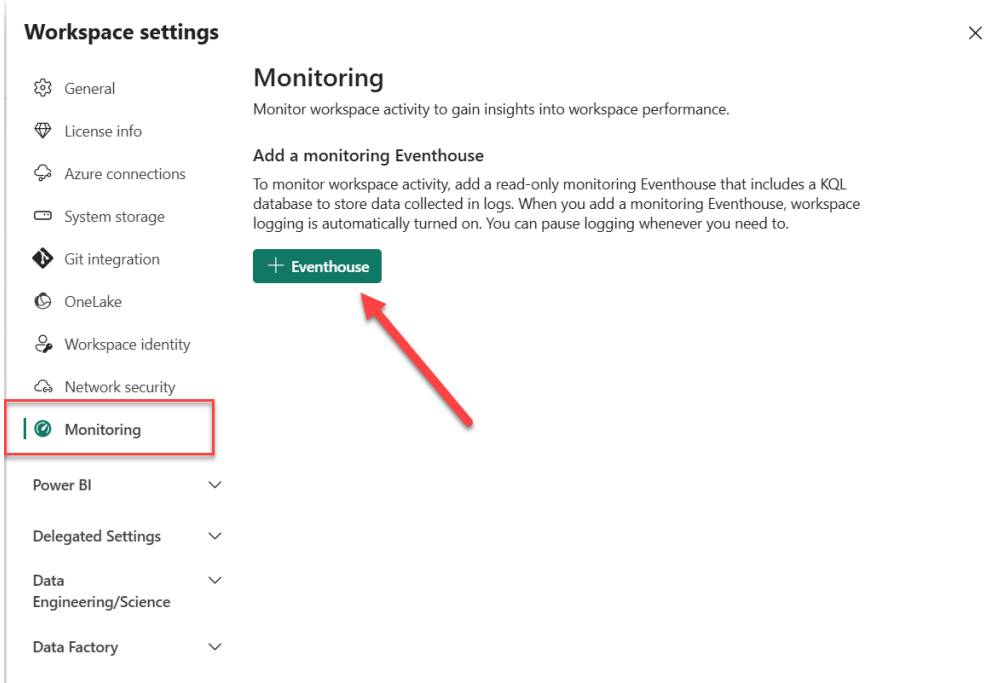
Capacity Metrics App Report



Workspace	Item kind	Item name	CU (s)	Duration (s)	Users	Rejected count	Billing type
Data Mozart [Production]	EventStream	FundamentalsMSFabricEventStreamBi...	257,242.0303	4,632,000.0000	1	0	Billable
Data Mozart	EventStream	FundamentalsMSFabricEventStreamBi...	257,242.0243	4,632,000.0000	1	0	Billable
Data Mozart [Test]	EventStream	FundamentalsMSFabricEventStreamBi...	256,709.3788	4,624,800.0000	1	0	Billable
My workspace	Activator	My Power BI Alerts	76,883.0400	3,463,200.0000	1	0	Billable
DP-700 Playground	SynapseNote...	Data Transformation	31,727.2880	3,719.6900	1	0	Billable
Data Mozart	EventStream	OneLakeEventStream	27,603.3784	5,780,402.6000	2	0	Billable
My workspace	Activator	My Power BI Activator Alerts	25,808.7772	1,165,231.4900	1	0	Billable
Data Mozart	EventStream	fabric_event_stream	25,601.0400	5,778,000.0000	2	0	Billable
DP-700 Playground	SynapseNote...	Create Sample Tables	15,664.6065	1,755.0870	1	0	Billable
DP-700 Playground	SynapseNote...	Missing and Duplicates	13,388.7410	1,491.6630	1	0	Billable
DP-700 Playground	KustoEventH...	EVH_DP700	12,705.0000	2,820.0000	1	0	Billable
DP-600 Bootcamp	KustoEventH...	FundamentalsMSFabricRTI	7,965.0000	3,540.0000	1	0	Billable
DP-600 Bootcamp	KustoDatabase	FundamentalsMSFabricRTI	1,189.6316	1.1490	1	0	Billable

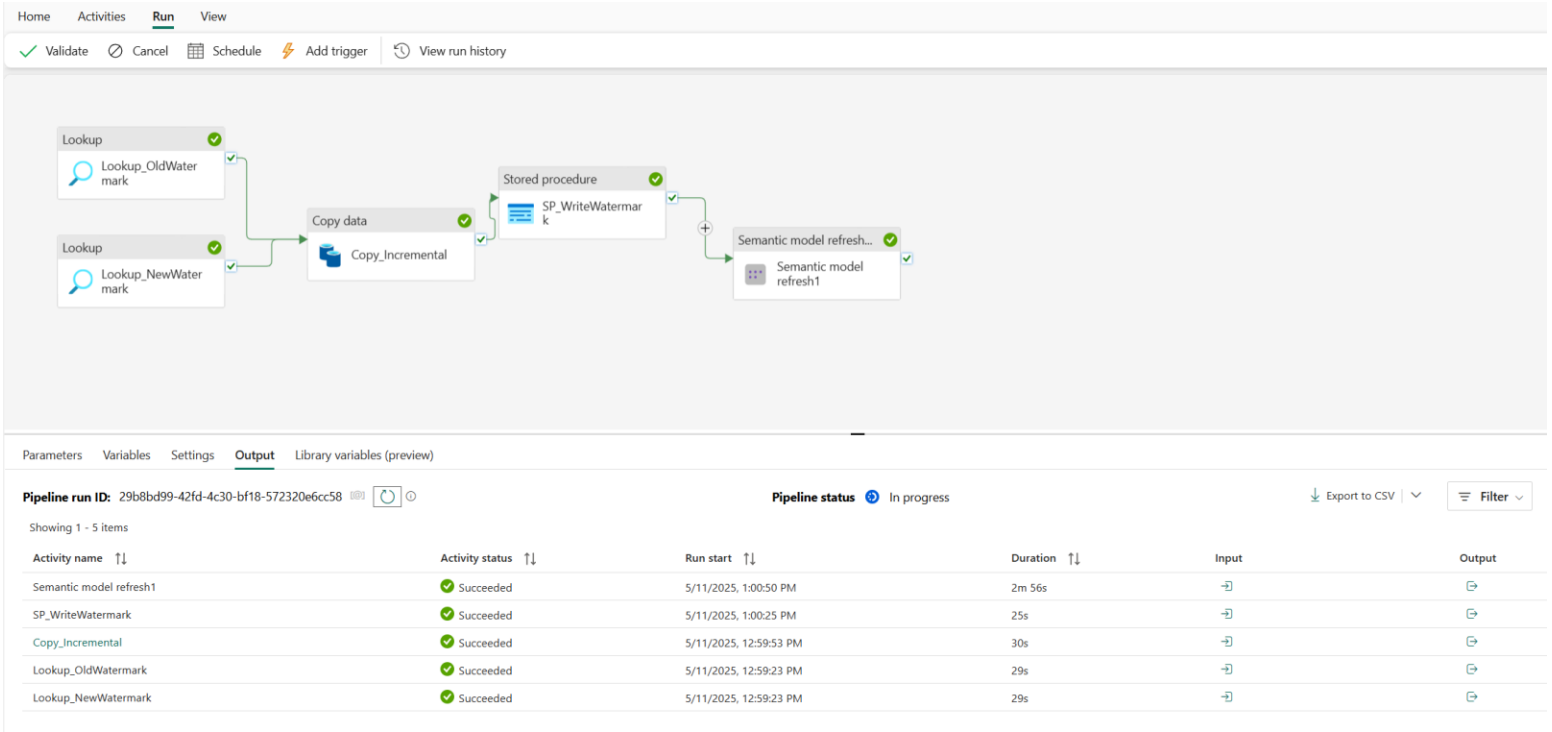
Workspace Monitoring Eventhouse

- **Creates special RTI items**
  - Eventhouse
  - Eventstream
  - KQL database
- **Read-only database**
- **At least the Contributor role**
- **Historical log analysis + real-time data streaming**
- **Currently supported items**
  - GraphQL
  - Eventhouse
  - Mirrored database
  - Semantic models



Item-level Monitoring

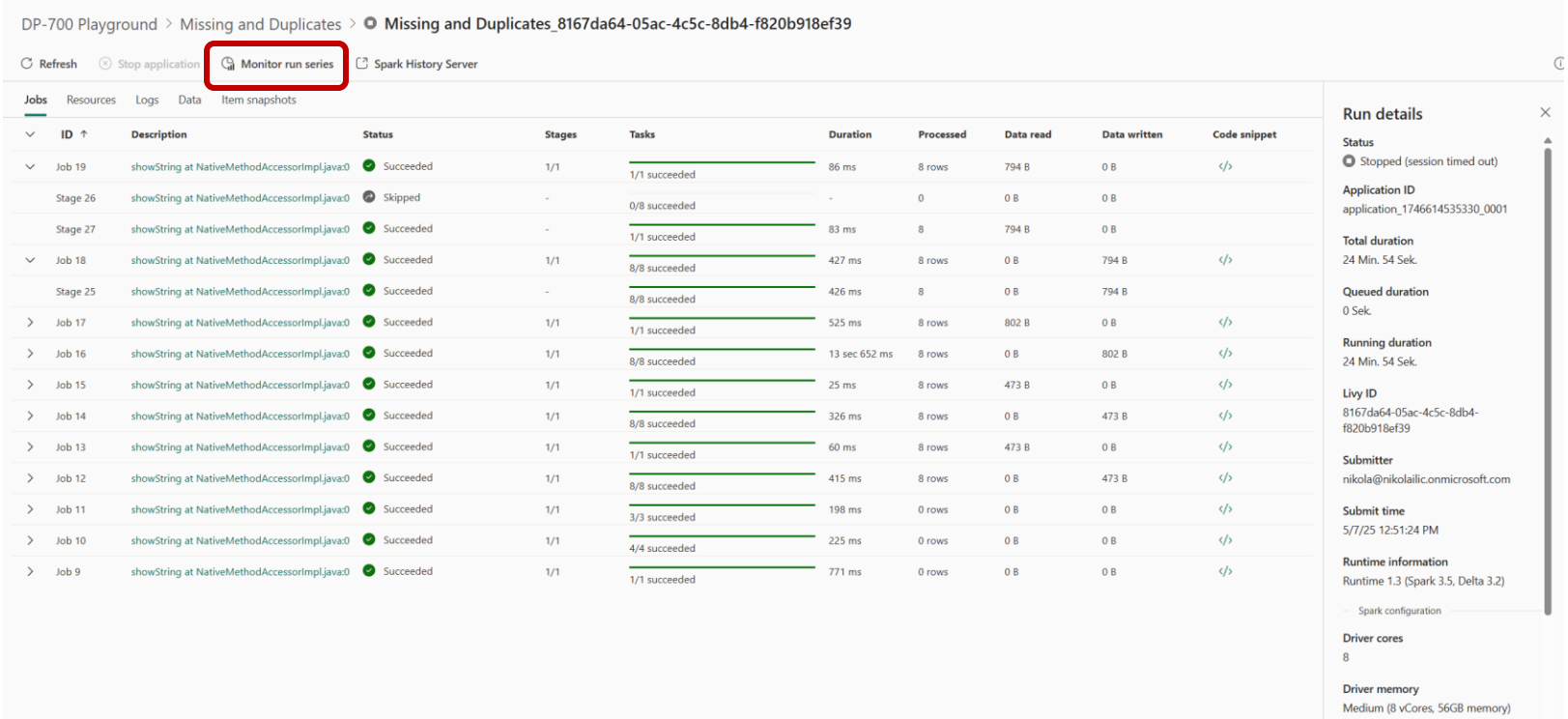
Pipeline



Activity name	Activity status	Run start	Duration	Input	Output
Semantic model refresh1	Succeeded	5/11/2025, 1:00:50 PM	2m 56s		
SP_WriteWatermark	Succeeded	5/11/2025, 1:00:25 PM	25s		
Copy_Incremental	Succeeded	5/11/2025, 12:59:53 PM	30s		
Lookup_OldWatermark	Succeeded	5/11/2025, 12:59:23 PM	29s		
Lookup_NewWatermark	Succeeded	5/11/2025, 12:59:23 PM	29s		

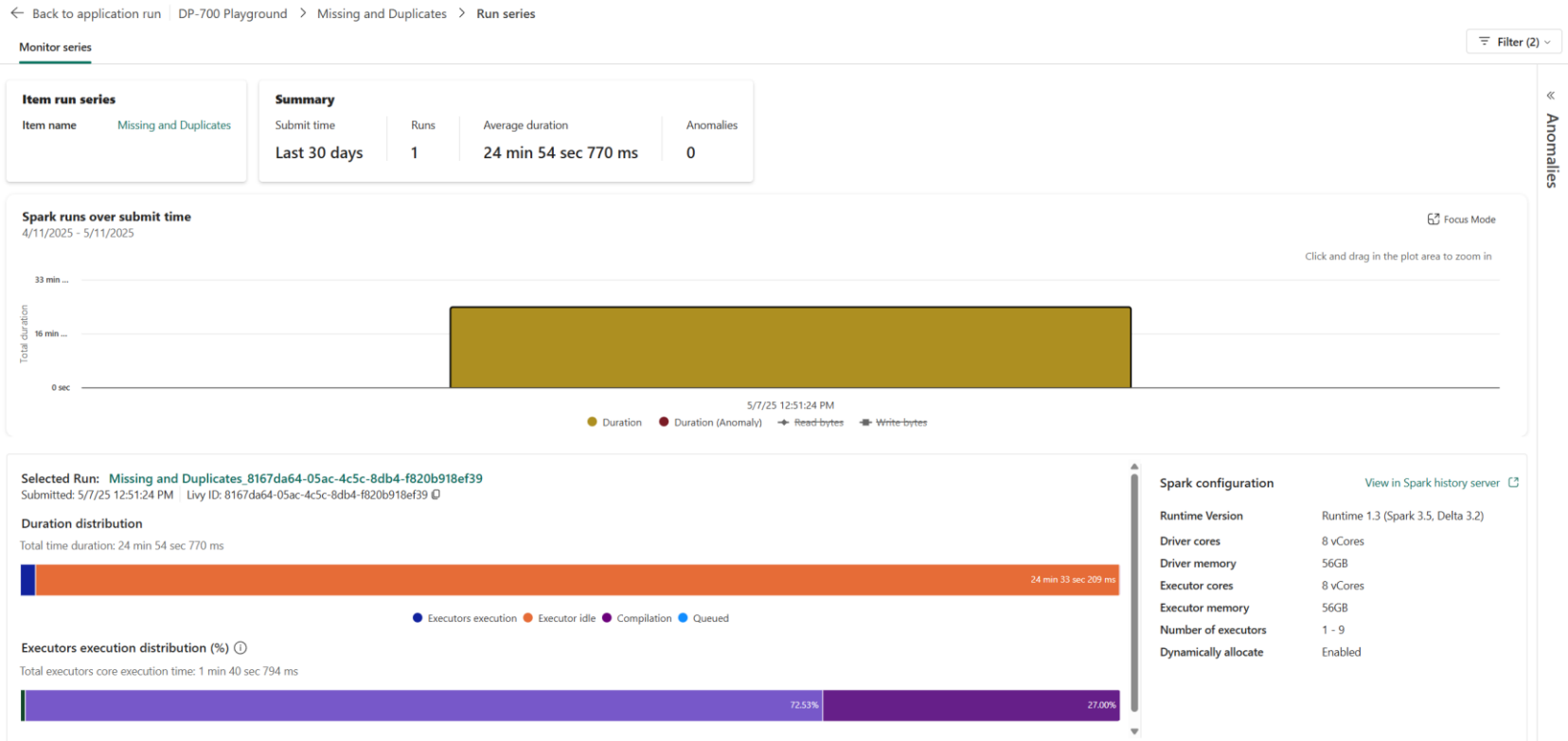
Item-level Monitoring

Notebook



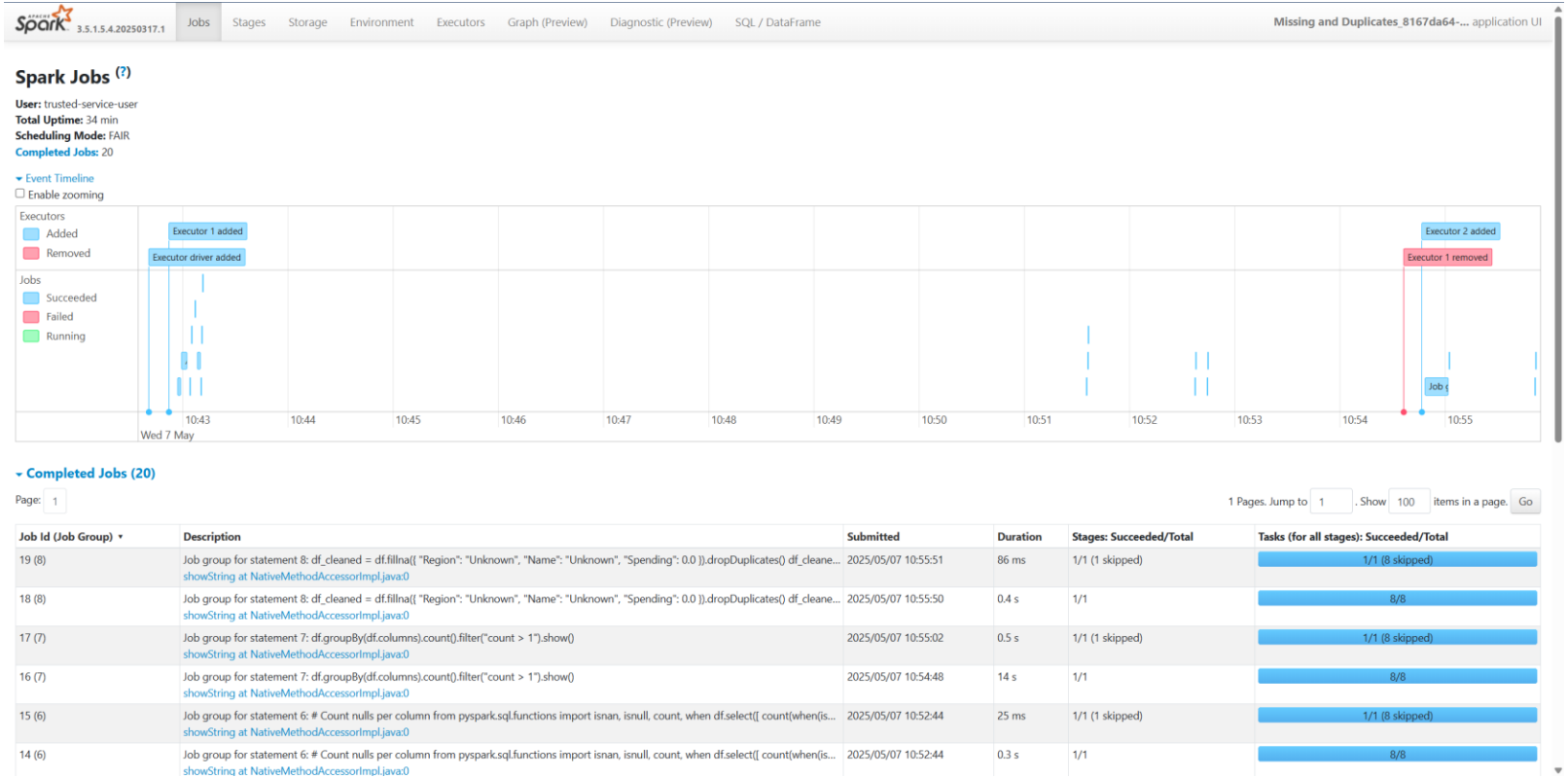
Item-level Monitoring

Notebook



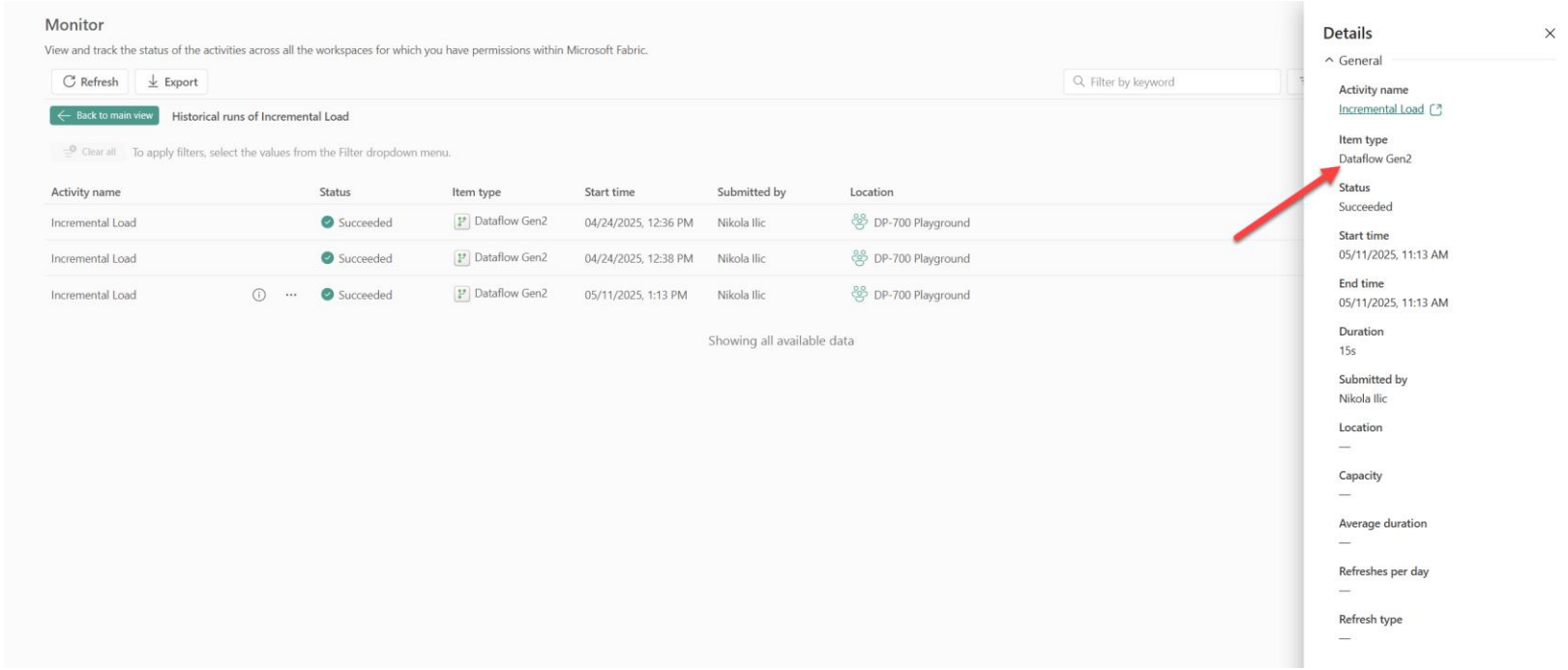
Item-level Monitoring

Notebook



Item-level Monitoring

Dataflows



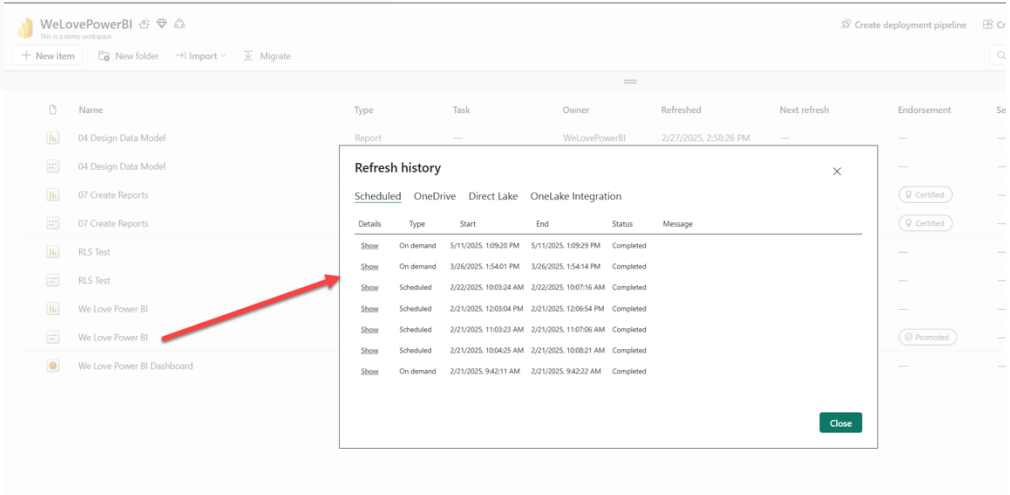
Item-level Monitoring

Semantic model

WeLovePowerBI > We Love Power BI > Refresh ID: 7cf25f21-b2e3-0de0-4af0-306a33af3e02

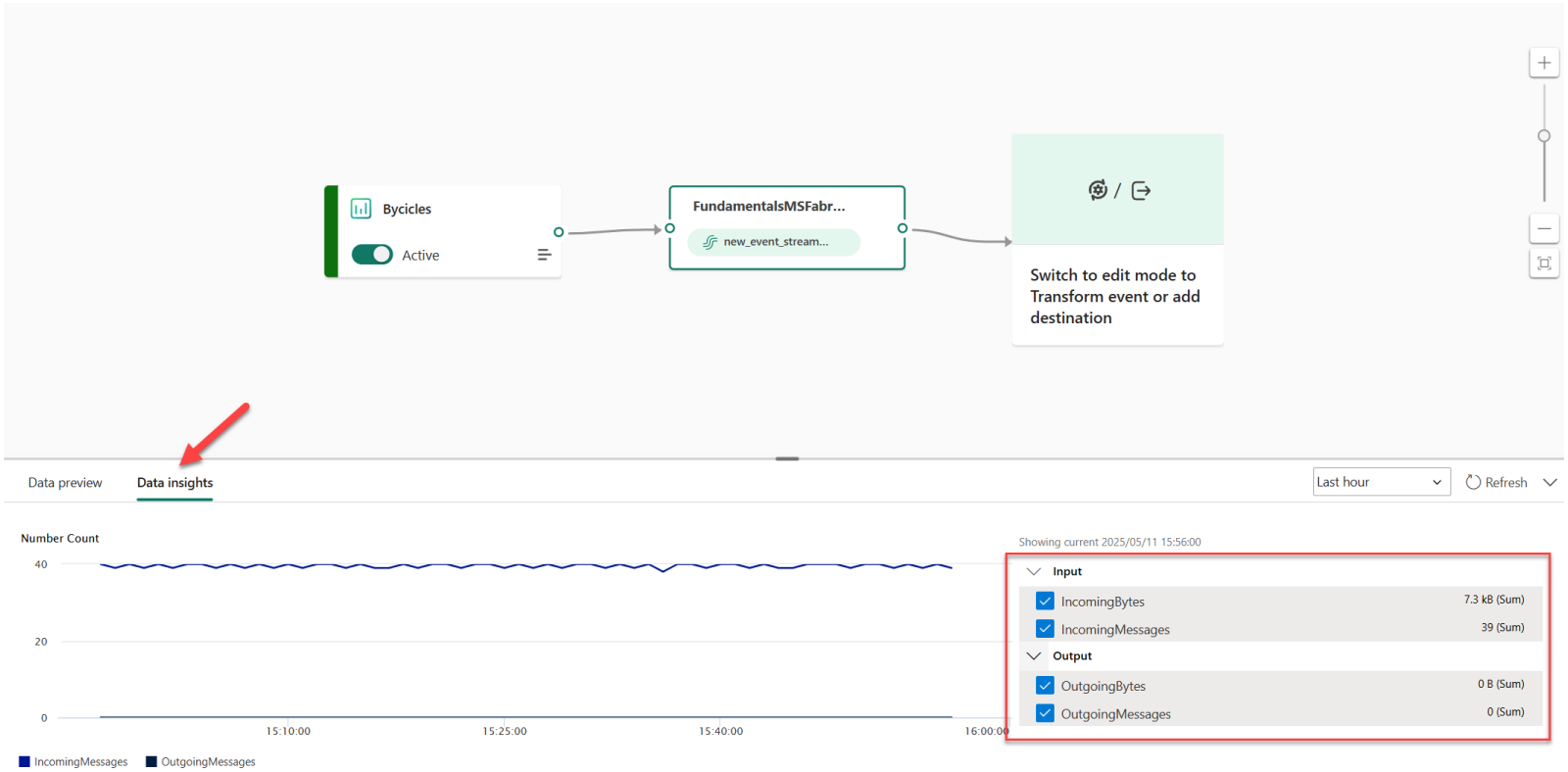
View details

Refresh attempt	Type	Start time	End time	Duration	Status	Execution details
1	Data	5/11/2025, 1:09:21 PM	5/11/2025, 1:09:26 PM	5s	Completed	Show
1	Query Cache	5/11/2025, 1:09:26 PM	5/11/2025, 1:09:26 PM	Less than 1s	Completed	Show



Item-level Monitoring

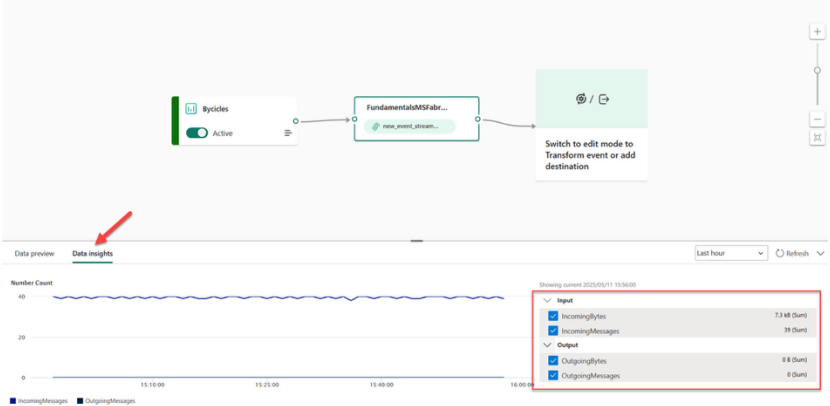
Eventstream



Item-level Monitoring

Eventstream

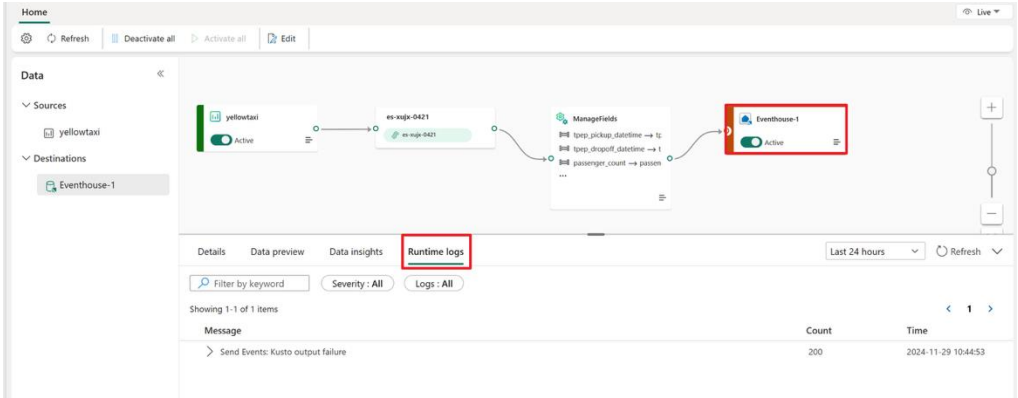
- Incoming messages – number of events sent to the eventstream in the specified period
- Outgoing messages – number of events going out from the eventstream
- Incoming bytes
- Outgoing bytes



Item-level Monitoring

Eventstream

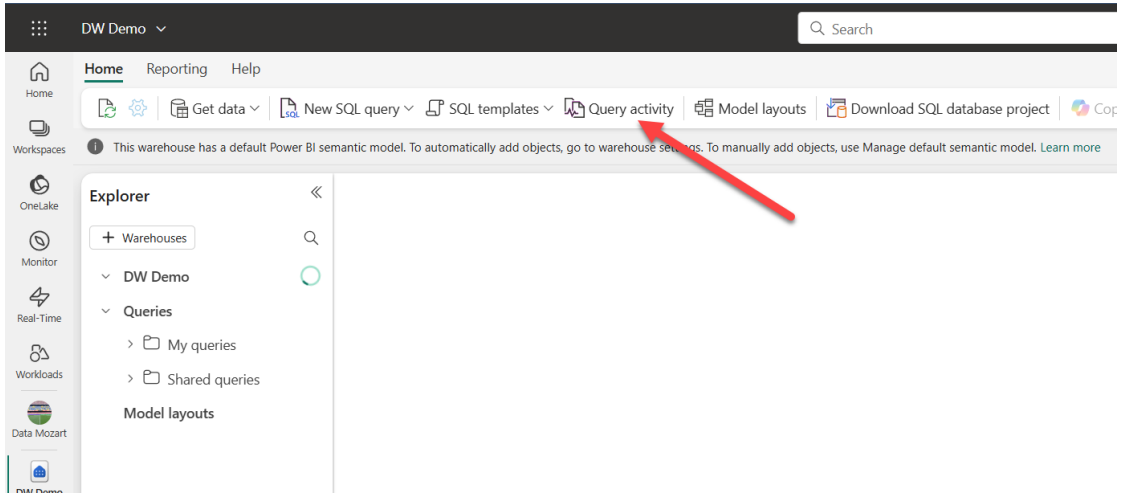
- Detailed logs
  - Warning
  - Error
  - Information



Item-level Monitoring

Warehouse

- Capacity Metrics App
- Query Activity in the Warehouse UI

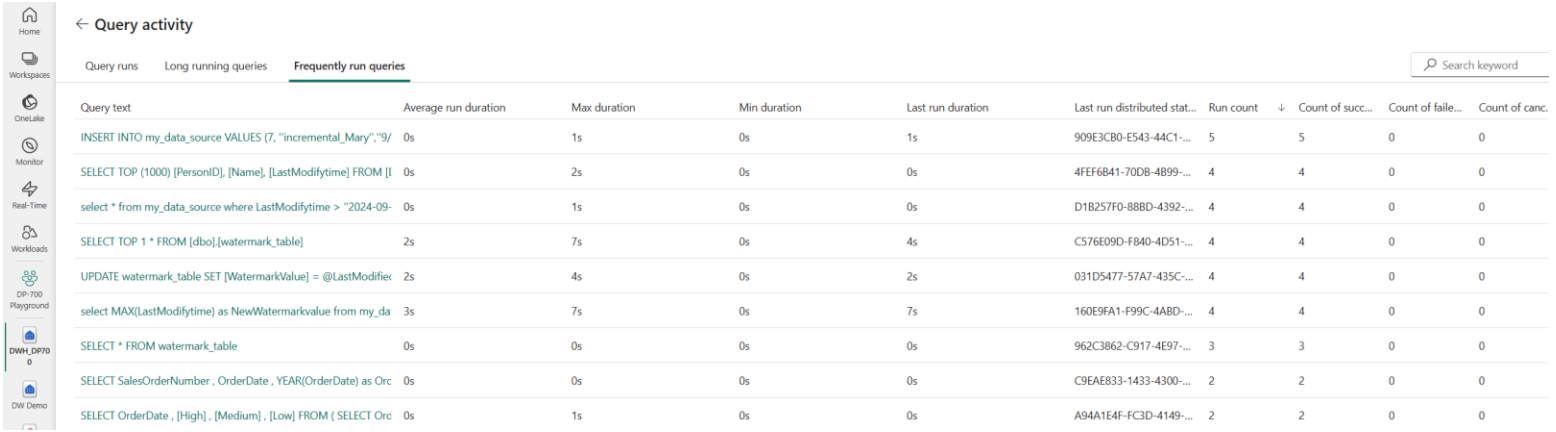


Item-level Monitoring

Warehouse

Fabric UI

- Query Activity in the Warehouse UI
  - Query runs
  - Long running queries
  - Frequently run queries



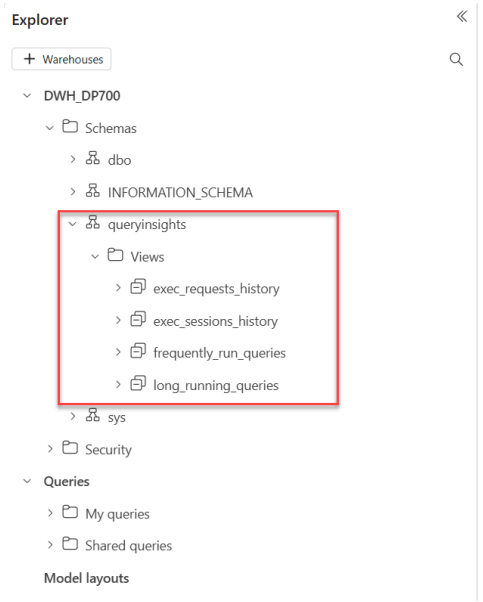
Query text	Average run duration	Max duration	Min duration	Last run duration	Last run distributed stat...	Run count	Count of succ...	Count of faile...	Count of canc.
INSERT INTO my_data_source VALUES (7, "incremental_Mary","9/	0s	1s	0s	1s	909E3CB0-E543-44C1-...	5	5	0	0
SELECT TOP (1000) [PersonID], [Name], [LastModifytime] FROM [I	0s	2s	0s	0s	4FEF6B41-70DB-4B99-...	4	4	0	0
select * from my_data_source where LastModifytime > "2024-09-	0s	1s	0s	0s	D1B257F0-88BD-4392-...	4	4	0	0
SELECT TOP 1 * FROM [dbo].[watermark_table]	2s	7s	0s	4s	C576E09D-F840-4D51-...	4	4	0	0
UPDATE watermark_table SET [WatermarkValue] = @LastModifiec	2s	4s	0s	2s	031D5477-57A7-435C-...	4	4	0	0
select MAX(LastModifytime) as NewWatermarkvalue from my_da	3s	7s	0s	7s	160E9FA1-F99C-4ABD-...	4	4	0	0
SELECT * FROM watermark_table	0s	0s	0s	0s	962C3862-C917-4E97-...	3	3	0	0
SELECT SalesOrderNumber , OrderDate , YEAR(OrderDate) as Orc	0s	0s	0s	0s	C9EAE833-1433-4300-...	2	2	0	0
SELECT OrderDate , [High] , [Medium] , [Low] FROM (SELECT Orc	0s	1s	0s	0s	A94A1E4F-FC3D-4149-...	2	2	0	0

Item-level Monitoring

Warehouse

Query-based

- queryinsights schema in the warehouse
- DMVs



DEMO

- Explore Monitor Hub
- Explore Capacity Metrics App
- Monitor individual Fabric items

Q&A

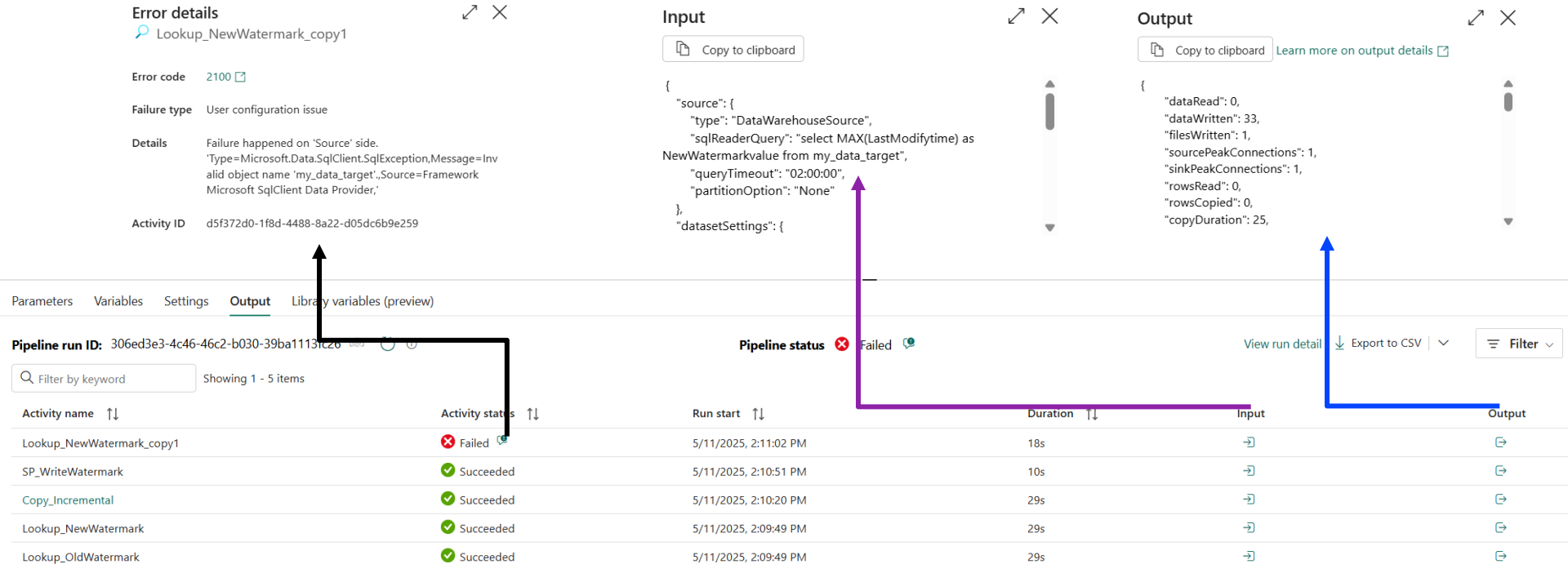
Identify and Resolve Errors

Identify and resolve errors

- Pipeline errors
- Dataflow errors
- Notebook errors
- Eventhouse errors
- Eventstream errors
- T-SQL errors

Identify and Resolve Errors

Pipeline

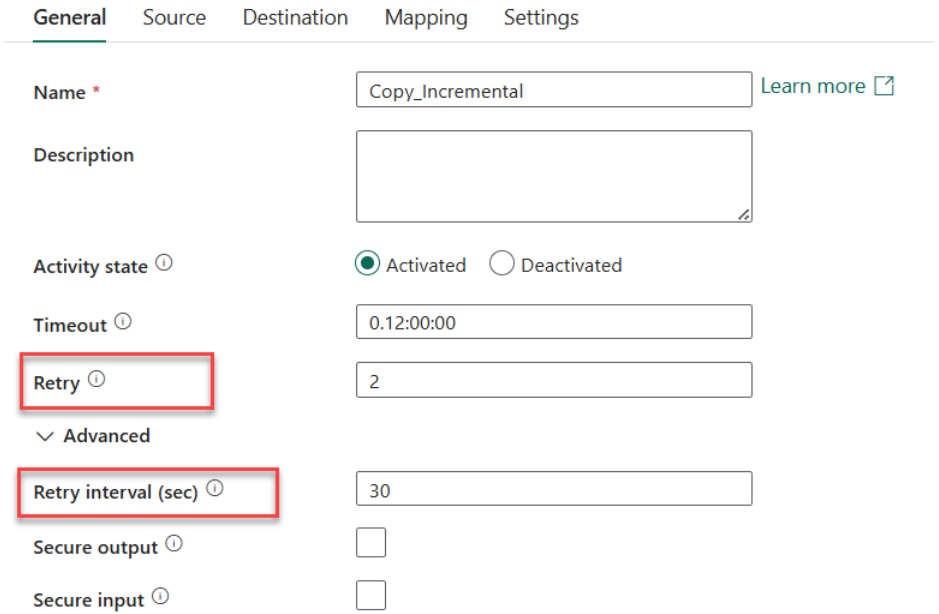


Identify and Resolve Errors

Pipeline

Make a pipeline more resilient

- Set the Retry property
- Failure -> Retry execution

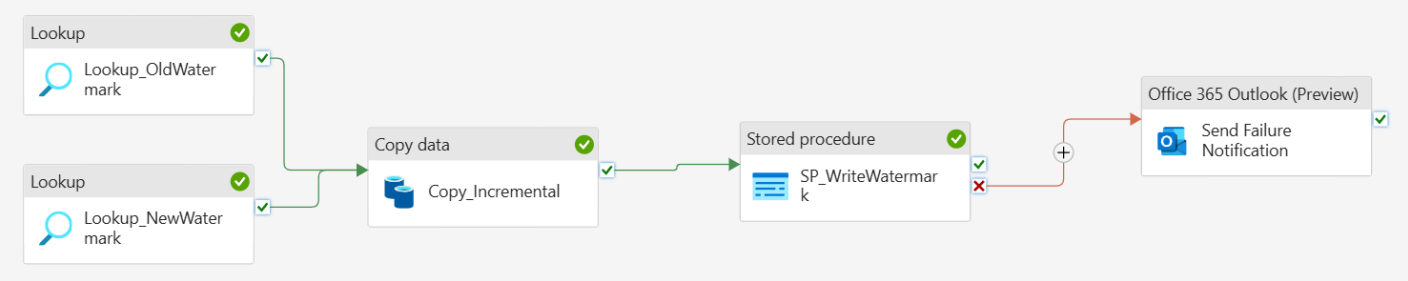


Identify and Resolve Errors

Pipeline

Make a pipeline more resilient

- **Send notifications when the activity fails**
  - **Outlook**
  - **MS Teams**



Parent pipeline → **Invokes** → **Child pipeline**

- Send notification

- Capture all failures

Q&A

Break

Optimize Performance

Optimize performance

- Lakehouse table
- Pipeline
- Data warehouse
- Eventsreams and eventhouses
- Spark performance
- Query performance

Optimize Performance

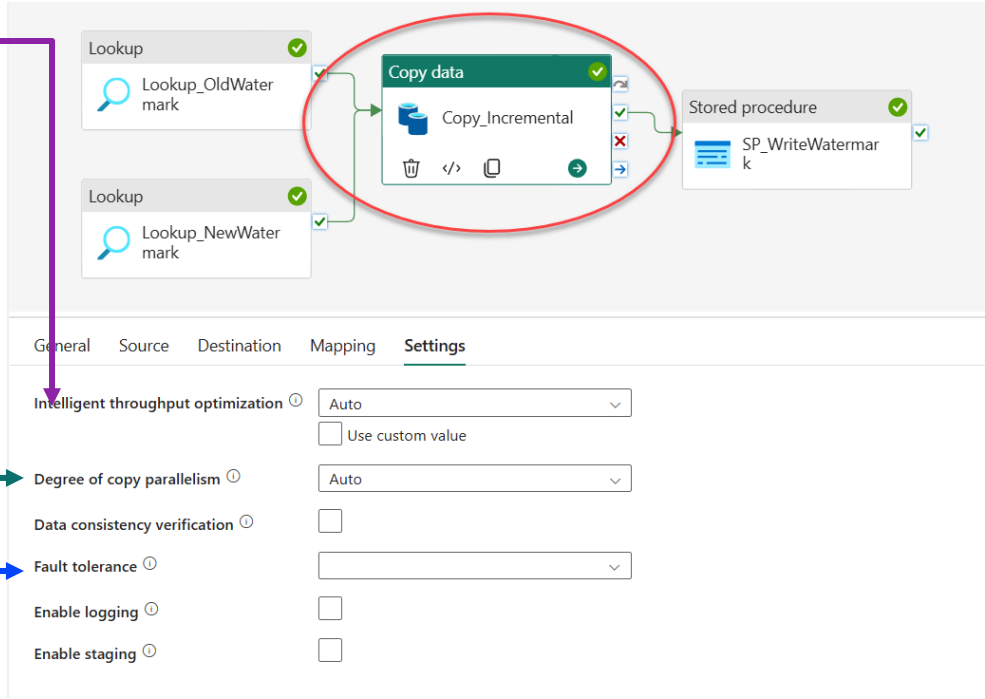
Pipeline

Amount of CPU assigned to a task (4-256)

Parallelize the copy to reduce the load time (128 by default)

Skip rows/files that can't be copied

Copy Data activity

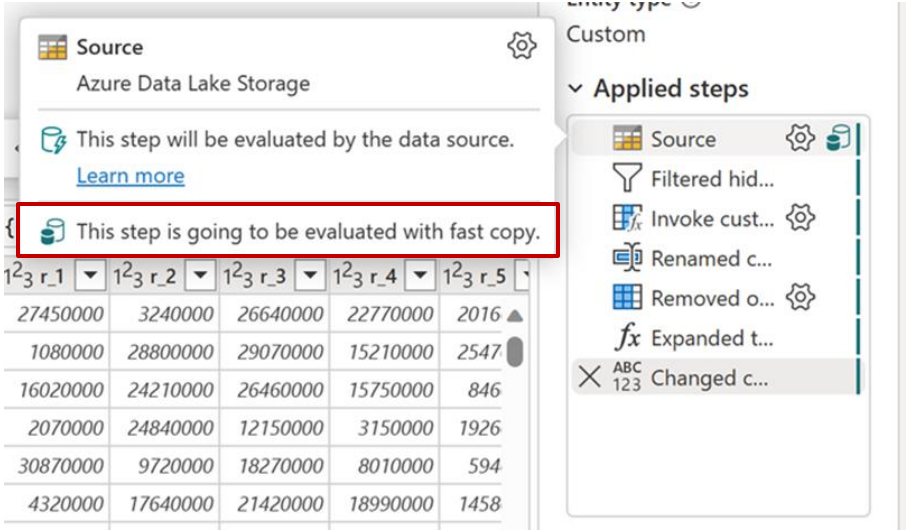


Optimize Performance

Dataflow

Usually consumes a lot of CUs!

- **Fast Copy (if data source supports)**
  - **In the background, it's a Copy Data activity from the data pipeline**
- **Require fast copy setting**
- **Not all transformations supported**
  - **Combine files**
  - **Select columns**
  - **Change data types**
  - **Rename/Remove column**

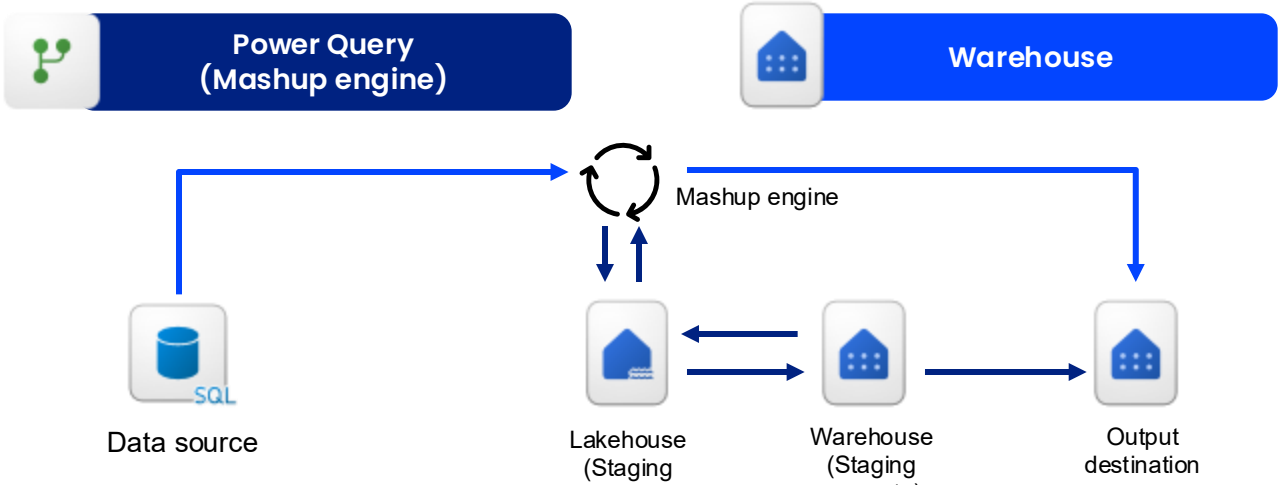


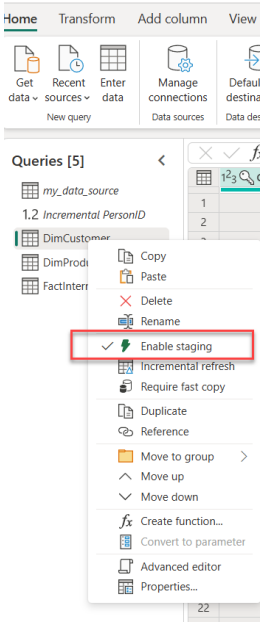
Optimize Performance

Dataflow

Enable/Disable staging

- Staged -> Query output written to a staging lakehouse/warehouse (hidden)
- Enabled by default
- Which engine performs transformations?





Optimize Performance

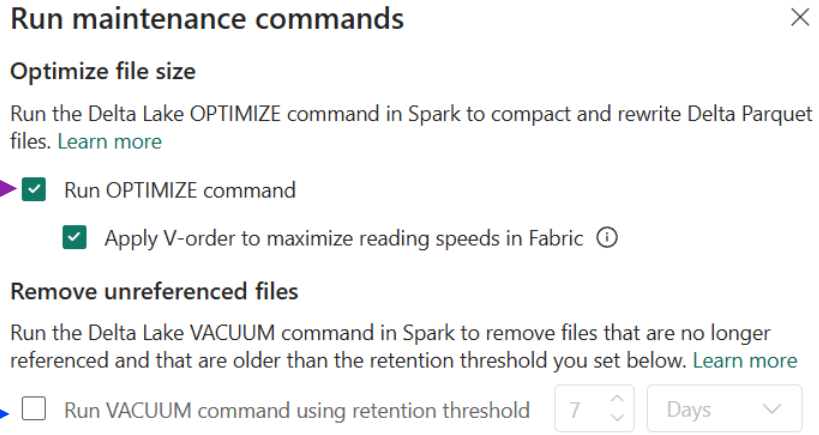
Lakehouse

Table maintenance - UI

- **Manually manage and maintain individual tables**

Compaction of smaller parquet files

Remove unreferenced parquet files

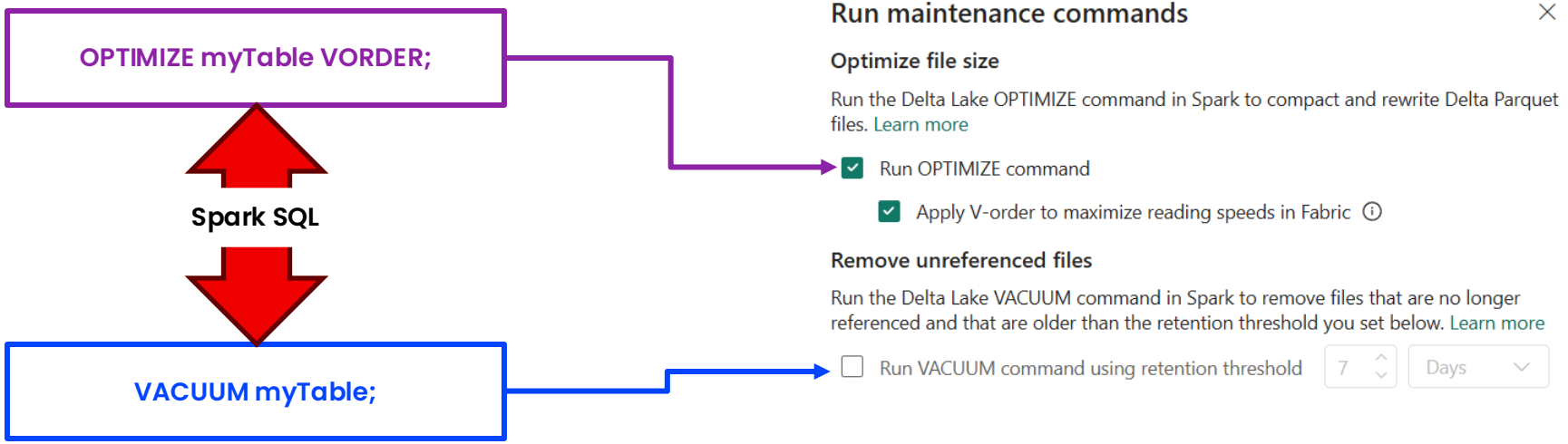


Optimize Performance

Lakehouse

Table maintenance – Code based

- **Automate the workflow with notebooks/jobs**



[Delta optimization and V-order]

[Optimizing Spark compute for medallion architecture]

Optimize Performance

Lakehouse

Table partitioning

- Break down larger tables into smaller chunks

```
#write data into lakehouse using partitioning

df_output.write.mode("overwrite").partitionBy('year','month','day').parquet(output_path)
```

Be careful! Sometimes, partitioning may slow down the query performance

To V-order or Not?

- Microsoft's proprietary algorithm that optimizes the way data is written in the Delta table
- Puts more pressure on the data writing process
- The data reading process is (usually) faster
- Enabled by default for all Fabric engines

Disable V-order

Warehouse

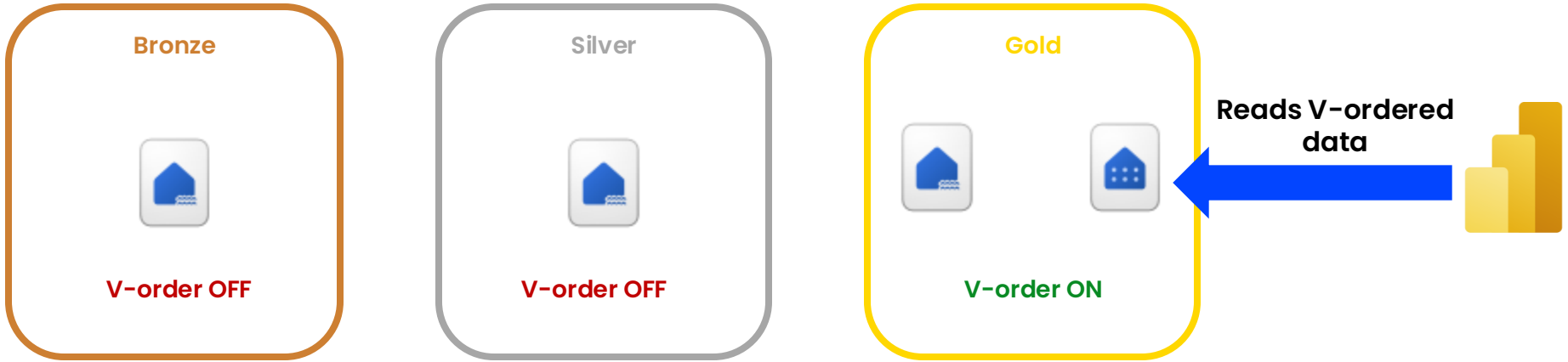
```
ALTER DATABASE CURRENT
SET VORDER = OFF;
```

Enabled/Disabled on a warehouse level!

Lakehouse

Configuration	Default value	Description
spark.sql.parquet.vorder.default	true	Controls session level V-Order writing.
TBLPROPERTIES("delta.parquet.vorder.default")	false	Default V-Order mode on tables
Dataframe writer option: parquet.vorder.default	unset	Control V-Order writes using Dataframe writer

To V-order or Not?

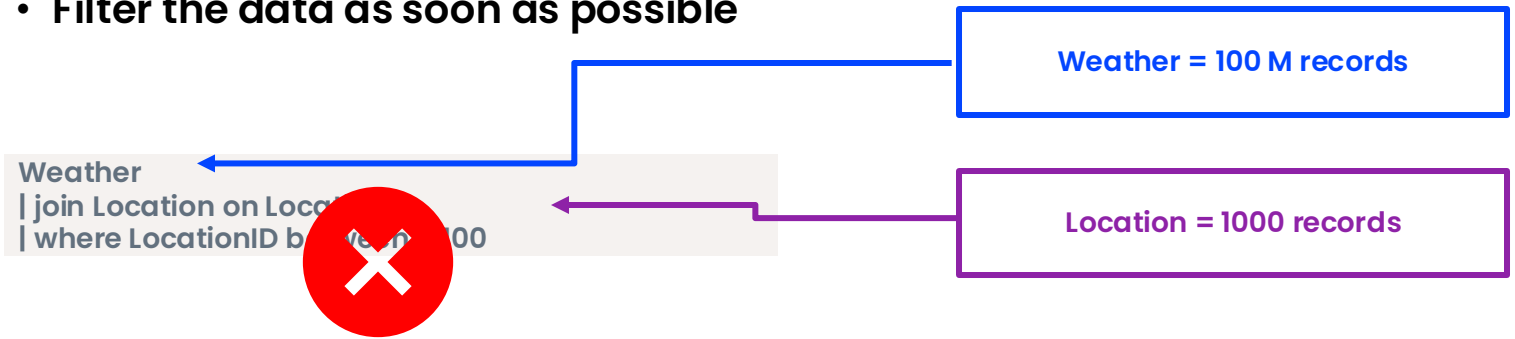


Optimize Performance

Eventhouse

Order of operations is important!

- **Filter the data as soon as possible**




```
Weather
| where LocationID between 1..100
| join Location on LocationID
```

```
Weather
| where timestamp > ago (12h)
| join Location on LocationID
```

Optimize Performance

Eventhouse

Has vs. Contains

- has – exact match (searched terms are indexed)
- contains – search for the string, but also substrings -> slower performance

[Best practices for KQL queries]

DEMO

- Optimize lakehouse with OPTIMIZE and VACUUM
- Enable/disable staging for Dataflows
- Enable/disable V-ordering

Q&A

Practical Exam Tips, Quiz, and Closing

Practical Exam Tips

- Make sure to prepare the room (for online takers)
- You can use Microsoft Learn during the exam☺…
- …but, don't let this waste too much of your time!
- Take time to understand case studies and explore the exhibits/datasets before answering (there is no way back)
- Focus on specific functions (PySpark, T-SQL, KQL) and their ORDER of execution
- Understand WHEN to complete a specific task in a certain way (ingest data with pipeline vs. notebook vs. shortcut vs. mirroring)

Quiz

Join the Quiz!

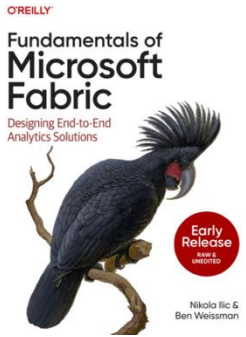
https://app.sli.do/event/caGLziwTYvnKGrnsHBwErx

Group Discussion

- Which benefits do you expect from passing the DP-700 exam?

Summary

Takeaways



- Check the official Microsoft DP-700 study guide:
  - Official study guide
- DP-700 Certification page with practice assessment
  - Take the practice assessment
  - Experience demo
- Microsoft Fabric courses and books @ O'Reilly platform
  - Fundamentals of Microsoft Fabric
  - Fabric Analytics Engineer Bootcamp (DP-600)

Q&A


O'REILLY®