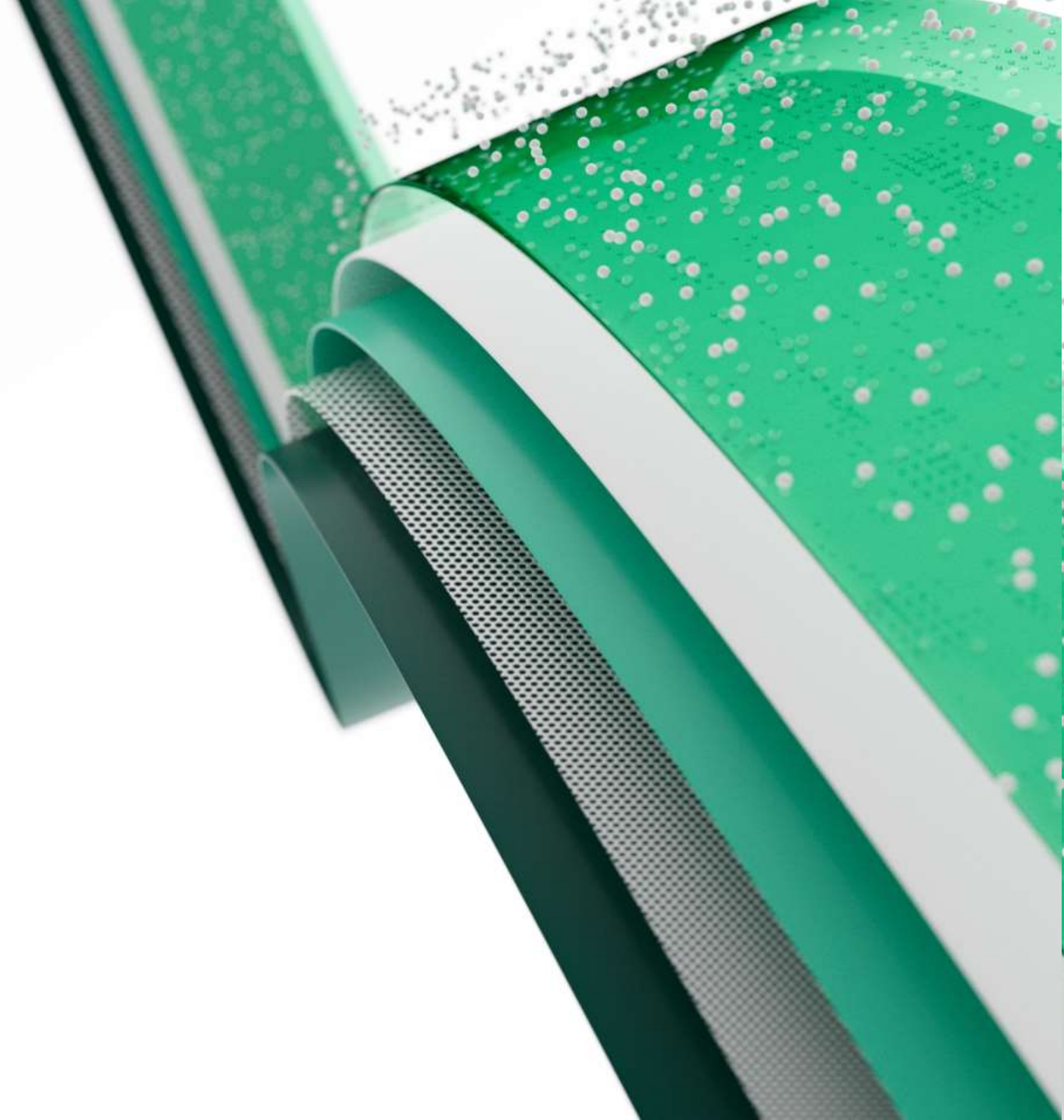
Microsoft

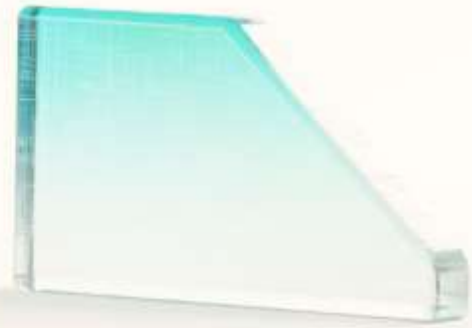
Fabric Certification Academy Day



WARNING

This deck is a “pre-release”. We’re working on a cohesive deck that you can leverage for re-delivery of this content.

This preview is targeted to Fabric Certification Academy Day attendees from the Microsoft France session an April 4th 2024.



Microsoft

Thank you!

Agenda (times are approximate and will be fluid with the class)

Morning

09:30 AM – 09:45 AM	Welcome
09:45 AM – 10:00 AM	Study guide for exam DP-600
10:00 AM – 10:45 AM	Module 1: Plan, implement, and manage a solution for data analytics
10:45 AM – 11:00 AM	Break
11:00 AM – 12:30 PM	Module 2: Prepare and serve data
12:30 PM – 02:00 PM	Break for lunch

Afternoon

02:00 PM – 02:45 PM	Module 3: Implement and manage semantic model
02:45 PM – 03:30 PM	Module 4: Explore and analyze data
03:30 PM – 03:45 PM	Break
03:45 PM – 04:45 PM	Practice exam
04:45 PM – 05:00 PM	Prepare for Exam Day
05:00 PM – 05:30 PM	Q&A Session

Fabric Certification Academy Day

Meet the audience

Raise your hand if…

- You're working for a Microsoft Partner
- You already passed a Microsoft Certification
- You already played with Fabric
- You have a Proof of Concept in progress with Fabric
- You already have a production workload on Fabric
- You are a member of a Fabric/Power BI user group

Microsoft Fabric Hands-on Day

Meet your proctors

EXAM

Implementing Analytics Solutions Using Microsoft Fabric

Passed

March 13, 2024 • Online • Provided by PearsonVue

View details on provider site

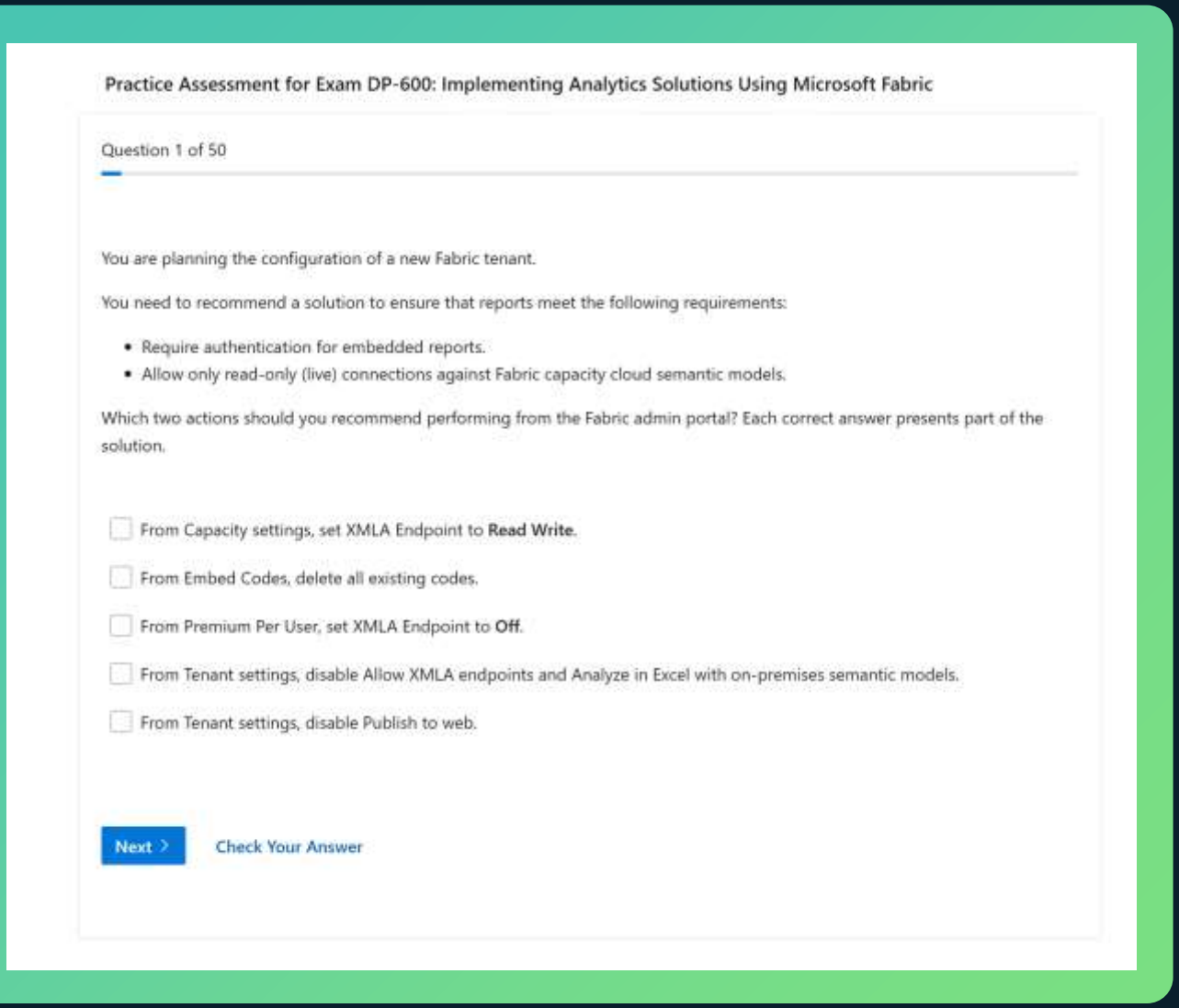
Frédéric Gisbert
Cloud Solutions Architect
Partenaires

Christopher Maneu
Cloud Advocate Data
Azure Engineering

DP-600 Practice test

Access To the Practice test

Aka.ms/dp600-practice



Fabric Certification Academy Day resources

Access Wifi

SSID:

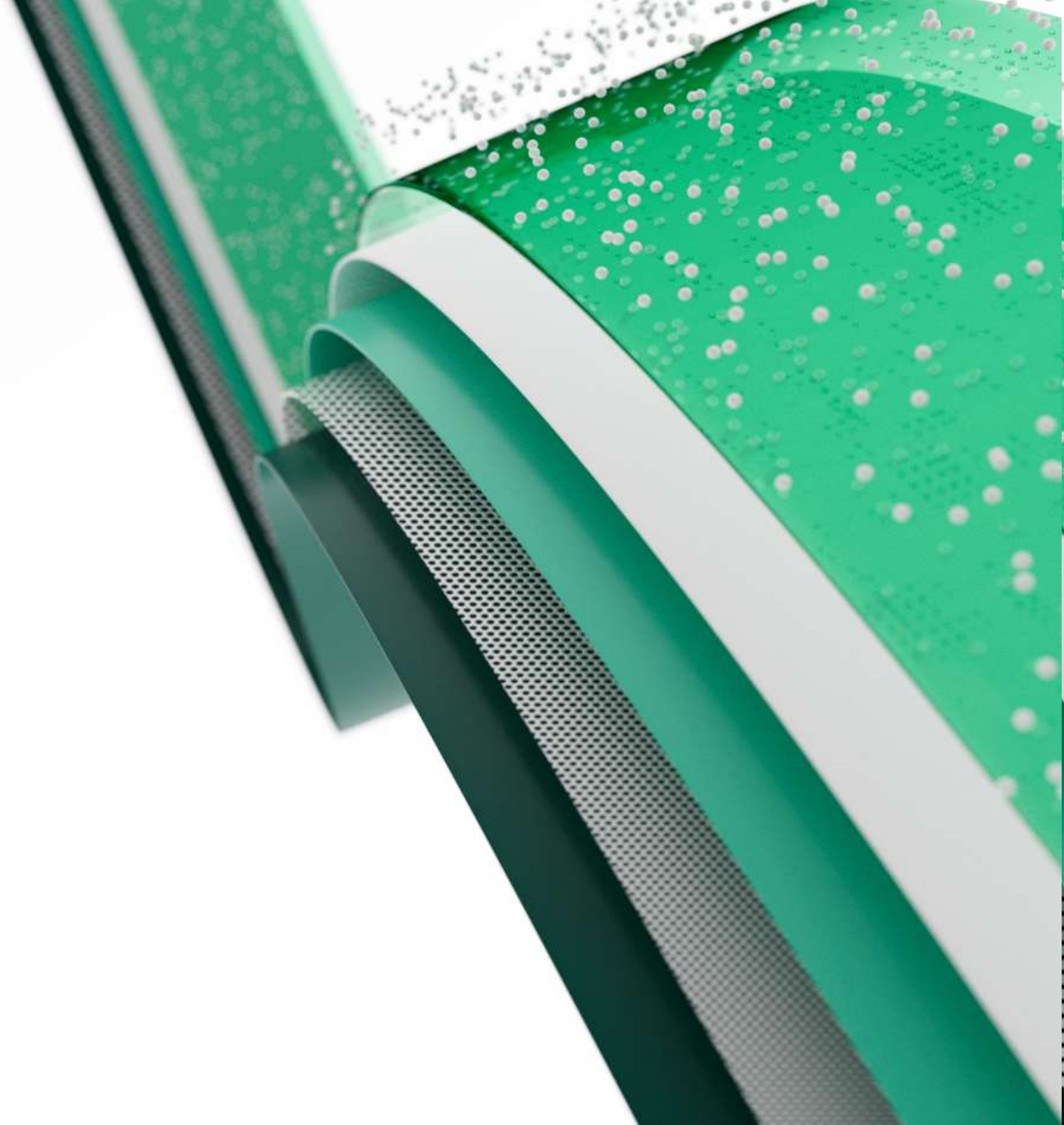
Event Code:

Get resources

Aka.ms/fabric-cert-day

Microsoft

DP-600 Exam overview



Exam DP-600
Implementing Analytics Solutions Using Microsoft Fabric

Skills measured

Plan, implement, and manage a solution for data analytics (10–15%)

Prepare and serve data (40–45%)

Implement and manage semantic models (20–25%)

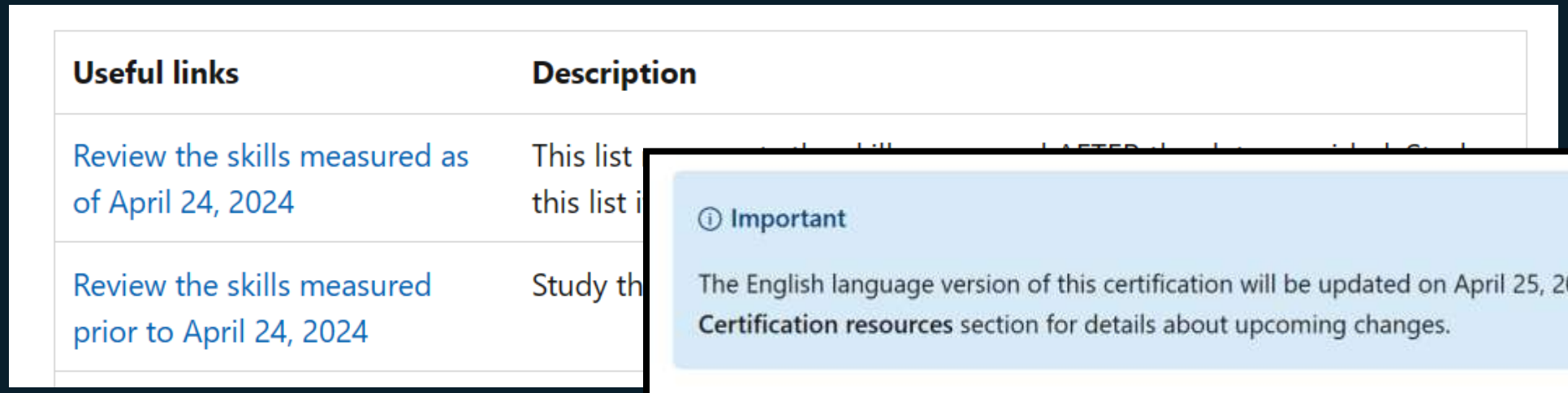
Explore and analyze data (20–25%)

Certification versus Reality

We ship new product features every week

The certification “training date” is March 15th.

If the content is updated, it will be notified in the study guide.



Useful links	Description
Review the skills measured as of April 24, 2024	This list
Review the skills measured prior to April 24, 2024	Study th

Important

The English language version of this certification will be updated on April 25, 2024. Review the study guide linked in t
Certification resources section for details about upcoming changes.

1

Module 1
Plan, implement, and manage a solution for data analytics

Plan, implement, and manage a solution for data analytics (10-15%)

Plan a data analytics environment

Identify requirements for a solution, including components, features, performance, and capacity stock-keeping units (SKUs)

Recommend settings in the Fabric admin portal

Choose a data gateway type

Create a custom Power BI report theme

Implement and manage a data analytics environment

Implement workspace and item-level access controls for Fabric items

Implement data sharing for workspaces, warehouses, and lakehouses

Manage sensitivity labels in semantic models and lakehouses

Configure Fabric-enabled workspace settings

Manage Fabric capacity

Manage the analytics development lifecycle

Implement version control for a workspace

Create and manage a Power BI Desktop project (.pbip)

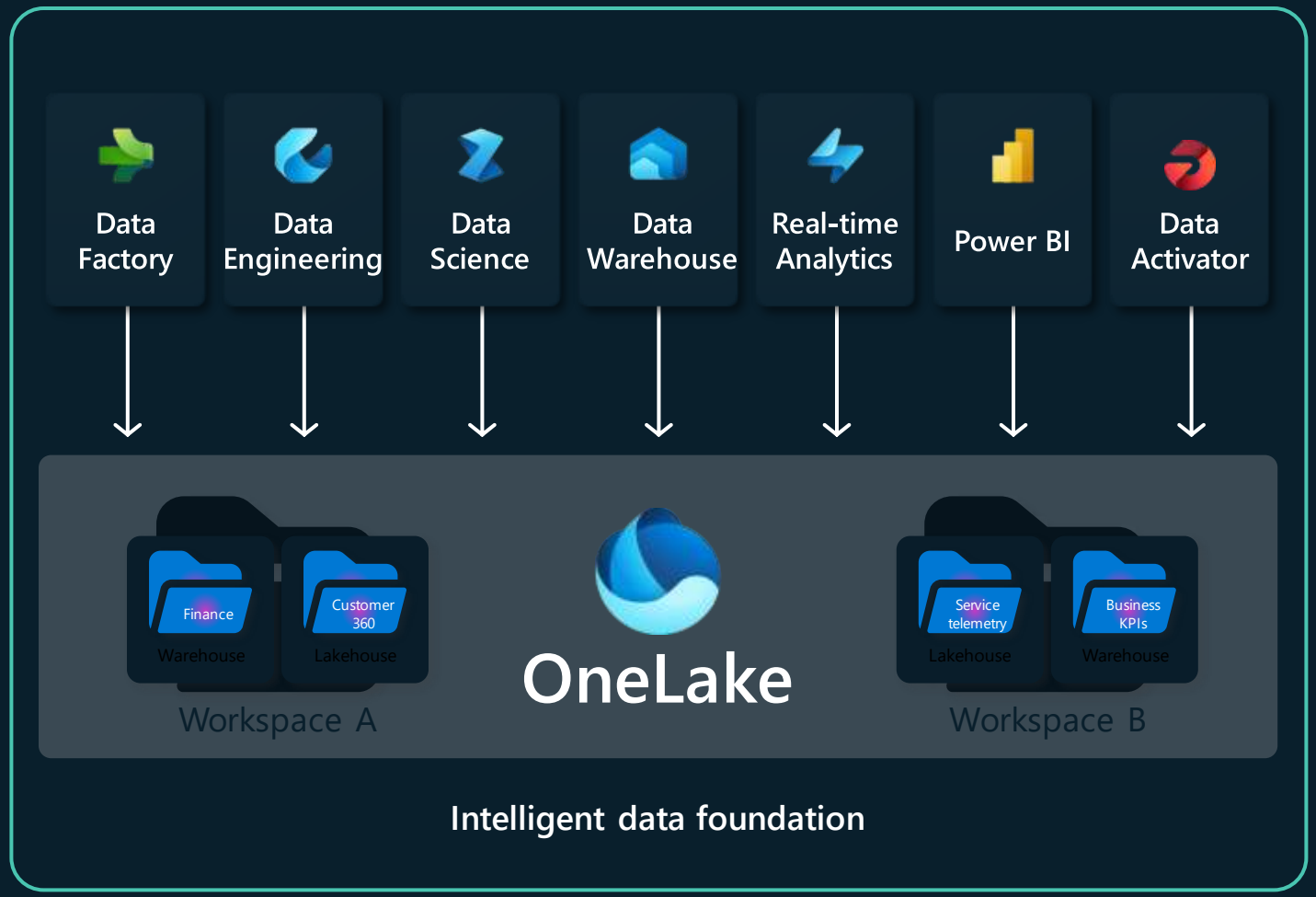
Plan and implement deployment solutions

Perform impact analysis of downstream dependencies from lakehouses, data warehouses, dataflows, and semantic models

Deploy and manage semantic models by using the XMLA endpoint

Create and update reusable assets, including Power BI template (.pbit) files, Power BI data source (.pbids) files, and shared semantic models

Understand Fabric architecture



Single SaaS lake

Provisioned automatically with the tenant

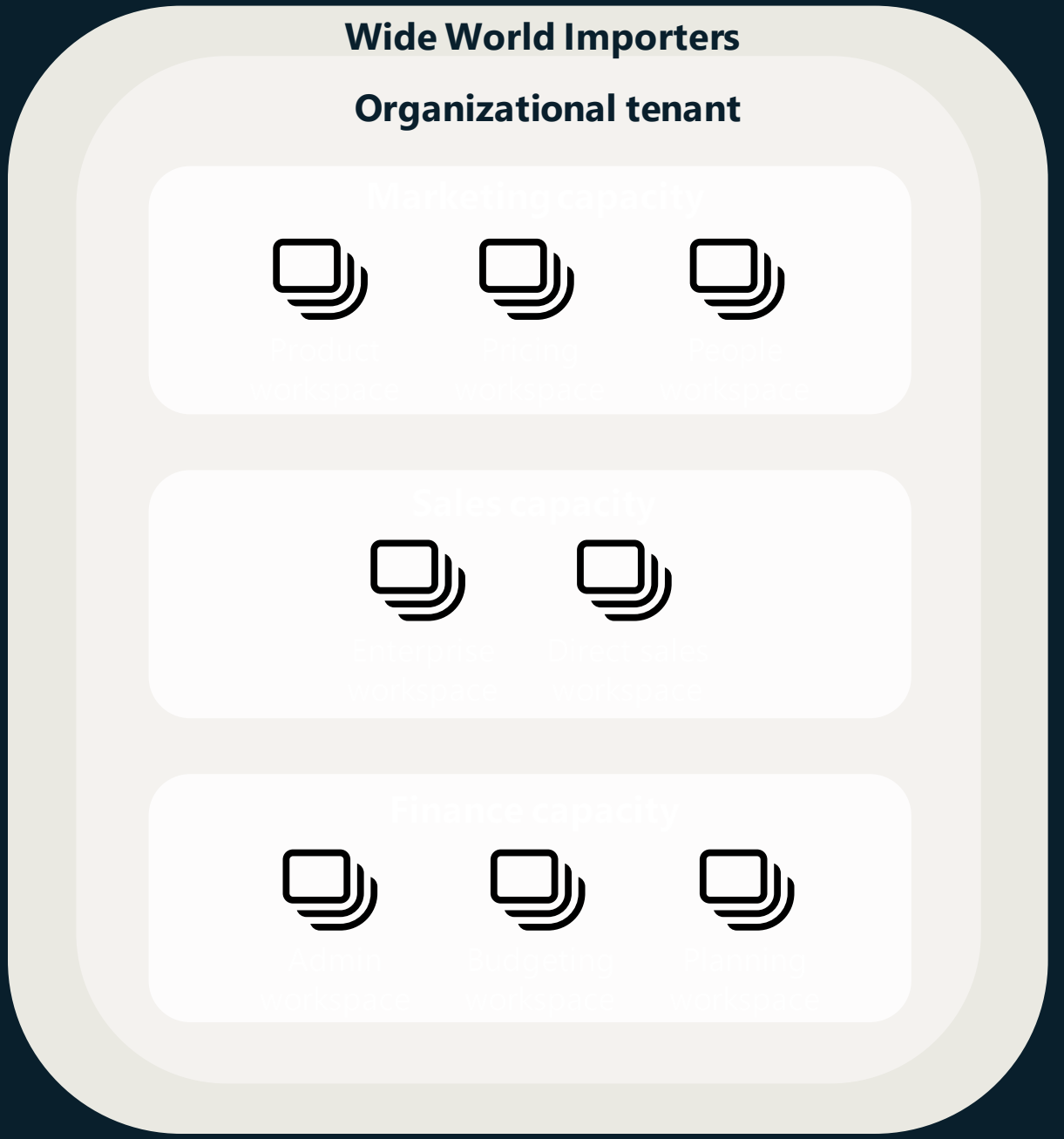
All workloads automatically store data in OneLake workspace folders

All data is organized in a hierarchical namespace

Data in OneLake is automatically indexed for discovery, MIP labels, lineage, PII scans, sharing, governance and compliance

Understand Fabric concepts

- Tenant
- Capacity
- Domain
- Workspace
- Items



Fabric admin tasks

Security and access control

Data governance

Customization and configuration

Monitoring and optimization

Fabric admin tools

Fabric admin portal

PowerShell cmdlets

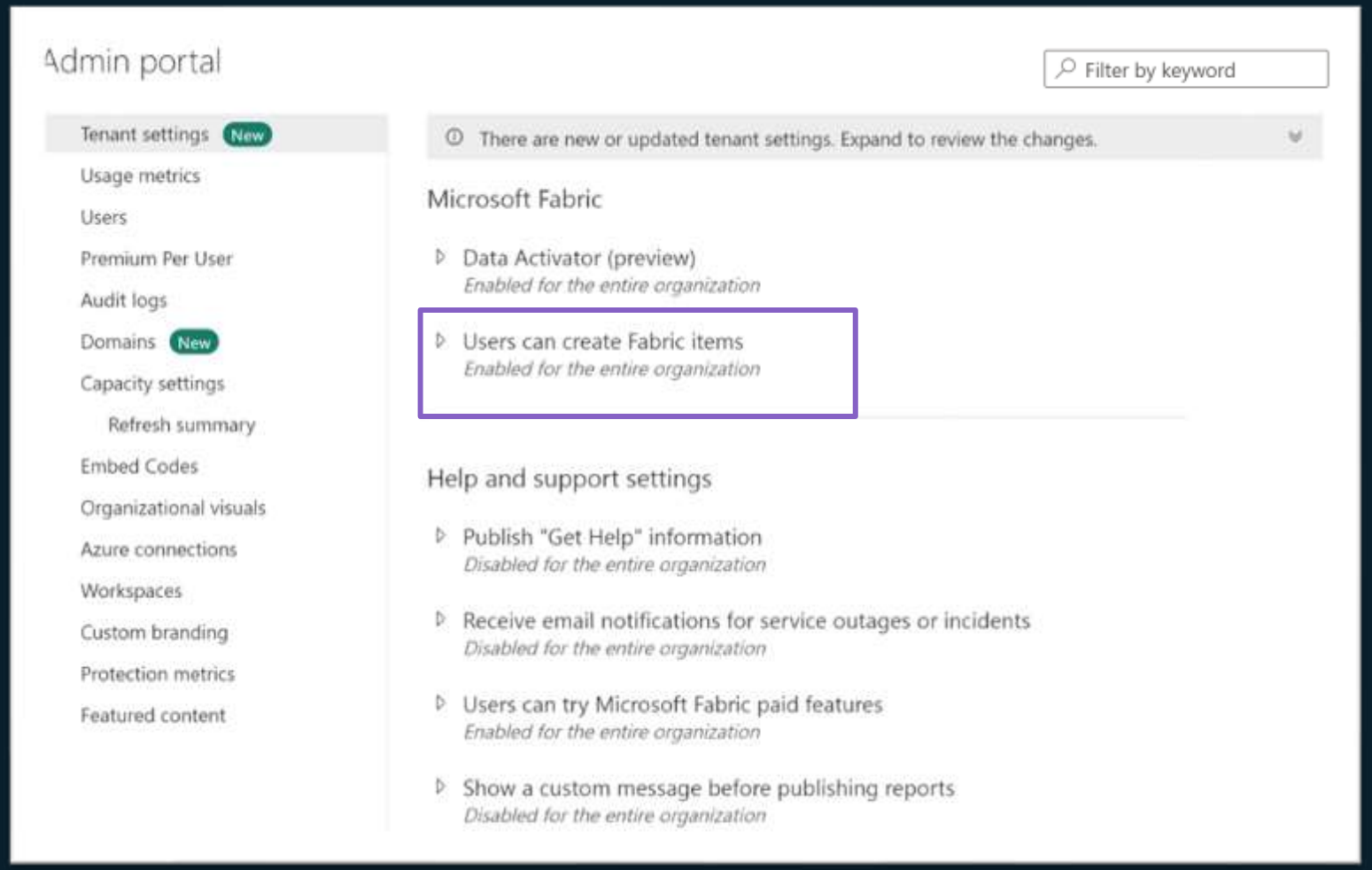
Admin APIs and SDKs

Admin monitoring workspace

Fabric capacity monitoring app

Fabric admin portal

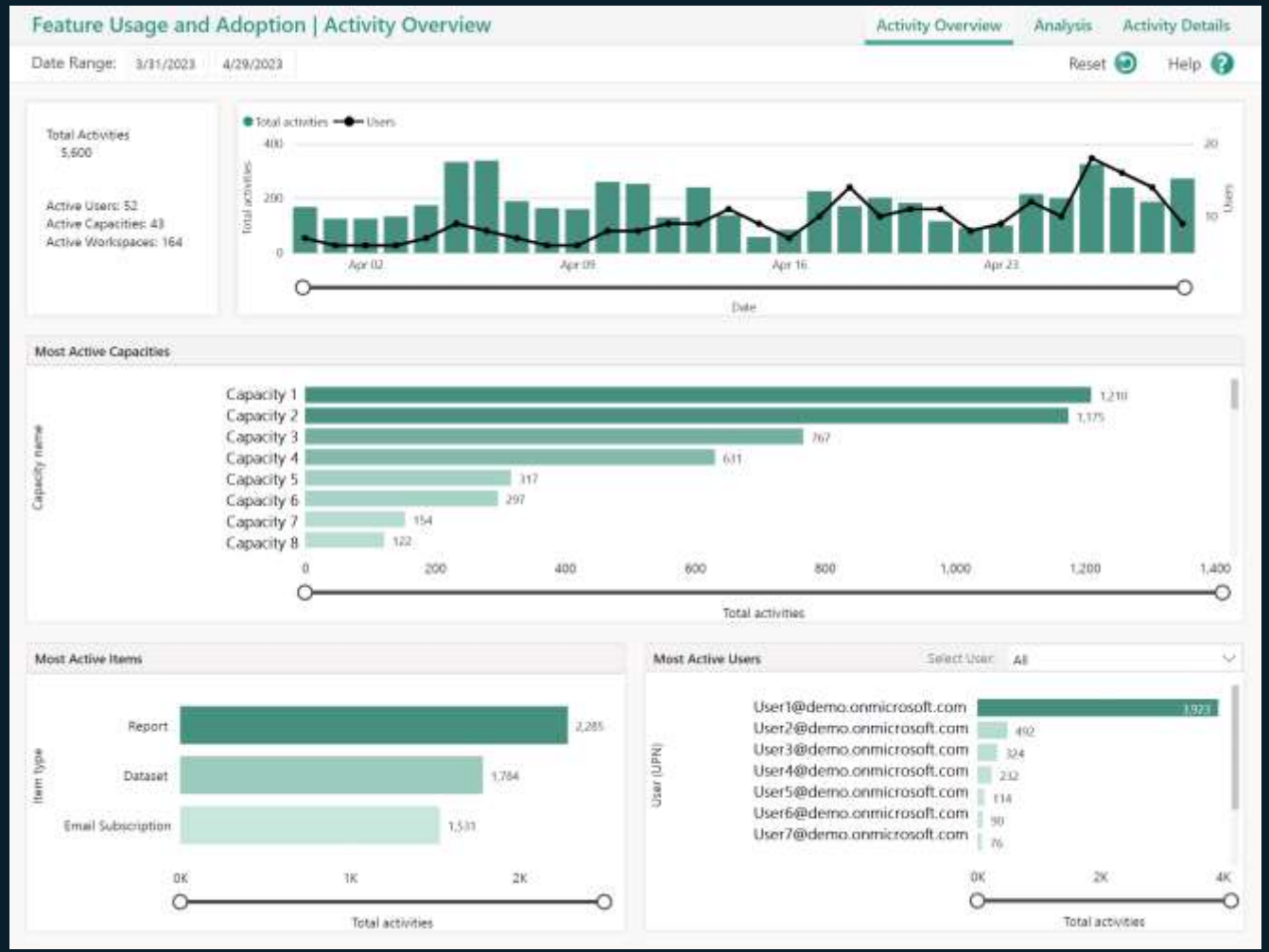
Centrally manage, review, and apply settings for the entire tenant or by capacity



Fabric admin monitoring workspace

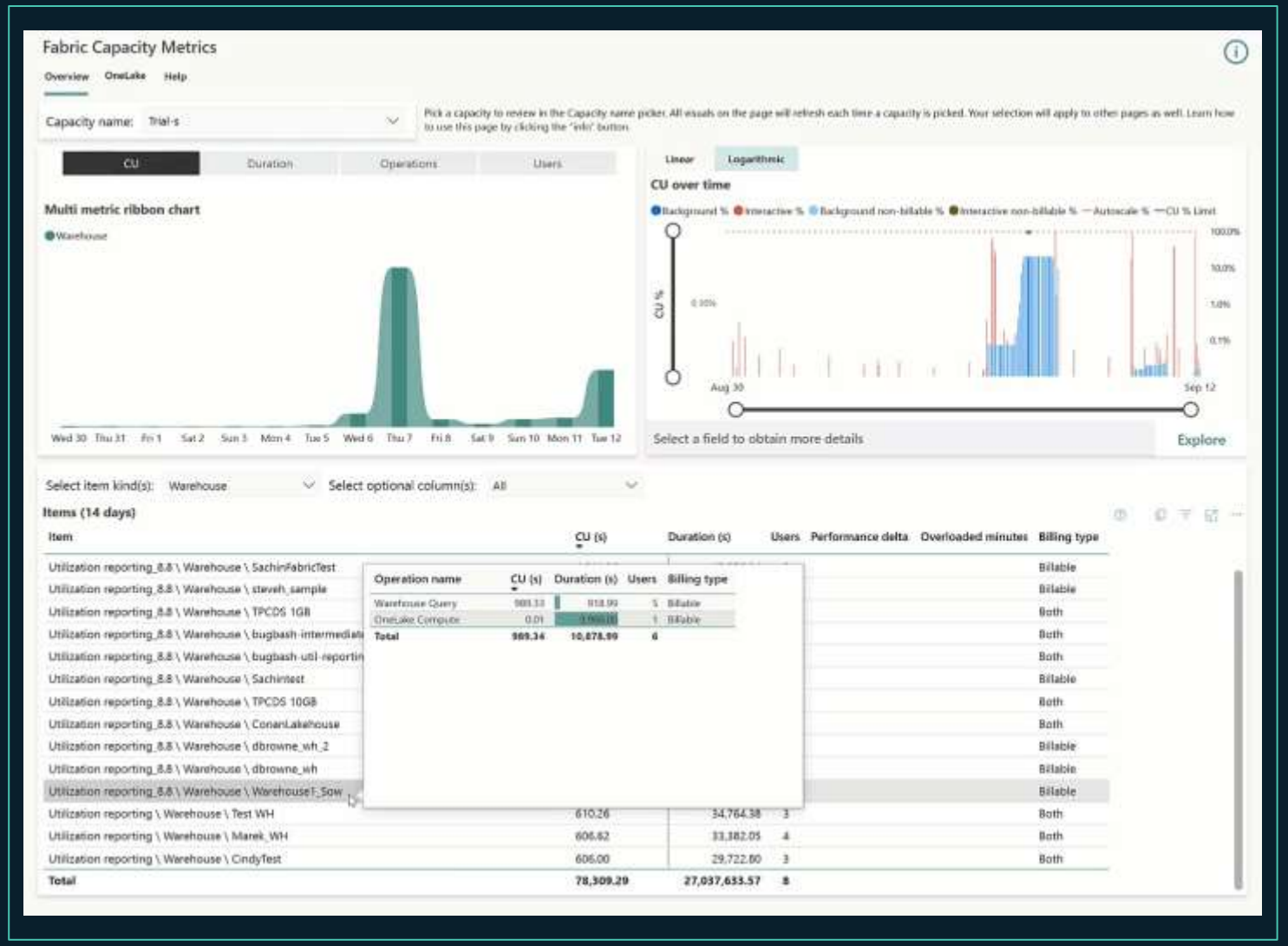
Monitor your environment including active:

- capacities
- workspaces
- items
- activities
- users



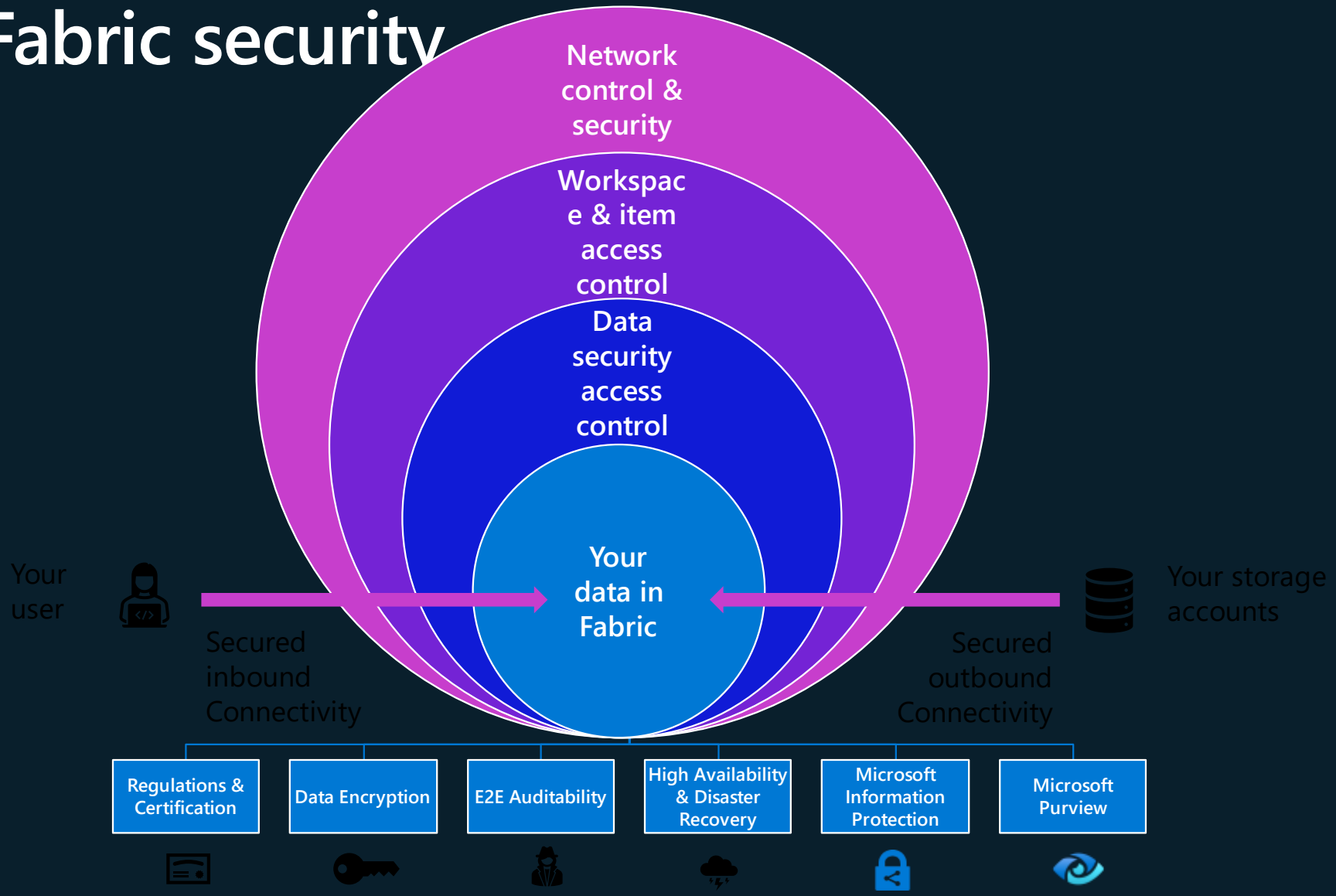
Fabric capacity metrics app

- Charts display capacity usage over a 14-day period
- Helps you understand which workloads use more capacity



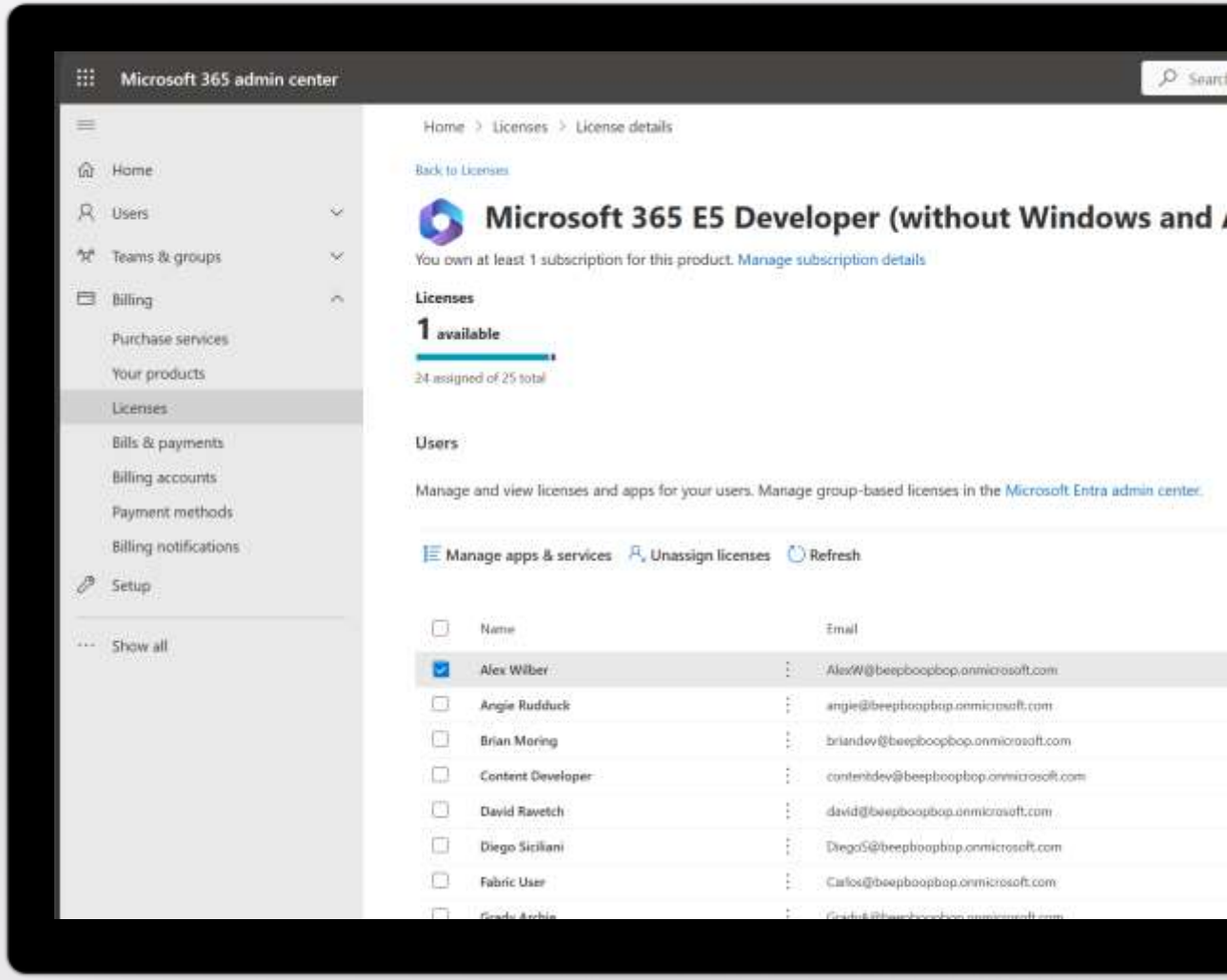
Admin
& workspace-level settings

Manage Fabric security



Manage users: assign and manage licenses

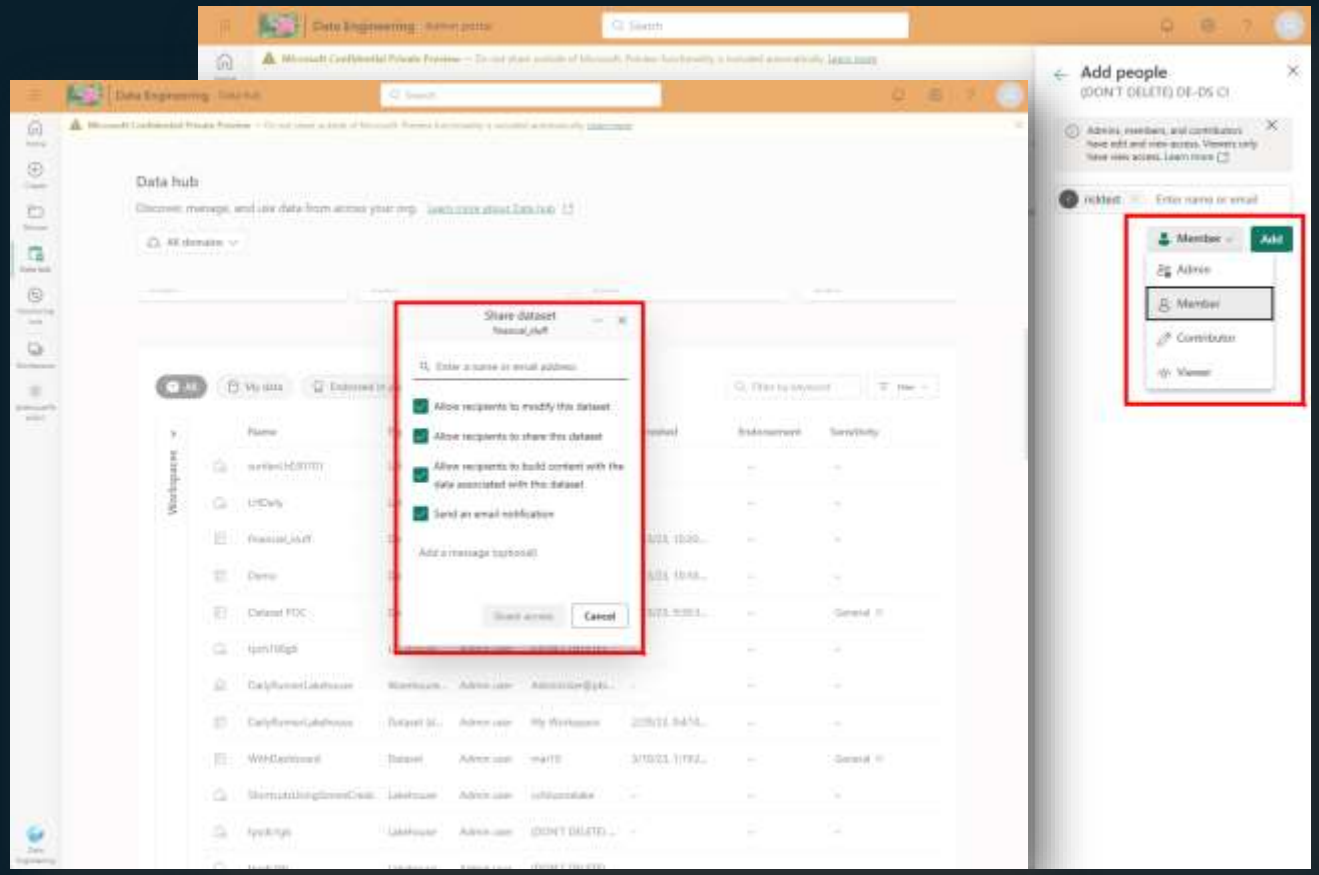
- Licenses control the level of access and functionality that users have within the Fabric environment.
- License management for Fabric is located in the M365 admin center.
- The next step beyond licenses is workspace management.



Manage items and sharing

Admins can manage how users interact with Fabric in terms of sharing and distribution of content.

- Fabric workspace roles define default permissions on workload items
- Item permissions can be modified and managed via sharing
- Universal Security defines access policies on the delta tables directly and all workload compute engines will respect such policies



Workspace and item permissions

Workspaces

- Containers, store related items
- Roles grant workspace access
  - Admin
  - Contributor
  - Member
  - Viewer

Items

- Lakehouses, warehouses, semantic models, reports, dashboards, etc.
- Item-level access in workspace
- SQL analytics endpoint access
  - Read
  - ReadData
  - ReadAll

Granular security permissions

User-specific permissions using T-SQL

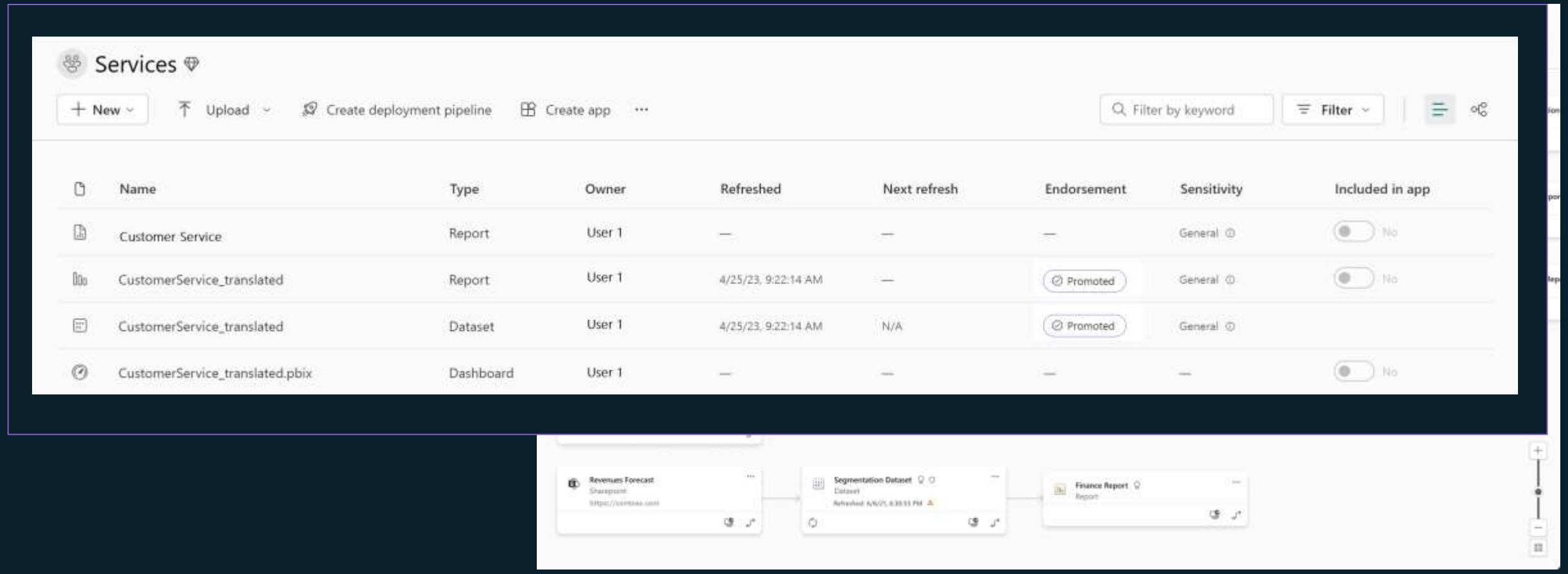
- Object-level security
- Column-level security
- Row-level security
- Dynamic data masking

User access recommendations

- Limit workspace access to collaborators.
    - Use Viewer role and grant read on specific objects for read-only.
- Grant access to specific Fabric item(s), then on objects for limited use cases.
    - Use Microsoft Entra ID groups, when possible.

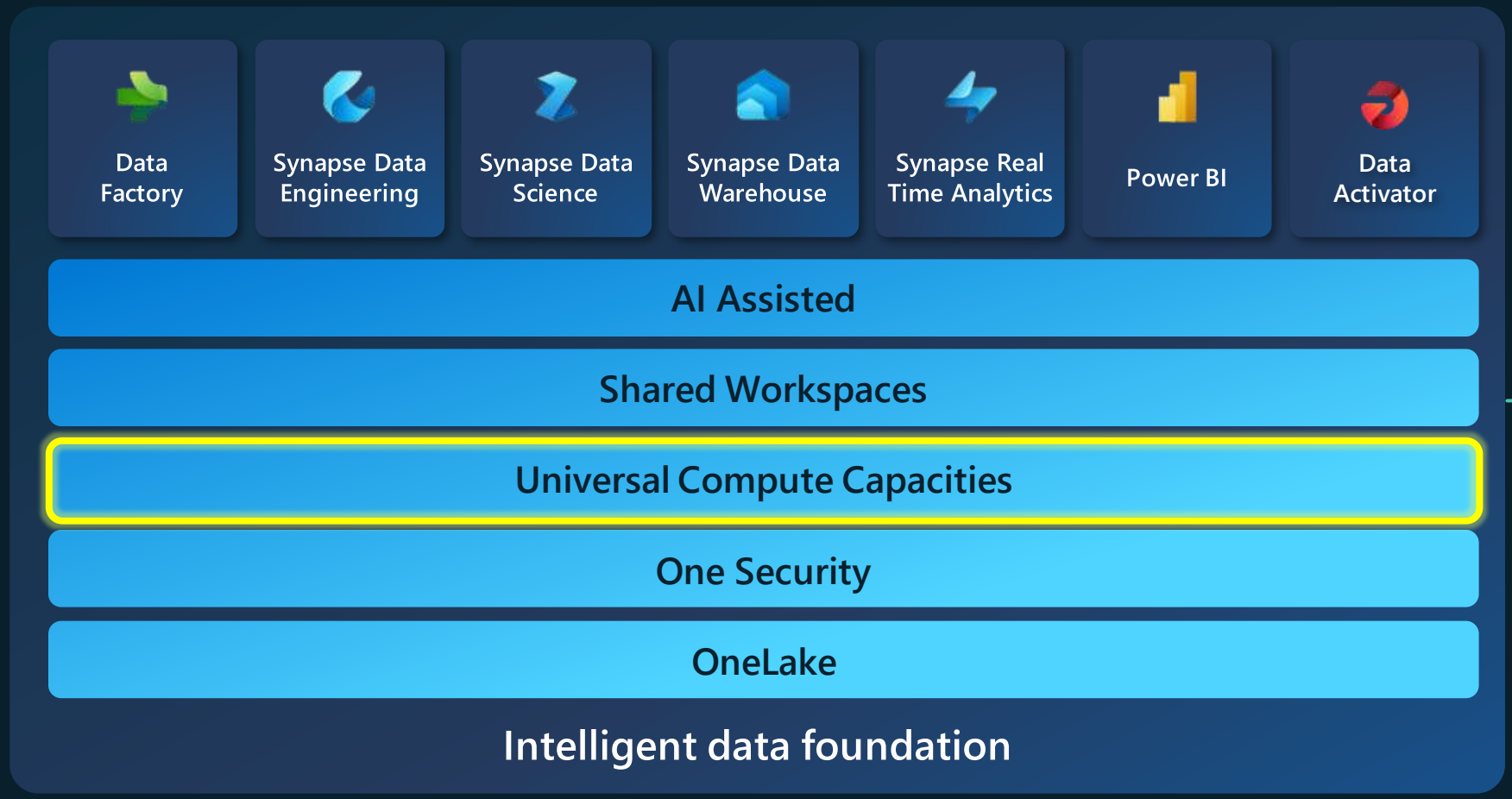
Govern data in Fabric

- Endorse Fabric content



Fabric Capacities Introduction

Microsoft Fabric



Single…

- Onboarding and trials
- Sign-on
- Navigation model
- UX model
- Workspace organization
- Collaboration experience
- Data Lake
- Storage format
- Data copy for all engines
- Security model
- CI/CD
- Monitoring hub
- Governance & Capacity Metrics
- Data Hub

Microsoft Fabric is a SaaS product

What does that mean?

Software As A Service

You rent the software + the infrastructure and pay only for what you use

Ease of use

- No infrastructure to manage
- Start in minutes, without IT knowledge
- Same UI as Power BI, Microsoft 365

Scalability

Add or remove users as you wish

Scale up or down processing power as needed

Bursting & smoothing helps you pass the Monday 9am rush

Security

Hosted on Microsoft Cloud Infrastructure

Unified security: same rights across tables, lake files, ... Coming Soon

Cost effective

No long-term commitments

But you can use them to save money

Scale up and down when you need

You can scale up for a few hours – for end of period calculations – and then scale down

Accessibility

Works on any PC or tablet*

You just need a web browser*

*There are some prerequisites, like hardware power and a modern browser, up to date. Some features are only available on Windows (in Power BI Client or OneLake Explorer)

Capacities are to Fabric what CPUs are to PCs

Personal Computing

When you purchase a PC you choose the number of CPU cores. The more CPU cores the more load the PC can handle.

The CPU cores are dynamically shared across all applications with no need to pre-allocate by app.

The total consumption of the CPU across all the apps cannot exceed the number of cores. CPU overload causes a slowdown.

Fabric Capacities

In Fabric, you provision a Capacity with a number of "capacity units". The more capacity units provisioned, the more compute load handled.

Unlike the PC, capacity units can be scaled up or down as needed.

The capacity units are dynamically shared across all the Fabric workloads, with no pre-allocation necessary.

A single capacity can simultaneously drive BI, DW, Spark, ML and every other compute engine in Fabric

The total consumption of the capacity across all the workloads cannot exceed the capacity units provisioned.

Overloading the capacity will throttle it (slow down).

Auto scale can dynamically increase the available compute units avoiding the slowdown.

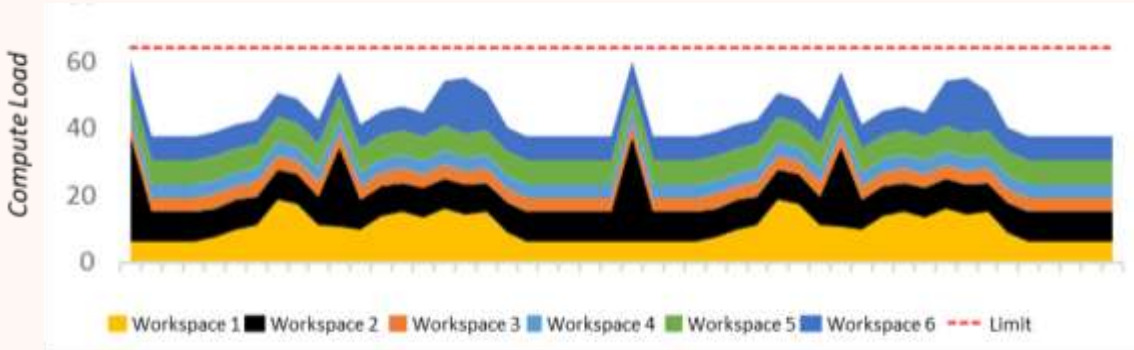
Capacities are a shared resource

Shared across workloads
A single capacity is providing the compute power for all Fabric workloads.

There is no need to allocate compute for each workload separately.

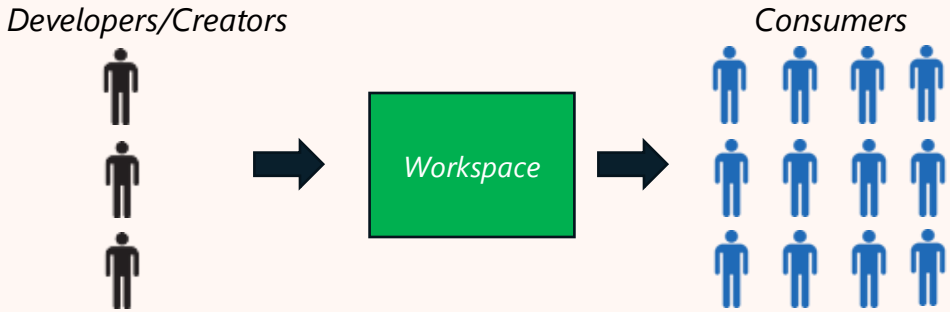
Shared Across Projects
A single capacity typically supports dozens of separate projects simultaneously, each managed in its own workspace.

It is rare to have a capacity dedicated to a single project



Shared across users
For each project, many developers will share a workspace where collaborative development and consumption at scale is managed.

Each creator can provision any artifact and run any job without the need for any pre-approval or planning



Capacities are a SaaS resource

Always on auto-pilot

Capacities are a self-managed system that alleviates day-to-day or hour-to-hour resource management by capacity administrators

Providing complete transparency into the usage of the capacities at any grain: projects/workspaces, artifacts, queries, users and jobs

Eliminate workload management

Large jobs can be scheduled at any time without impacting any other jobs running

Protection against a single user hogging all the resources by carelessly running a very large query

Smoothing automatically prevents spikey loads from creating temporal overload of the system – thereby you are purchasing for average load vs. max load.

Always peak performance

Bursting allows all jobs and queries to always run at peak performance regardless of any other compute activity happening on the capacity

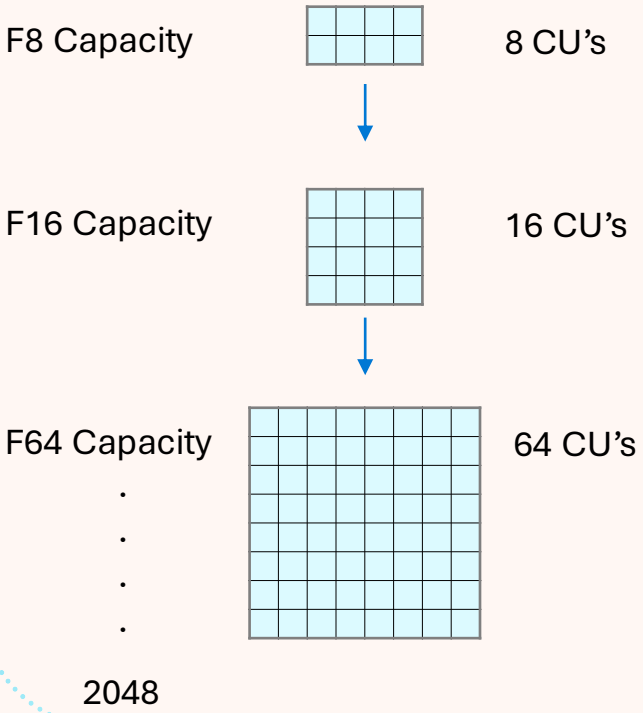
Auto scale can automatically adjust the compute units provisioned to the capacity

Capacities are flexible building blocks for growth

Capacities can be configured in endless ways to meet scale, usage and governance requirements while tuning to minimize TCO and performance goals

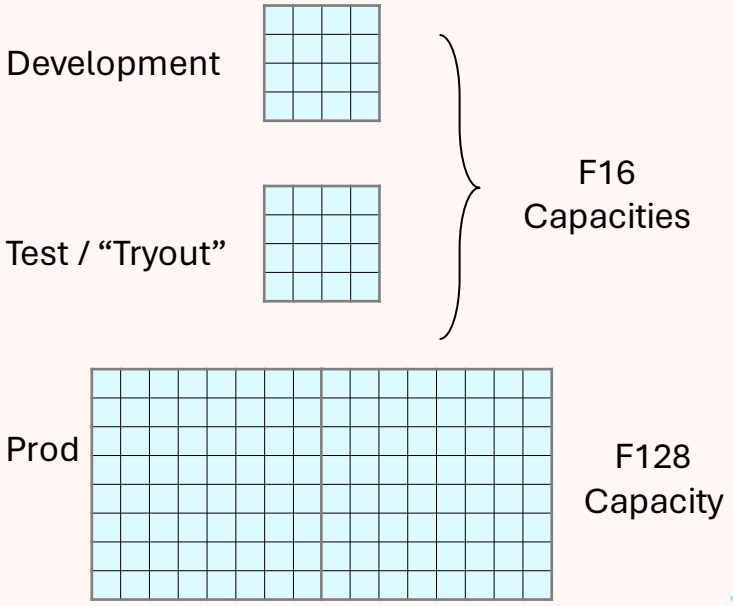
Scale Vertically

Increased capacity size provides more throughput



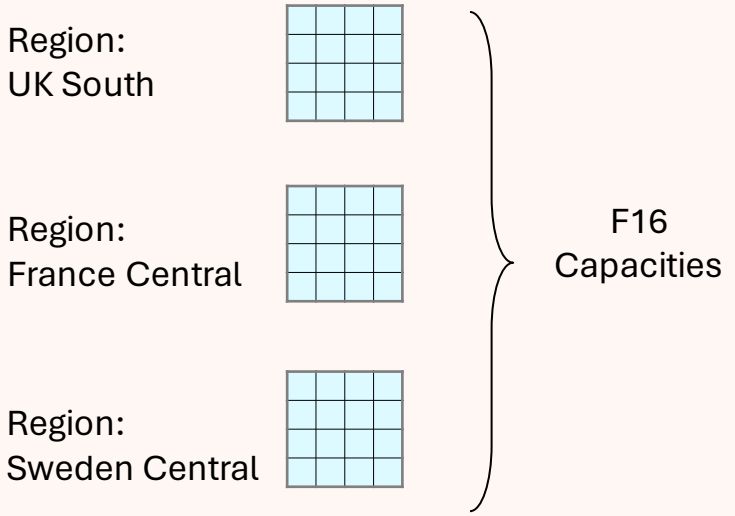
Scale Horizontally

Scale horizontally using the benefits of modular design for hardened isolation and governance



Regional Availability

Use different capacities for different regions to support GDPR / Data residency requirements



Provisioning and Deploying Capacities

Purchased in Azure

- **Purchased** either as a PAYG or RI resource
- **Provisioned with a certain amount of compute** units, analogous to CPU cores.
- The **more capacity units are provisioned, the more load** the capacity can support
  - Multiply SKU size by 30s to match platform evaluation in metrics app
- Capacities are **priced at a fixed hourly rate**, based on capacity units provisioned
- The RI commitment (1-year reserved instance) enjoys a **40% discount**

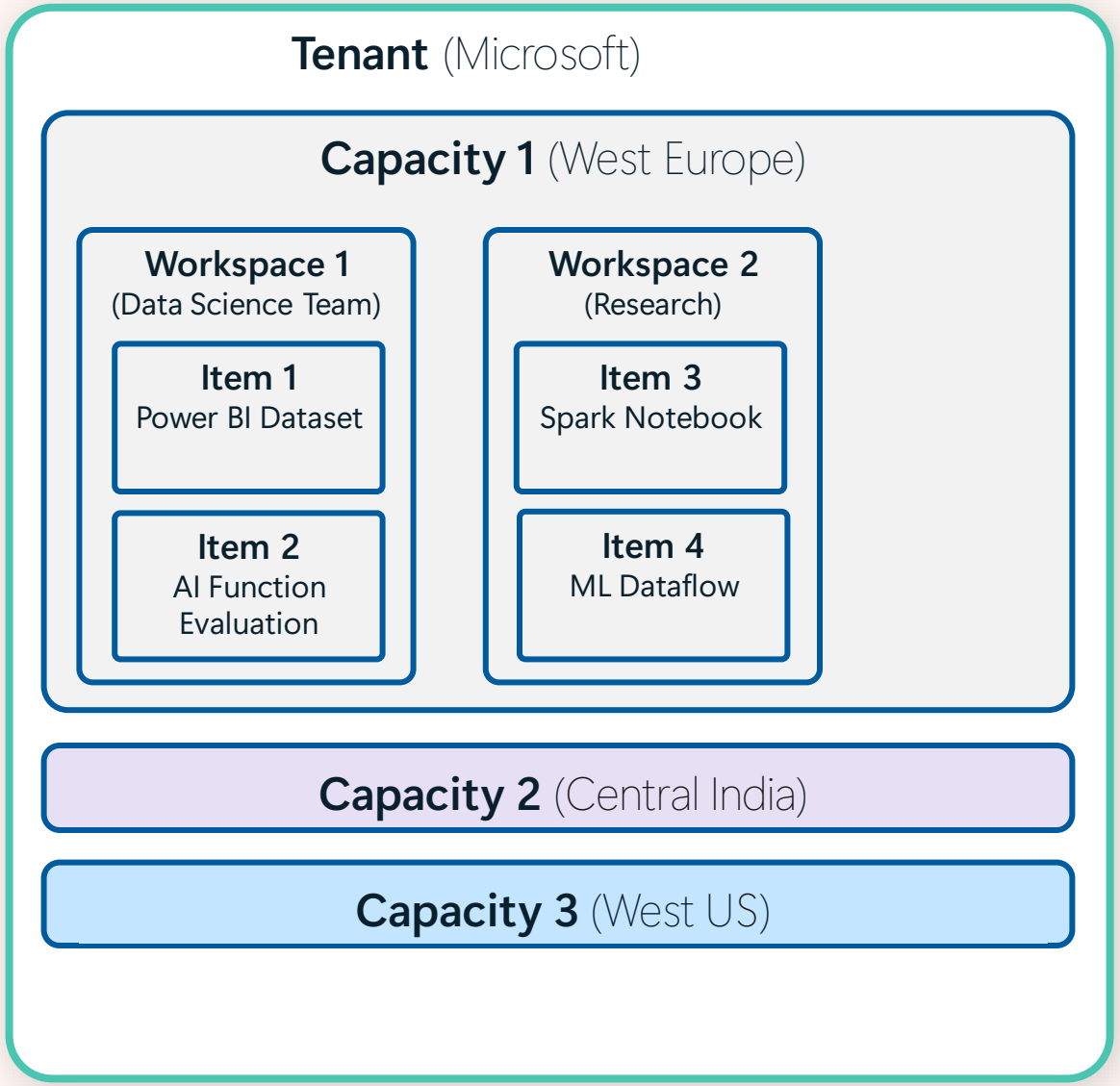
Universal Compute Capacities SKU Sizing

SKU	Capacity Units (CU)	CU's (per 30s)	Power BI SKU	Power BI V-cores
F2	2	60	-	0.25
F4	4	120	-	0.5
F8	8	240	A1	1
F16	16	480	A2	2
F32	32	960	A3	4
F64	64	1920	P1	8
F128	128	3840	P2	16
F256	256	7680	P3	32
F512	512	15360	P4	64
F1024	1024	30720	P5	128
F2048	2048	61440	-	256

Provisioning and Deploying Capacities

Deployed to Regions

- Each capacity **resides in a specific region of the buyers' choice** where both the data & compute reside
- **Workspaces are assigned to a capacity** that provides the compute and storage for all the workspace artifacts
- Multiple capacities can be purchased, deployed and managed by **different owners** residing in a single tenant allowing each business unit to pay for their own consumption



Smoothing intro and benefits

Load stabilization	Smoothing helps capacities self-stabilize by flattening large spikey loads into a smooth load profile, eliminating temporal spikes
Eliminates Scheduling contention	Large/scheduled Jobs usage (not execution) are smoothed over 24 hours, eliminating the need to decide the timing and order of job execution *Performance now, pay later. Demo to follow.*
Bad actor protection	Interactive operations smoothed over several minutes, preventing a single user with a very demanding query from hogging the entire capacity

What is Bursting?

Job acceleration

Bursting provides extra compute resources to jobs and queries to accelerate their completion

Go beyond

The extra resources of bursting allow jobs to **utilize far more resources than "face value"**

Instead of running a job on 64 CU and completing in 60 seconds, bursting could use 256 CUs to complete the job in 15 seconds.

Same amount of work, just completed faster

No hassle, No overload

Bursting is automatic when the system reasons it can accelerate the job by applying extra resources. No settings are required.

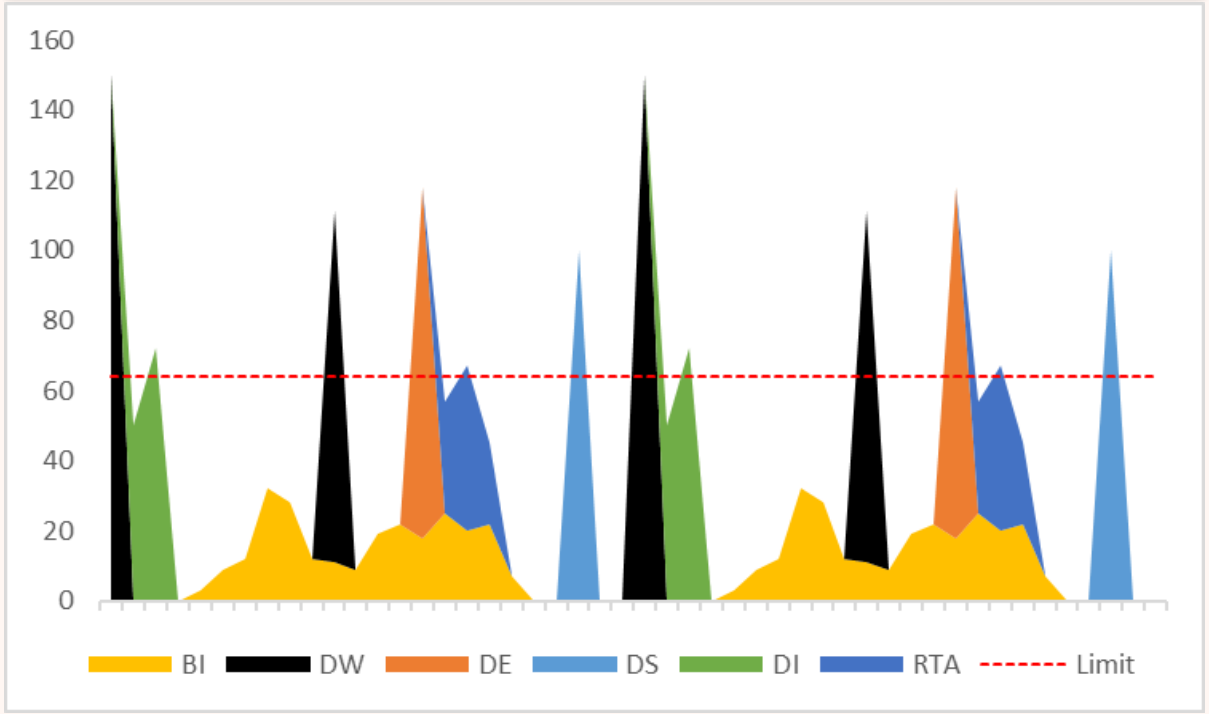
Bursting never causes an overload as the *smoothing* mechanism will always flatten the resource burst

Bursting and smoothing | before and after

Looking at an example of a 64 CU capacity, running multiple workloads over a couple of days…

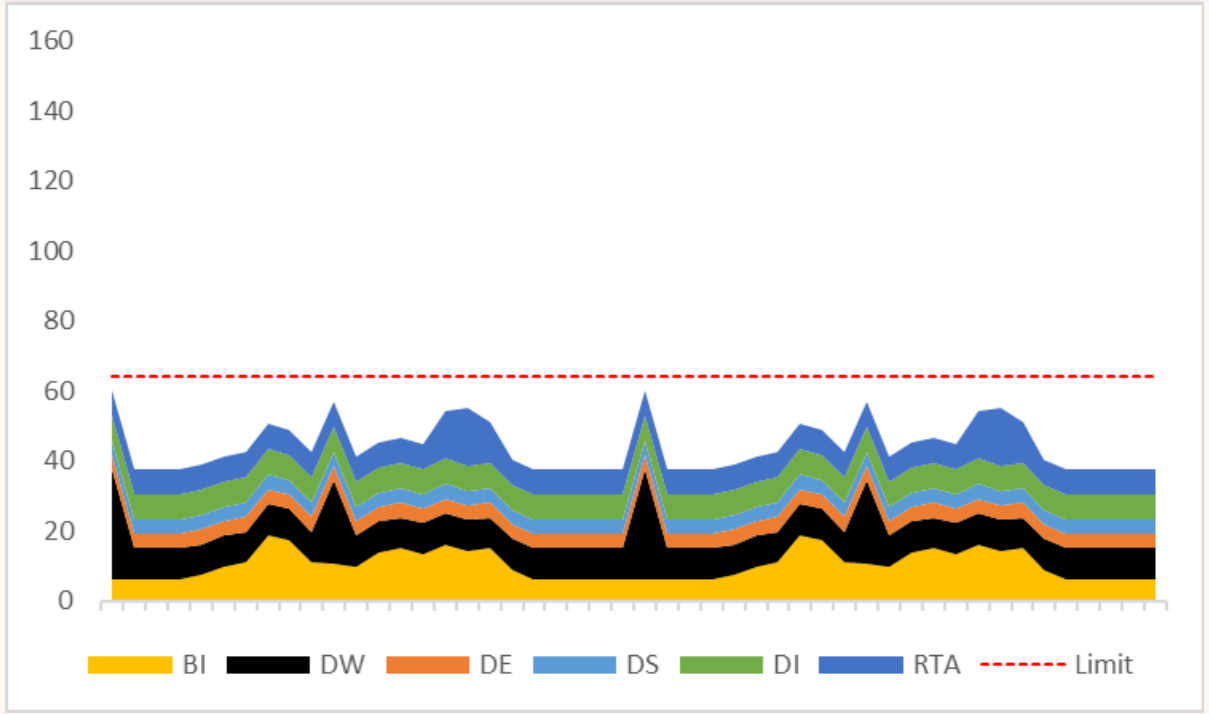
Before Smoothing

- Actual load as executed on the capacity before smoothing
- *Bursting* accelerates jobs execution by resource boosting
- The capacity could be overloaded 25% of the time
- Some of the overloads are more than 2x the limit
- There are periods of no/low usage



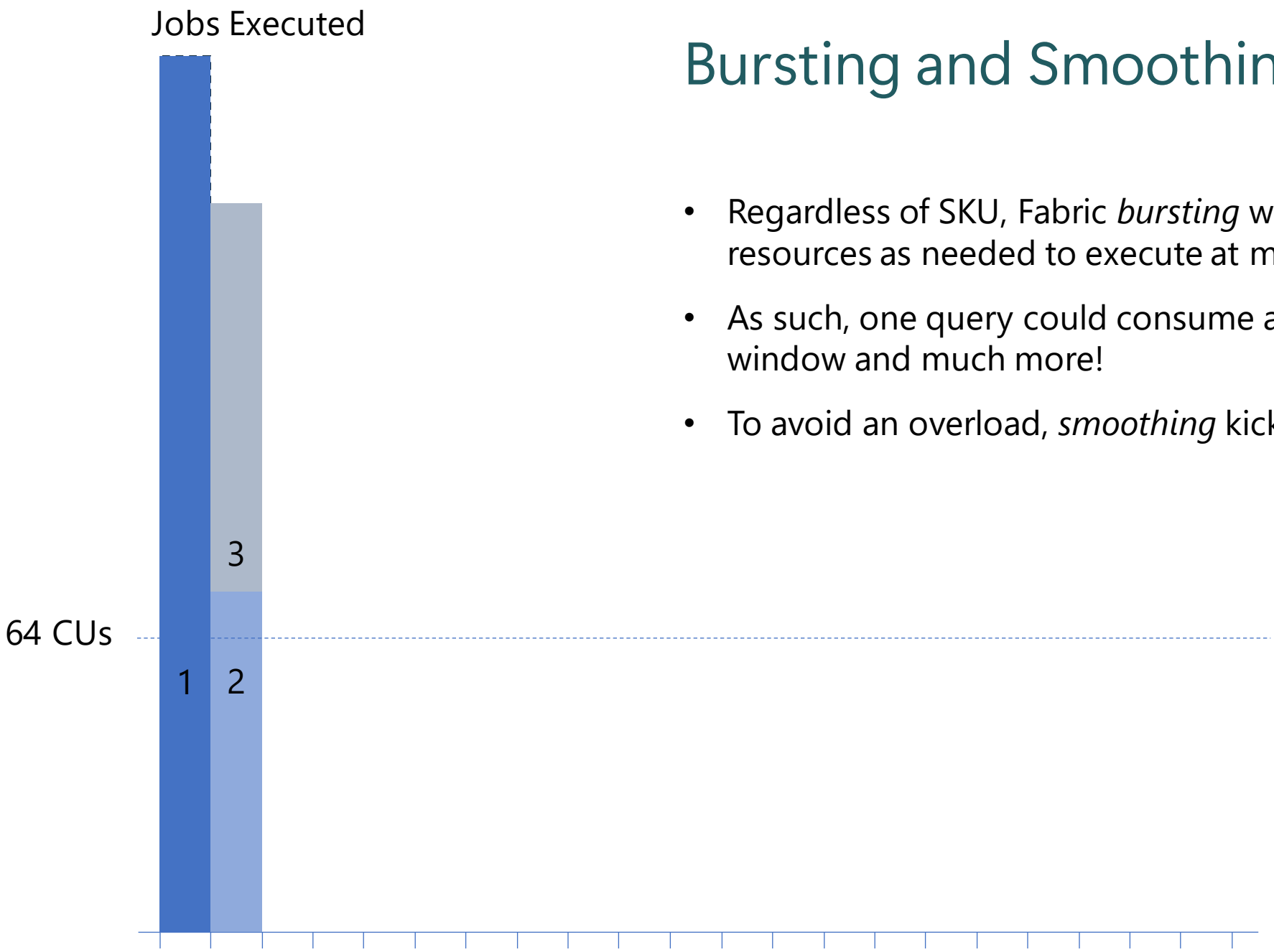
After Smoothing

- Shows the reported load (not runtime execution) against the capacity limits
- There is NO overload, and consumption is more stable
- The smoothing of usage fills in gaps

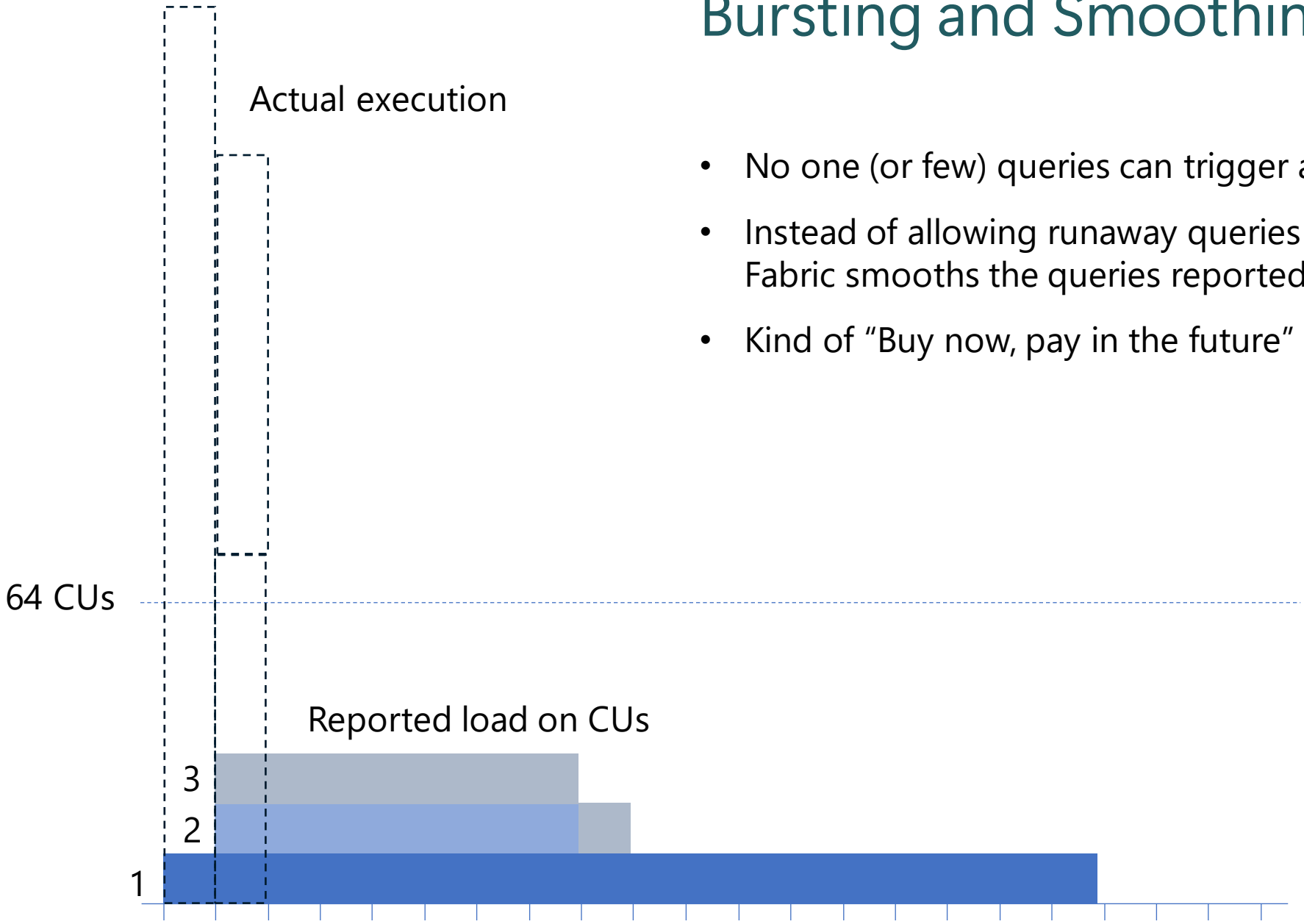


Bursting and Smoothing

- Regardless of SKU, Fabric *bursting* will automatically allocate resources as needed to execute at maximum performance
- As such, one query could consume all the quota of a single time window and much more!
- To avoid an overload, *smoothing* kicks in

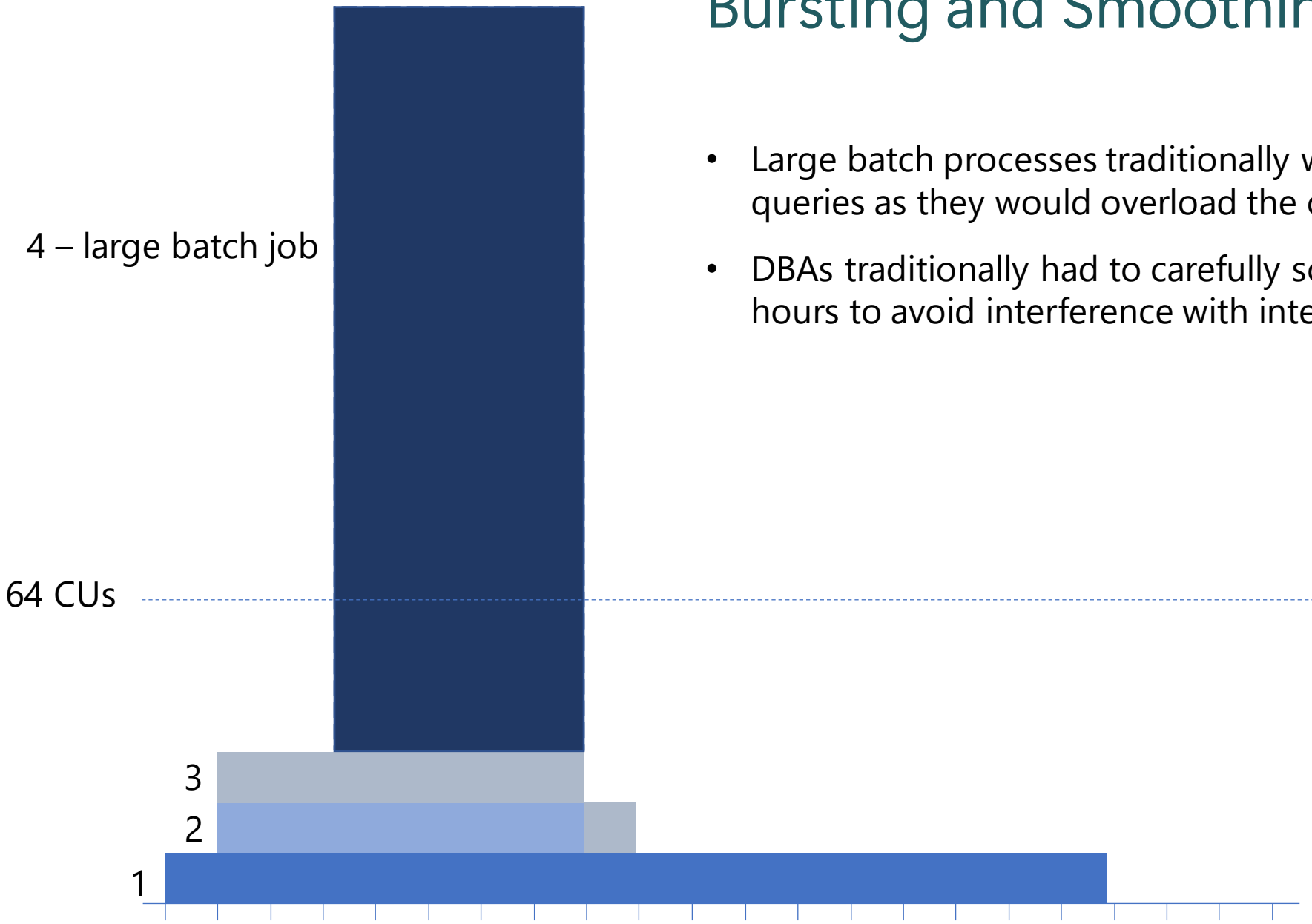


Bursting and Smoothing



- No one (or few) queries can trigger an overload
- Instead of allowing runaway queries to create a local overload, Fabric smooths the queries reported usage to future time windows
- Kind of “Buy now, pay in the future” installment plan

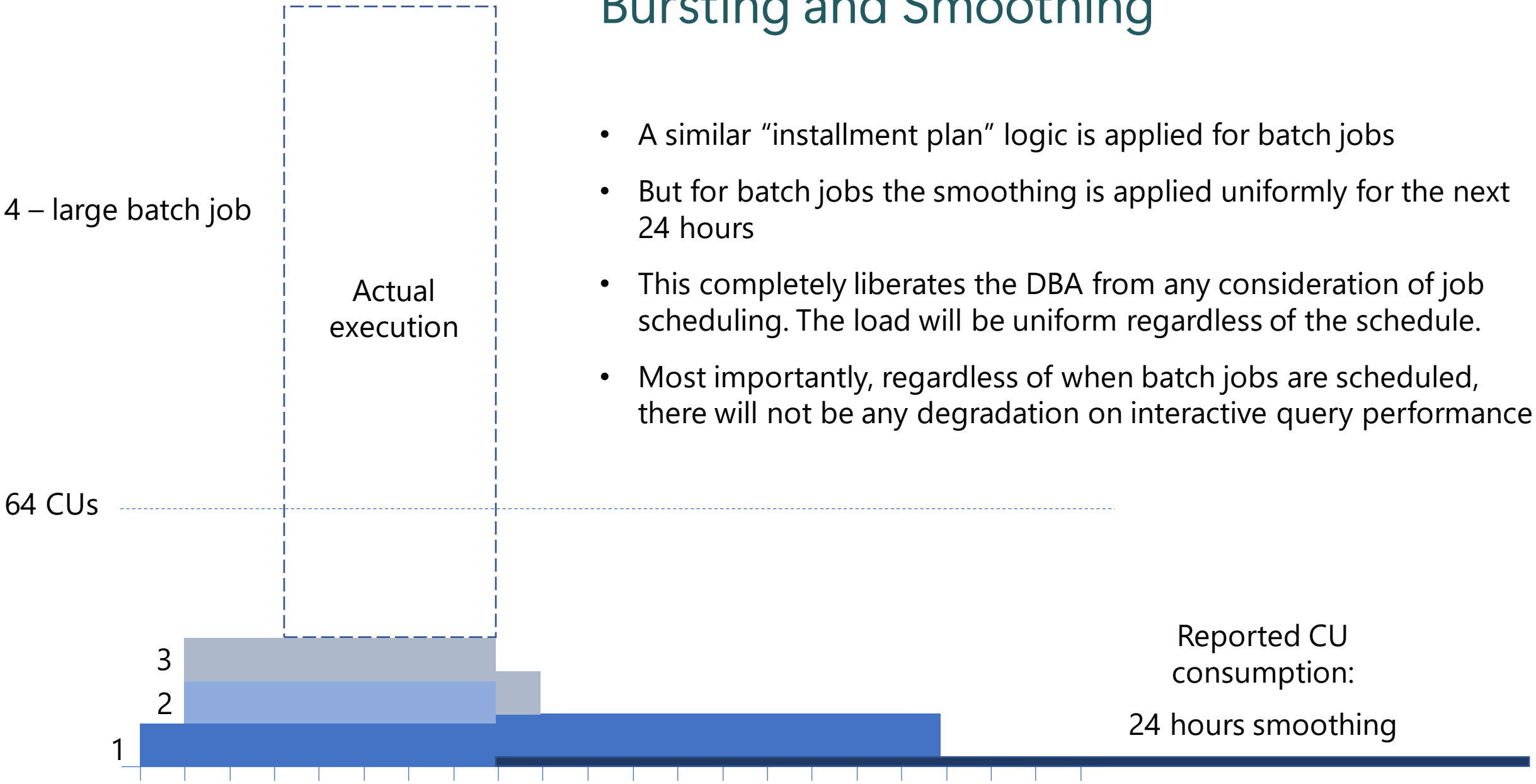
Bursting and Smoothing



- Large batch processes traditionally were a threat to interactive queries as they would overload the compute resource
- DBAs traditionally had to carefully schedule these jobs to off-hours to avoid interference with interactive user experiences

Bursting and Smoothing

- A similar “installment plan” logic is applied for batch jobs
- But for batch jobs the smoothing is applied uniformly for the next 24 hours
- This completely liberates the DBA from any consideration of job scheduling. The load will be uniform regardless of the schedule.
- Most importantly, regardless of when batch jobs are scheduled, there will not be any degradation on interactive query performance

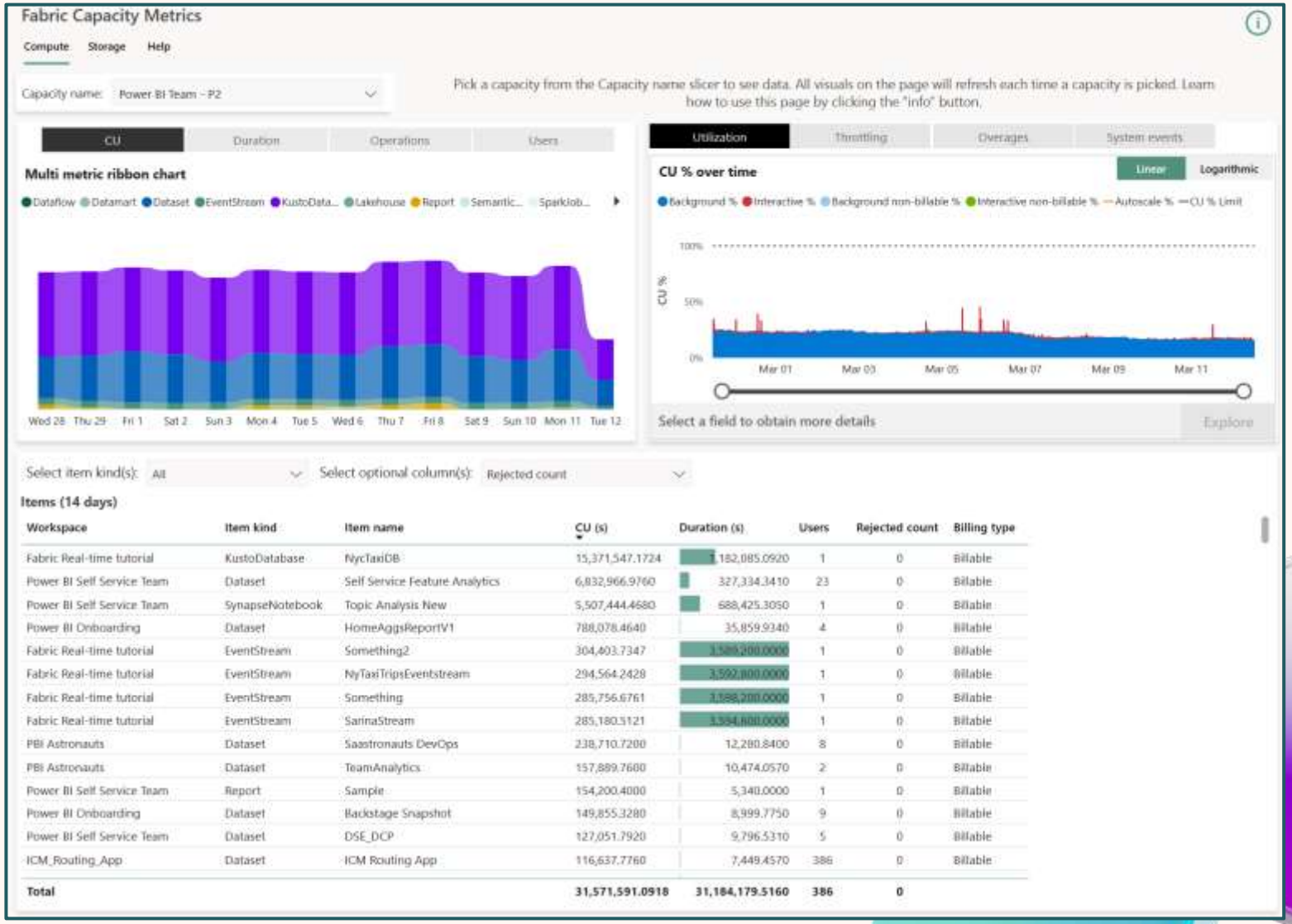


Monitoring with Capacity Metrics

Capacity Metrics

Monitor Capacities and Plan capacity scale-up with confidence

- Tenant wide visibility into capacity usage for all Fabric experiences
- Identify resource usage trends and their impact to autoscale & throttling
- View preview workload usage alongside production workloads to make data-driven capacity sizing decisions

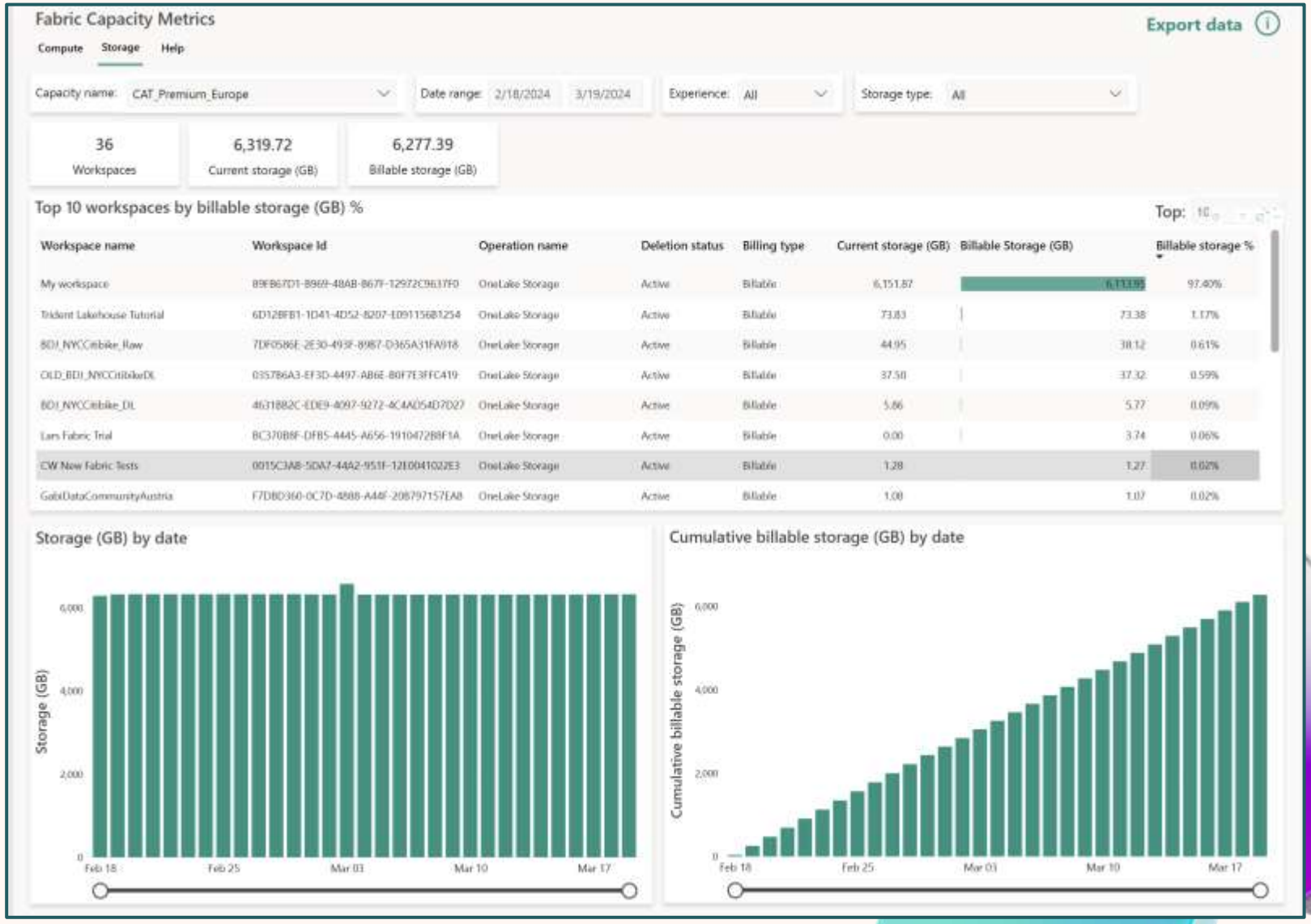


Capacity Metrics

Monitor OneDrive consumption

Measure the trends of workspace storage consumption against capacity limits, by day or hour

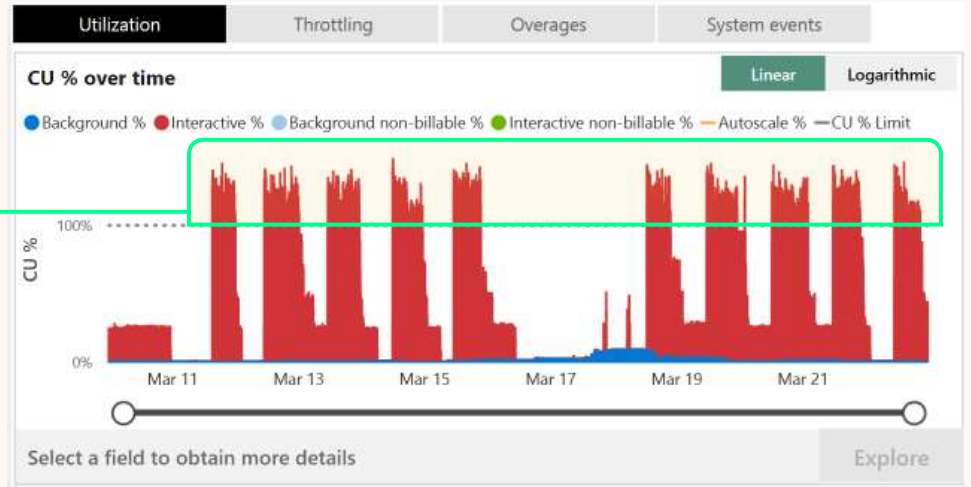
Reconcile costs with internal chargeback processes

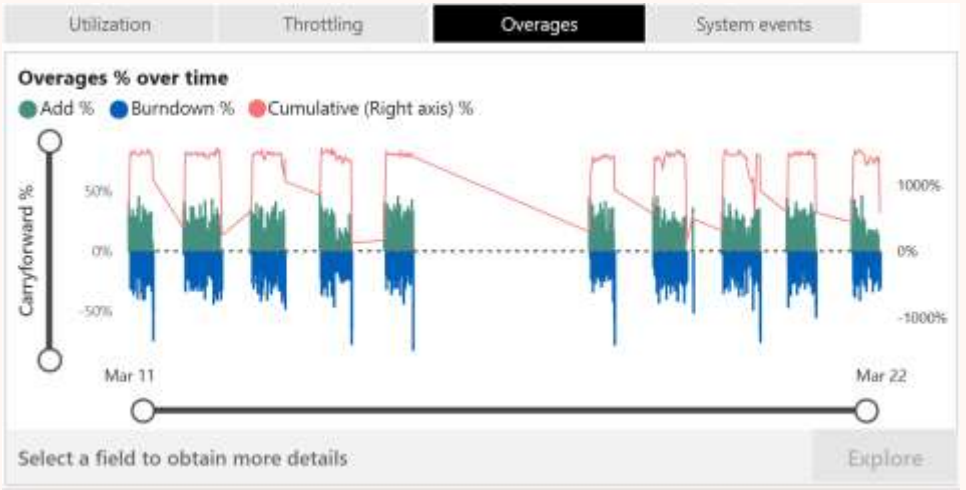


Capacity Throttling Policies

Throttling intro

- Throttling is the platform policy for managing consumption that exceeds throughput is provided by SKU choice
- When workloads exceed the throughput of a capacity a cumulative debt is tracked to be burned down
- Cumulative debt is used to determine throttling policies and is burned down when resources are free

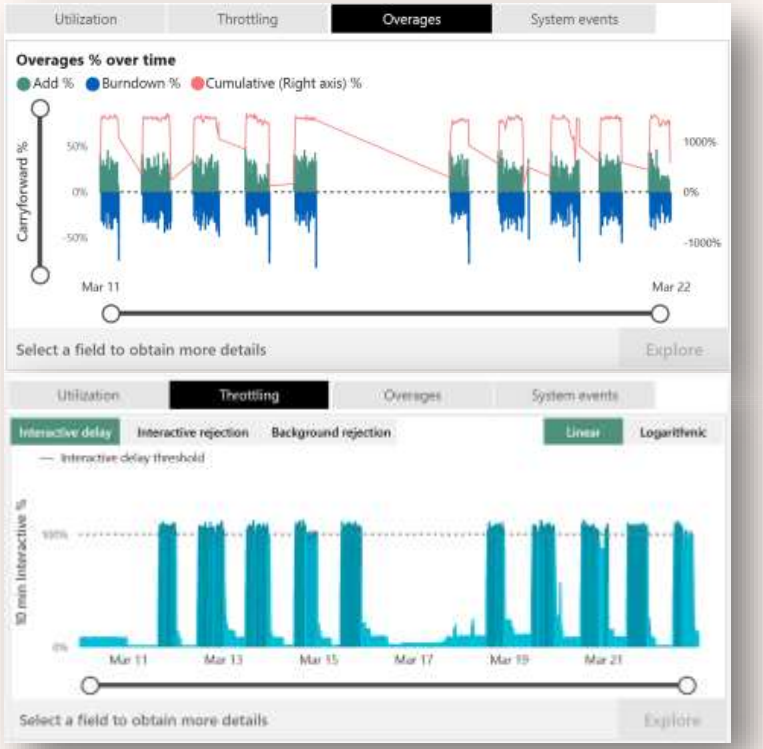




Overage Operation	Description
Overages - Added	• Timepoint when job requests exceed the throughput of a capacity, overages are was added to the cumulative buffer to burn down. • This graph simplifies identification of the optimal timepoint to load timepoint drill to analyze the user operations that contributed to an overage.
Overages - Burndown	• Overages being reconciled when future capacity is free to burn down
Overages - Cumulative	• The total amount of queued work on the capacity to be burned down in the future when the capacity is not fully utilized

Capacity throttling evolution for Fabric

- For Fabric, throttling policies were refined to deliver multiple benefits
  - **Reduced throttling** for capacities that only experience occasional spikes
  - **Added overage protection** – rejection policies prevent overloaded capacities from irrecoverable overload
  - **Optimizations for long-running jobs:** We're optimizing the platform for long-running jobs, so if a job exceeds capacity limits, it will run to completion and the overage will be burned down against future capacity

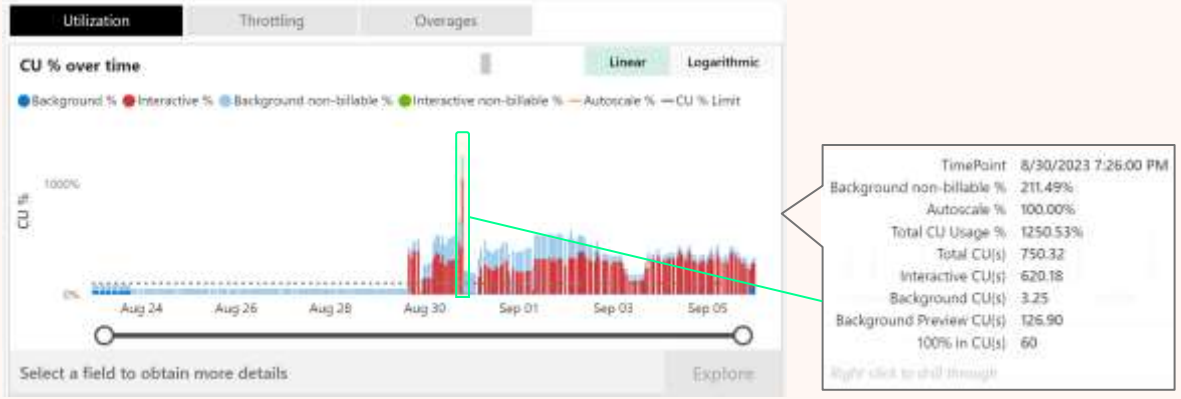


Smoothed Capacity - Future Use	Platform Policy	Customer Impact
<= 10m	Overage Protection	Jobs can consume 10 minutes of future capacity use without throttling
> 10m → <= 60m	Interactive Delay	User requested interactive type jobs will be throttled
> 60m → <= 24h	Interactive Rejection	User requested interactive type jobs will be rejected
> 24h	Background Rejection	User Scheduled background jobs will be rejected from execution

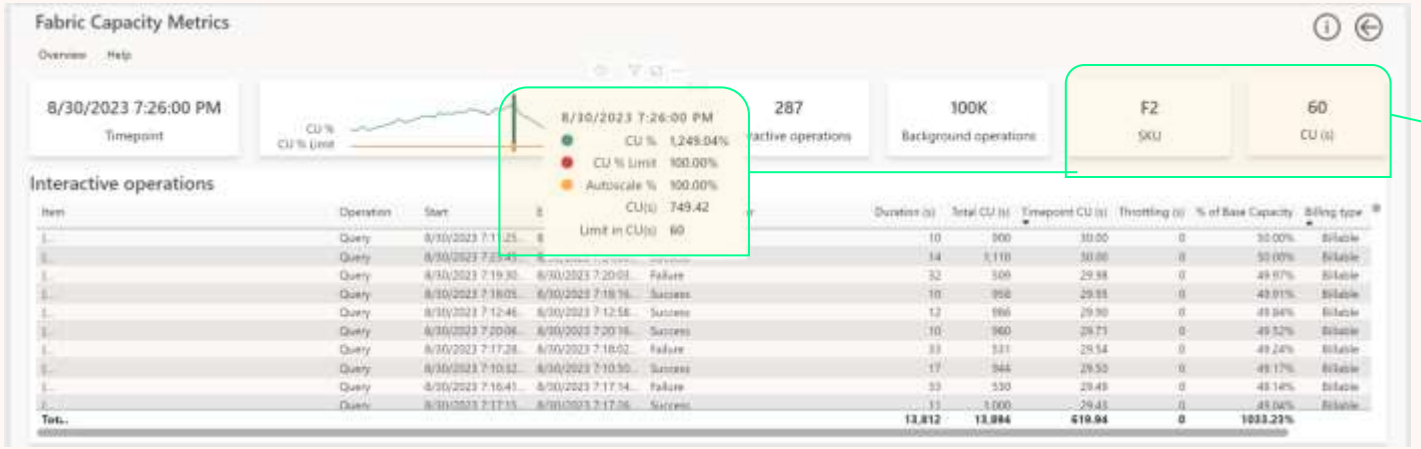
Capacity Planning with Capacity Metrics

Capacity planning case study - measurement

Start with a test or trial capacity to evaluate the load of specific Fabric Experiences i.e., Power BI Datasets, Spark Notebooks or a Datawarehouse



If usage is above the current capacity limits , choose the desired utilization rate to accommodate via capacity scale up



Load Capacity Metrics timepoint drill to analyze :

- Total CU's consumed : 749 CU(s)
- Capacity Size : (F2)
- CU(s) available on your capacity : 60 CU(s)

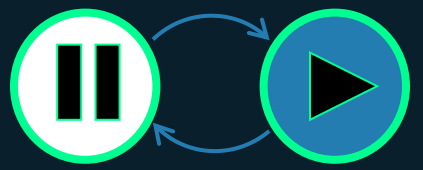
Capacity planning case study – SKU selection

To accommodate a 749 CU(s) load the admin can purchase an F32 capacity providing 960 CU(s) of throughput

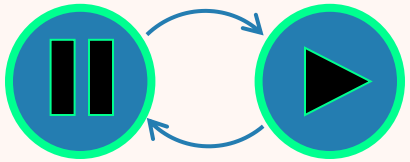
Universal Compute Capacities SKU Sizing

SKU	Capacity Units (CU)	CU's (per 30s)	Power BI SKU	Power BI V-cores
F2	2	60	-	0.25
F4	4	120	-	0.5
F8	8	240	A1	1
F16	16	480	A2	2
F32	32	960	A3	4
F64	64	1920	P1	8
F128	128	3840	P2	16
F256	256	7680	P3	32
F512	512	15360	P4	64
F1024	1024	30720	P5	128
F2048	2048	61440	-	256

Pausing and Resuming Capacities



Introduction to Pausing and Resuming Capacities



Overview and Benefits

Pause and Resume lets you manage compute costs on F SKU capacities by suspending the execution of all workloads running on the capacity

- When a capacity administrator pauses a capacity:

Workloads stop execution

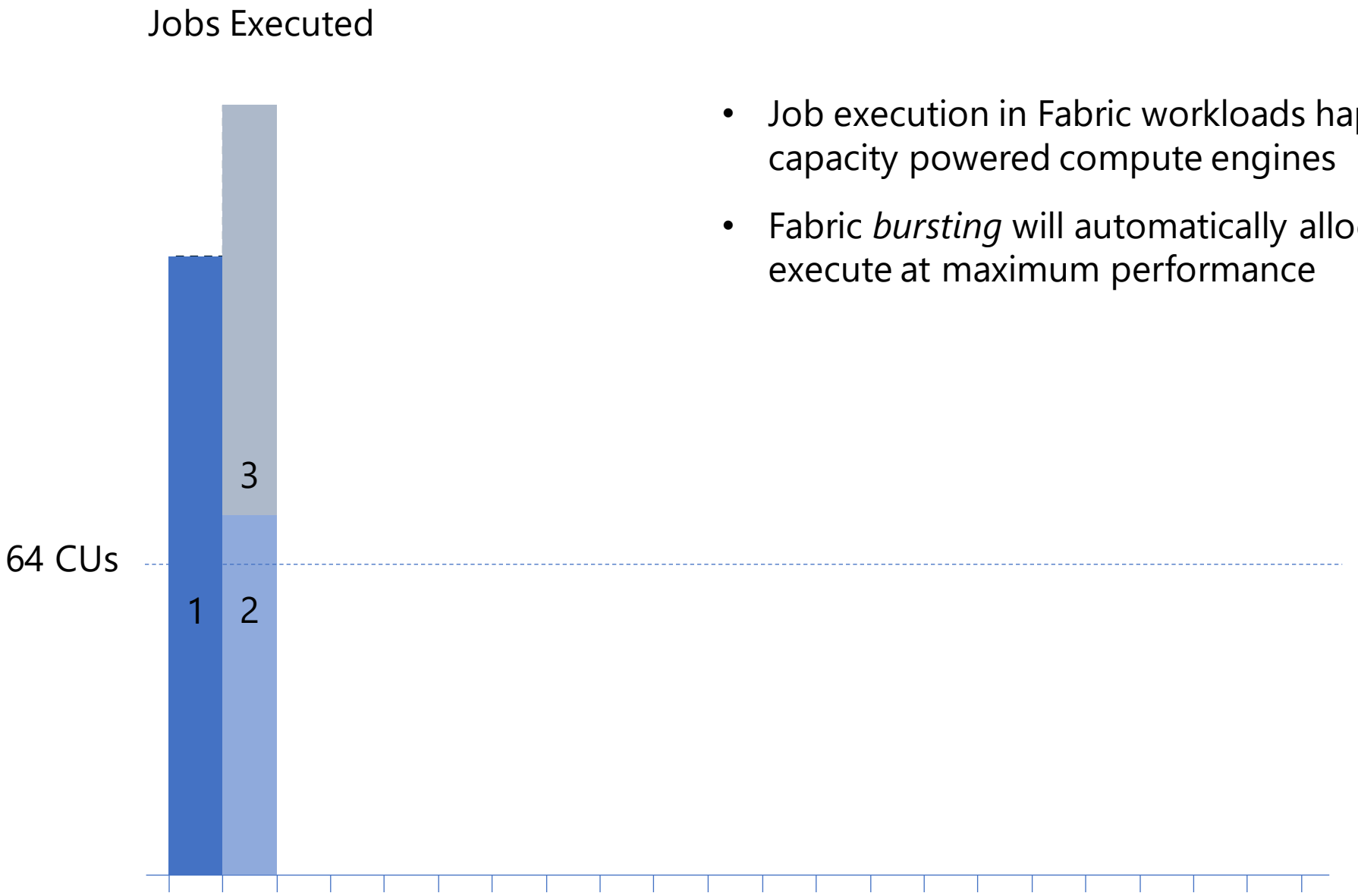
New requests are not run

Smoothed usage will be reconciled

(details in the next demo)

Note: OneLake storage will remain active and billable while a capacity is paused

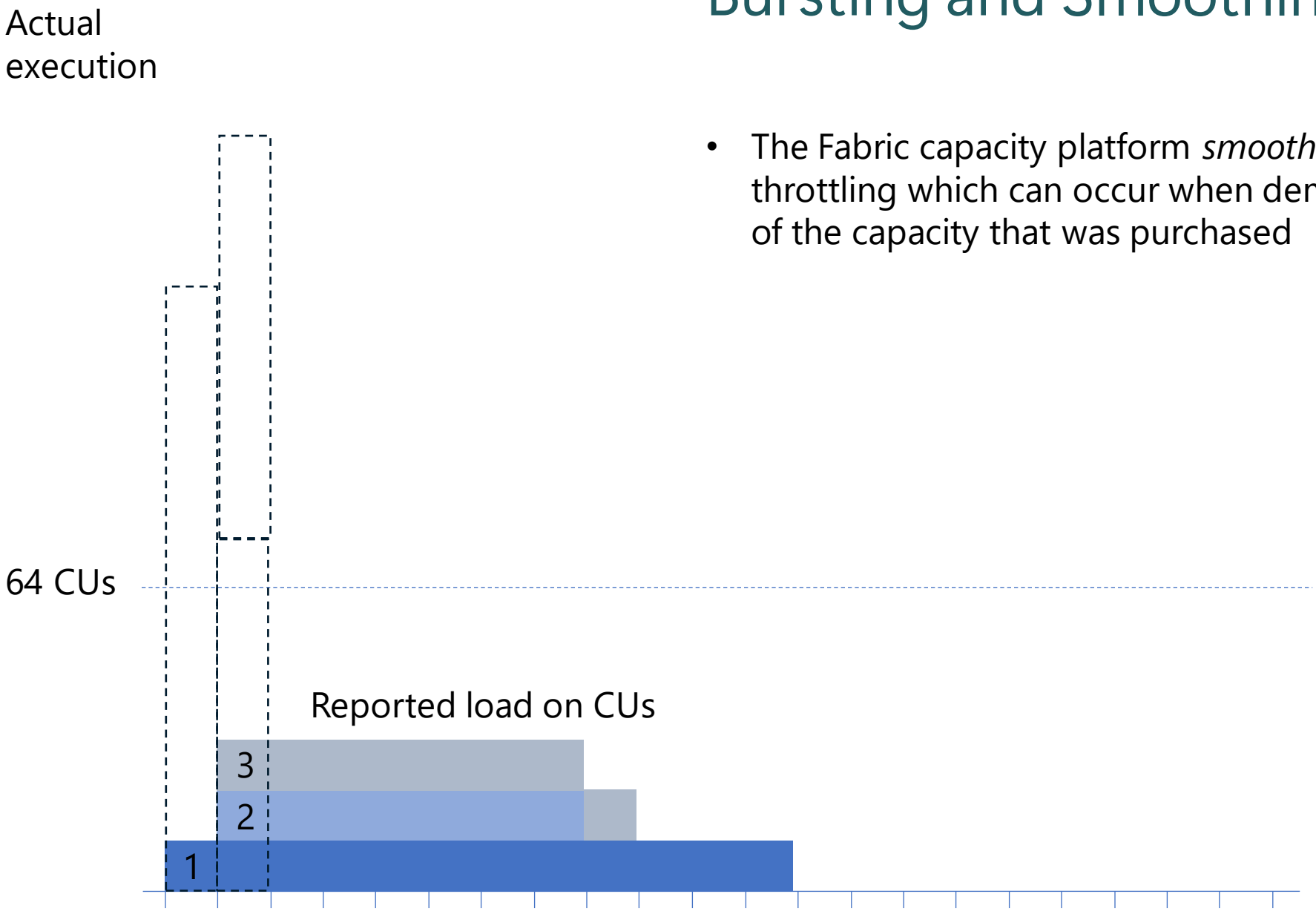
Bursting and Smoothing



- Job execution in Fabric workloads happens on-demand via capacity powered compute engines
- Fabric *bursting* will automatically allocate resources as needed to execute at maximum performance

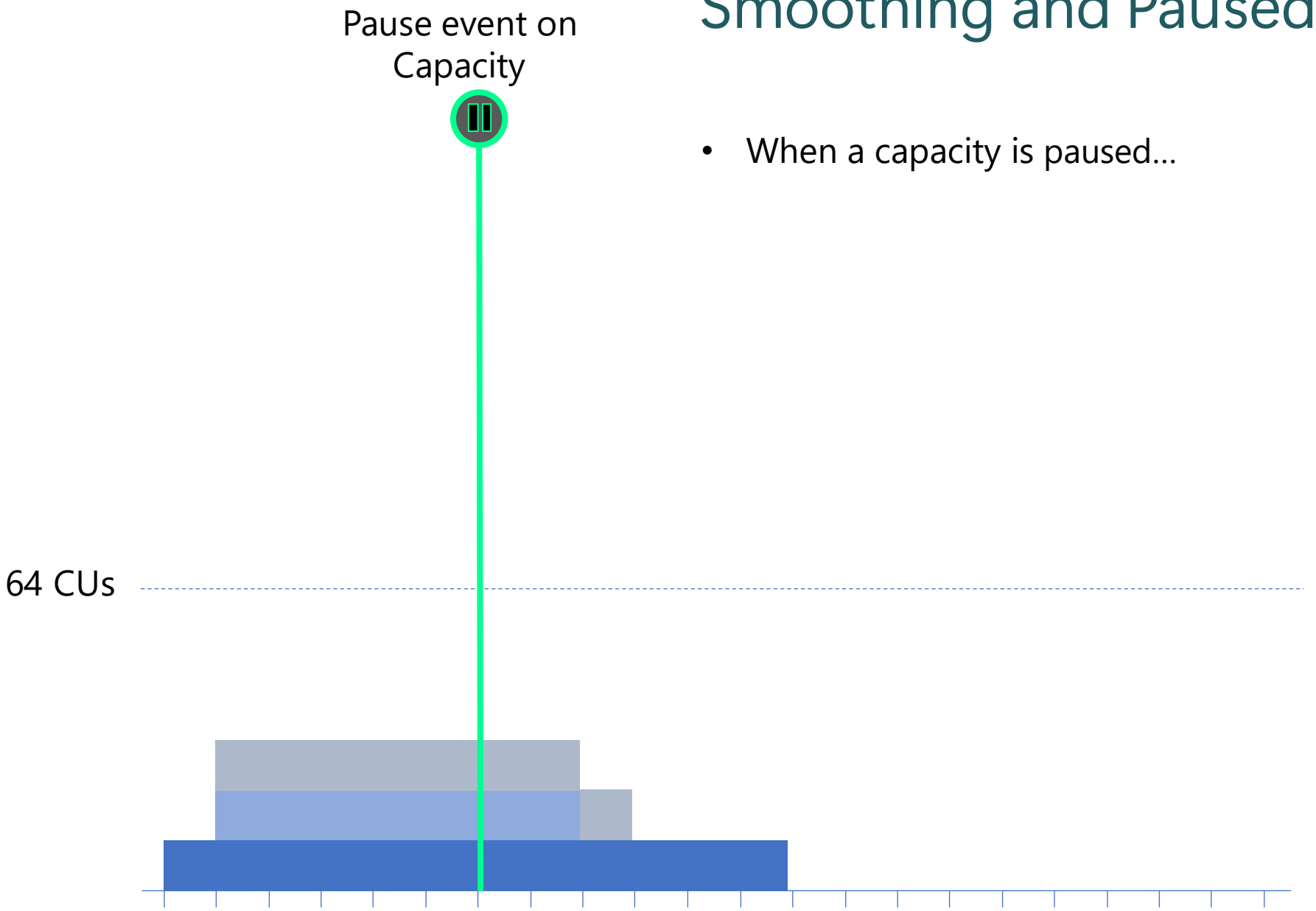
Bursting and Smoothing

- The Fabric capacity platform *smooths* usage out to reduce throttling which can occur when demand exceeds the throughput of the capacity that was purchased

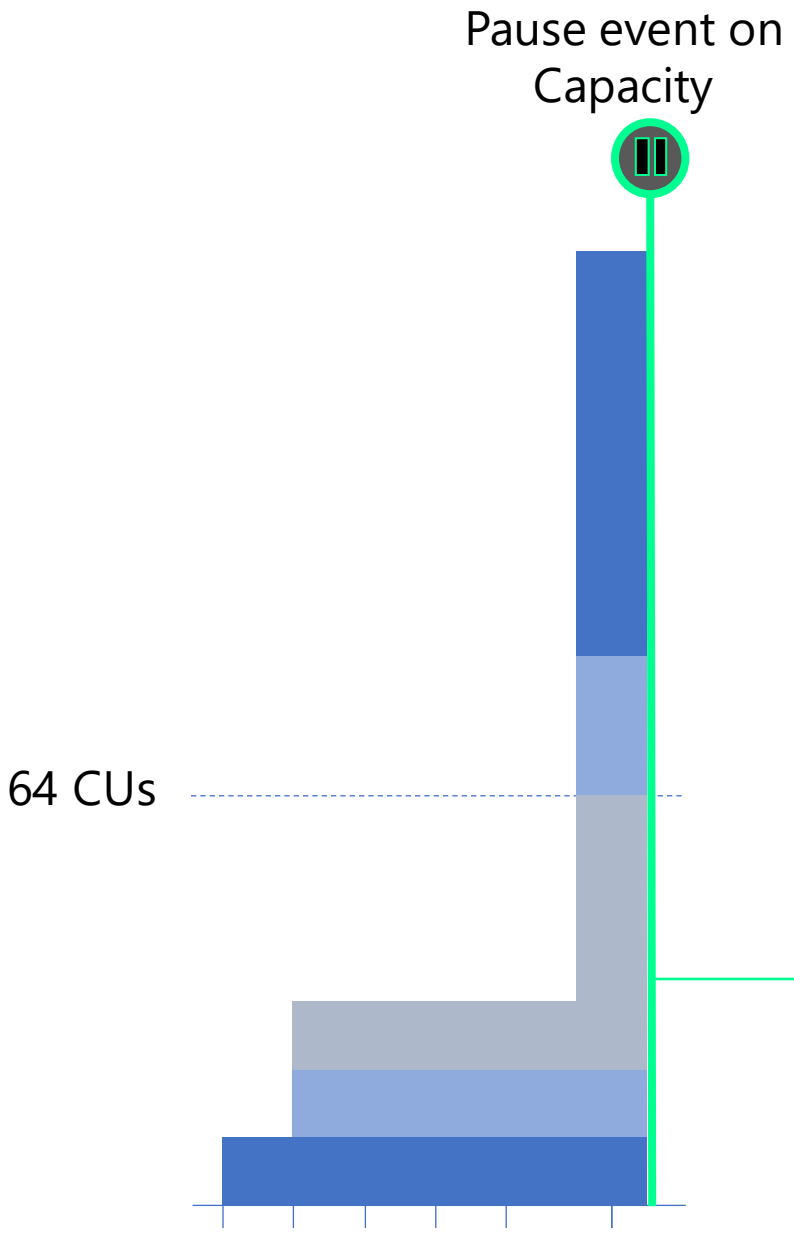


Smoothing and Paused Capacities

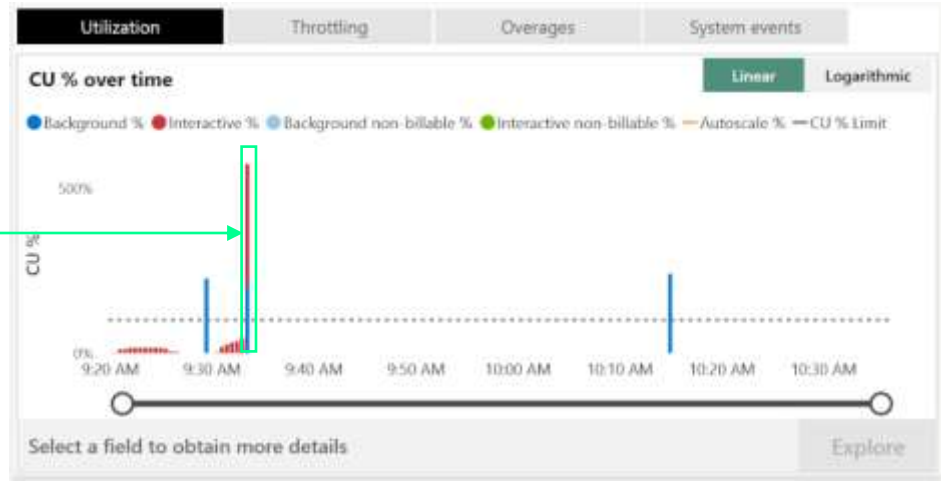
- When a capacity is paused…



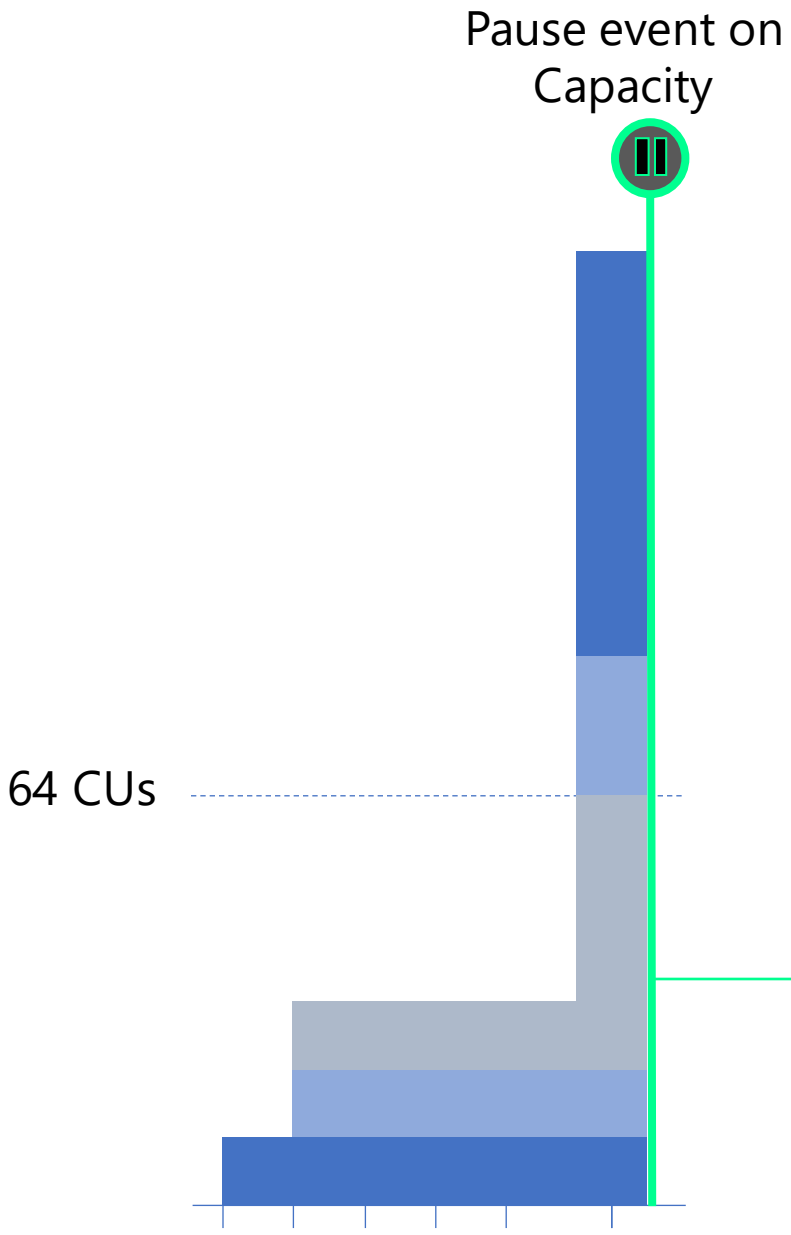
Smoothing and Paused Capacities



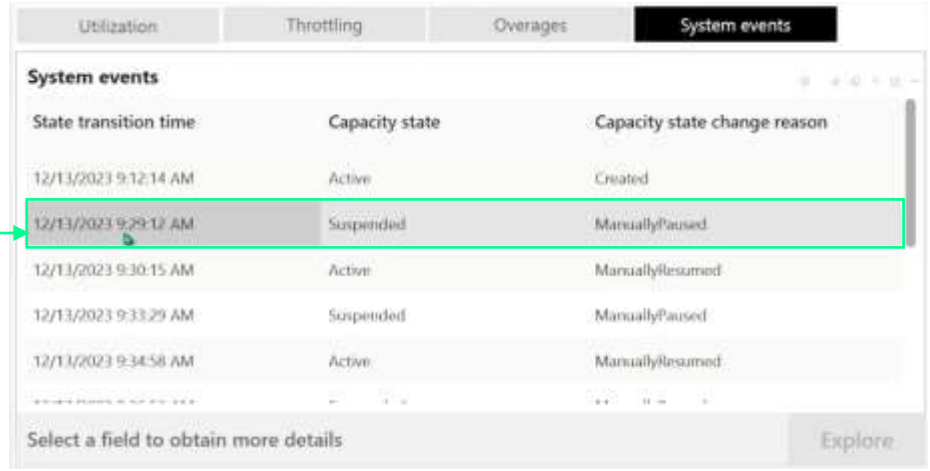
- When a capacity is paused…
- Usage that was smoothed into the future will be “reconciled” and charged against the capacity at the timestamp the capacity was paused
- Reconciled usage will show up as a spike in capacity metrics



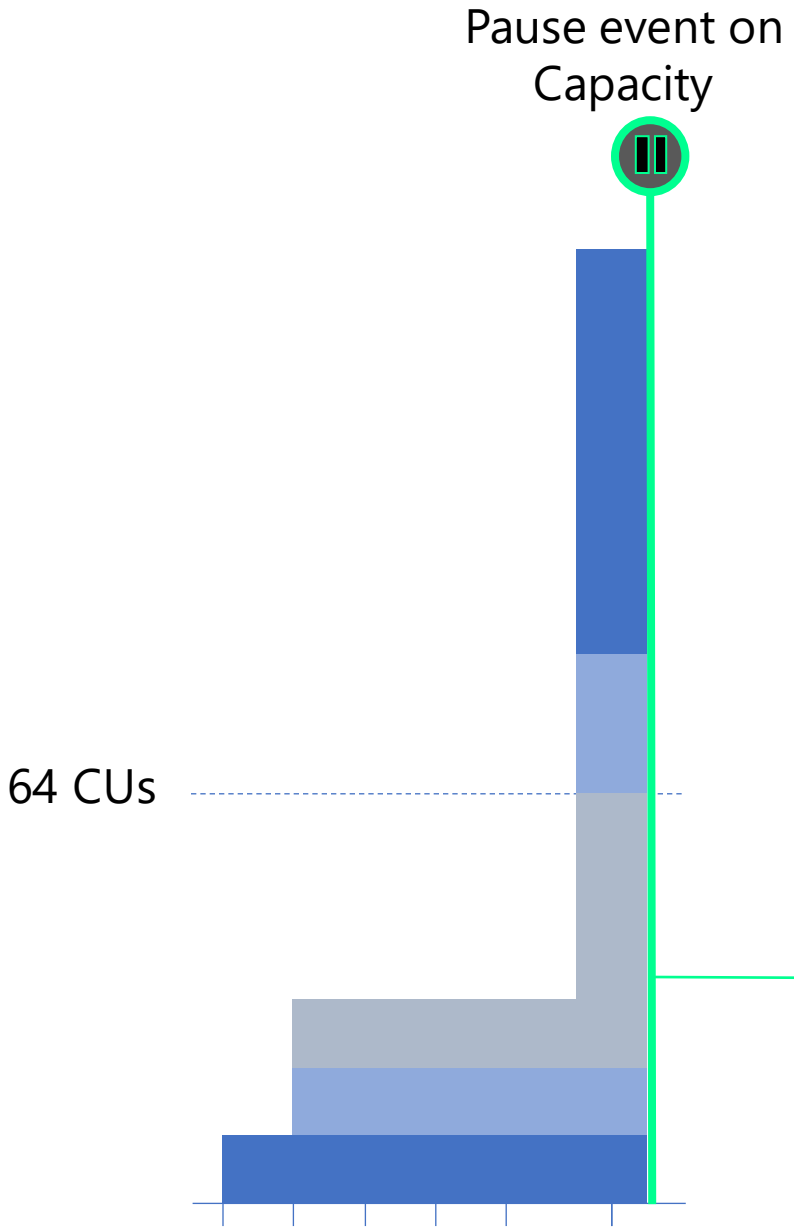
Smoothing and Paused Capacities



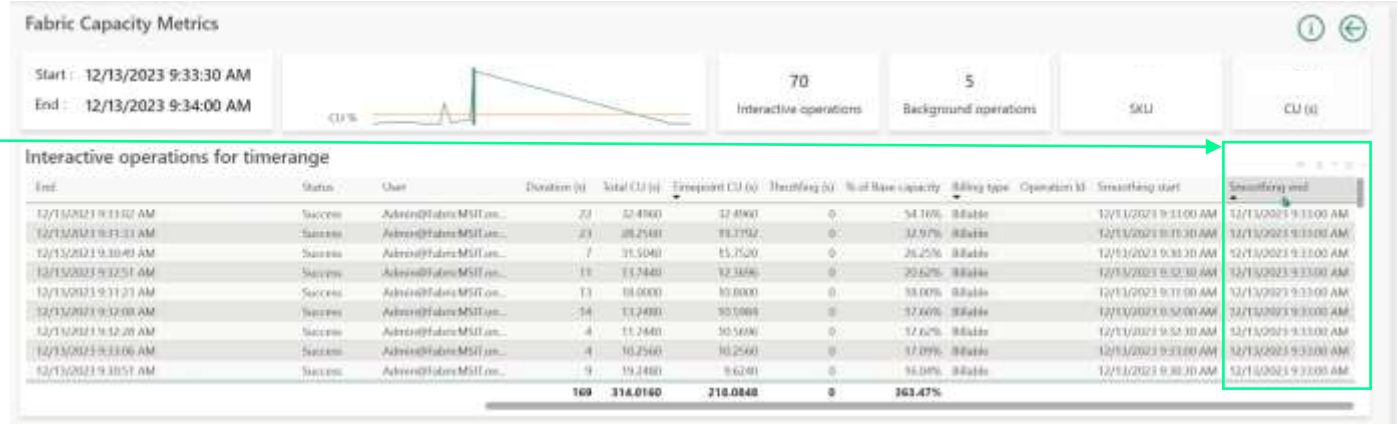
- When a capacity is paused…
- Usage that was smoothed into the future will be “reconciled” and charged against the capacity at the timestamp the capacity was paused
- Pause events can be viewed in the new System events tab



Smoothing and Paused Capacities



- When a capacity is paused…
- Usage that was smoothed into the future will be “reconciled” and charged against the capacity at the timestamp the capacity was paused
- Pause events timestamp is shown in the smoothing end field in timepoint drill views



Git integration and Deployment Pipelines

Git integration

- Git integration enables source control capabilities for developers to integrate their development processes, tools, and best practices directly into the Fabric platform
- Sync workspaces with Azure DevOps Git directly through Microsoft Fabric

Scenarios enabled by Git

- Backup and version development work
- Revert to previous stages
- Collaborate with others or work alone using git branches
- Apply the capabilities of familiar source control tools to manage Fabric items
  - Commit changes
  - Sync changes from Git
  - Manage conflict resolutions
  - …and more!!

SQL Database Projects

Extension to design, edit, and publish schemas from a source-controlled environment

- Local representation of SQL objects that comprise the schema for a single database such as tables, stored procedures, or functions

- Enables T-SQL analytics engineers with capabilities including
  - Source code integration – Stored in control systems like Git to keep track of changes and perform code reviews easily
  - Database testing – Support the addition of “unit test” to validation functionality of the database
  - Schema validation – “Build” operation can be used to compile and validate T-SQL code in the project

Deployment Pipelines

What are deployment pipelines?

- Deployment pipelines in Fabric provide a tool to manage the life cycle of organizational content
- Enables creators to develop and test content in the service before it reaches end-users in a production environment

What scenarios do deployment pipelines enable?

- Clone content from one stage in the development pipeline to another – typically from dev to UAT/test to production
- Review and compare differences in schema/item definitions between stages

How can I automate deployments?

- Use deployment pipelines REST APIs to deploy content programmatically between stages
- Use new or existing Azure DevOps pipelines in conjunction with APIs

Power BI Projects, themes & DevOps

2

Module 2
Prepare and serve data

Prepare and serve data (40-45%)

Create objects in a lakehouse or warehouse

Ingest data by using a data pipeline, dataflow, or notebook

Create and manage shortcuts

Implement file partitioning for analytics workloads in a lakehouse

Create views, functions, and stored procedures

Enrich data by adding new columns or tables

Copy data

Choose an appropriate method for copying data from a Fabric data source to a lakehouse or warehouse

Copy data by using a data pipeline, dataflow, or notebook

Add stored procedures, notebooks, and dataflows to a data pipeline

Schedule data pipelines

Schedule dataflows and notebooks

Transform data

Implement a data cleansing process

Implement a star schema for a lakehouse or warehouse, including Type 1 and Type 2 slowly changing dimensions

Implement bridge tables for a lakehouse or a warehouse

Denormalize data

Aggregate or de-aggregate data

Merge or join data

Identify and resolve duplicate data, missing data, or null values

Convert data types by using SQL or PySpark

Filter data

Optimize performance

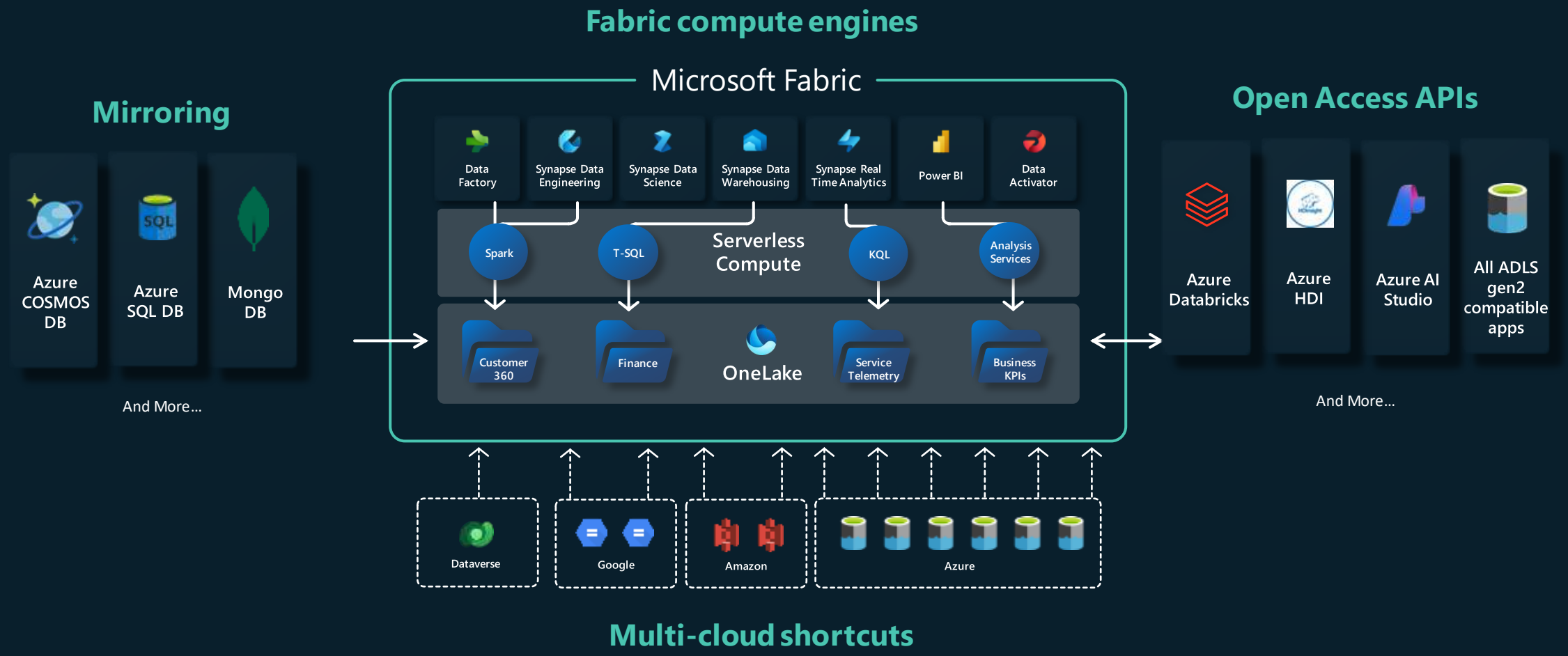
Identify and resolve data loading performance bottlenecks in dataflows, notebooks, and SQL queries

Implement performance improvements in dataflows, notebooks, and SQL queries

Identify and resolve issues with Delta table file sizes

All roads lead to OneLake

Creating Data Gravity in OneLake



Dataflow Gen2 benefits and limitations

Benefits:

- Allow self-service users access to a subset of data warehouse separately.
- Optimize performance with dataflows, which enable extracting data once for reuse, reducing data refresh time for slower sources.
- Simplify data source complexity by only exposing dataflows to larger analyst groups.
- Ensure consistency and quality of data by enabling users to clean and transform data before loading it to a destination.

Limitations:

- Not a replacement for a data warehouse.
- Row-level security isn't supported.
- Fabric capacity workspace is required.

Who is using a Medallion architecture?

‘Uniform data architecture’
From data “Spaghetti to Lasagna”

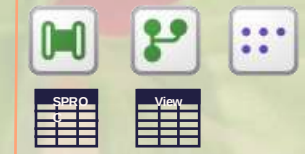
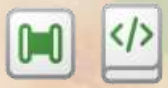
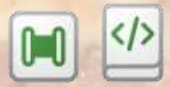
Medallion Architecture

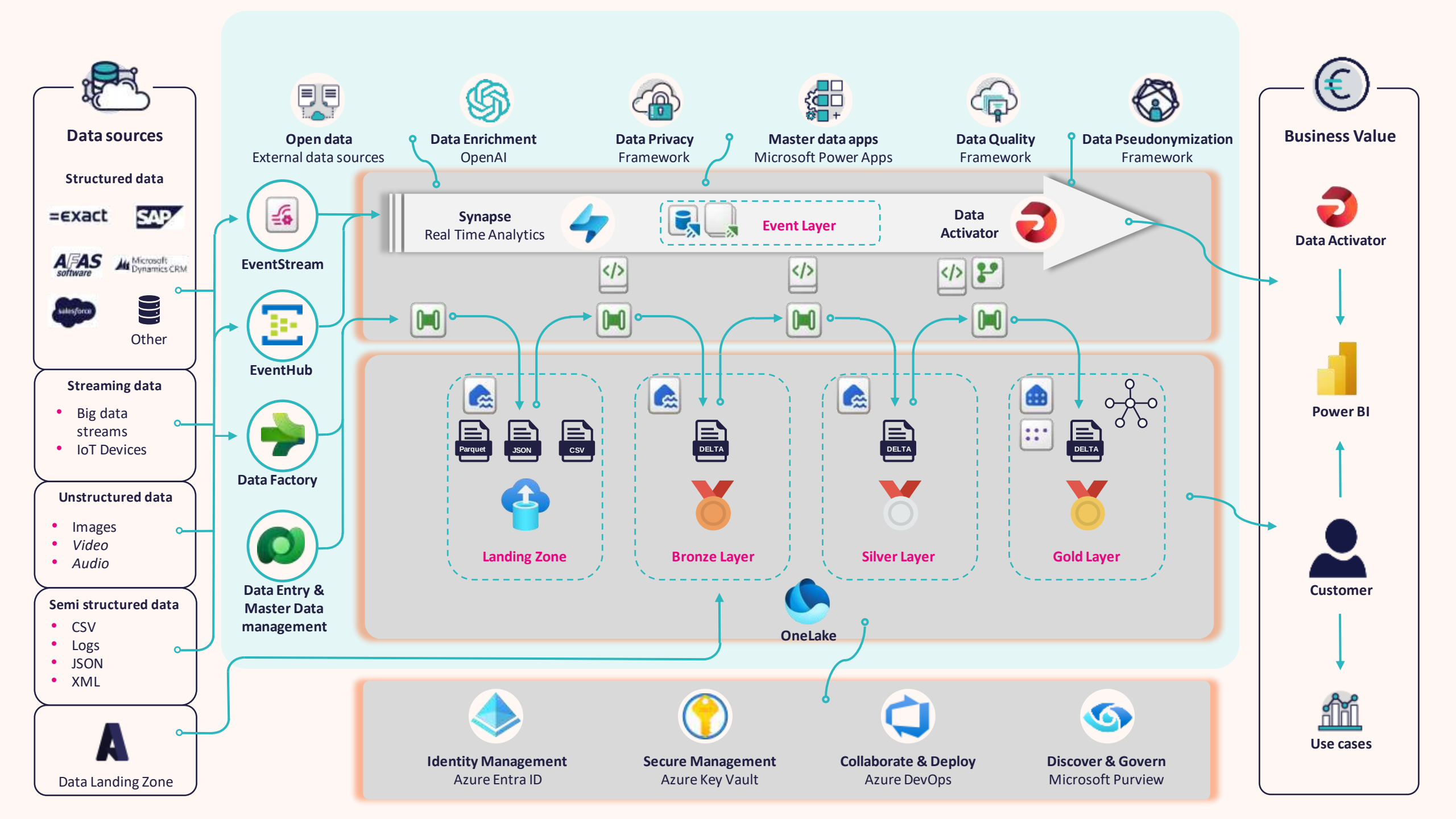
'Data processing in different stages'

Stages

Medallion Architecture

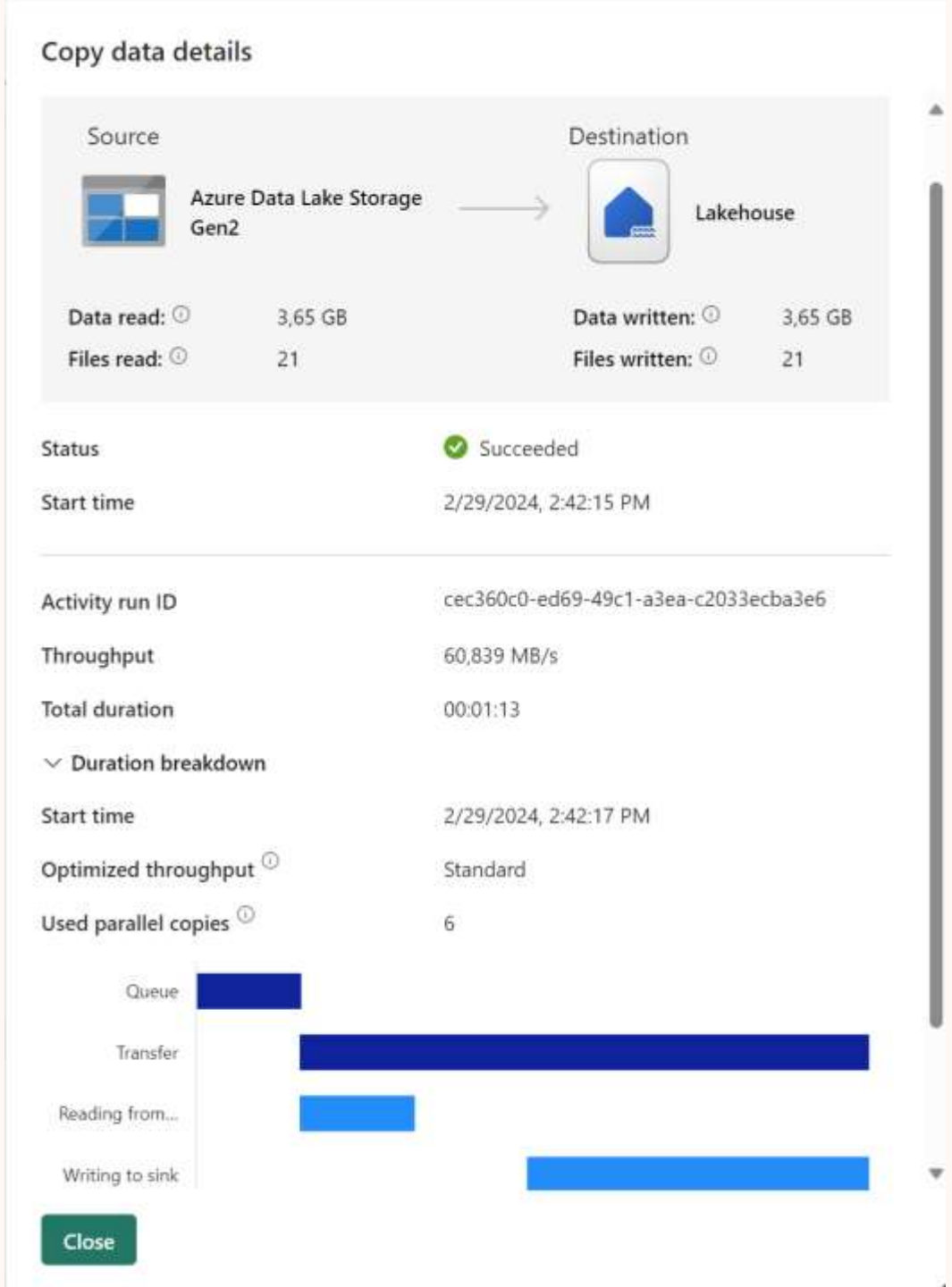
'Data processing in different stages'

Stage:	Definition:	Filetype:	Files/Tables:	Fabric:
Gold layer	• Dimensions & Facts (Star Schema) • Historical Analysis • Business rules • Documentation • Aggregated data • Logical table names			
Silver layer	• Historical Data (Type 1 or 2) • Data quality rules • Data Cleansing • Validated data • No business model/data			
Bronze layer	• Deduplicate data • Add datatypes • Data can be inconsistent • Mostly a copy of the source • Schema			
Landing zone	• Structured data • Unstructured data • Incremental loads • Data as is • Stored in Datetime folder structure • No Schema			


Data sources
Structured data
Other
Streaming data
Big data streams
IoT Devices
Unstructured data
Images
Video
Audio
Semi structured data
CSV
Logs
JSON
XML
Data Landing Zone
Open data
External data sources
Data Enrichment
OpenAI
Data Privacy
Framework
Master data apps
Microsoft Power Apps
Data Quality
Framework
Data Pseudonymization
Framework
EventStream
EventHub
Data Factory
Data Entry & Master Data management
Synapse
Real Time Analytics
Event Layer
Data Activator
Parquet
JSON
CSV
DELTA
DELTA
DELTA
Landing Zone
Bronze Layer
Silver Layer
Gold Layer
OneLake
Identity Management
Azure Entra ID
Secure Management
Azure Key Vault
Collaborate & Deploy
Azure DevOps
Discover & Govern
Microsoft Purview
Business Value
Data Activator
Power BI
Customer
Use cases

Copy Activity

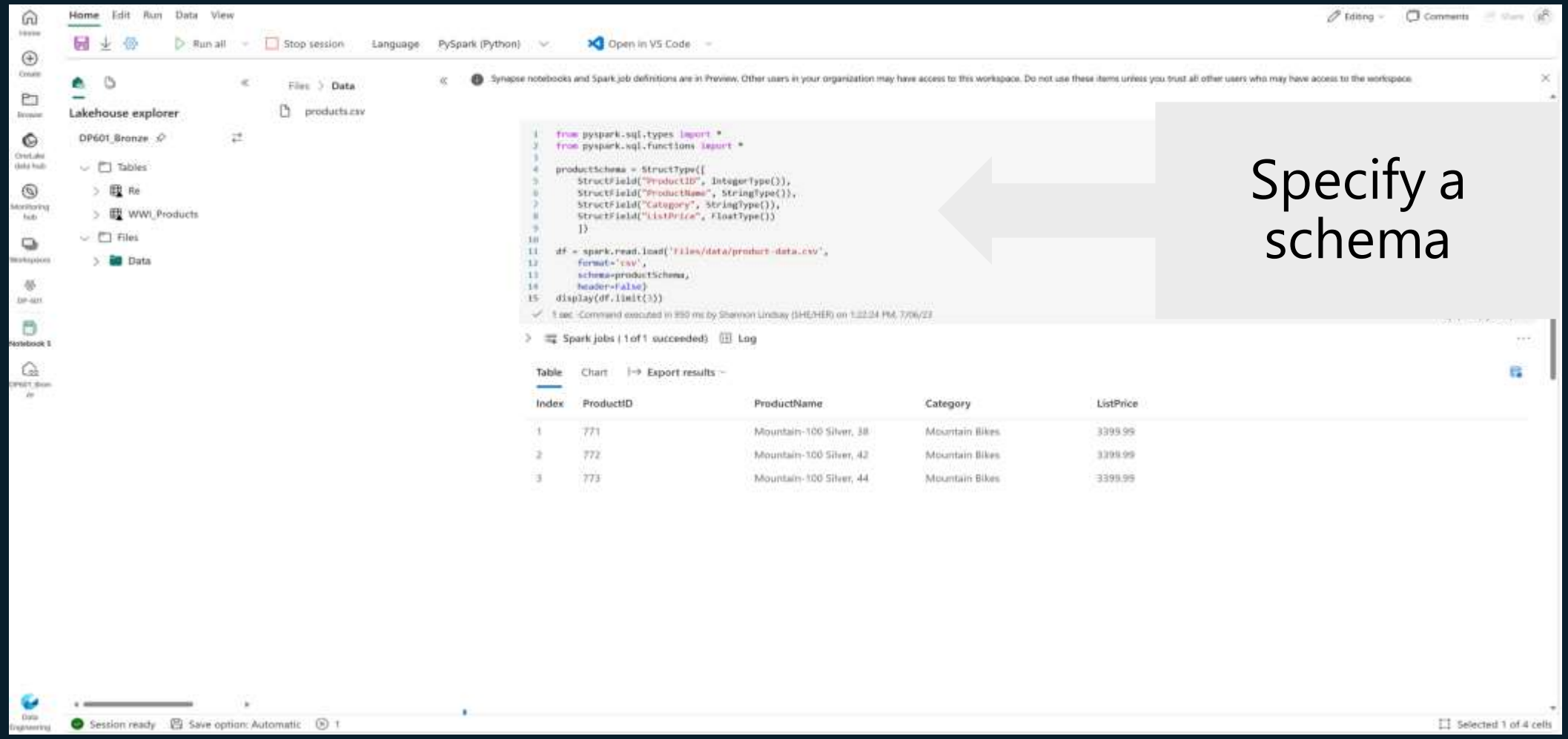
- Queue 11 seconds
- Reading from Source 12 seconds
- Writing to Sink 35 seconds
- Transfer 60 seconds



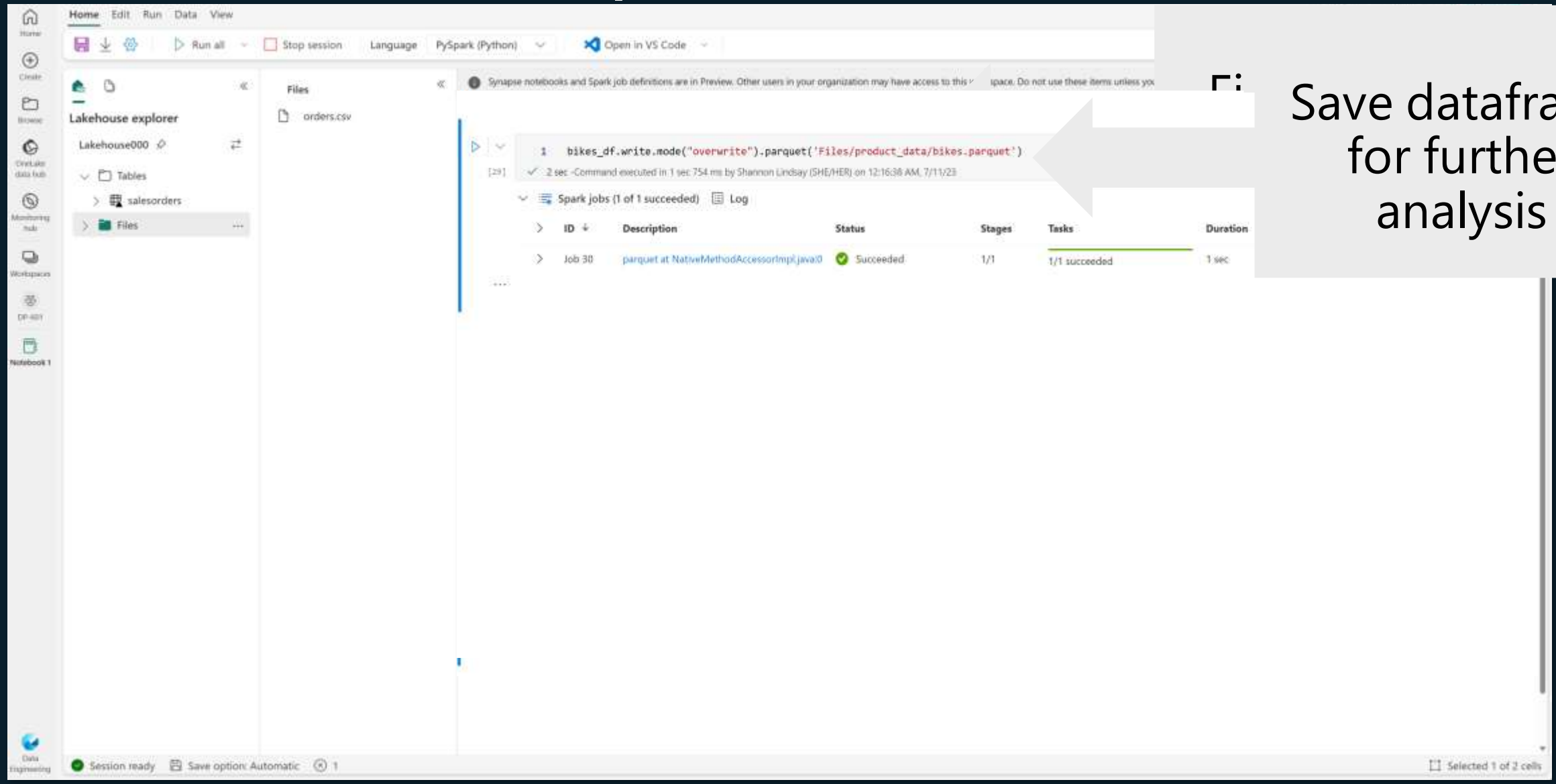
Delta Lake in Microsoft Fabric

Load data in a Spark Dataframe

Schema can be either inferred or specified



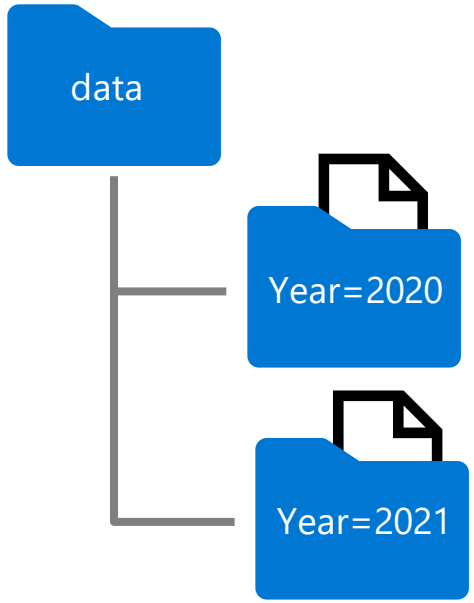
Transform data in a Spark Dataframe



Save dataframe for further analysis

Partition the output file

- Partition data by one or more columns
- Distributes data to improve performance and scalability




```
df.write.partitionBy("Year").mode("overwrite").parquet("/data")
```

Work with data using Spark SQL

Use the metastore to define tables and views

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

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

Create external tables

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

Query data using the Spark SQL API

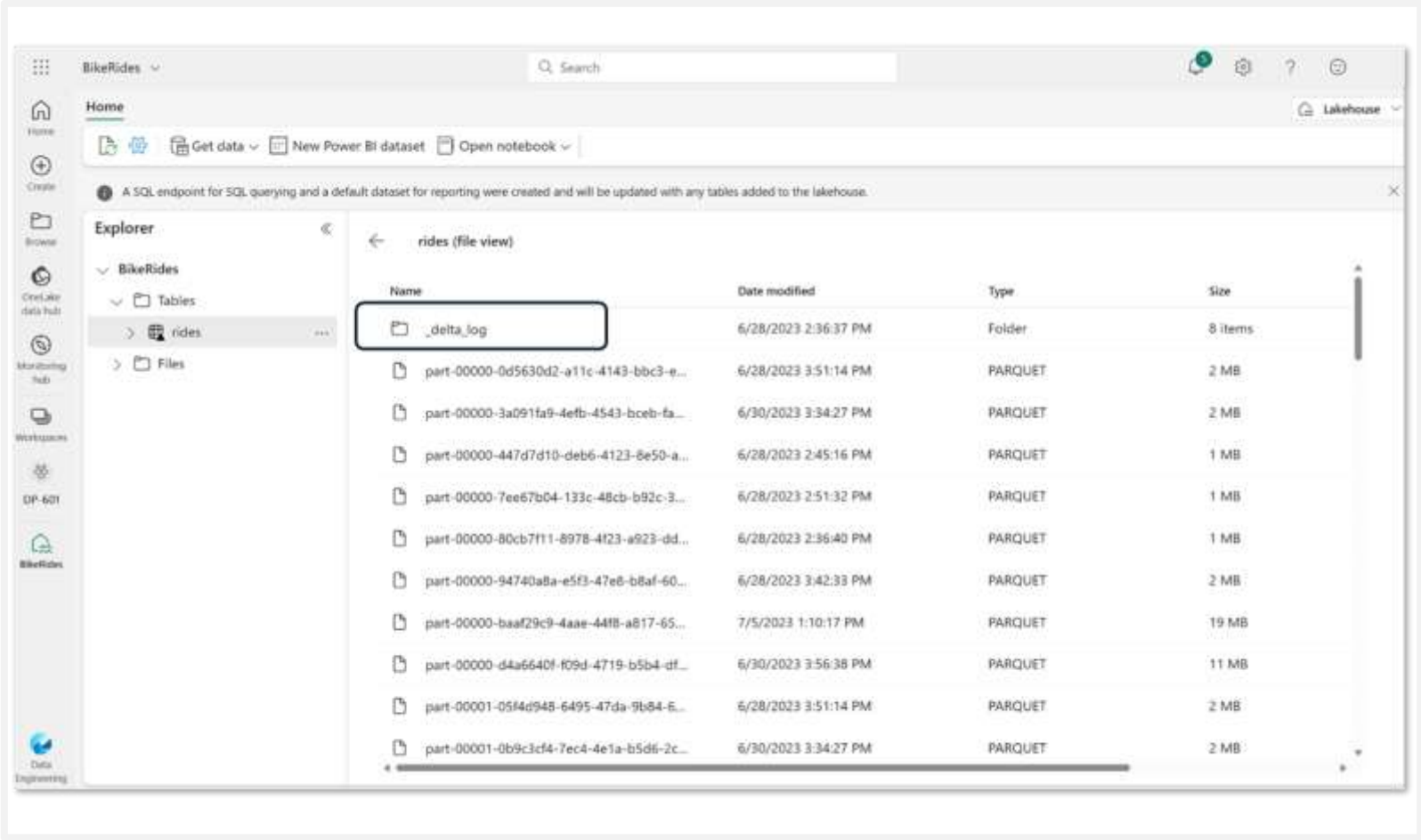
Use the Spark SQL API in code written in any language to query data in the catalog

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

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

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

%%sql

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

Understand Delta Lake

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



Create Delta tables using code in Spark

1. Save a dataframe as a managed table

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

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

2. Use Spark SQL

```
%%sql

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

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

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

Managed vs external tables

Managed tables

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

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

External tables

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

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

Work with Delta tables in Spark

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

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

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

```
%%sql

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

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

```
from delta.tables import *
from pyspark.sql.functions import *

# Create a DeltaTable object
delta_path = "Files/mytable"
deltaTable = DeltaTable.forPath(spark, delta_path)

# Update the table (reduce price of accessories by 10%)
deltaTable.update(
    condition = "Category == 'Accessories'",
    set = { "Price": "Price * 0.9" })
```

Data versioning and time travel

- Modifications made to delta tables are logged in the transaction log for the table.
- You can use the logged transactions to view the history of changes made to the table and to retrieve older versions of the data (known as time travel).

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

%%sql

DESCRIBE HISTORY products

# Use SQL to see the history of an external table, specifying folder location

%%sql

DESCRIBE HISTORY 'Files/mytable'
```

Use Delta tables with streaming data

Use Delta lake table as a streaming source

```
from pyspark.sql.types import *
from pyspark.sql.functions import *

# Load a streaming dataframe from the Delta Table
stream_df = spark.readStream.format("delta") \
    .option("ignoreChanges", "true") \
    .load("Files/delta/internetorders")

# Now you can process the streaming data in the dataframe
# for example, show it:
stream_df.show()
```

Use Delta lake table as a streaming sink

```
# Create a stream that reads JSON data from a folder
inputPath = 'Files/streamingdata/'
jsonSchema = StructType([
    StructField("device", StringType(), False),
    StructField("status", StringType(), False)
])
stream_df = spark.readStream.schema(jsonSchema).option("maxFilesPerTrigger", 1).json(inputPath)

# Write the stream to a delta table
table_path = 'Files/delta/devicetable'
checkpoint_path = 'Files/delta/checkpoint'
delta_stream = stream_df.writeStream.format("delta").option("checkpointLocation", checkpoint_path).start(table_path)
```

Inside the Apache Parquet format It's Open Source!

In traditional databases (Postgres, SQL Server*) and file types (CSV, JSON), Data is stored as rows...

first_name	last_name	street	isPremiumCustomer	Next Month Buying Probability
Abigail	Hayes	4009 9th Street	1	1
Amanda	Clark	87 5th Street	0	2
Melissa	Cl			
Sara	Ben			
Isaac	Hend			
Emma	Ste			
Sean	Mo			
Jason	San			
Alexander	Wa			
Megan	Mo			

Customers.csv

```
first_name,last_name,street,isPremiumCustomer,nextmbuyprob
Abigail,Hayes,4009 9th Street,1,0.26340968441351104
Amanda,Clark,87 5th Street,0,0.17177893540625921
Melissa,Clark,12650 5th Street North,1,0.5438465585898219
Sara,Bennett,203 State Street,0,0.15856312140709694
Isaac,Henderson,732 Magnolia Drive,1,0.19389081059172666
Emma,Stewart,47 Route 32,1,0.8833057932776623
Sean,Morris,254 Lincoln Avenue,0,0.4156333382834478
Jason,Sanchez,26050 Madison Avenue,0,0.5862717219123177
Alexander,Walker,54051 Valley Road,1,0.5941814808482648
Megan,Morris,31 Delaware Avenue,1,0.6289626353900487
```

...which is not optimal for analytical needs:

- 👎 For some file types (like CSV), there is no embedded schema
- 👎 Compression is not good (you're kinda limited to text compression)
- 👎 You need to scan(read) all the rows

*Unless you're using columnstore tables :)

In Parquet, data is stored as columns

Customers.parquet

```
first_name Abigail Amanda Melissa Sara Isaac Emma Sean Jason

last_name Hayes Clark Clark Bennett Henderson Stewart Morris Sanchez

isPremiumCustomer 1 0 1 0 1 x2     0 x2

Category 0.263409684 0.171778935 0.543846559 0.158563121 0.193890811
0.883305793 0.415633338 0.586271722

Col 1 STRING  Col 2 STRING  Col 3 BOOLEAN  Col 4 UINT_32
```

File structure is simplified, omitting details like row groups.

- 🔍 Parquet embeds the data Schema
- Want the number of premium customers? You can compute that aggregate without reading any other unnecessary data!
- 🗜️ We can efficiently compress repetitive data like this one

With Fabric, you can easily read parquet files stored in OneLake or in other clouds

@cmaneu

Delta Transaction Log

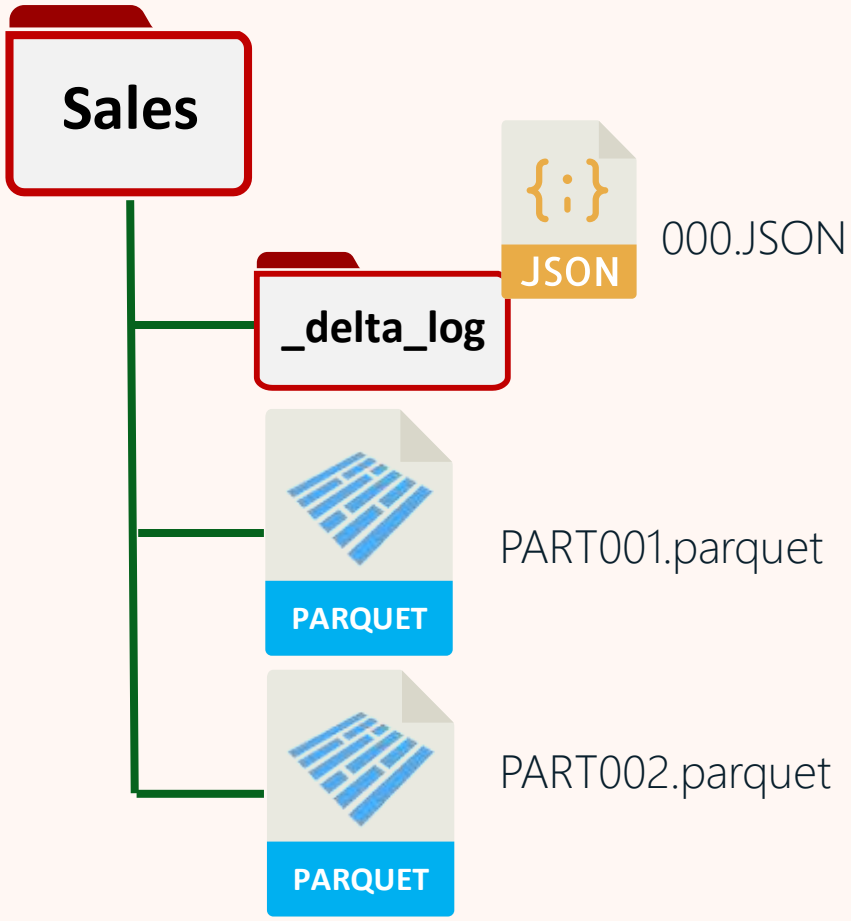


Table Version	File Changes
0	+PART001.parquet +PART002.parquet

Delta Transaction Log

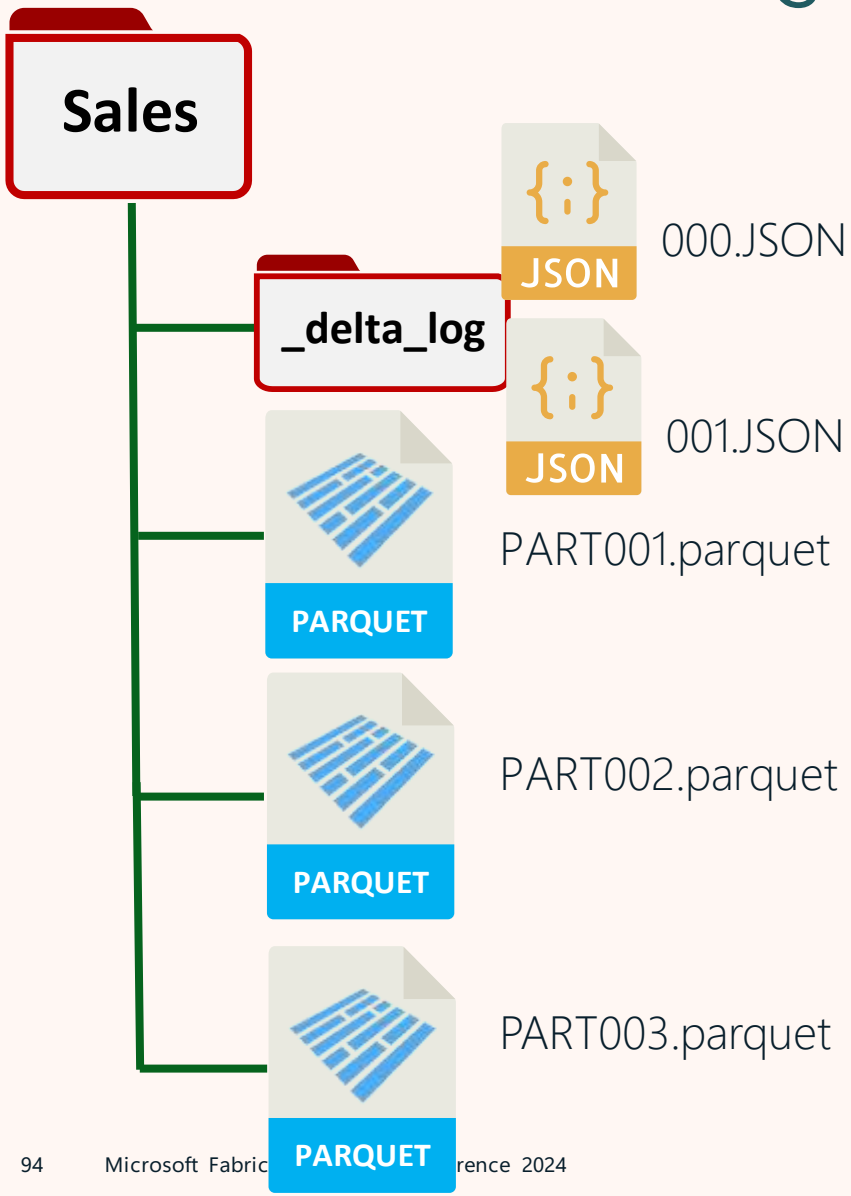


Table Version	File Changes
0	+ PART001.parquet + PART002.parquet
1	- PART001.parquet - PART002.parquet + PART003.parquet

DELETE * **FROM** SALES
WHERE Segment = 3

Reading Delta

SELECT * FROM SALES

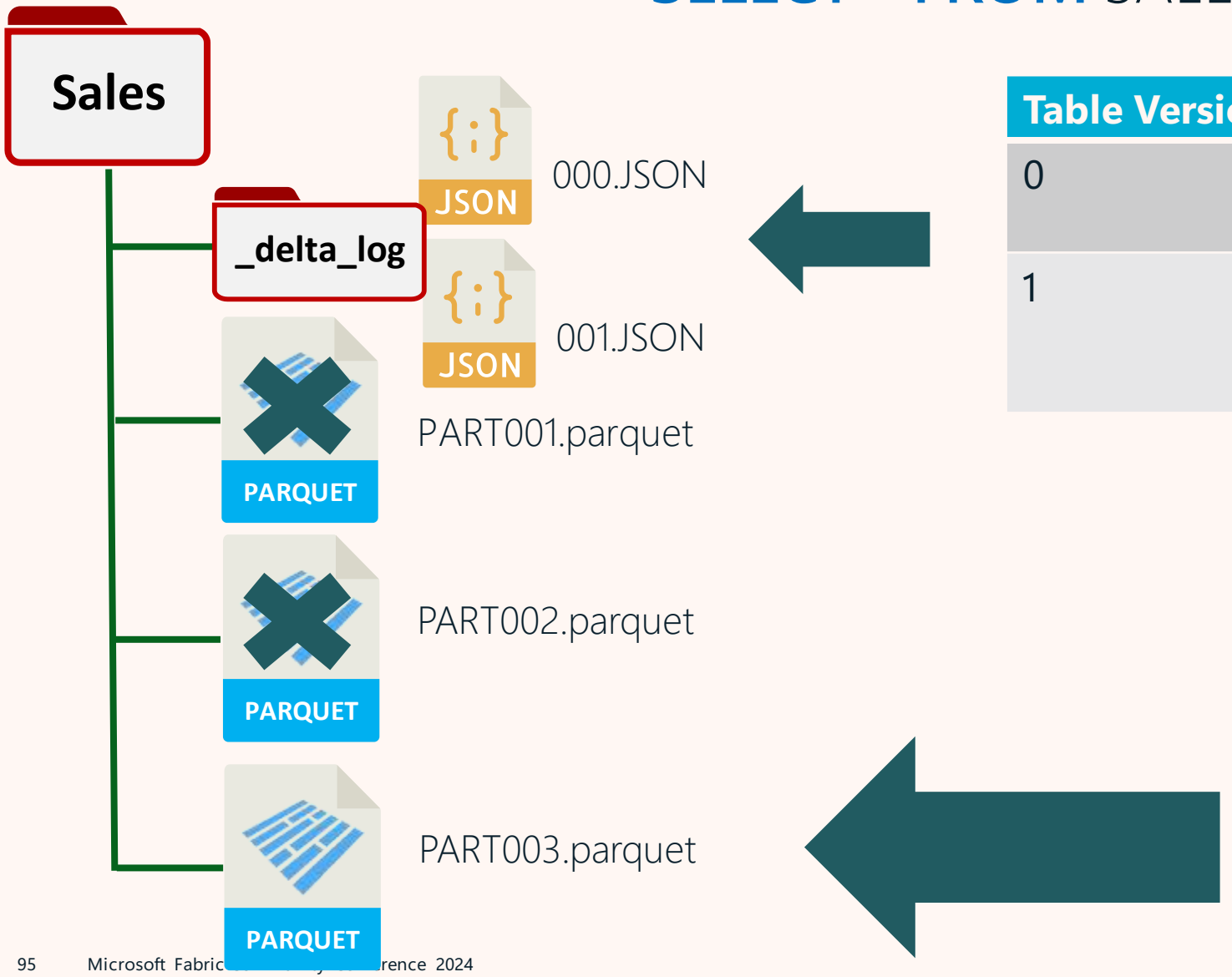


Table Version	File Changes
0	+ PART001.parquet + PART002.parquet
1	- PART001.parquet - PART002.parquet + PART003.parquet

Time Travel

SELECT * FROM SALES VERSION AS OF 0

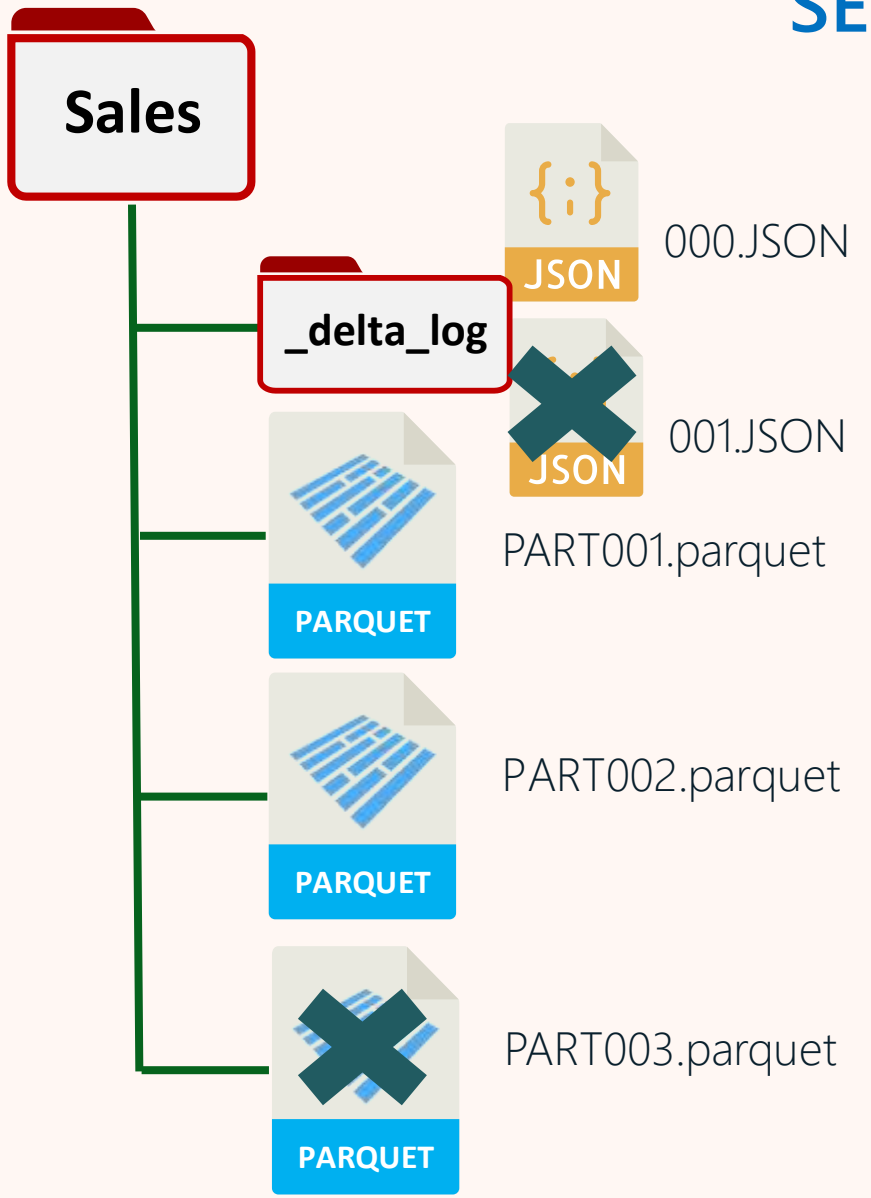
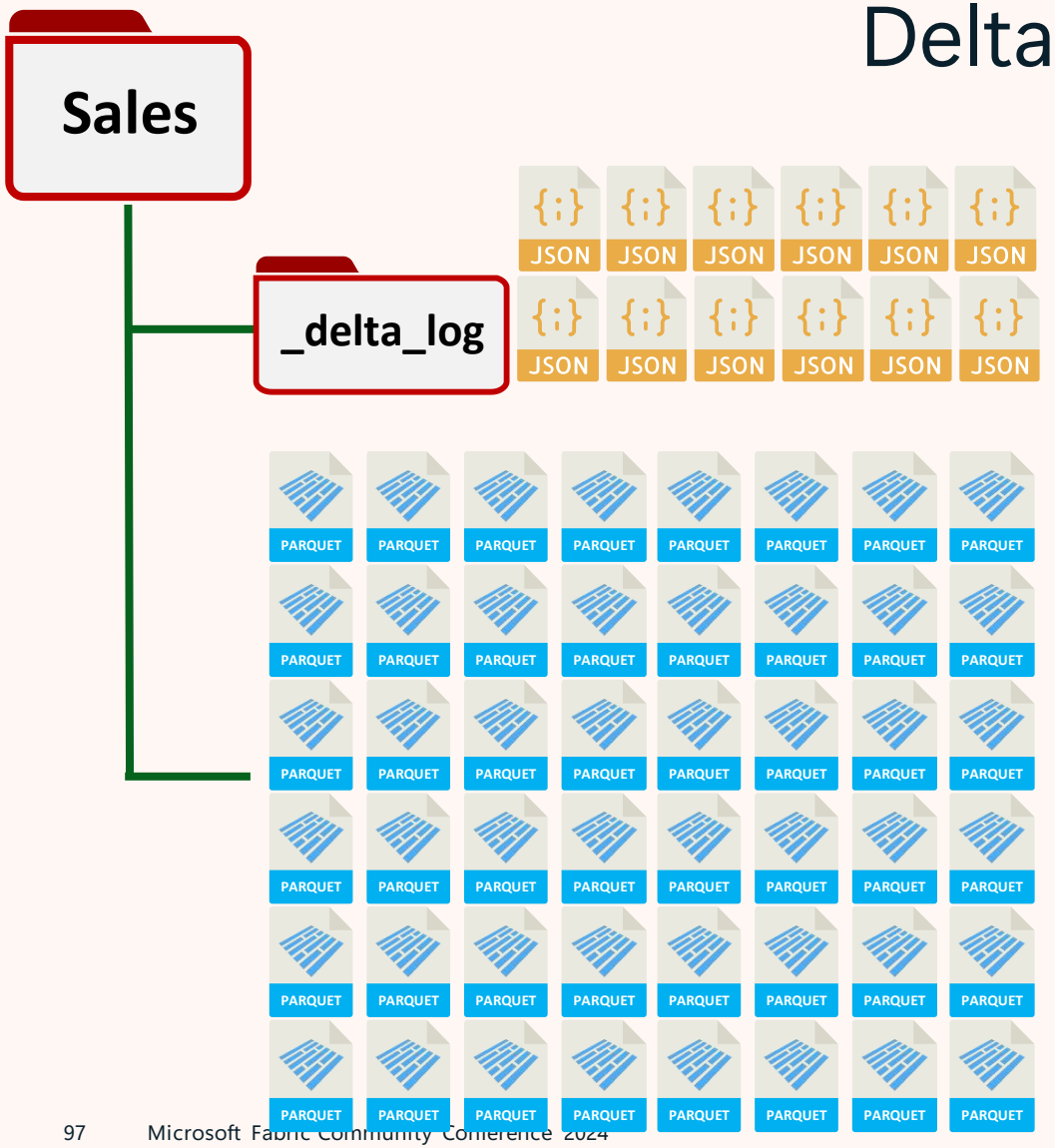


Table Version	File Changes
0	+ PART001.parquet + PART002.parquet
1	- PART001.parquet - PART002.parquet + PART003.parquet

Vacuum

To remove obsolete history files, Delta has the **VACUUM** command



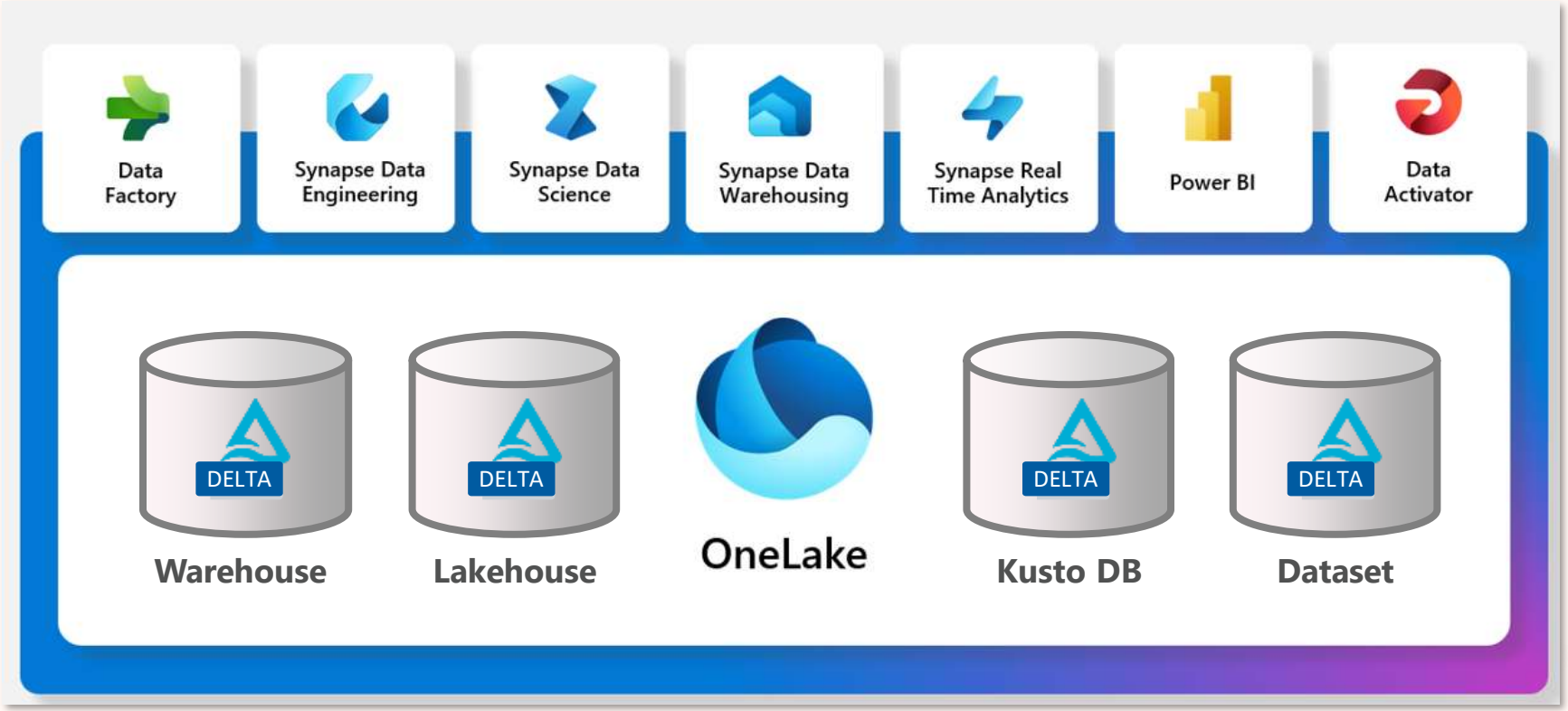
This command physically deletes data files older than a specified date

You CANNOT time travel past dates where history has been vacuumed

Delta Lake table format in Microsoft Fabric

Delta Lake tables are the **unifying table format in Microsoft Fabric**.

Multiple experiences, with multiple compute engines, **OneCopy** of your data.



Microsoft

Principles and challenges

If one compute engine in Microsoft Fabric **writes** a Delta Lake table, **all other engines must be able to read it.**

Full interoperability and limitations documentation.

We align to the **Delta Lake open-source** protocol specification.

We contribute back to the OSS project.

Like we do to other projects such as Apache parquet, Spark, and others.

Twelve engines (as of today) in Microsoft Fabric are aligned on Delta Lake protocol,

We need to go beyond the OSS provided libraries, to develop and integrate internal C++ and .NET Delta Lake protocol implementations to our products.

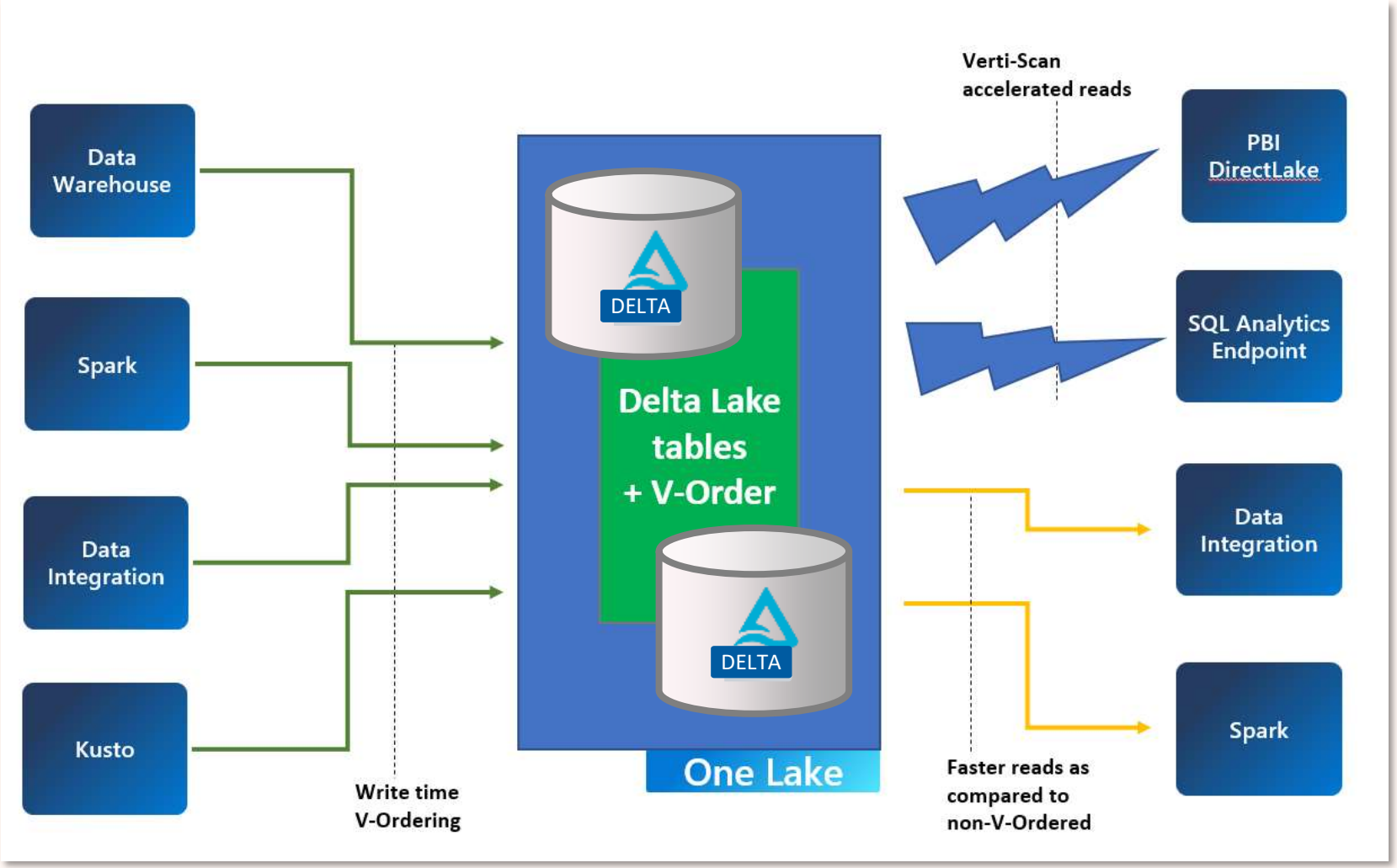
Microsoft

What is V-Order?

V-Order is a write time optimization to the parquet file format that enables lightning-fast reads under the Microsoft Fabric engines.

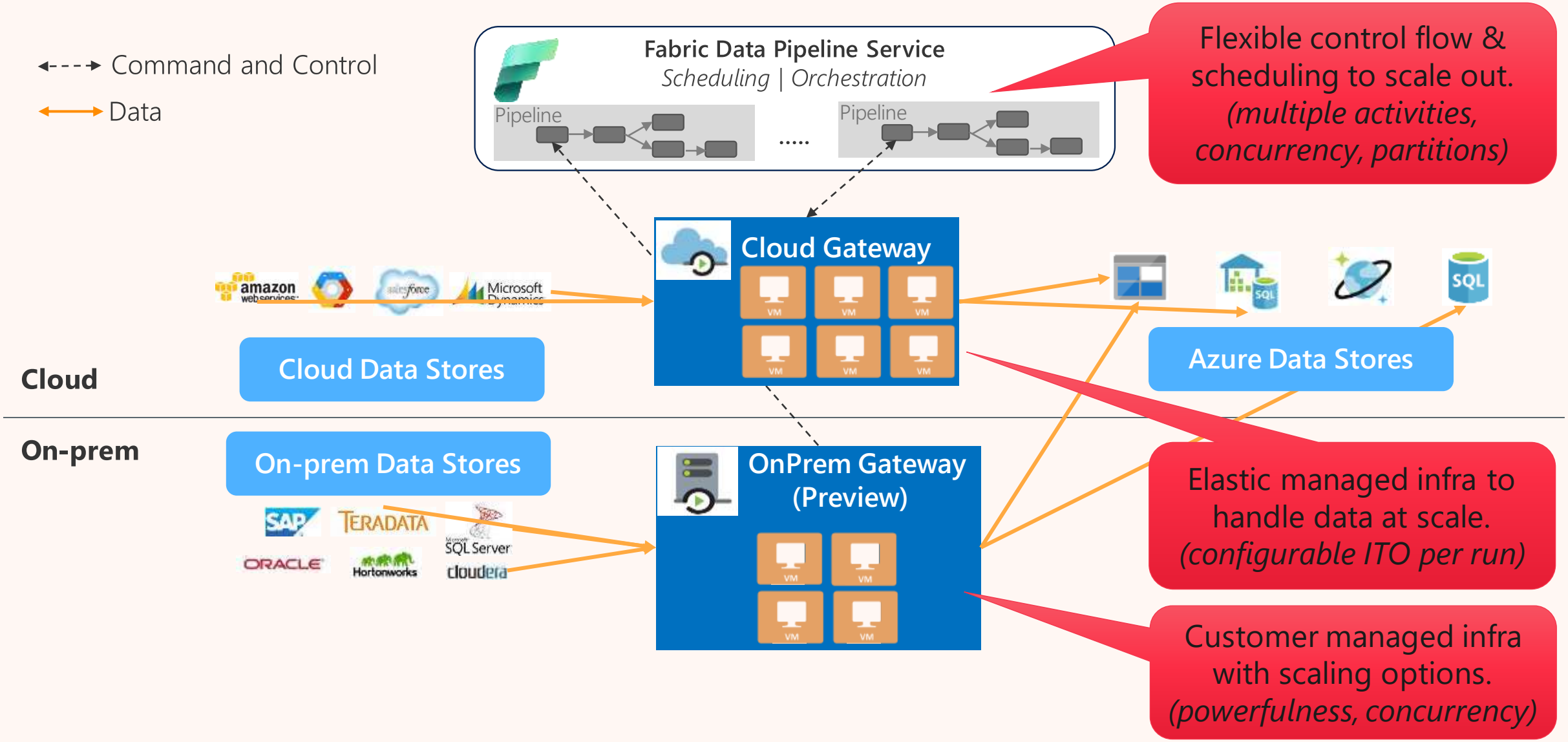
Microsoft Fabric engines **write V-Ordered** parquet files **by default**.

Table APIs are provided to manage V-Order on Lakehouse



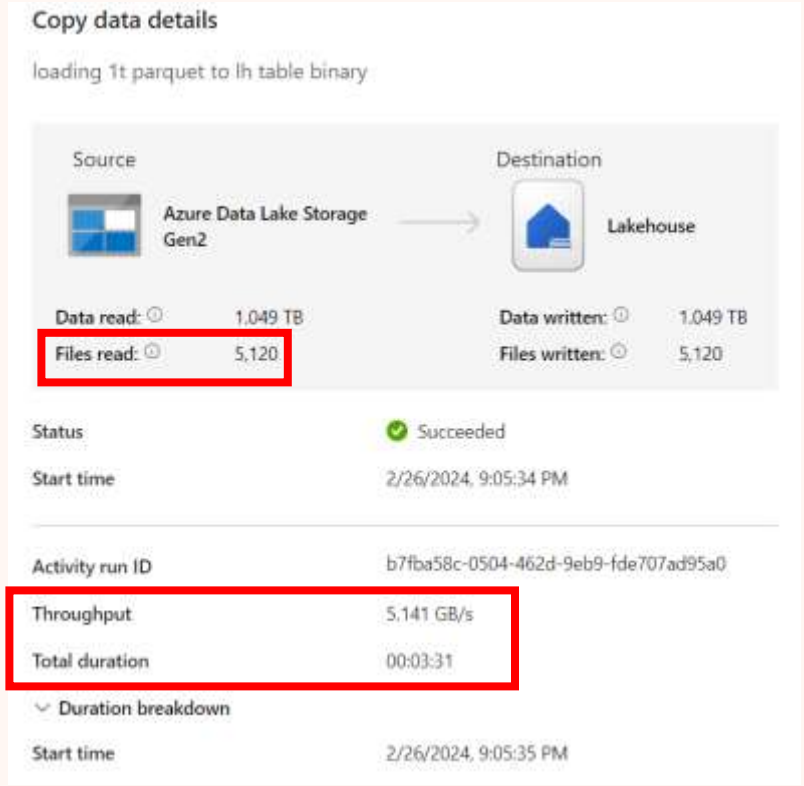
Pipeline Architecture

Understand How Pipeline Scales

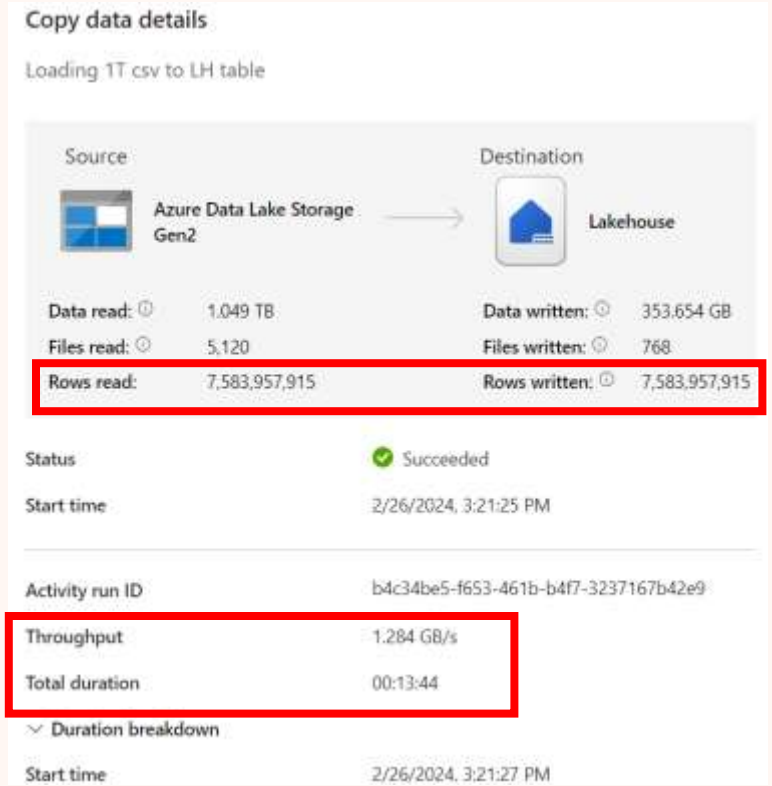


Copy Performance Metrics

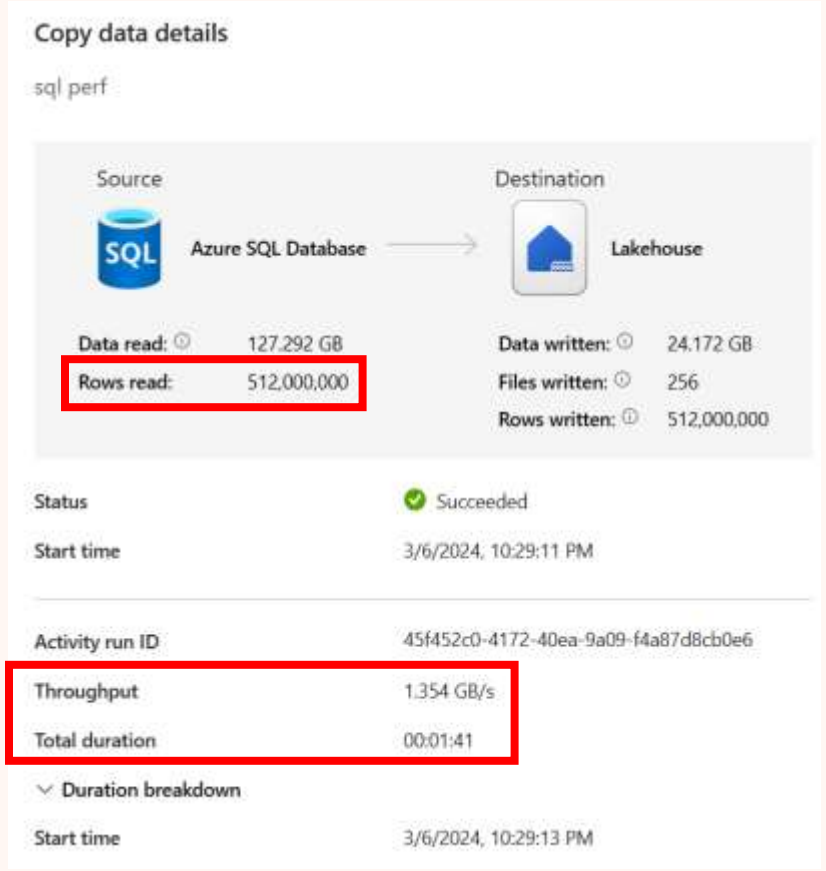
Parquet -> LH (binary)



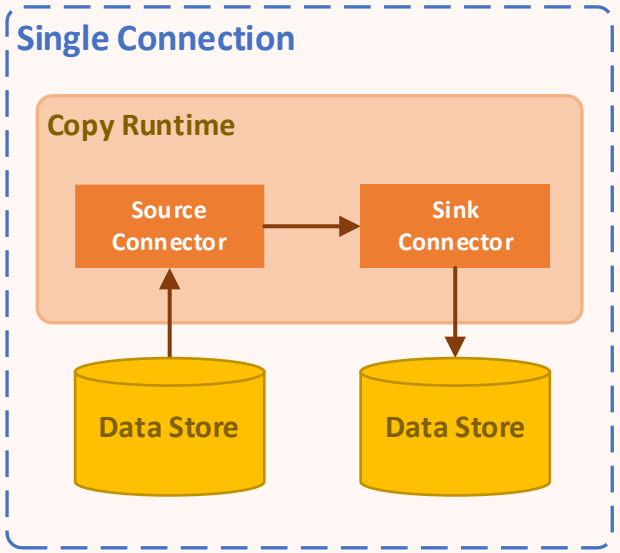
CSV -> LH



SQL -> LH

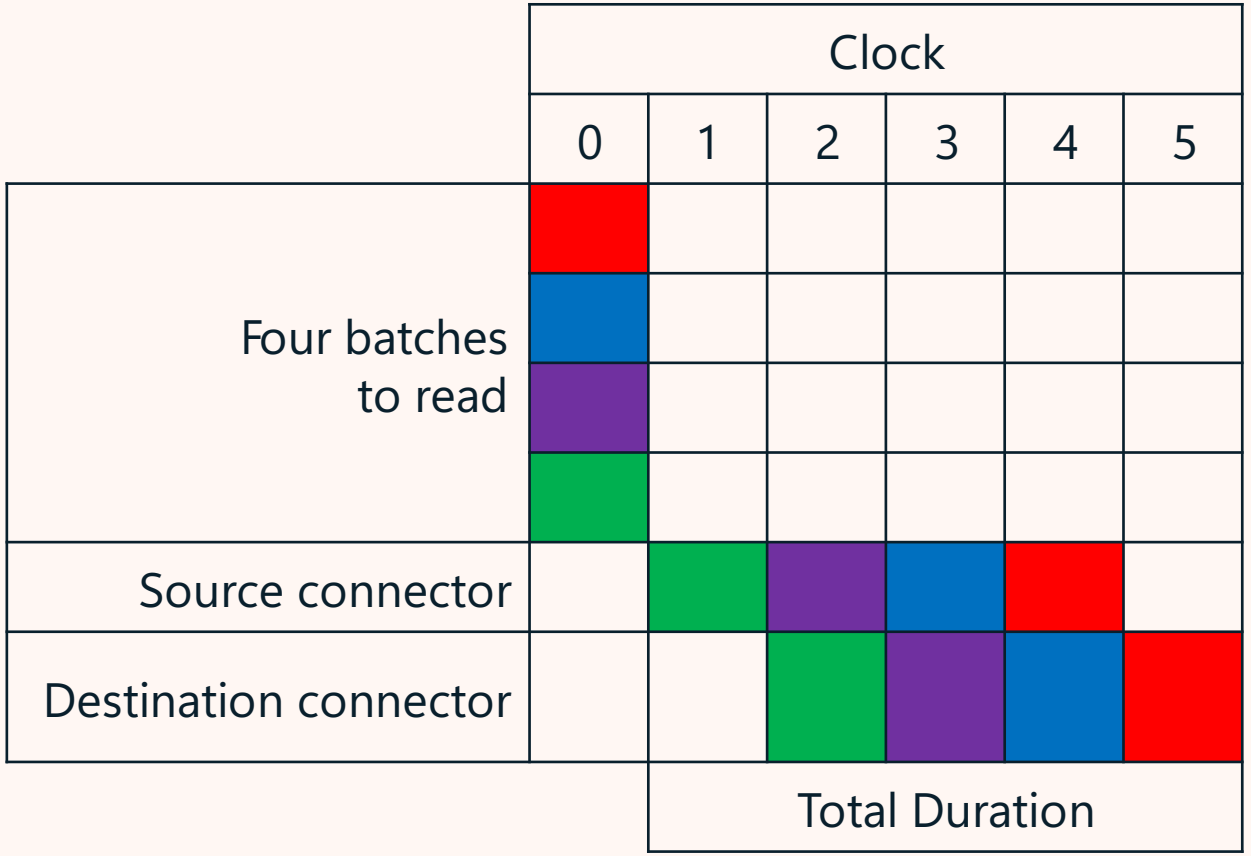


How Copy Scales – Connection Level

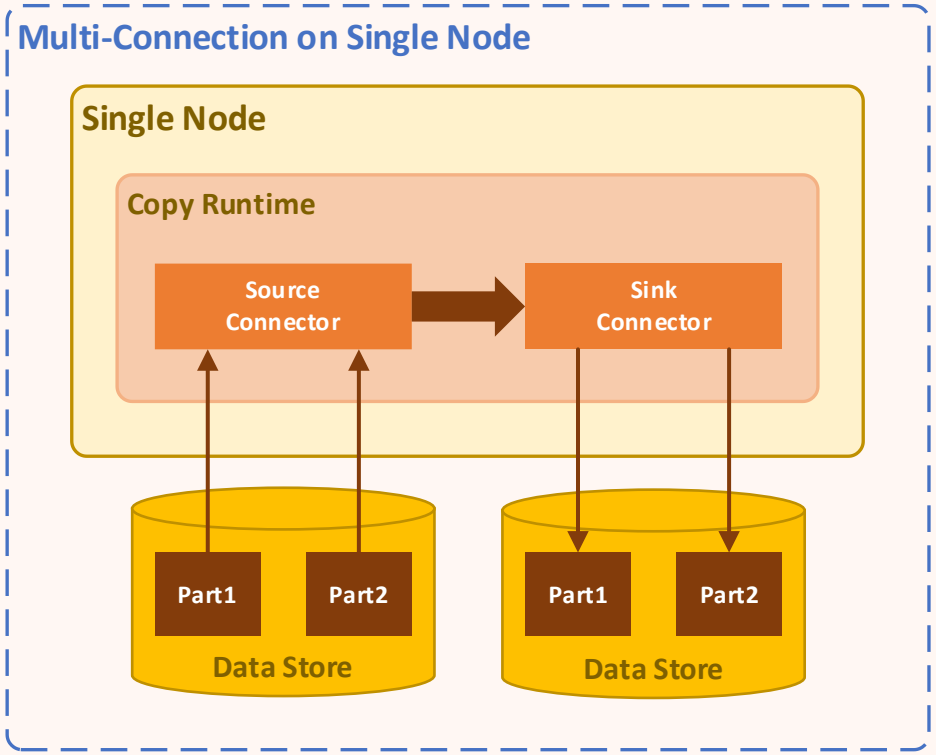


Pipeline processing

- **Less memory**: No need to load everything in memory and then write
- **Less total duration**: Read and Write are in parallel

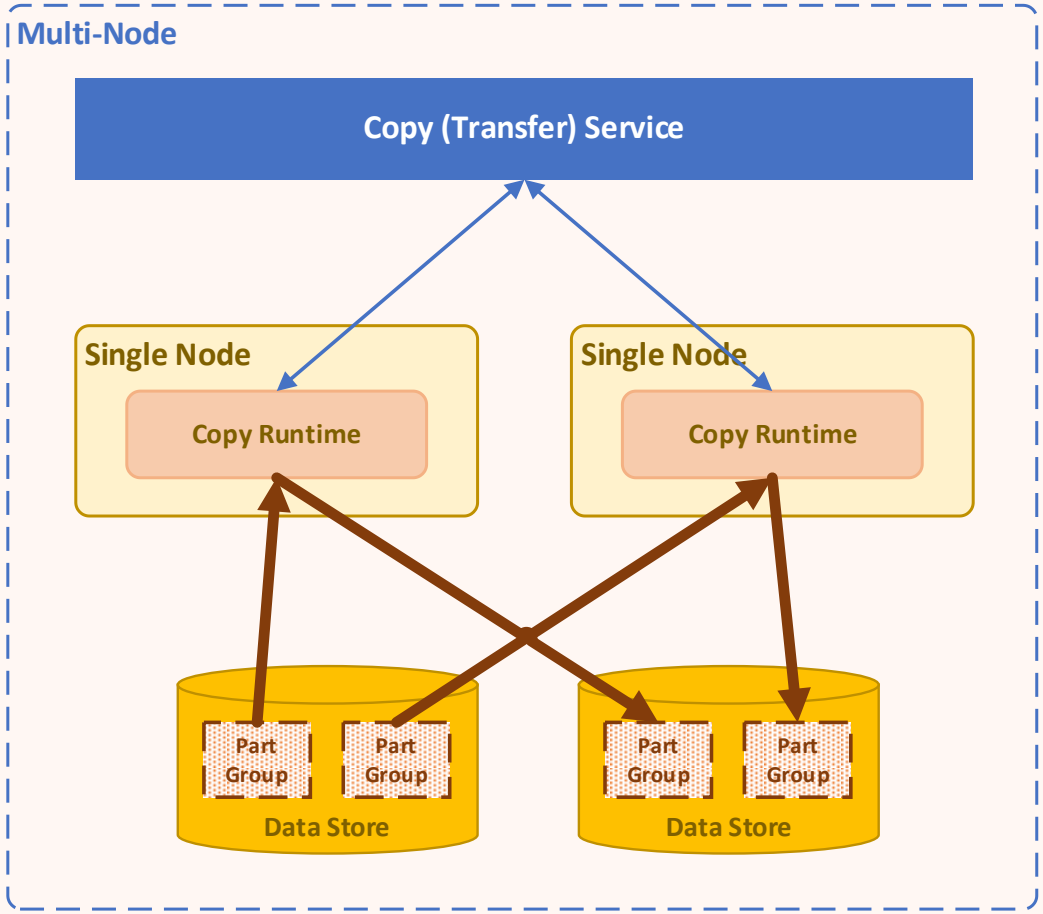


How Copy Scales – Node Level



- Producer & Consumer design
  - Data is partitioned for multiple concurrent connections (even for single large file)
  - Full node utilization: Data can be partitioned differently between source and destination to avoid any starving / idle connection
- Partitions come from
  - Physical partitions setup on DB side
  - Dynamic partitions from different queries
  - Multiple files
  - Multiple parts from a single file

How Copy Scales – Multi-Node Level



- **Copy(Transfer) Service**: Manages Copy activities
  - Copy State Management / Monitoring
  - Cross Machine Parallelism
  - Billing
  - Manages Compute Clusters and tasks running on them
- Easily scale out to multiple stateless nodes when available partitions are more than what one node can afford.
- Theoretically no upper limit on the performance.

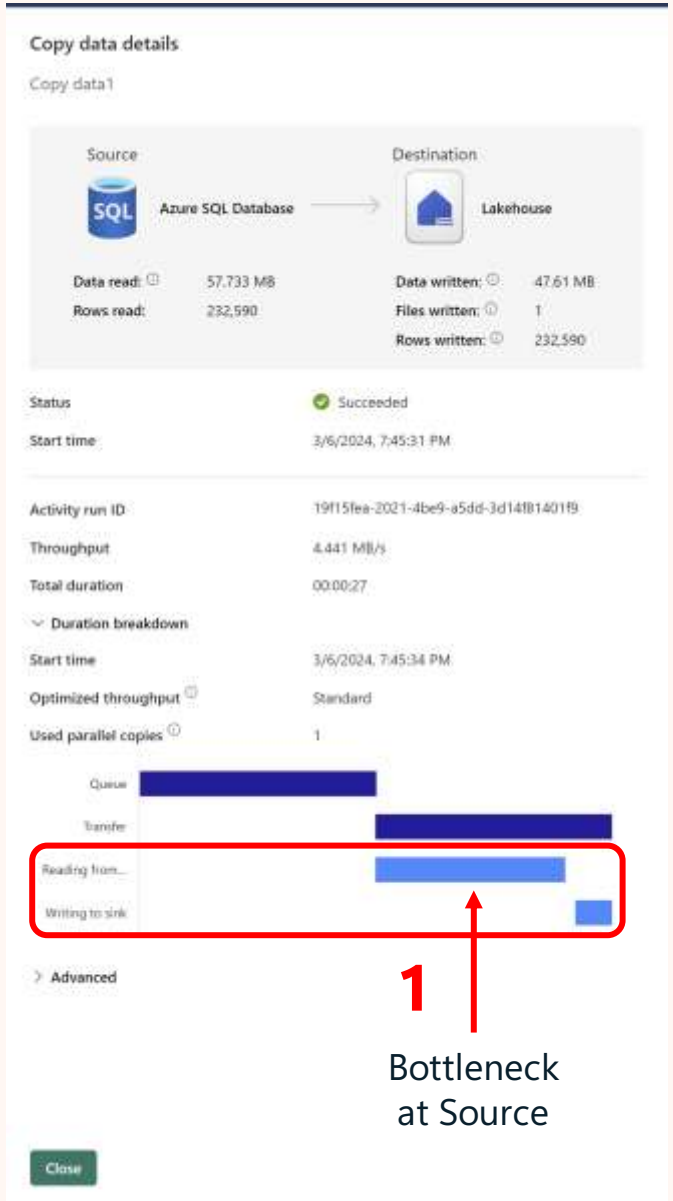
Copy Design Considerations

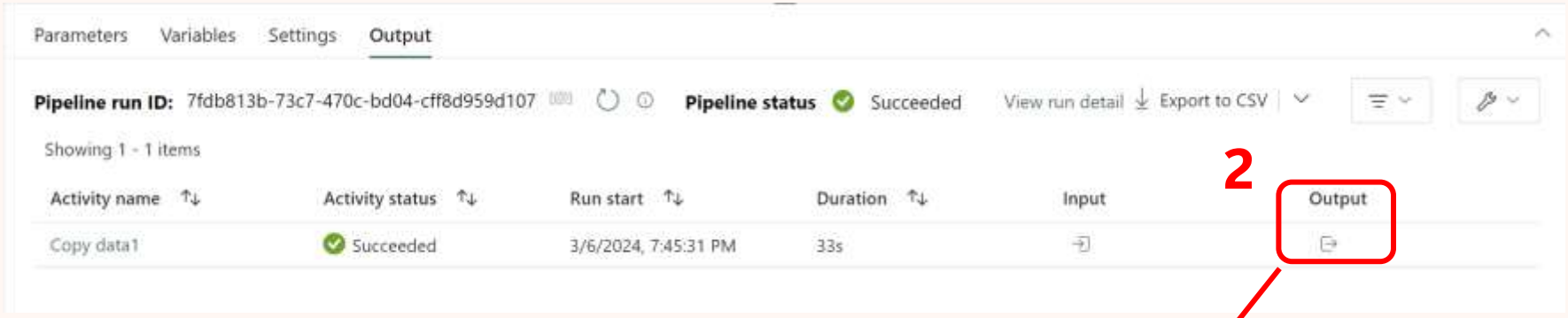
- Concepts
  - **Intelligent Throughput Optimization (ITO)**: A measure that represents the power (a combination of CPU, memory, and network resource allocation) used for a single Copy activity.
  - **Parallel Copy**: The maximum number of threads within the Copy activity that read from source and write to destination in parallel.
  - **Max Concurrent Connections**: The upper limit of concurrent connections established to the data store during the activity run. (Usually used to avoid throttling)
- Implementation details
  - Files based connectors
    - 1 file = 1 partition
  - Tabular connectors that support Partition read
    - SQL family, Oracle, Teradata, Azure PostgreSQL, Netezza, SAP HANA, SAP Table, SAP Open Hub
  - **Default perf settings**
    - Intelligent throughput optimization: **Auto** (Use maximum available VM resources)

Troubleshoot Performance Issues

- Common bottlenecks
  - Network (cross region/OnPre)
  - Source/Destination data stores
    - Busy neighbors on the same data store
    - Low service/compute tier
    - Throttling mechanism on data store
    - Complex/unoptimized query (huge timeToFirstByte)

Troubleshoot Performance Issues – Example





Indicating that most time was spent on preparing/running the query on DB side. Optimize the query or upgrade DB tier to eliminate this bottleneck.

Dataflows

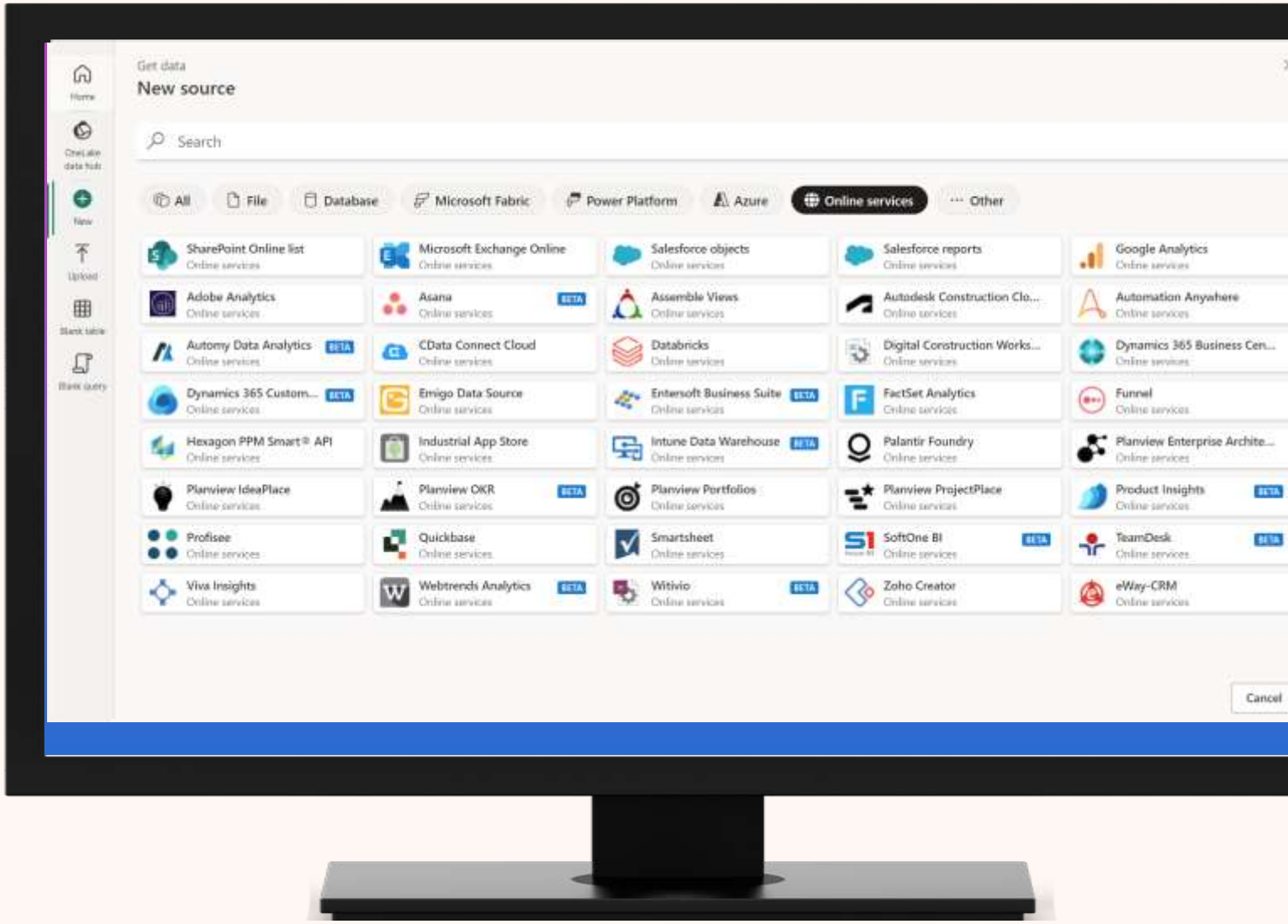
Dataflows Gen 2

- Dataflow Gen 2 in **Microsoft Fabric** is a powerful data preparation technology that allows you to create, transform, and load data into Fabric and Azure destinations. Here's what you can do with it:

- **Create Dataflows:**
  - Dataflows are self-service, cloud-based tools for data preparation.
- **Get Data:**
  - Dataflows enable you to retrieve data from 100s of on-premise and cloud data sources.
- **Apply Transformations:**
  - Once you've connected to your data source, it's time to shape it according to your needs.
  - Use the Power Query editor to apply transformations. For instance:
    - Calculate the total number of orders per customer using the Group By feature, combine, remove columns, etc.
- **Configure Destinations:**
  - Specify a destination to store the results of the query and transformations.
- **Publish Dataflows:**
  - After applying transformations, you can publish your dataflow so that it can start processing data.

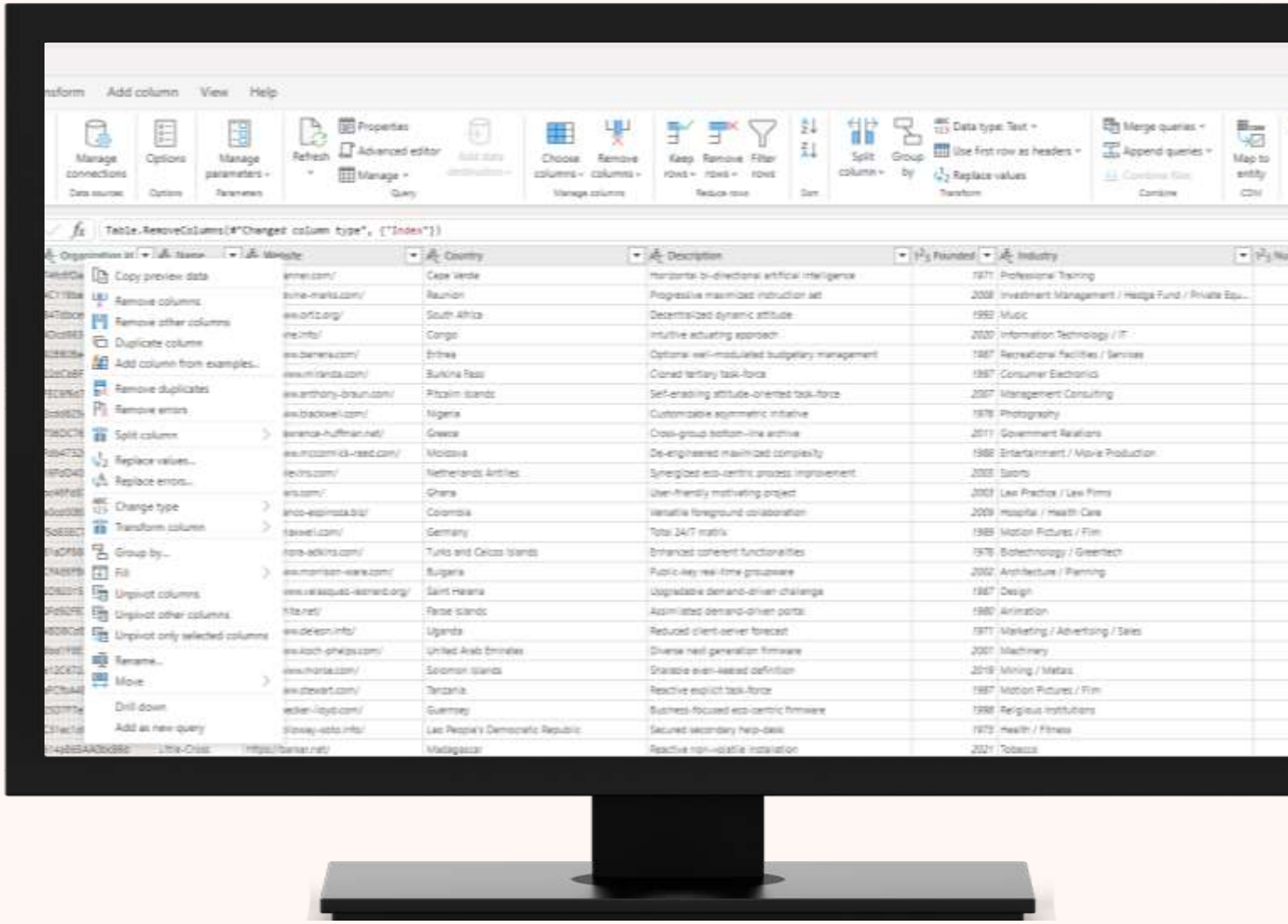
Get Data

- Fabric
- Database
- Azure
- ~175 connectors



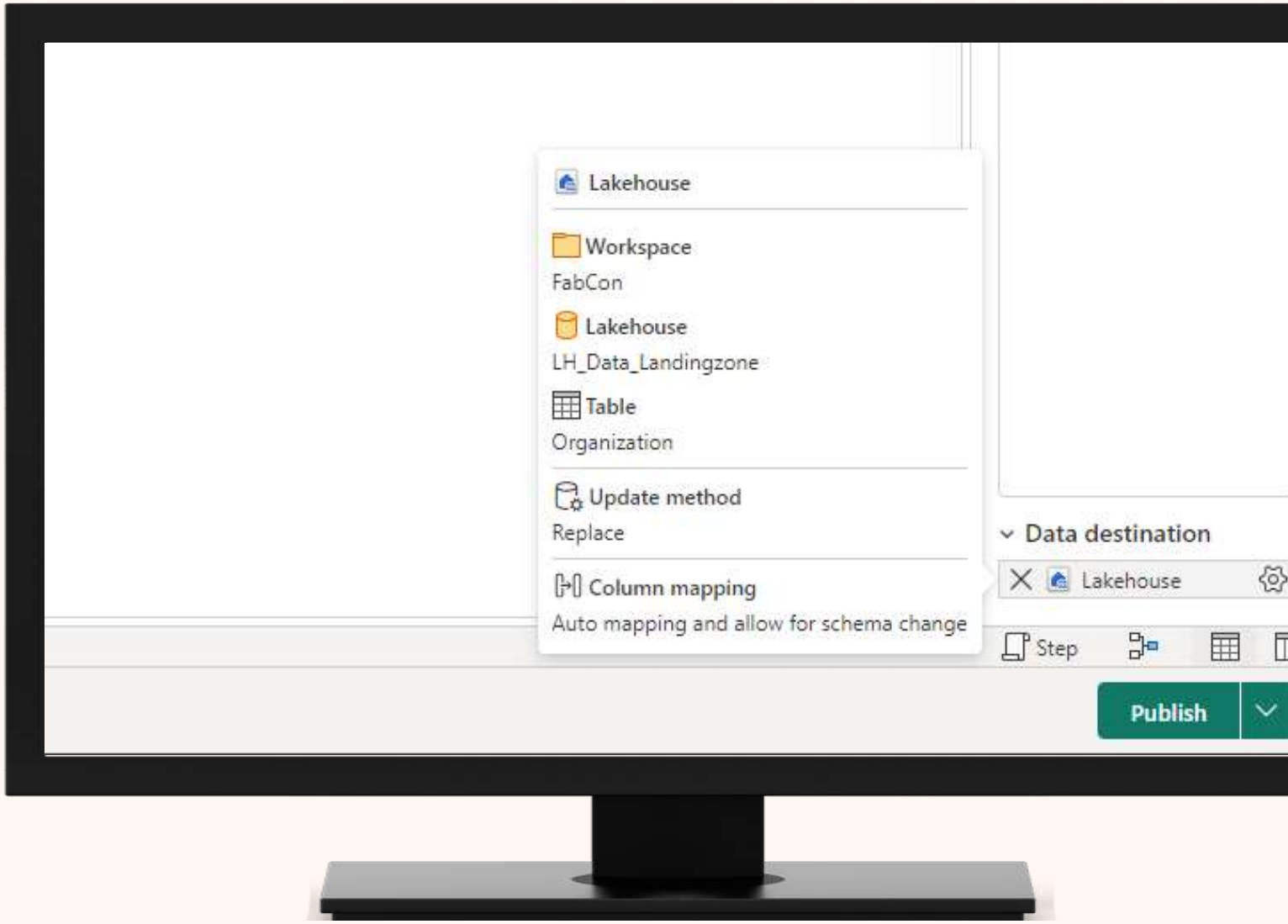
Apply Transformations

- Change data types
- Remove Columns
- Aggregations



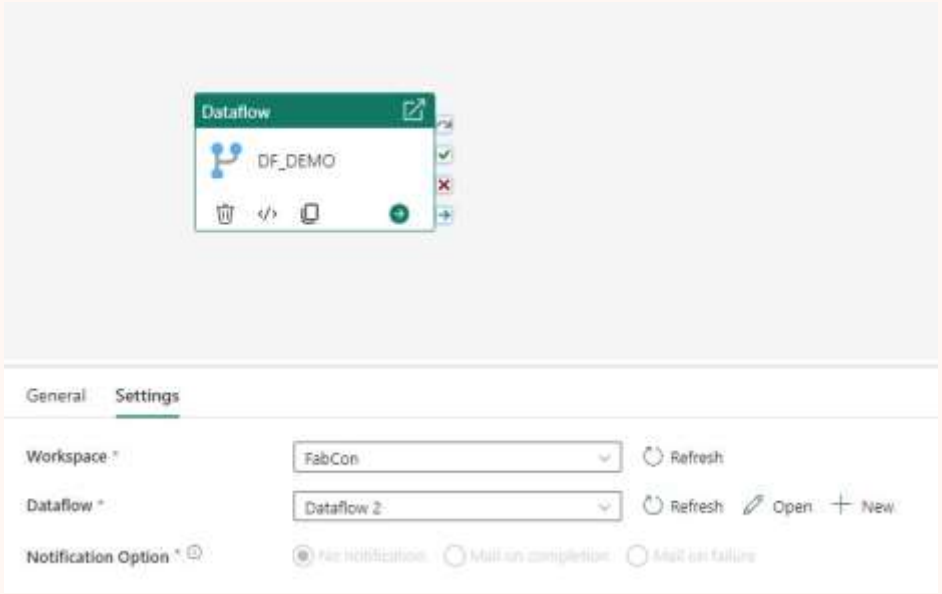
Publish Dataflows

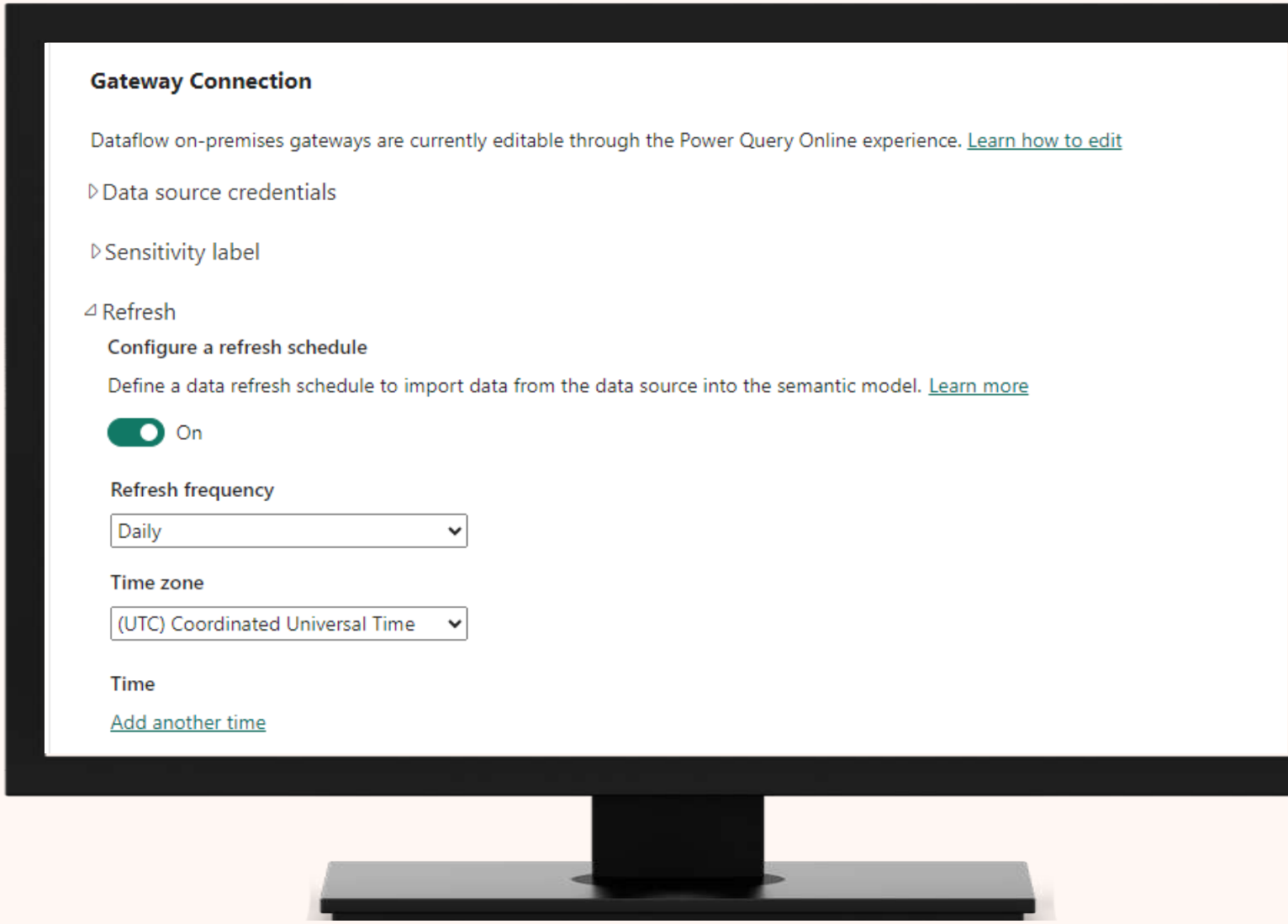
- Select Destination
  - Lakehouse
  - Warehouse
  - Azure Data Explorer
  - Azure SQL Database
    - More to come



Refreshing Dataflows

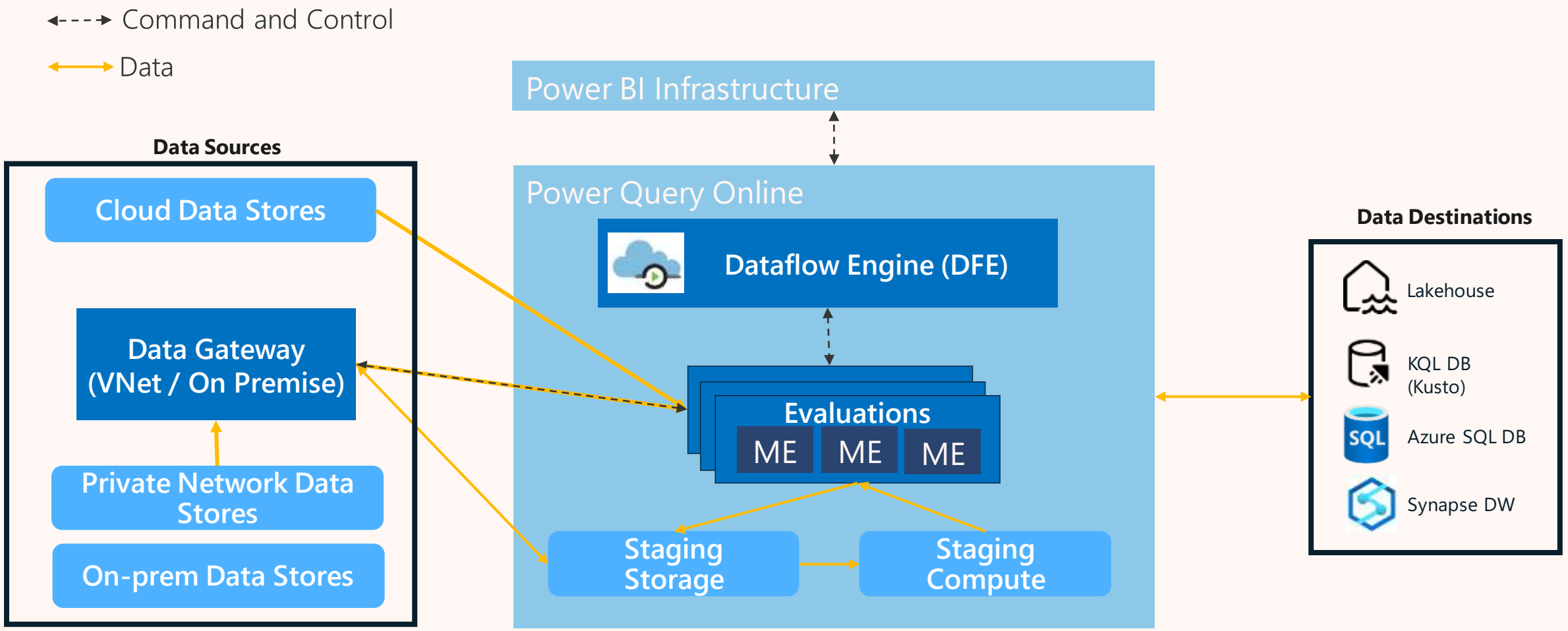
- Schedule dataflow refreshes for automatic operation
- Execute from a pipeline



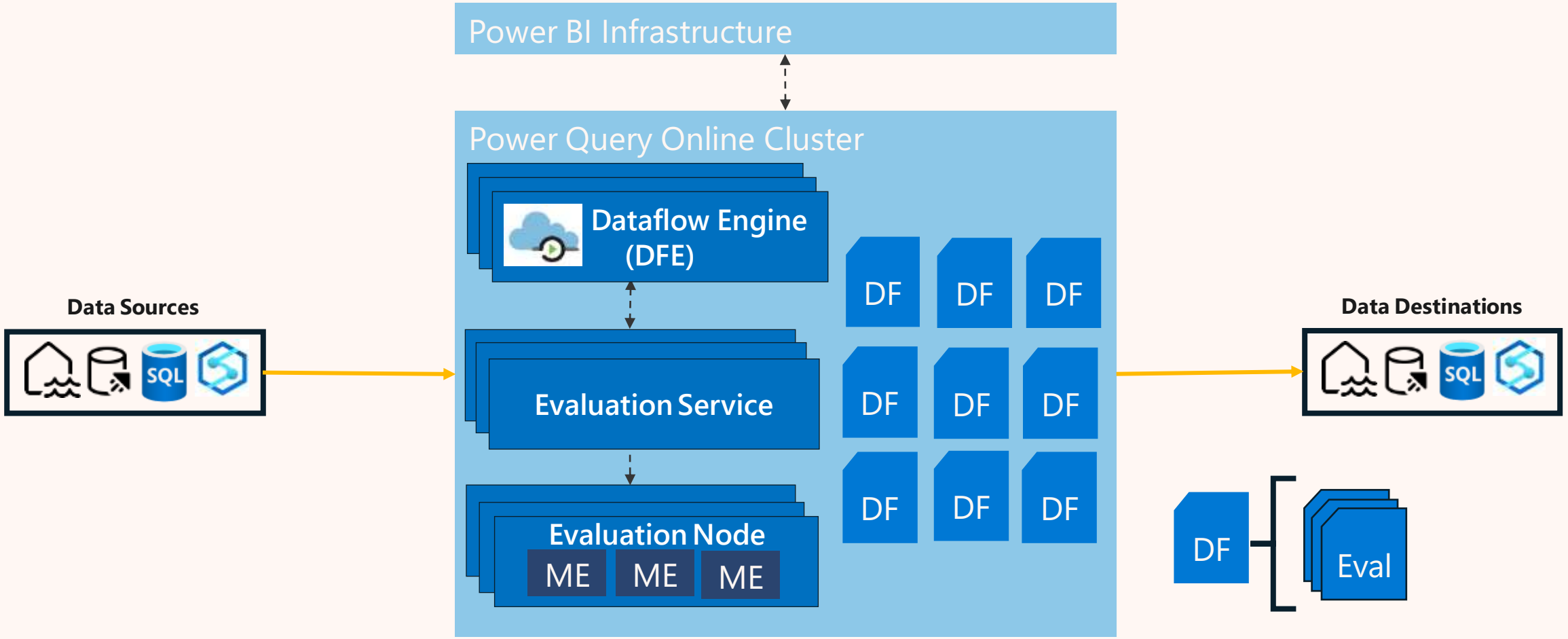


Dataflow Architecture

Fabric Dataflows (aka Gen2)

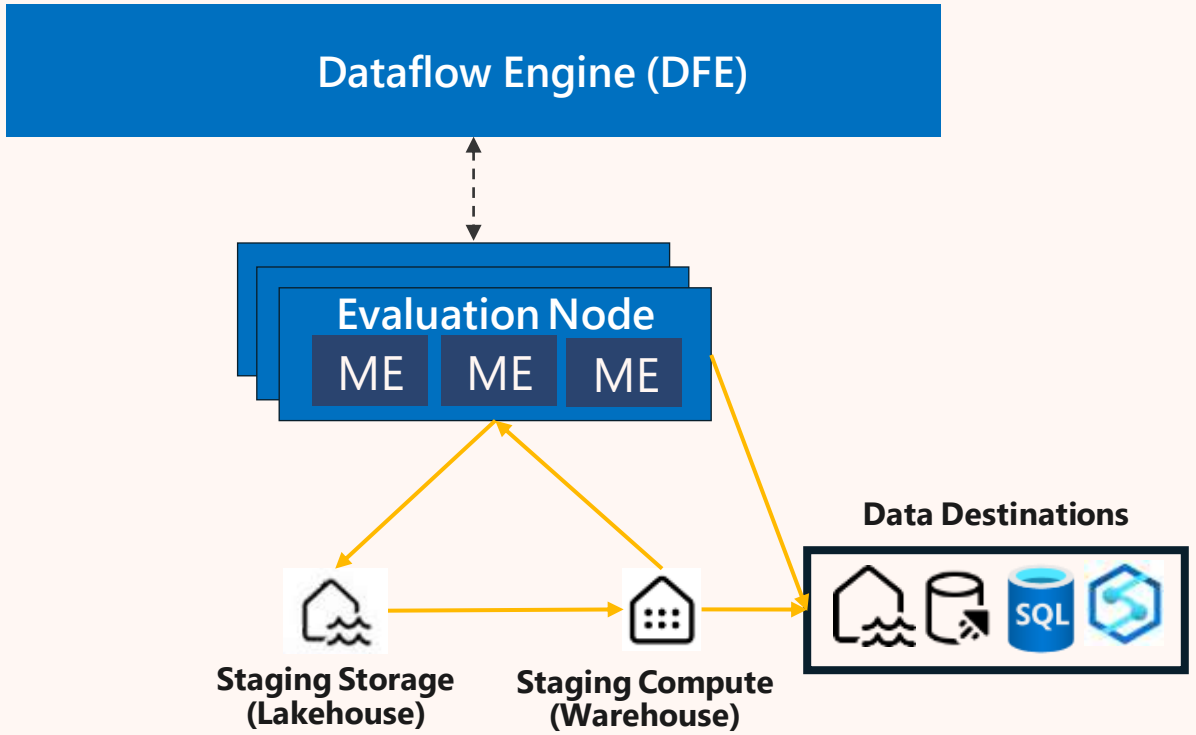


Fabric Dataflows Scale Out



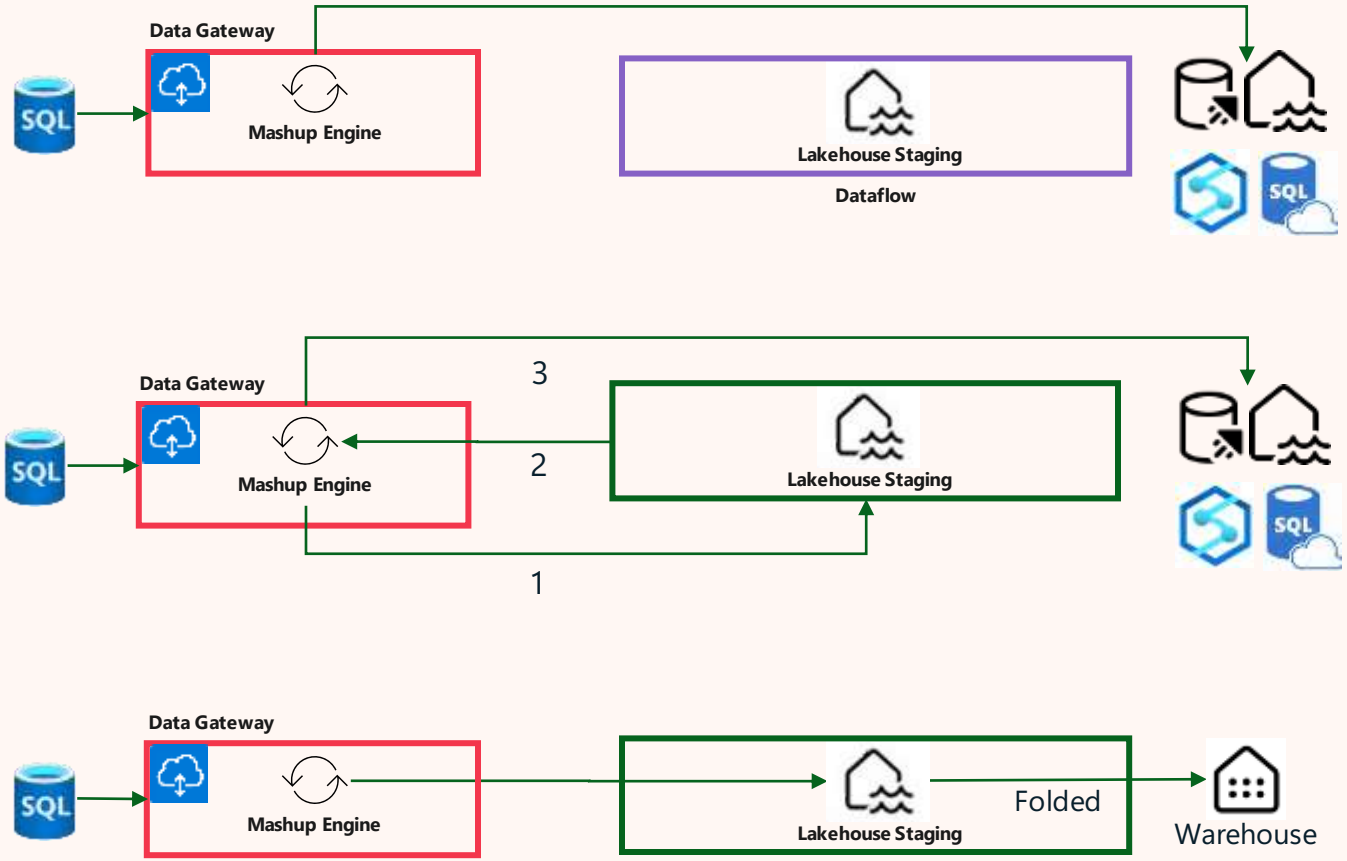
Staging

- Capabilities
  - Landing zone for incoming data
  - Provides a staging point tied to the dataflow / entity refresh
  - Provides WH Compute to speed up operations
- When to use it
  - Combining data from multiple sources
  - Stage data once for subsequent use
  - Writing to WH destination (required)
  - Merging, sorting, grouping, aggregating (uses WH compute)
- When to skip it
  - No value added for entities that write to non-WH destinations
  - See Gateway (next)



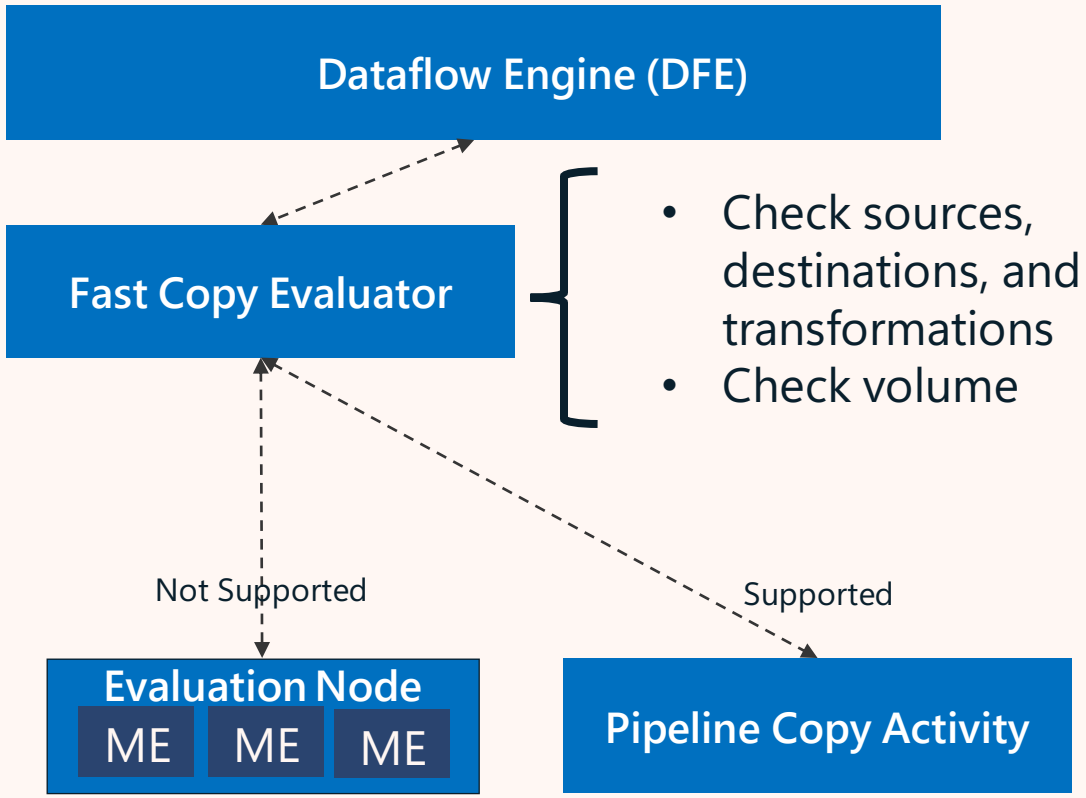
Data Gateway (VNET / On Premise)

- Capabilities
  - Enables access to data sources without exposing them to the internet
  - Data is processed locally
- When to use it
  - Data source is inside your private network
  - Move compute closer to source
  - Control where evaluations happen
- When to skip it
  - Entities that stage data and write to destination (excluding WH)
    - Split Ingest and Destination into separate dataflows (consider where to Transform)
    - Don’t stage



Fast Copy

- Capabilities
  - Leverages Pipeline Copy Activity for large performance boost in ingest
  - Automatically used based on pattern matching and volume
  - Transparent (no pipeline to manage)
- When to use it
  - Whenever possible
    - Defer transformations to post ingest if they affect Fast Copy use
  - Enable in Options..Scale..Allow use of fast copy connectors
- When to skip it
  - Don't – if it's an option, use it
  - Mark as "Require fast copy" to enforce



Leveraging the Architecture

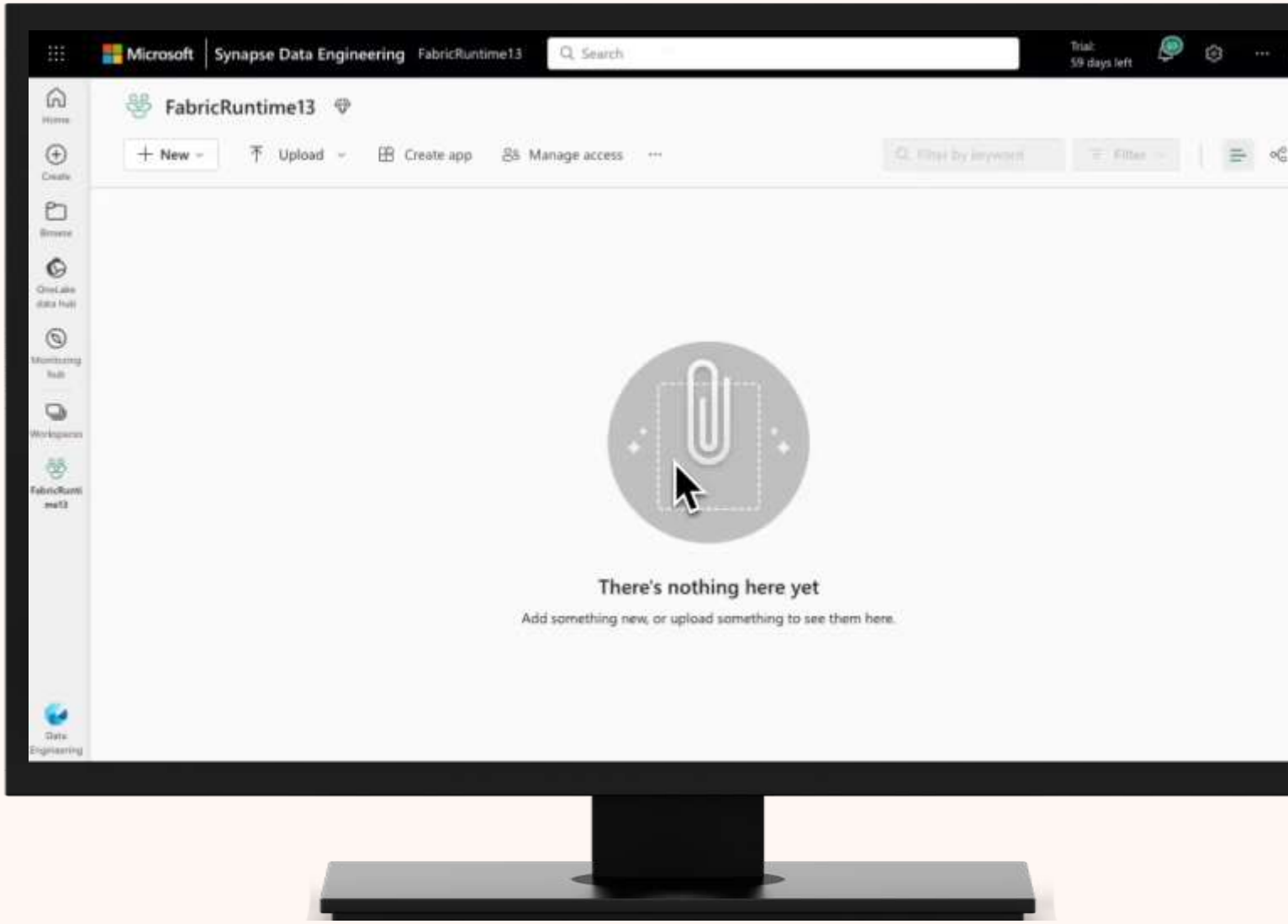
- Carefully consider ETL vs ELT when laying out dataflows
- Know why you are using staging
  - Accelerates some operations, but adds no value to others
- Avoid “double-hops” with Data Gateways
- Take advantage of Fast Copy wherever possible
  - As patterns are added, dataflows can automatically benefit
  - Enable “use fast copy connectors”
  - Ensure that queries fully fold to maximize fast copy usage
- Build for parallel processing
  - Evaluations, dataflows
  - Incremental refresh when available

Schedule data pipelines, dataflows and notebooks

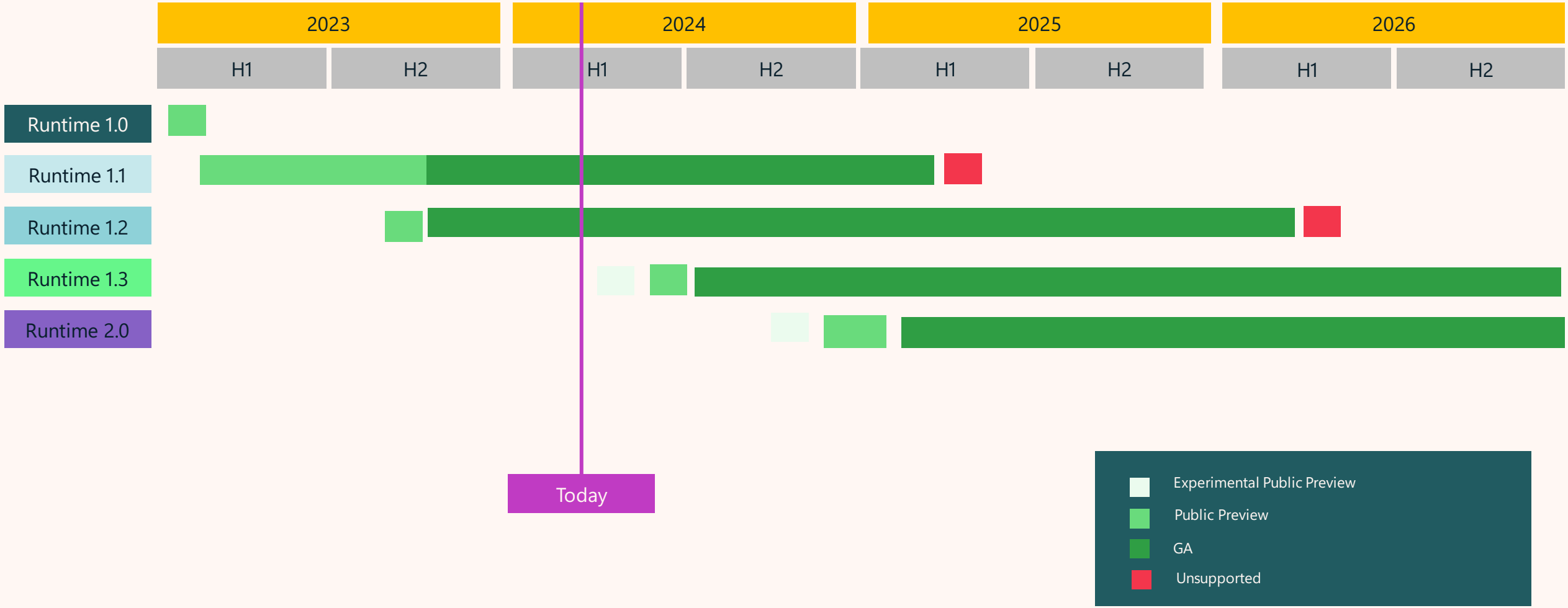
Spark Runtime

Fabric Spark Runtime

- Runtime 1.1: Spark 3.3 and Delta 2.2
- Runtime 1.2: Spark 3.4 and Delta 2.4
- Experimental Runtime: Spark 3.5 and Delta 3.0
- All Fabric artifacts automatically write Delta with V-Order optimization providing lightning-fast analytics using DirectLake in Power BI



Evolution of Fabric Runtimes



Packages and Versions Across Fabric Runtimes

	Runtime 1.1	Runtime 1.2	Runtime 1.3	Runtime 2.0
Apache Spark	3.3.1 -> 3.3.4	3.4.1 -> 3.4.2	3.5.0 -> 3.5.X	4.X
Operating System	Ubuntu 18.04	Mariner 2.0	Mariner 2.0	Mariner 3.0
Java	8	11	11	17
Scala	2.12.15	2.12.17	2.12.17	2.13
Python	3.10	3.10	3.11	3.12
Delta Lake	2.2	2.4	3.0	4.X
R	4.2.2	4.2.2	4.2.2	4.X.Y

Spark Autotune

Experience a significant boost in performance

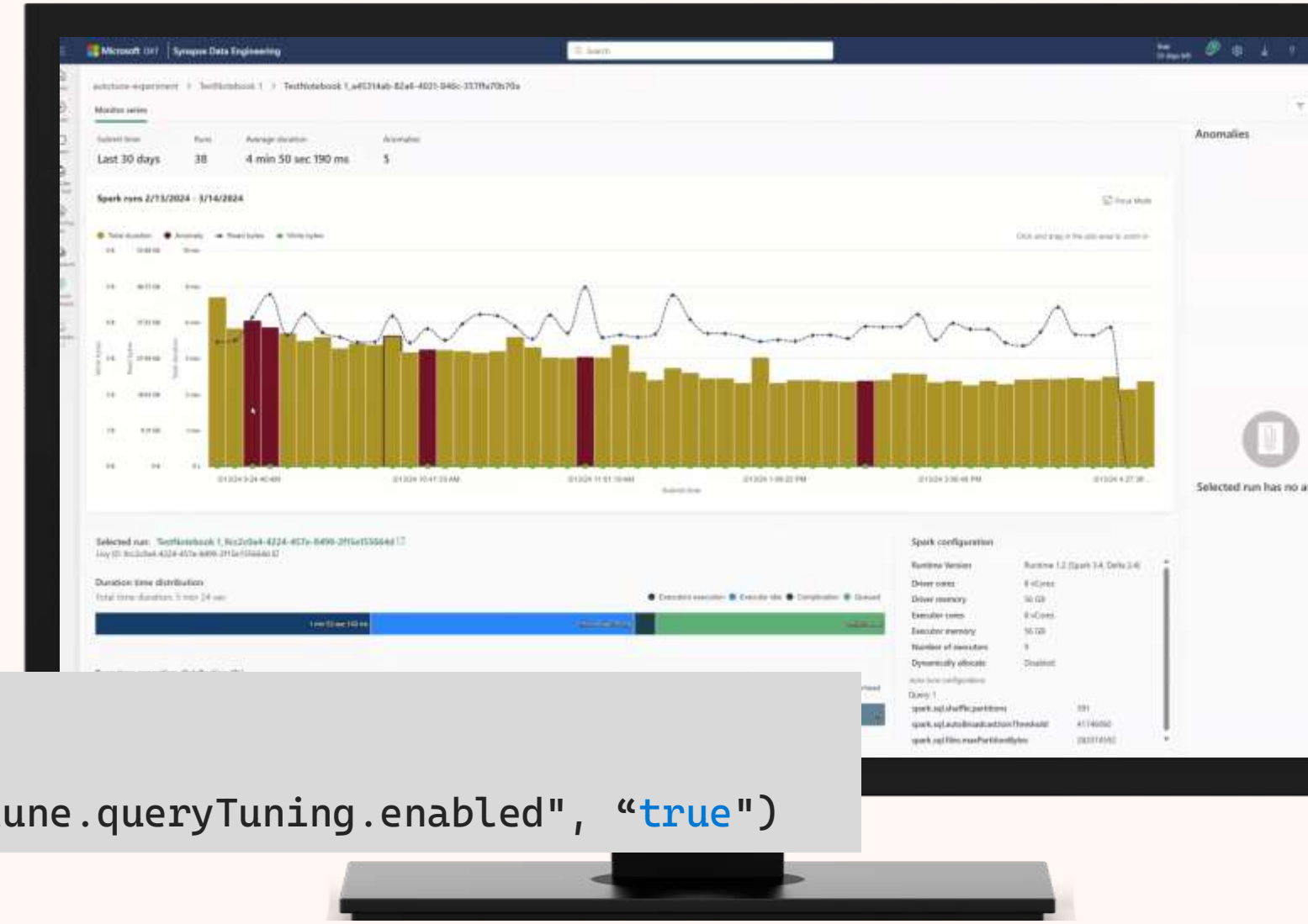
- Autotune configures three query-level Spark configurations:

✓ spark.sql.shuffle.partitions

✓ spark.sql.autoBroadcastJoinThreshold

✓ spark.sql.files.maxPartitionBytes

- Enable autotune with




```scala
spark.conf.set("spark.ms.autotune.queryTuning.enabled", “true")
```

Low Shuffle Merge

- Merge into command optimization enabled by default in Fabric
- Reducing amount of shuffle data for join
- Efficient when each underlying file size is large and MERGE updates only few rows in the file
- For large size job merge (newly added file size > 1GB), improved performance by 2-5x
- For small size job merge (newly added file size < 1GB), improved performance by 1-2x

```
Scala

spark.conf.set("spark.microsoft.delta.merge.lowShuffle
.enabled", “true")
```

```
SQL

SET
spark.microsoft.delta.merge.lowShuffle.enabled = true
```

Optimize Write

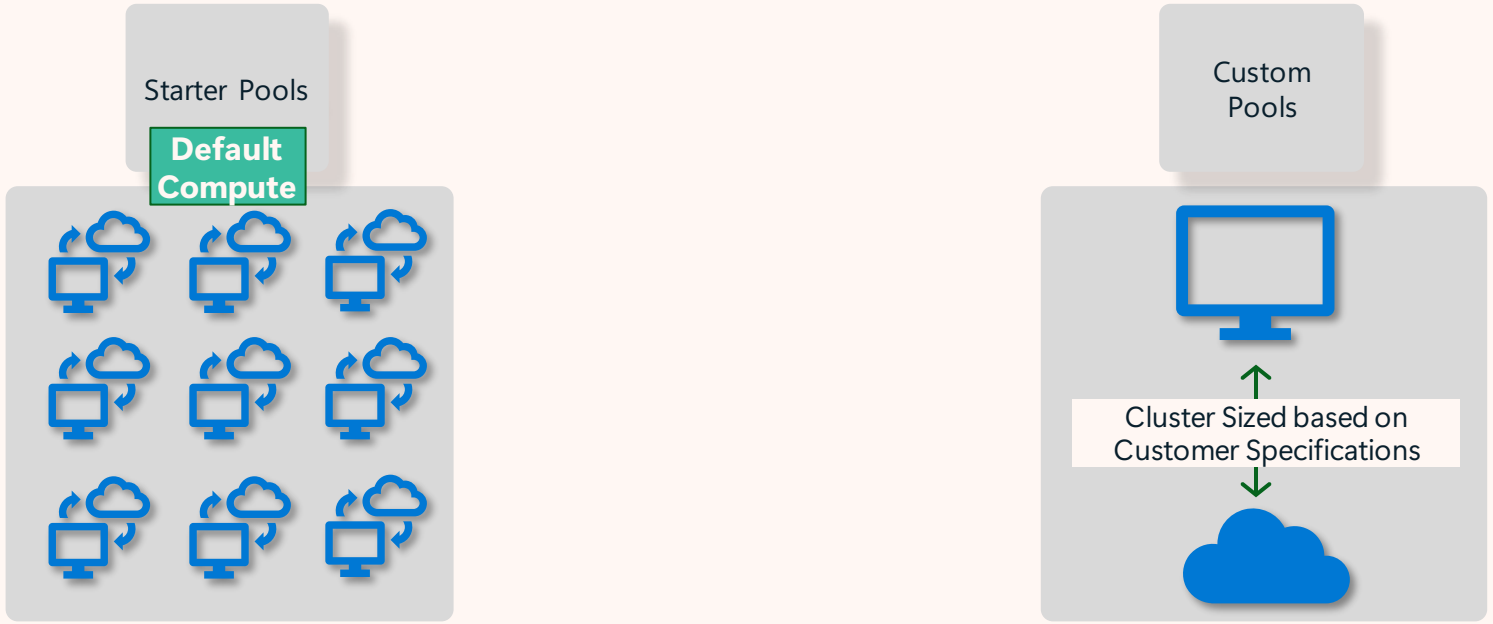
- Helps resolve many small files problem by shuffling data before writing
- Optimize Write is enabled by default in Fabric
- Targeting 1GB file size by default for VORDER query efficiency resulting in blazing fast performance when reading the tables from PowerBI or DW engines
- Write performance gain expected when a write job writing into many partition directories

```
Scala

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

```
SQL

SET
spark.microsoft.delta.optimizeWrite.enabled = false
```

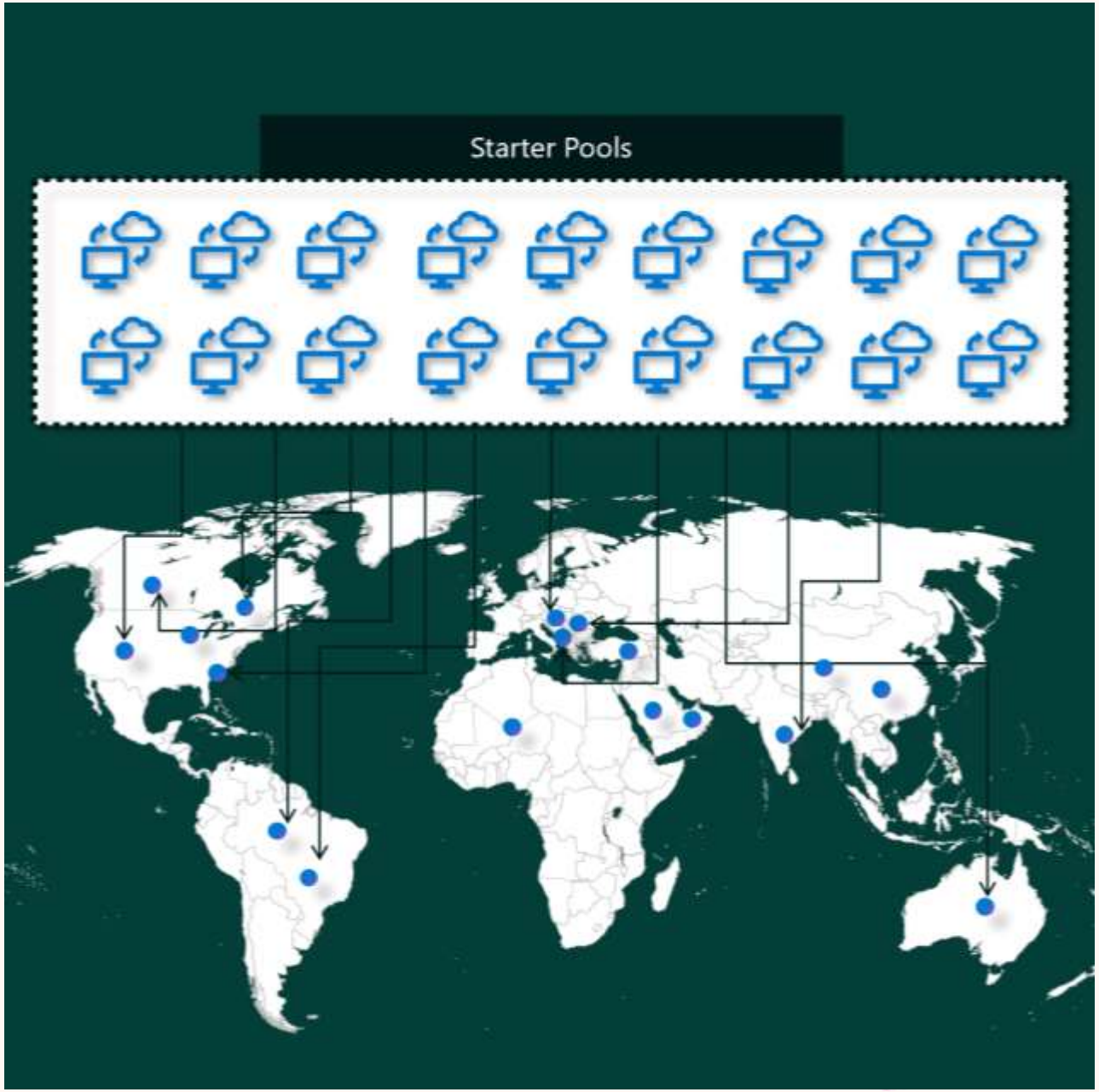
Spark Compute

- Spark Compute is a powerful and flexible way of using Spark on Microsoft Fabric.
- It enables you to perform various data engineering and data science tasks on your data stored in OneLake or other sources



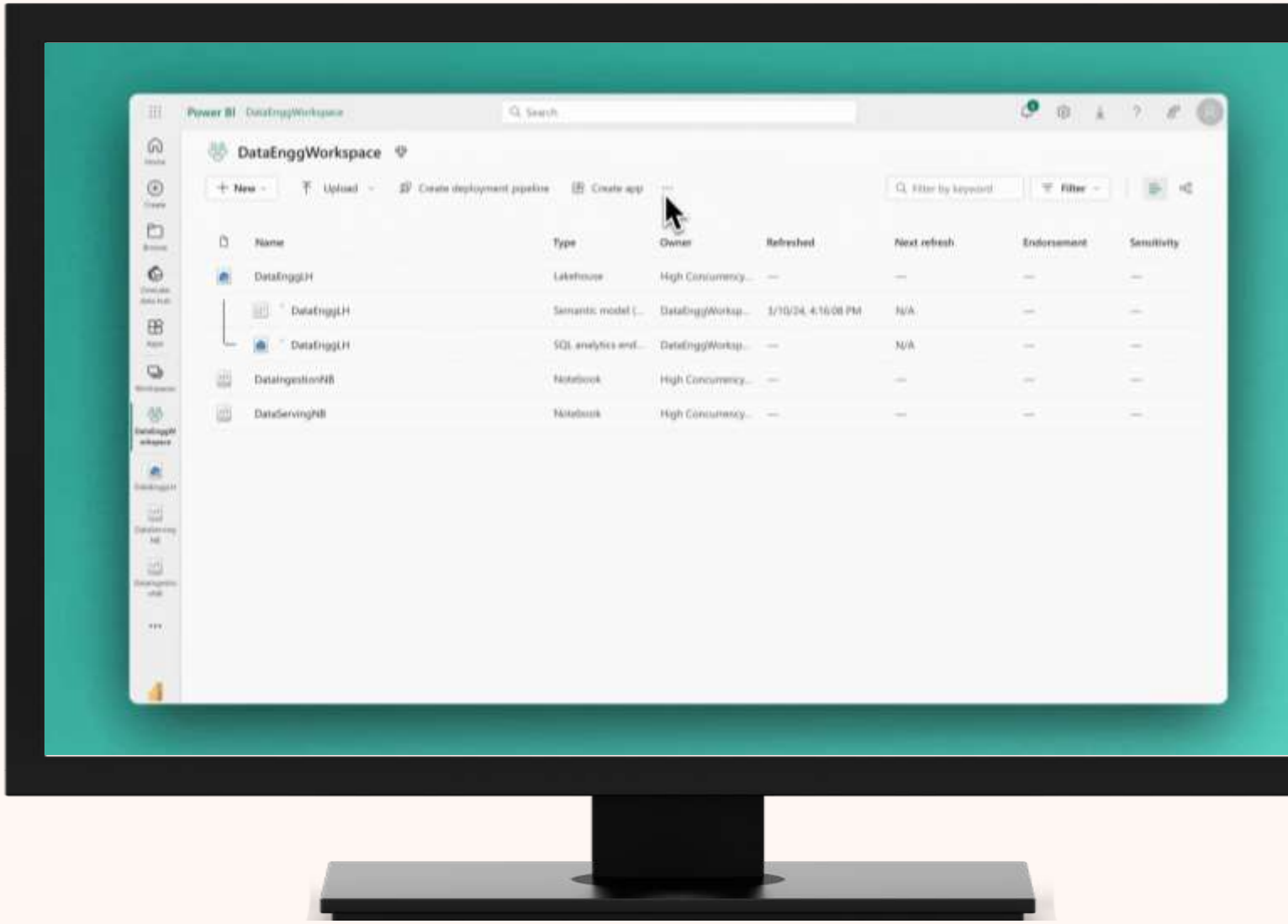
Starter Pools

- Starter pools are pre- hydrated clusters provide a spark jobs and sessions within 5 seconds
- Clusters in the Starter Pools are constantly rehydrated as they are assigned to incoming requests from users
- For workloads with smaller compute requirements - Users can set the pools to max node of 1 and still get the 5 Second session start experience.

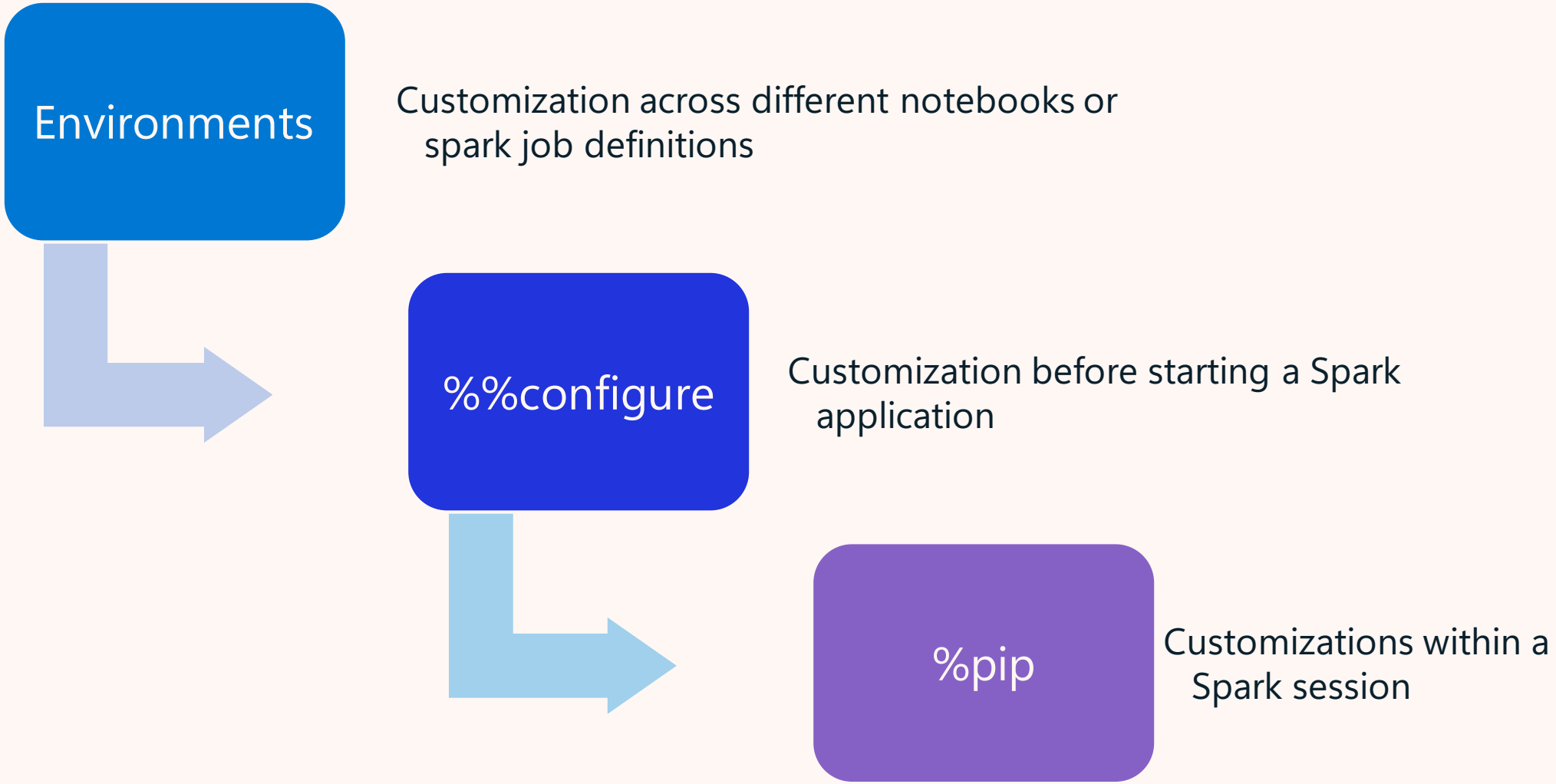


Custom Pools

- Custom Pools can be configured by workspace or capacity admins in Fabric allow customized compute options with different node sizes
- Pools created by admins can be selected at the workspace level or in an environment
- These spark clusters will take ~3 minutes for starting a spark session as they are acquired on-demand during job submission



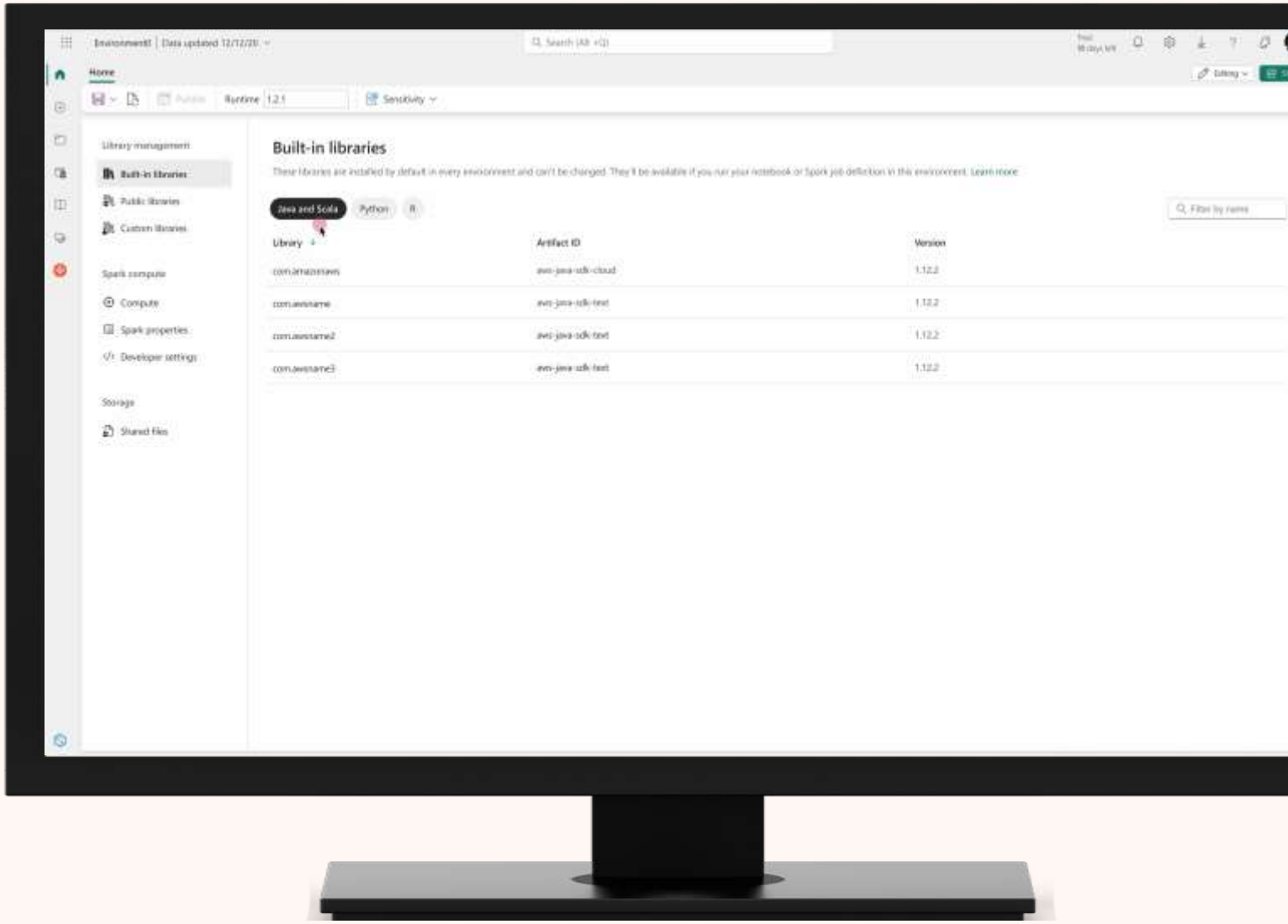
Session Customization



Customization with Environments and Library Management

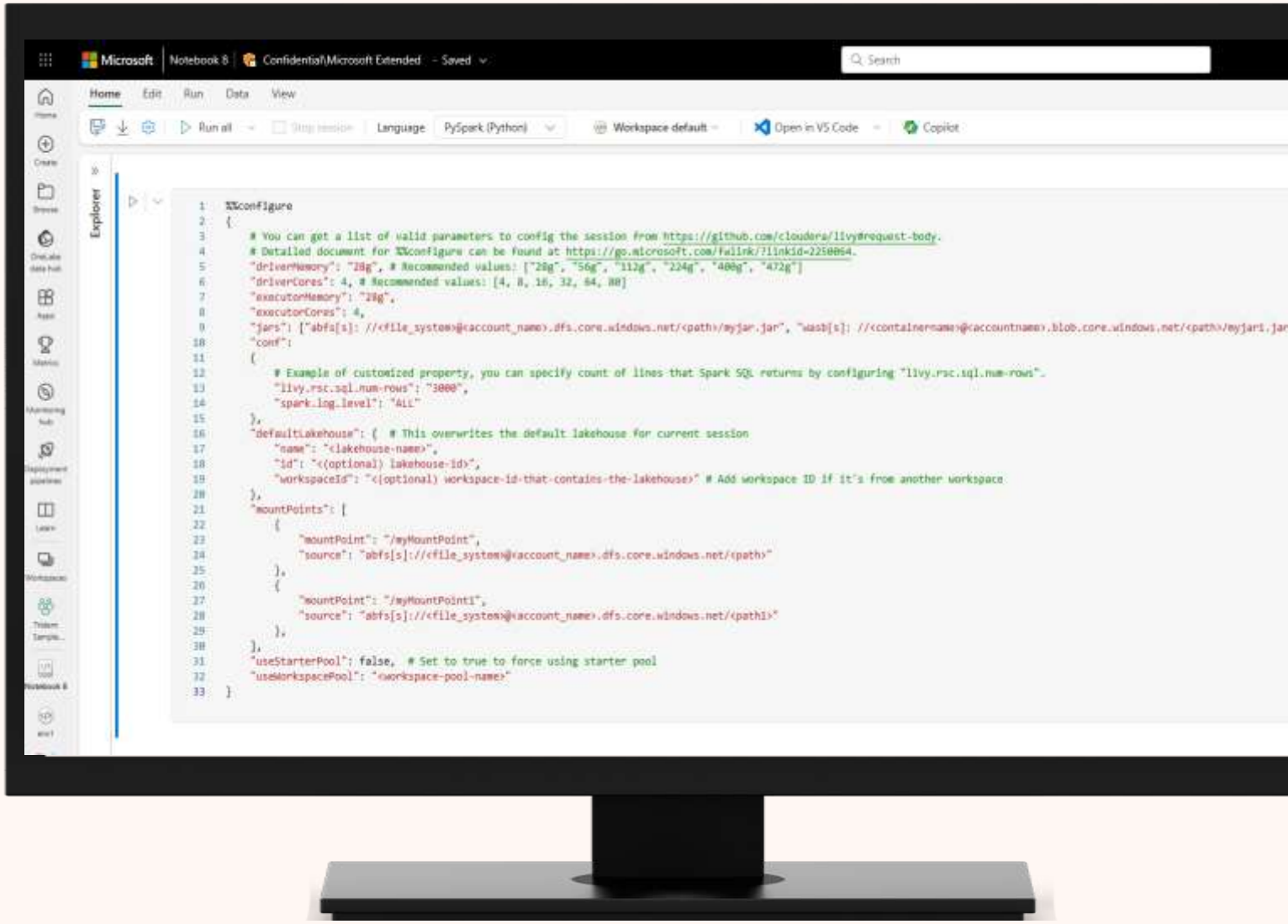
- The environment artifact allows users
  to configure all their Spark and library
  related settings in a single place

- Users can do the following (subject to the permissions they have been granted):
  - Install libraries
  - Choose from a pre-defined user pool
  - Set Spark properties



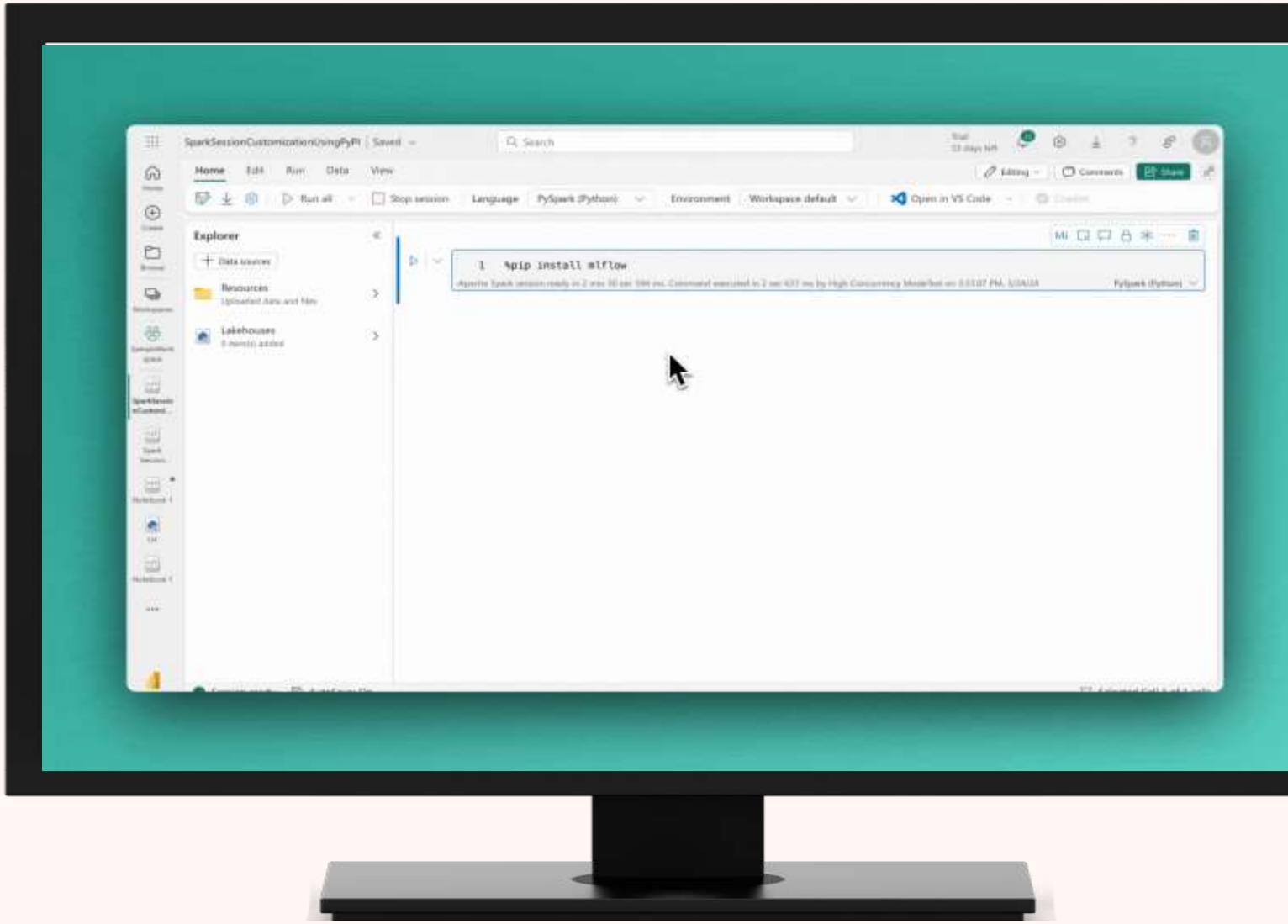
Customization using %%configure

- Customize the driverMemory, driverCores, executorMemory, executorCores
- Configure mounts, and default lakehouse and file-based dependencies
- Configure spark properties for your sessions
- Install jars to your Spark sessions



Customization using %pip

- Install library packages within active sessions
- %pip is recommended instead of !pip
  - As !pip will only install packages on the driver node
  - Packages that install through *!pip* will not affect when conflicts with built-in packages or when it's already imported in a notebook.



Datawarehouse database design

From relational to Star schema

Relational schemas are optimized for normalization (=reduce data redundancy) and speed of execution for transactional queries

Books
bookId (PK)
title
publishedDate
authorId (FK)
publisherId (FK)
price

n — 1

Publisher
publisherId (PK)
publisherName
publisherCountry

n — 1

Authors
authorId (PK)
authorName
firstName
lastName

PK: Primary Key. FK: Foreign Key.

Note: this schema is simplistic on purpose (e.g. a book could be authored by multiple authors)

How to move from relational to star schema?

1. Identify the business process
2. Determine the grain
3. Identify the dimensions
4. Design the fact table
5. Design dimension tables
6. Handle hierarchies
7. Generate surrogate keys
8. Manage Slowly Changing Dimensions
9. Optimize, Test, Document

Star schemas are optimized for analytical Query performances

FactBook
FactBookId(SK)
bookId (AK)
title
DimDateId (SK)
publishedDate
DimAuthorId (SK)
authorName
authorFirstName
authorLastName
DimPublisherId (SK)
publisherName
publisherCountry

Surrogate Keys (SK) is a unique identifier for each record in the datawarehouse. Acts as Primary Key

Alternate Keys (AK) maintain a connection to the original data sources

Fact tables contains the measurement of "fact" that you want to analyze

Most of star schema have a "date" dimension that is common across all entities

DimDate
DimDateKey
FullDate
DayOfWeek
DayOfMonth
MonthNumber
MonthName
Quarter
Year

DimPublisher
DimPublisherId (SK)
publisherId (AK)
publisherName
publisherCountry

DimAuthor
DimAuthorID (SK)
authorId (Ak)
authorName
firstName
lastName

Dimension tables contains attributes that are used to filter, group or label data contained in Fact tables

3

Module 3
Implement and manage a semantic model

Implement and manage semantic models (20–25%)

Design and build semantic models

Choose a storage mode, including Direct Lake

Identify use cases for DAX Studio and Tabular Editor 2

Implement a star schema for a semantic model

Implement relationships, such as bridge tables and many-to-many relationships

Write calculations that use DAX variables and functions, such as iterators, table filtering, windowing, and information functions

Implement calculation groups, dynamic strings, and field parameters

Design and build a large format dataset

Design and build composite models that include aggregations

Implement dynamic row-level security and object-level security

Validate row-level security and object-level security

Optimize enterprise-scale semantic models

Implement performance improvements in queries and report visuals

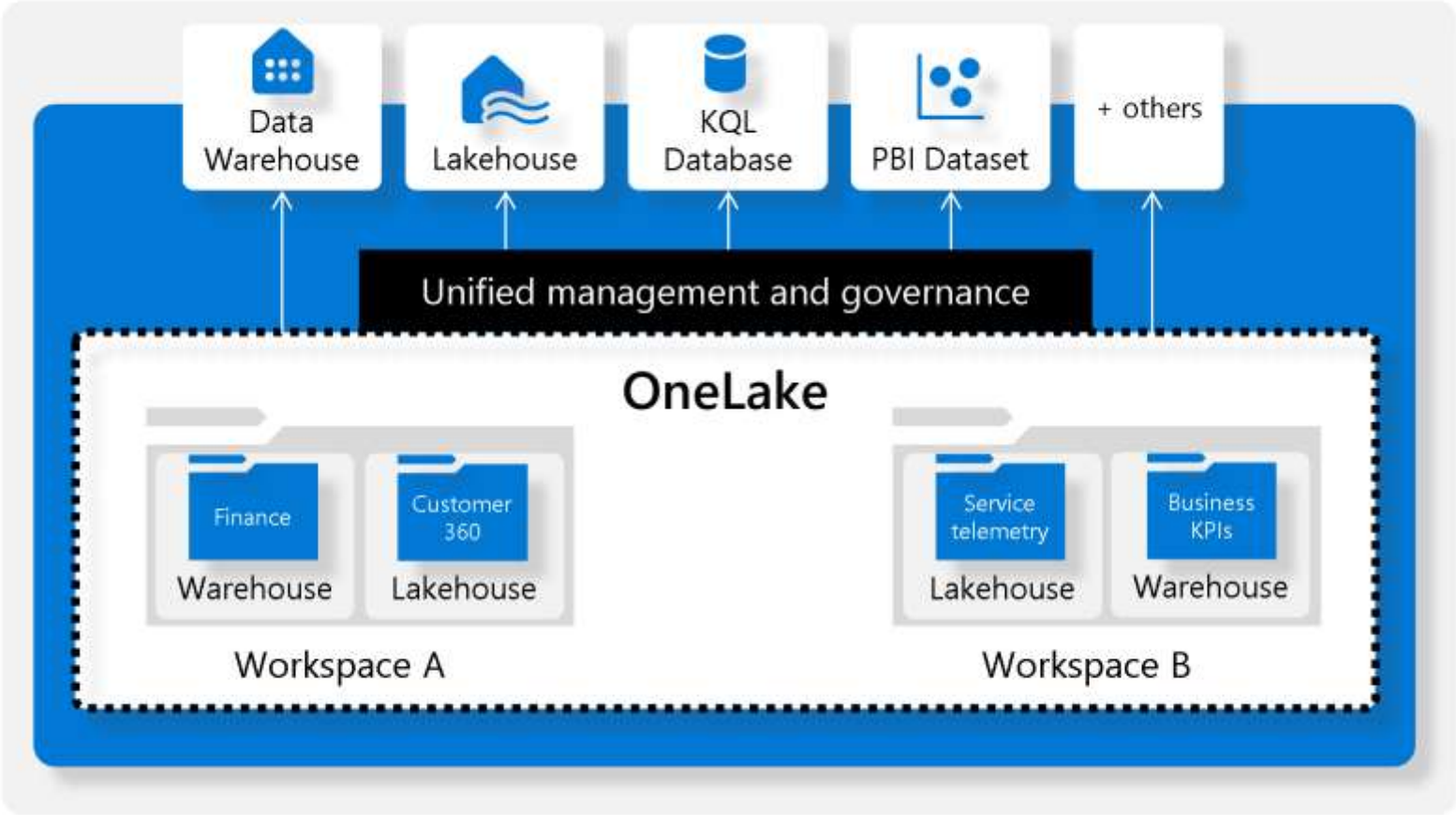
Improve DAX performance by using DAX Studio

Optimize a semantic model by using Tabular Editor 2

Implement incremental refresh

Understand Fabric warehouses

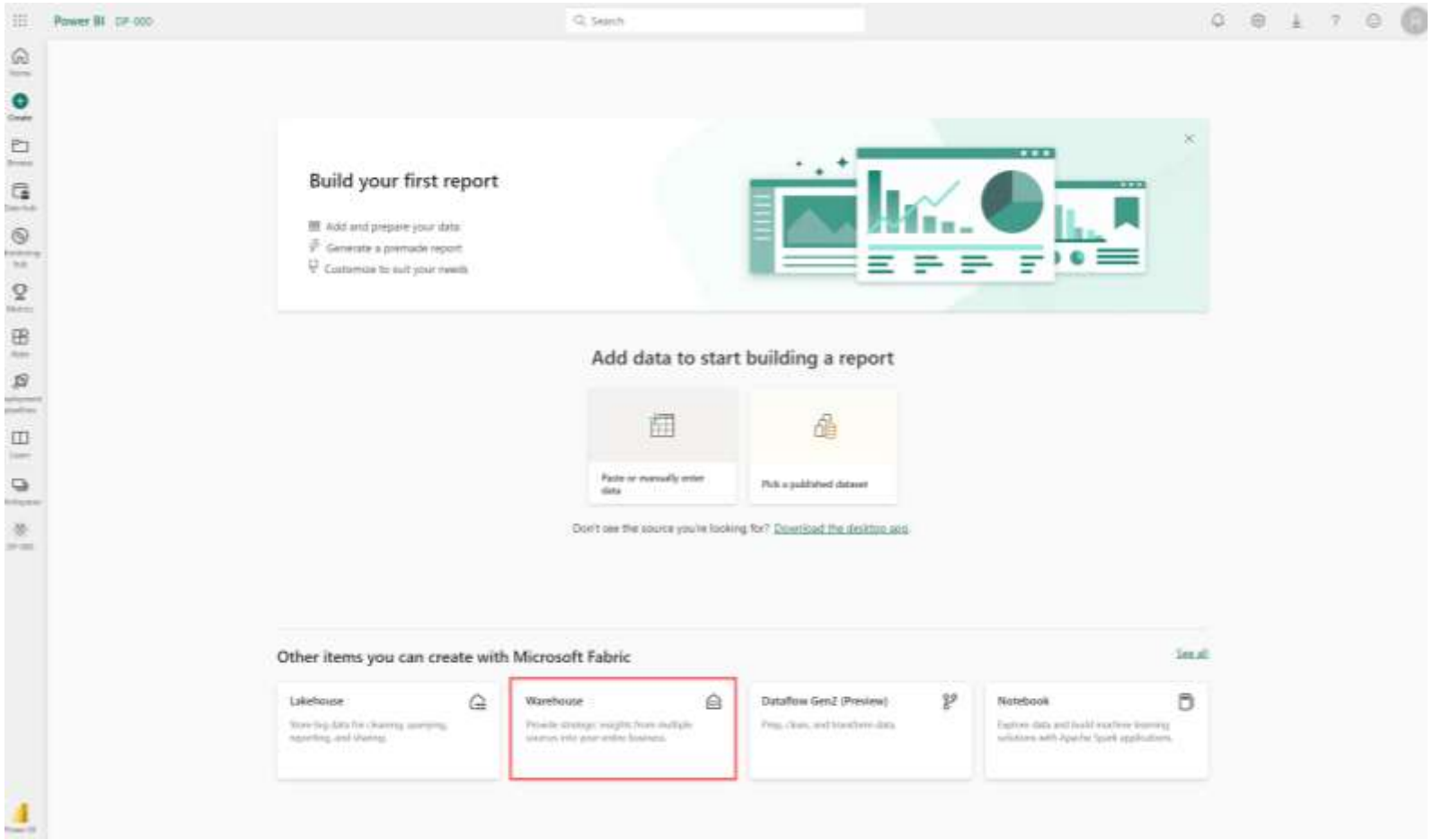
- Centered on single data lake
- Powered by Synapse Analytics
- Fully supports T-SQL
- Parquet file format



Create a data warehouse in Fabric

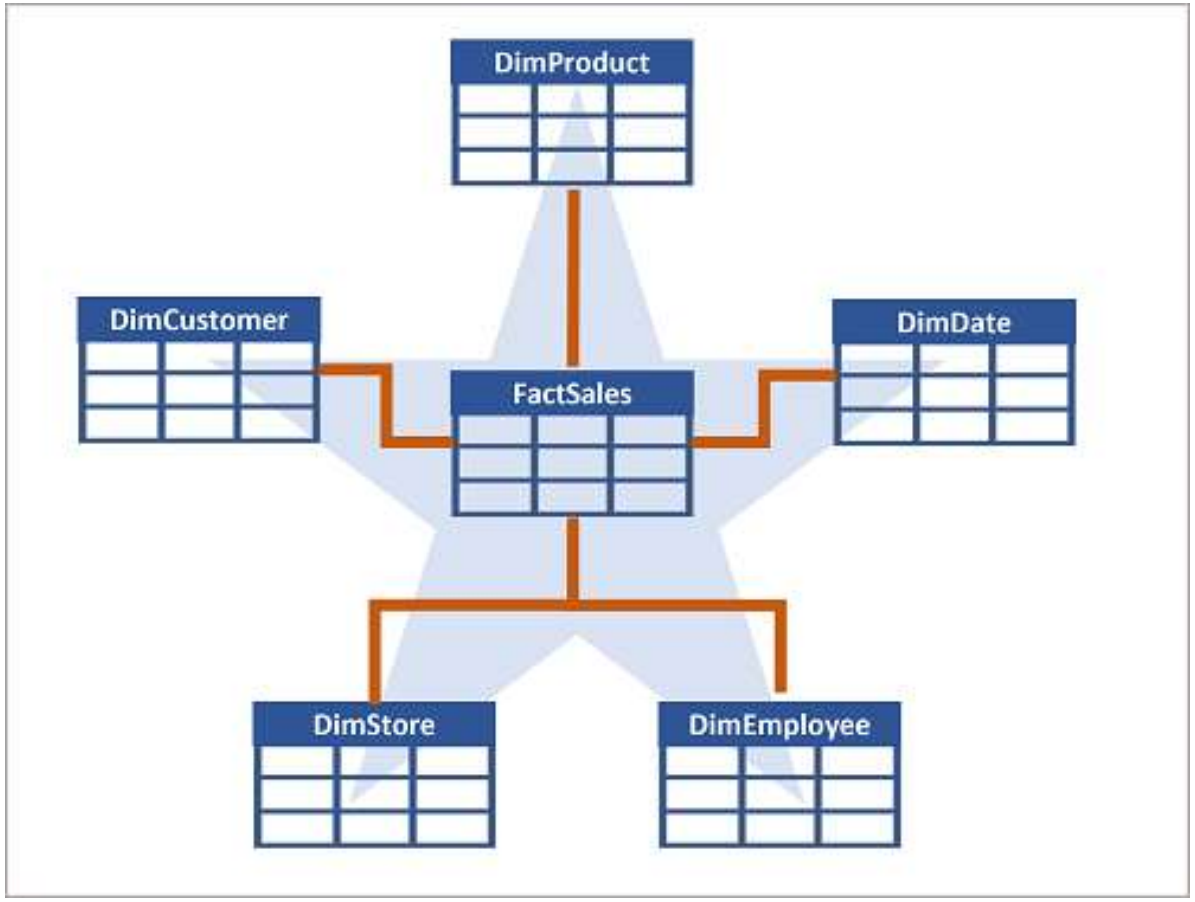
Within Fabric service portal

- Navigate to Fabric workspace
- Select new item > Warehouse
- Name warehouse



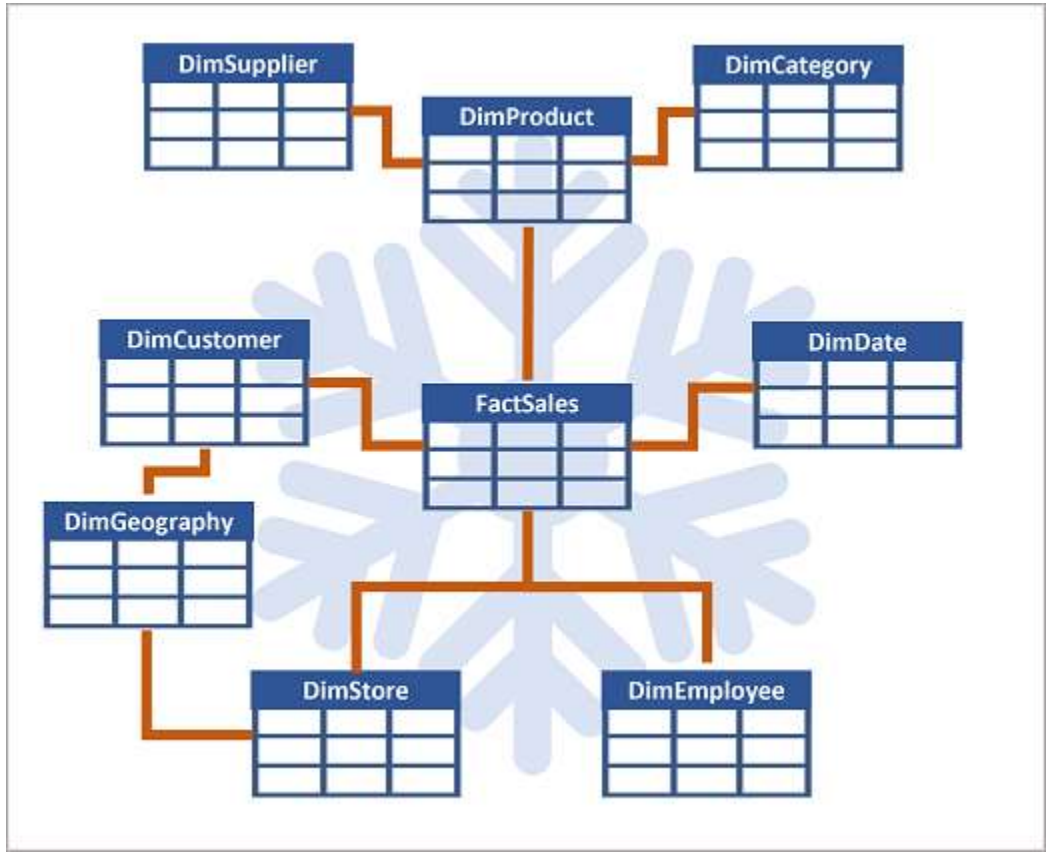
Design a dimensional warehouse

- Fact tables
- Dimension tables
- Unique keys
  - Surrogate key
  - Alternate key



Consider a snowflake schema

- Star schema, further denormalized
- More granular dimensions



Special types of dimension tables

Time dimensions

- Extensive date table
- Ideal for aggregation
- Columns may include:
  - Year
  - Quarter
  - Month
  - Day
  - Fiscal Year

Slowly changing dimensions

- Changes to attributes
- Analyze changes over time
- Changes may include:
  - Customer address
  - Product price

Ingest data into a data warehouse

Fabric Pipelines

Fabric Dataflow Gen2

Cross-database querying

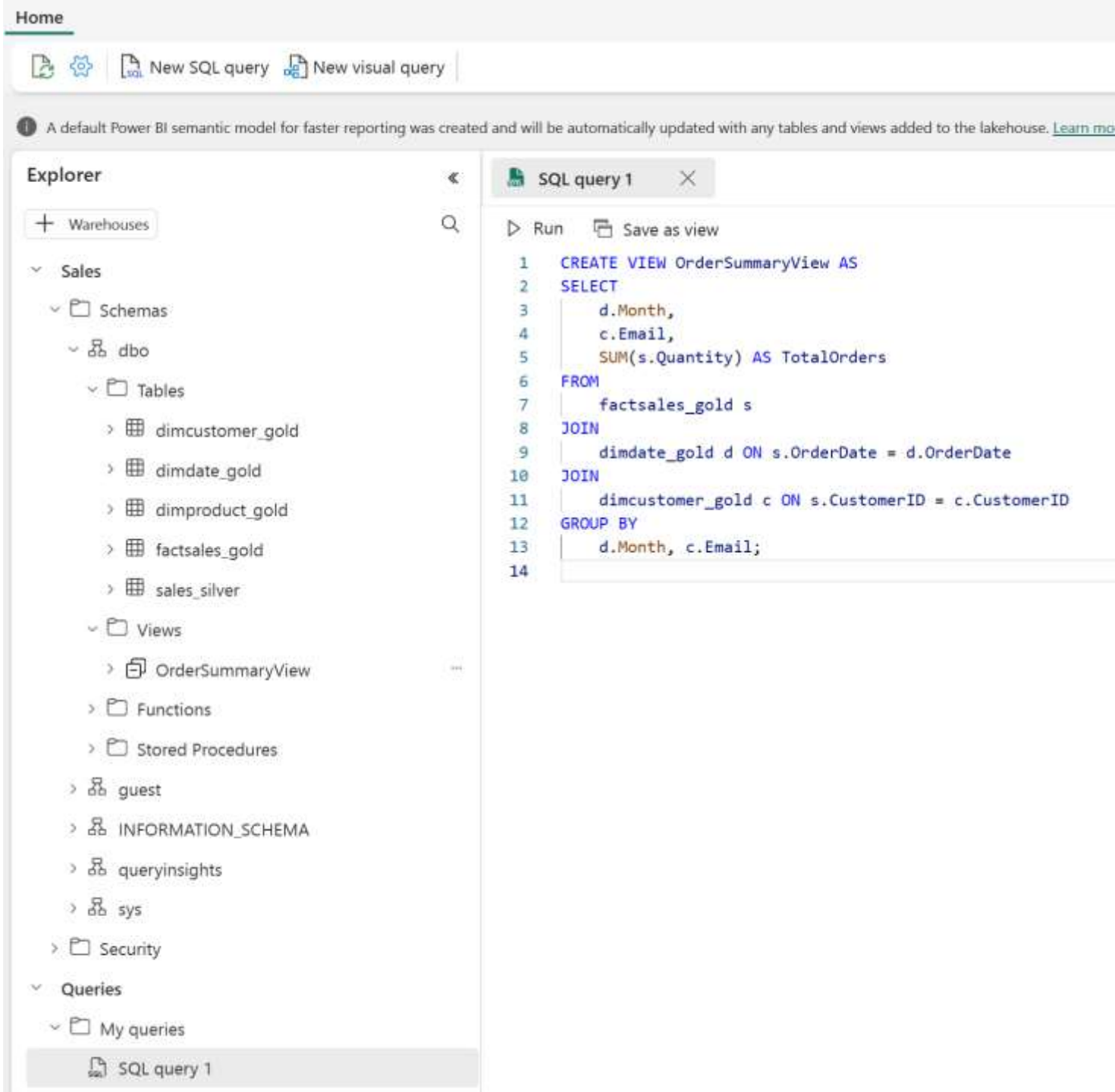
COPY INTO command

Query data in Fabric

Use the SQL query editor

Like using SQL Server Management Studio (SSMS) or Azure Data Studio (ADS).

- Intellisense
- Code completion
- Syntax highlighting
- Client-side parsing
- Validation

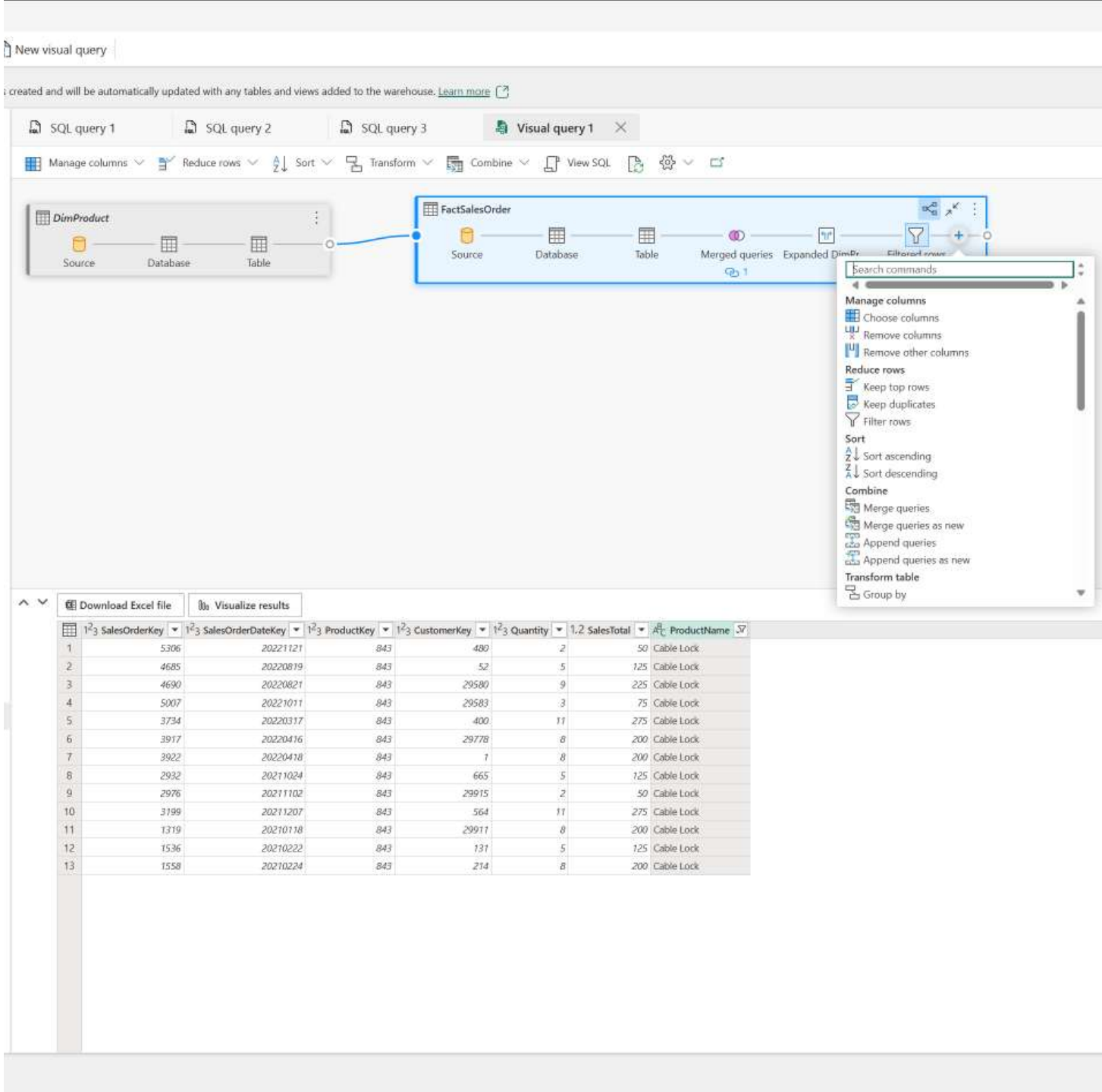


Visualize queries in Fabric

Create a visual query

Drag table to the canvas, then transform as needed:

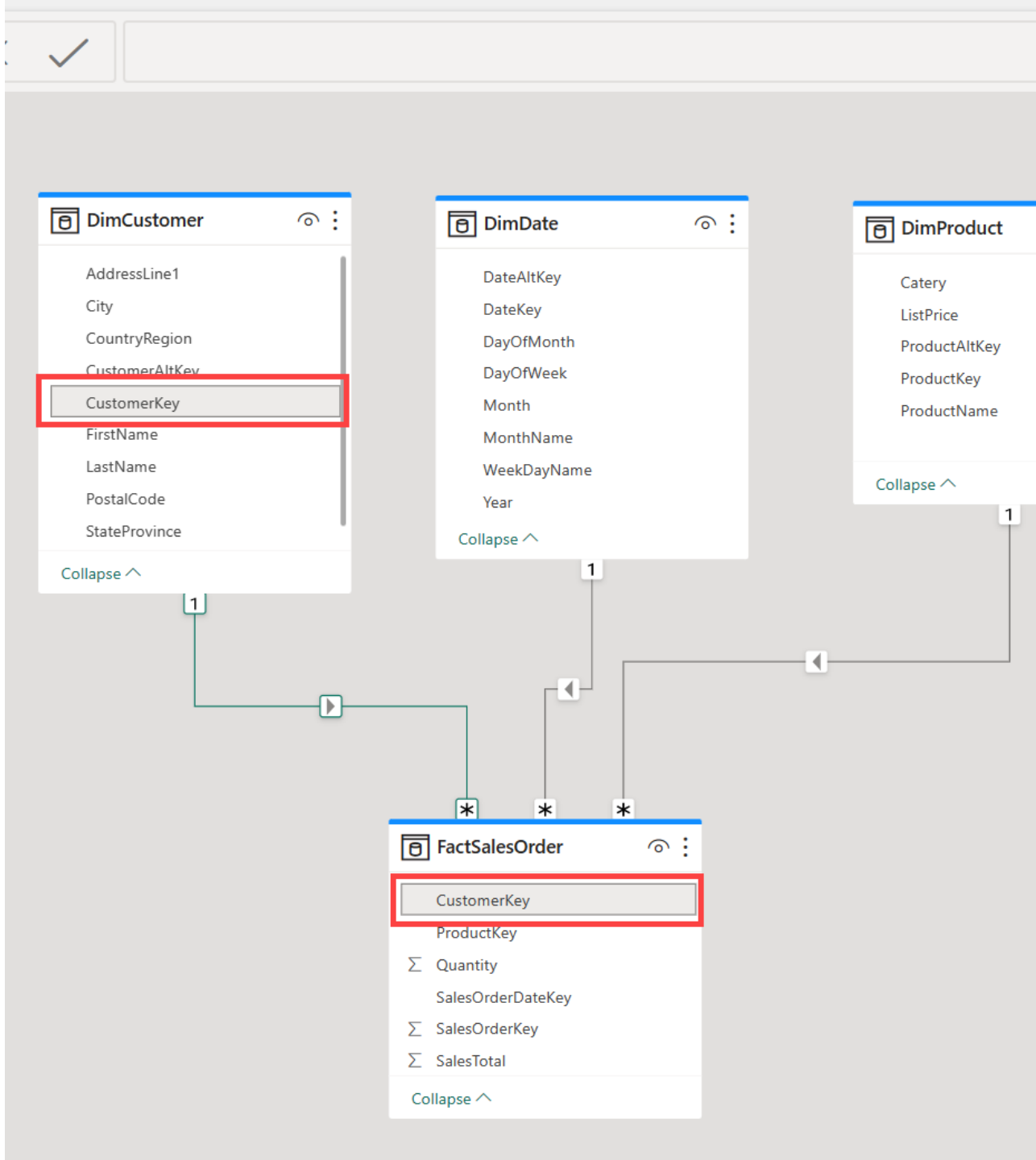
- Manage columns
- Filter rows
- Replace values
- Group by
- And more



Build relationships

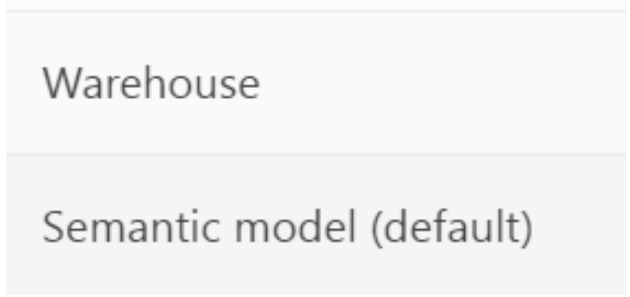
Connect fact and dimension tables

- Use **Model** view in warehouse in Fabric portal.
- Identify the appropriate key between tables.
- Drag columns from one table on top of the related column in another table.



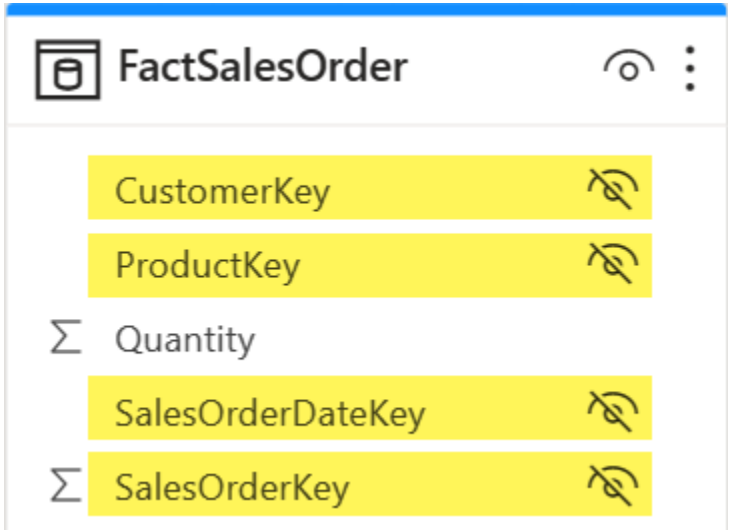
Understand the default semantic model

Fabric automatically creates a default semantic model for Power BI users to use for report creation.

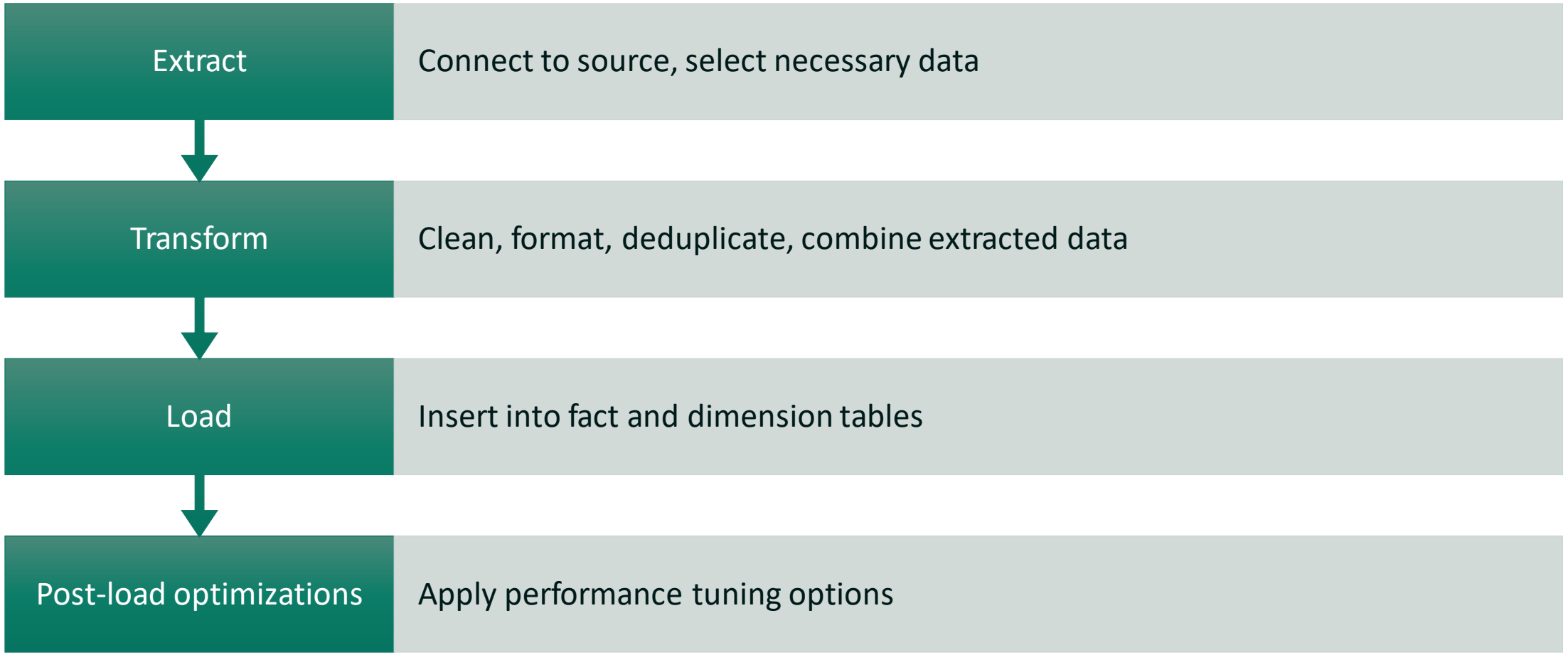


Consider shaping the semantic model to reduce downstream transformations.

- User-friendly column names
- Hide columns or tables from view

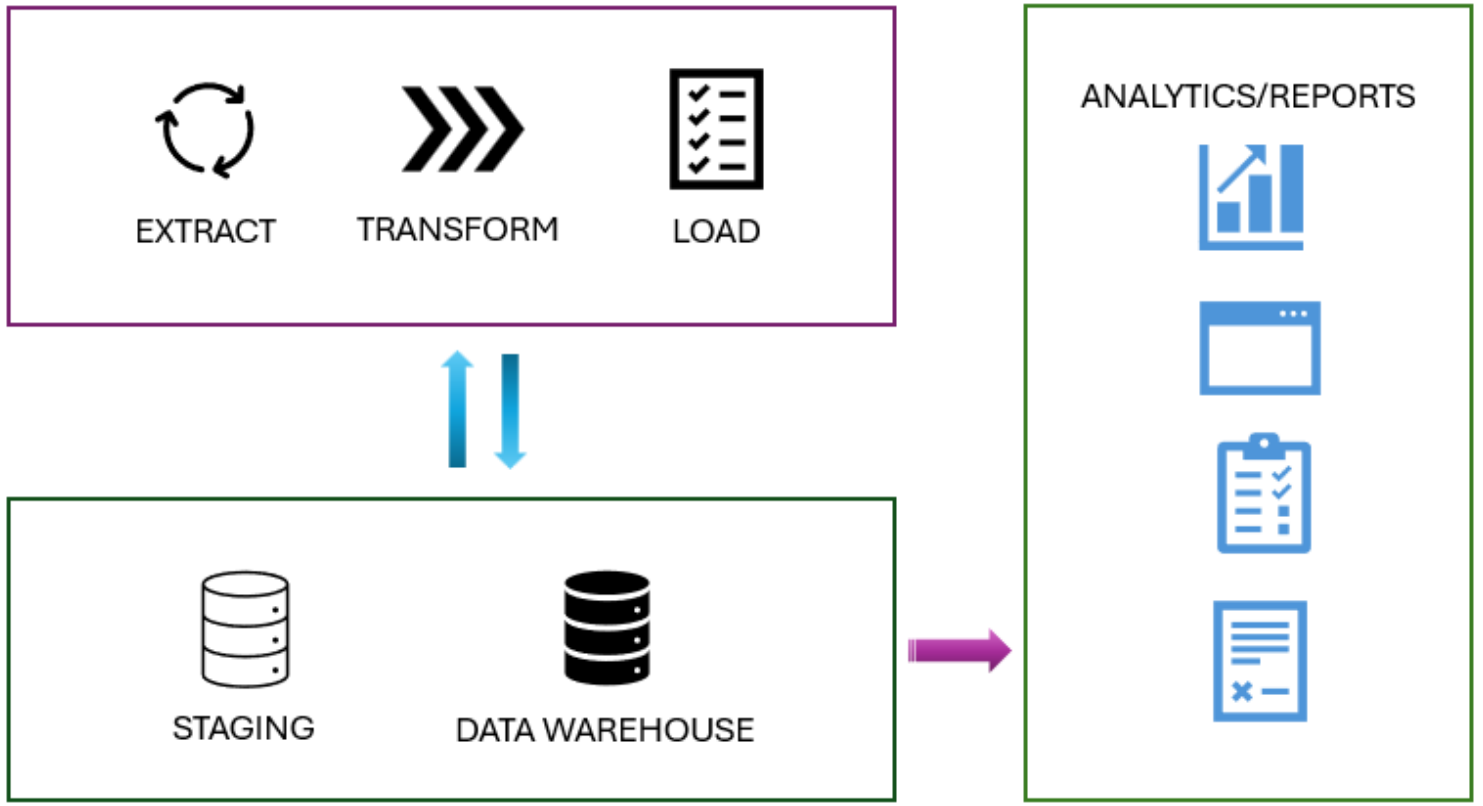


Understand ETL (Extract, Transform and Load)



Stage the data

- Temporary storage and transformation
- Simplifies load operations



Different data load types

Full (initial) load

- Truncate table and load data
- Longer load time
- No history stored
- Best for initial load or refresh

Incremental load

- Append data to tables
- Faster updates
- Preserves history (timestamp)
- Best for frequent updates

Create tables with T-SQL

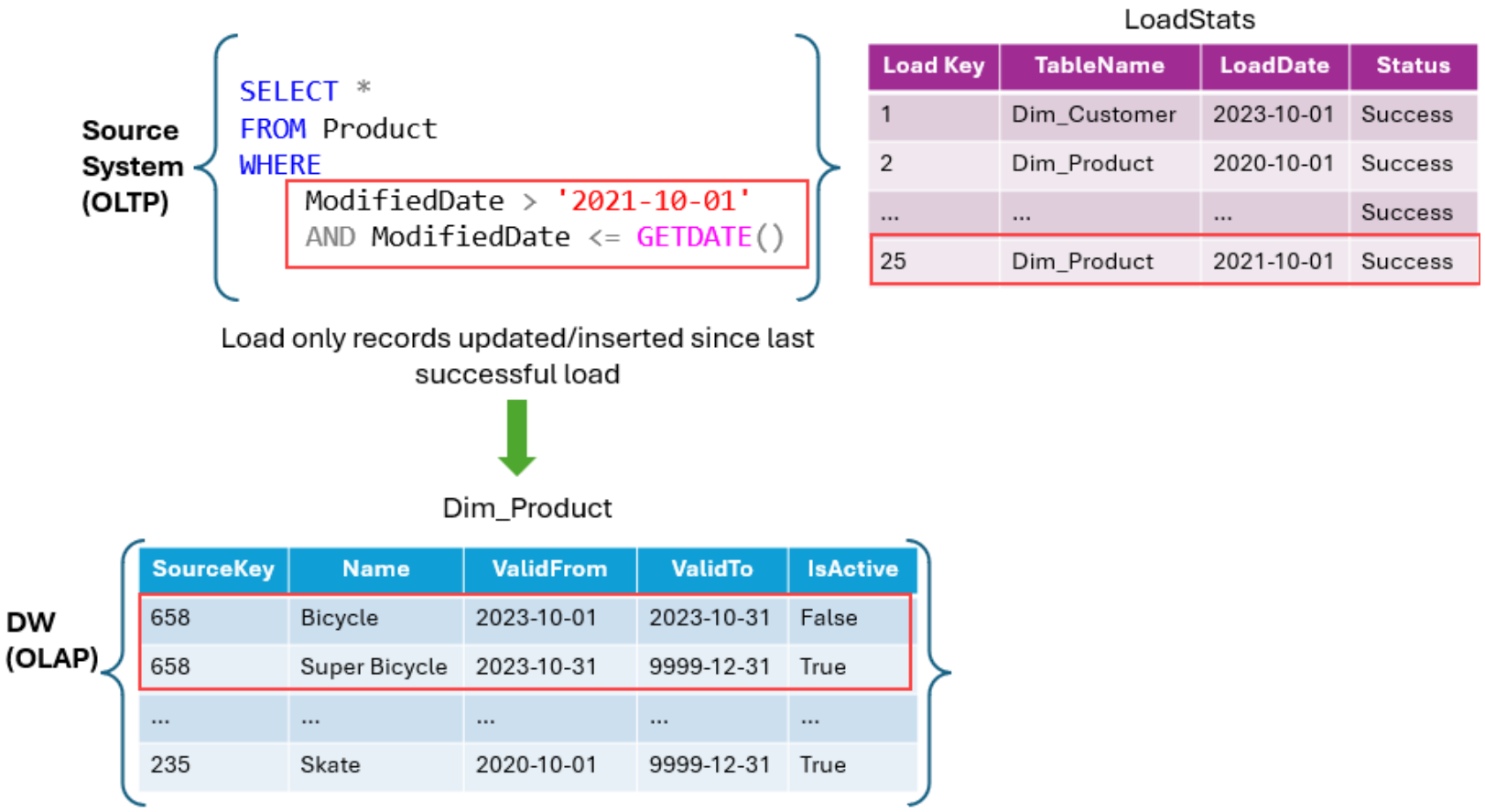
- SQL Server Management Studio
- Microsoft Fabric service portal

```
COPY INTO dbo.Region
FROM
'https://mystorageaccountxxx.blob.core.windo
ws.net/private/Region.csv' WITH (
            FILE_TYPE = 'CSV'
            ,CREDENTIAL = (
                IDENTITY = 'Shared Access
Signature'
                , SECRET = 'xxx'
                )
            ,FIRSTROW = 2
            )
GO
```

Load dimension tables

Slowly changing dimension types

- **0**: never changes
- **1**: overwrites, no history
- **2**: appends, tracks changes
- **3**: add history column
- **4**: add new dimension
- **5**: like type 2 for large changes
- **6**: combo of types 2 and 3



Example

Update table *Dim_Products,*

designed as a **Type 2 SCD.**

```
IF EXISTS (SELECT 1 FROM Dim_Products WHERE
SourceKey = @ProductID AND IsActive =
'True')

BEGIN

    -- Existing product record

    UPDATE Dim_Products

    SET ValidTo = GETDATE(), IsActive =
'False'

    WHERE SourceKey = @ProductID AND
IsActive = 'True';

END

ELSE

BEGIN

    -- New product record

    INSERT INTO Dim_Products (SourceKey,
ProductName, StartDate, EndDate, IsActive)

    VALUES (@ProductID, @ProductName,
GETDATE(), '9999-12-31', 'True');

END
```

Load fact tables

Frequent inserts and updates.

Common uses include:

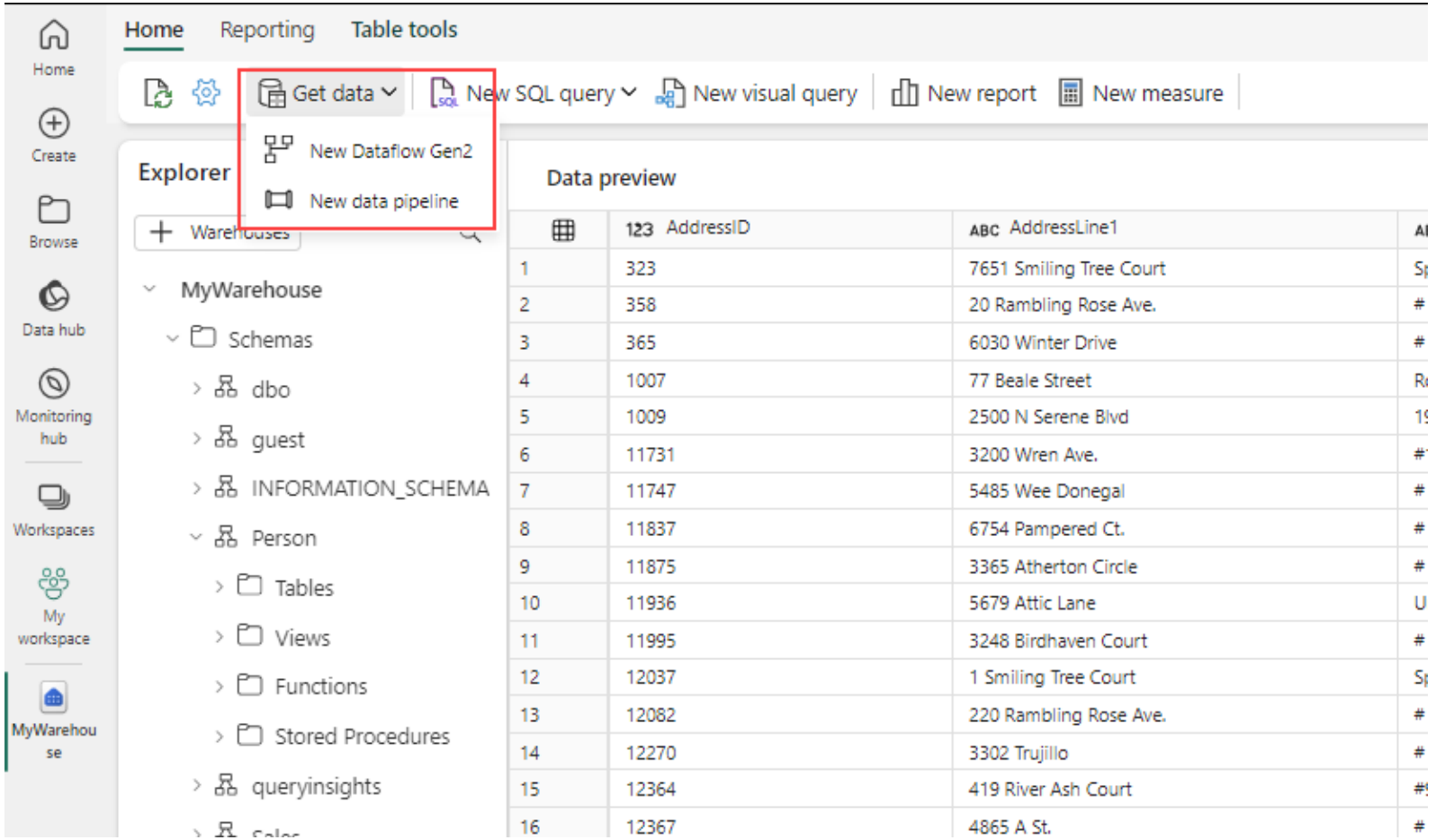
- Customers
- Claims
- Events
- Visits
- Calls
- Inventory

```
-- Lookup keys in dimension tables
INSERT INTO Fact_Sales
   (DateKey, ProductKey, OrderID, Quantity, Price,
   LoadTime)
SELECT d.DateKey
   , p.ProductKey
   , o.OrderID
   , o.Quantity
   , o.Price
   , GETDATE()
FROM Order_Detail o
JOIN Dim_Date d ON o.OrderDate = d.Date
JOIN Dim_Product p ON o.ProductID = p.ProductID;
```

Load data with Fabric pipelines

Orchestrate activities

- Copy Data
- Schedule
- Parameters
- Delete Data



Load data with T-SQL

Use COPY statement

- External Azure storage
- Specify file format (PARQUET / CSV)
- Error handling
- Multiple files

```
-- Load as CSV

COPY my_table

FROM
'https://myaccount.blob.core.windows.net/myblobcontainer/
folder0/*.csv,
https://myaccount.blob.core.windows.net/myblobcontainer/f
older1/'

WITH (FILE_TYPE = 'CSV',

    CREDENTIAL=(IDENTITY= 'Shared Access Signature',
SECRET='<Your_SAS_Token>')

    FIELDTERMINATOR = '|’)


-- Load as PARQUET

COPY INTO test_parquet

FROM
'https://myaccount.blob.core.windows.net/myblobcontainer/
folder1/*.parquet'

WITH (CREDENTIAL=(IDENTITY= 'Shared Access Signature',
SECRET='<Your_SAS_Token>'))
```

Load from other items

Combine into one warehouse

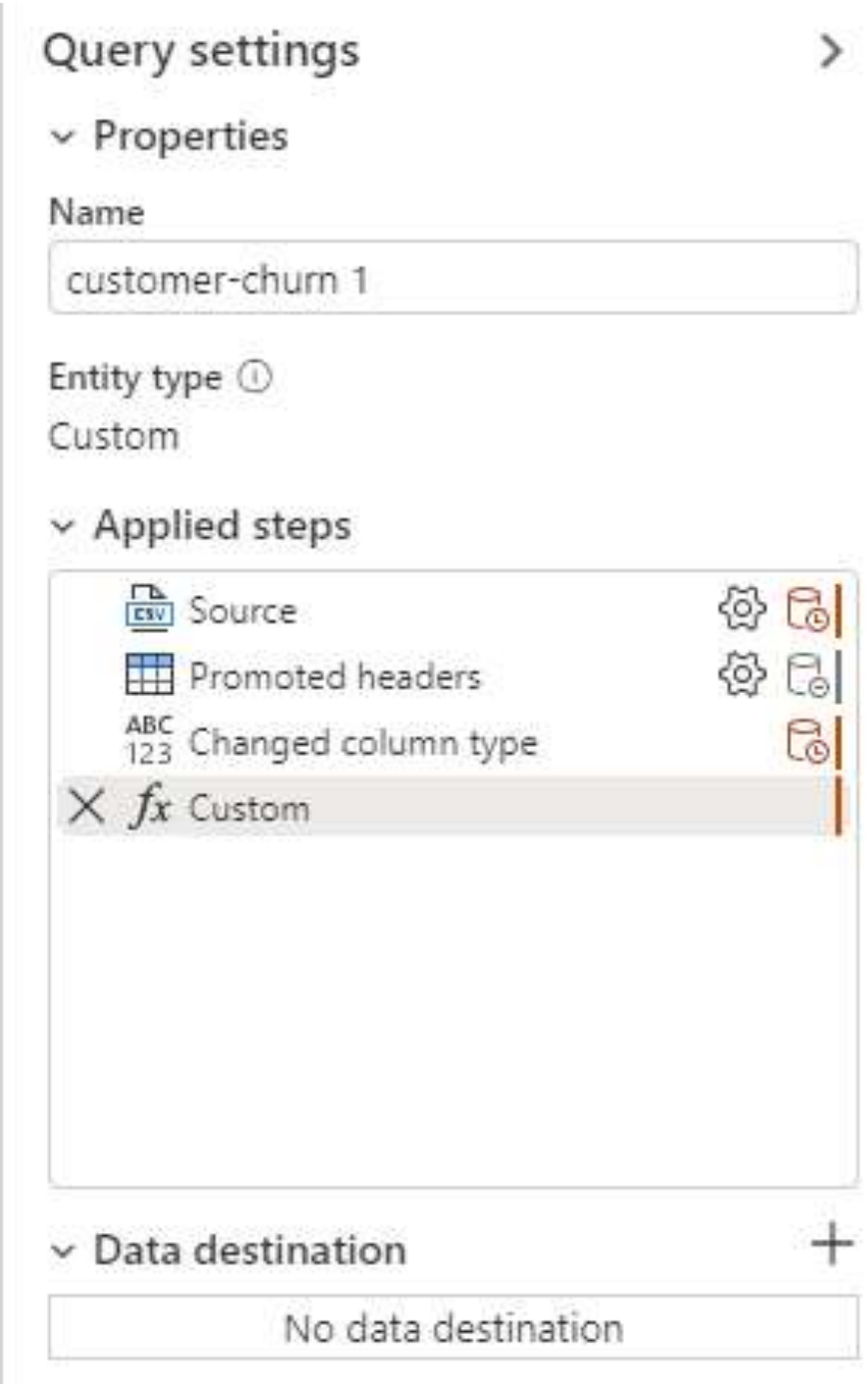
- Requires three-part-naming
- CREATE TABLE AS SELECT
- INSERT…SELECT

```
CREATE TABLE
[analysis_warehouse].[dbo].[combined_data]
AS
SELECT
FROM [sales_warehouse].[dbo].[sales_data] sales
INNER JOIN [social_lakehouse].[dbo].[social_data]
social
ON sales.[product_id] = social.[product_id];


INSERT INTO
[analysis_warehouse].[dbo].[combined_data]
SELECT
    sales.product_id
FROM [sales_warehouse].[dbo].[sales_data] sales
INNER JOIN [social_lakehouse].[dbo].[social_data]
social
ON sales.product_id = social.product_id;
```

Load data with Dataflow Gen2

Ingest and transform with Power Query

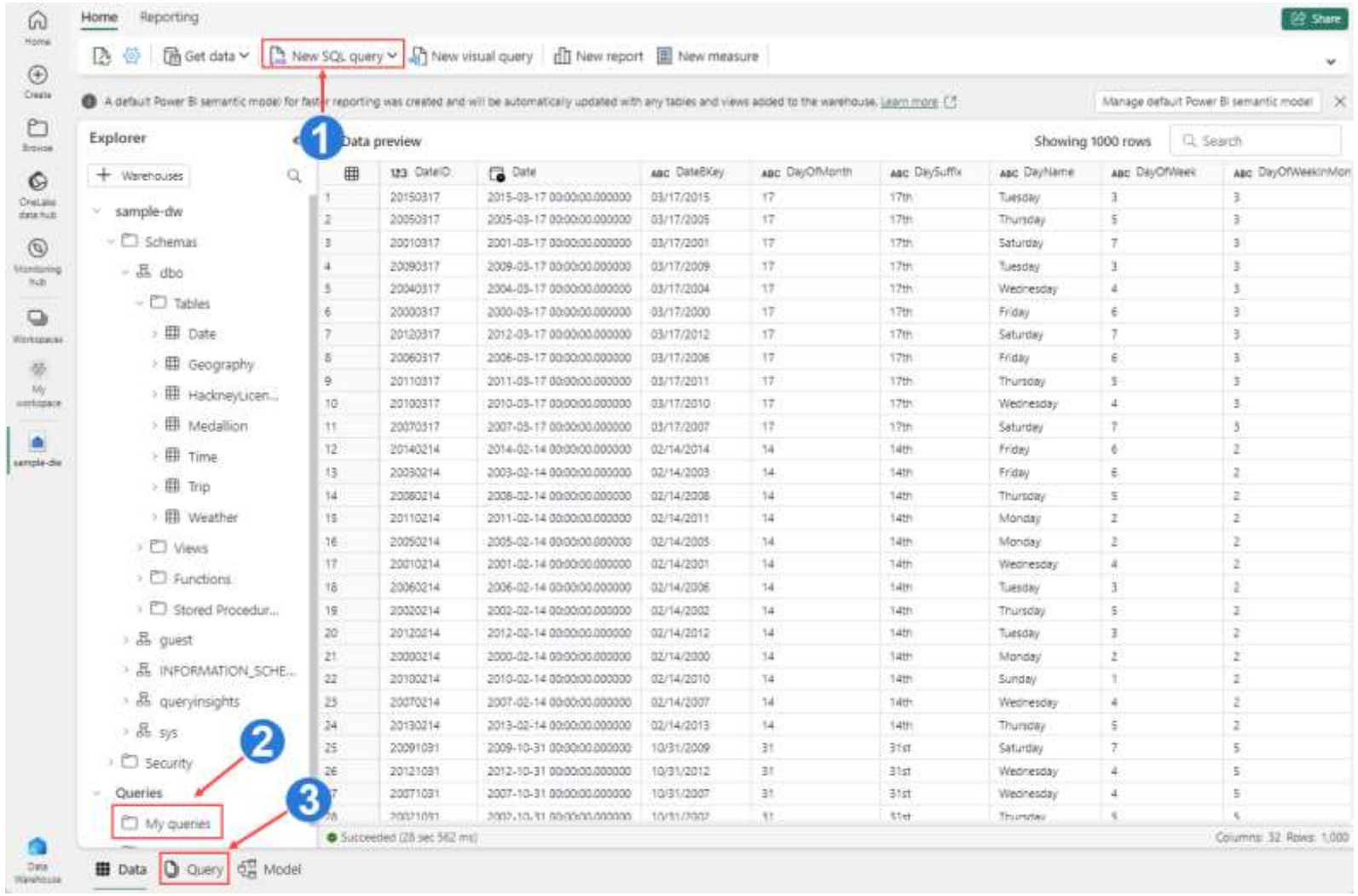
- Connect to data source
- Transform data – leverage copilot
- Append or replace
- Add data destination
  - Lakehouse
  - Warehouse
  - Azure SQL Database
  - Azure Data Explorer (Kusto)
  - Azure Synapse Analytics (SQL DW)



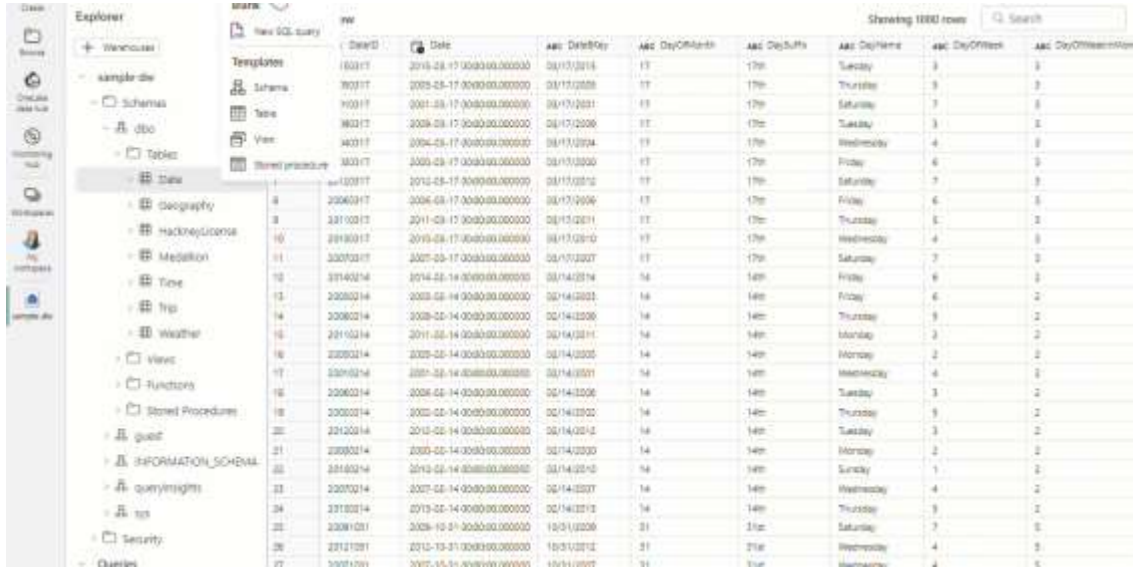
Use the SQL query editor

Warehouse explorer in Fabric portal

- Use T-SQL
- Supports Intellisense
- Stores in **My queries**



Run queries and export results



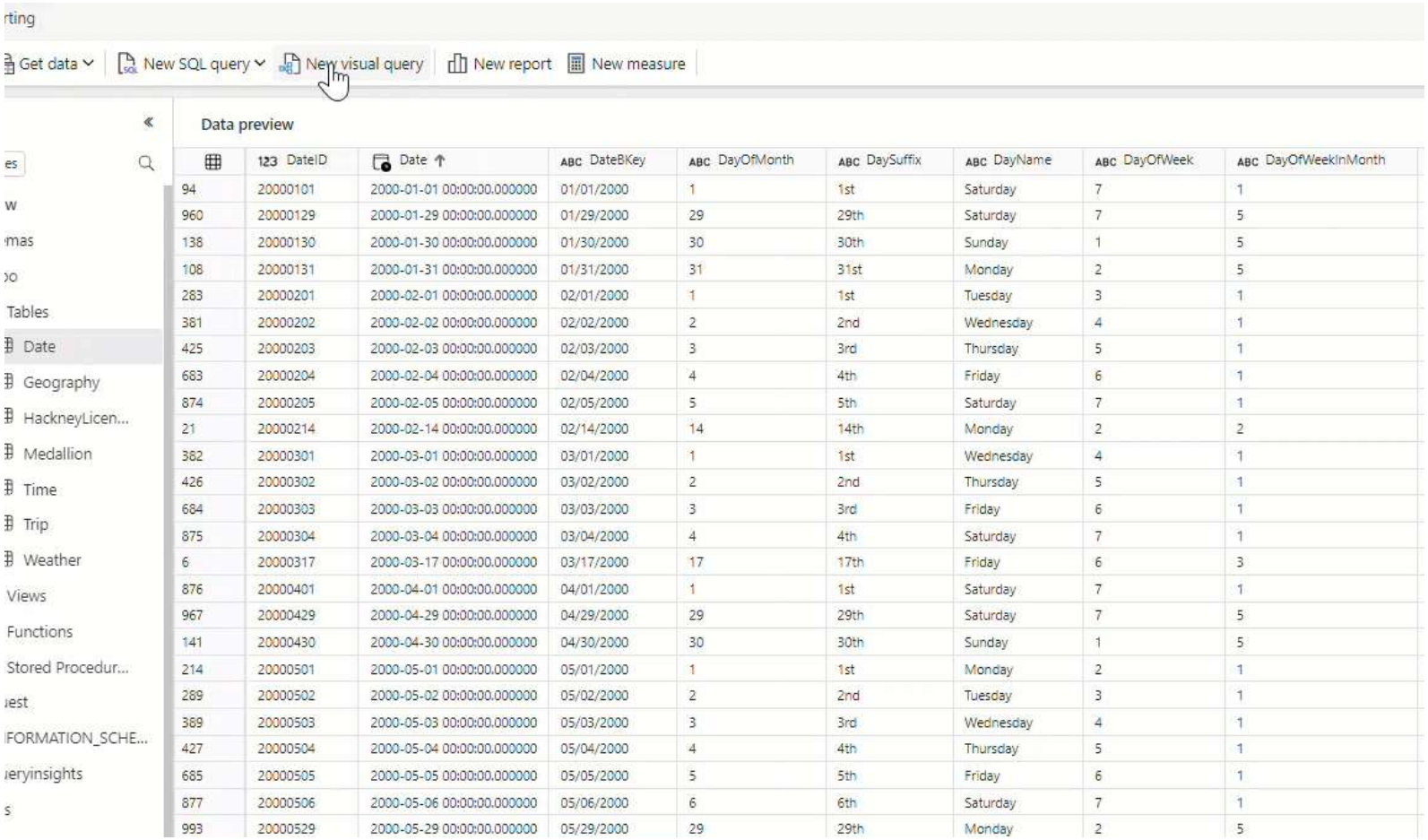
Run and save results as view or table

Download as Excel file

Explore visual query editor

Visually build queries

- Graphical interface
- Automatic query generation
- Save as table/view



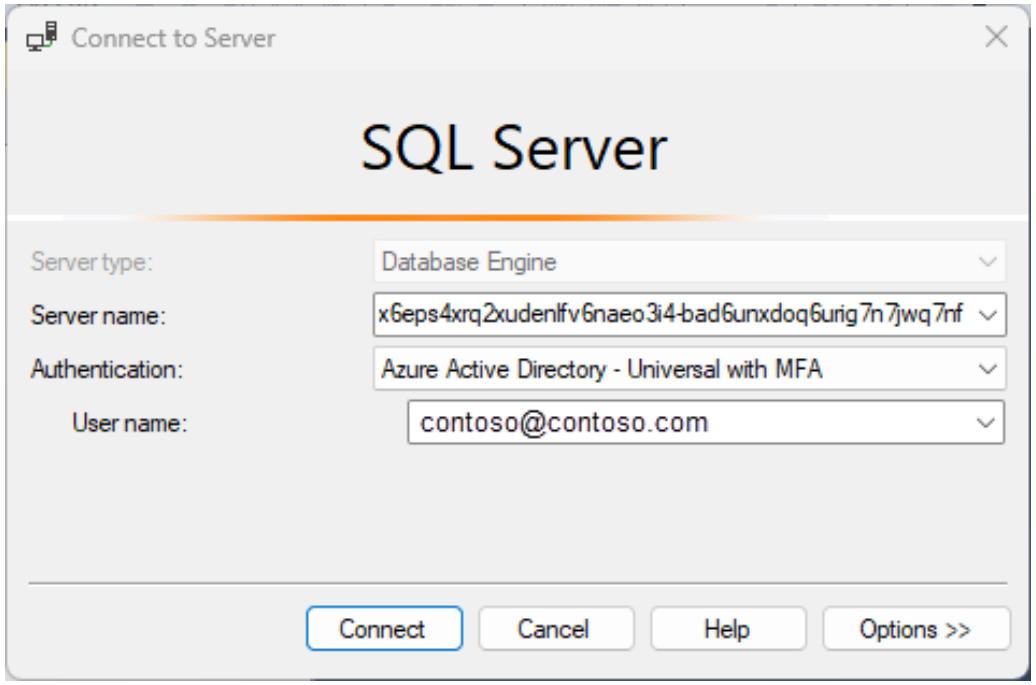
	DateID	Date	DateBKey	DayOfMonth	DaySuffix	DayName	DayOfWeek	DayOfWeekInMonth
94	20000101	2000-01-01 00:00:00.000000	01/01/2000	1	1st	Saturday	7	1
960	20000129	2000-01-29 00:00:00.000000	01/29/2000	29	29th	Saturday	7	5
138	20000130	2000-01-30 00:00:00.000000	01/30/2000	30	30th	Sunday	1	5
108	20000131	2000-01-31 00:00:00.000000	01/31/2000	31	31st	Monday	2	5
283	20000201	2000-02-01 00:00:00.000000	02/01/2000	1	1st	Tuesday	3	1
361	20000202	2000-02-02 00:00:00.000000	02/02/2000	2	2nd	Wednesday	4	1
425	20000203	2000-02-03 00:00:00.000000	02/03/2000	3	3rd	Thursday	5	1
683	20000204	2000-02-04 00:00:00.000000	02/04/2000	4	4th	Friday	6	1
874	20000205	2000-02-05 00:00:00.000000	02/05/2000	5	5th	Saturday	7	1
21	20000214	2000-02-14 00:00:00.000000	02/14/2000	14	14th	Monday	2	2
382	20000301	2000-03-01 00:00:00.000000	03/01/2000	1	1st	Wednesday	4	1
426	20000302	2000-03-02 00:00:00.000000	03/02/2000	2	2nd	Thursday	5	1
684	20000303	2000-03-03 00:00:00.000000	03/03/2000	3	3rd	Friday	6	1
875	20000304	2000-03-04 00:00:00.000000	03/04/2000	4	4th	Saturday	7	1
6	20000317	2000-03-17 00:00:00.000000	03/17/2000	17	17th	Friday	6	3
876	20000401	2000-04-01 00:00:00.000000	04/01/2000	1	1st	Saturday	7	1
967	20000429	2000-04-29 00:00:00.000000	04/29/2000	29	29th	Saturday	7	5
141	20000430	2000-04-30 00:00:00.000000	04/30/2000	30	30th	Sunday	1	5
214	20000501	2000-05-01 00:00:00.000000	05/01/2000	1	1st	Monday	2	1
289	20000502	2000-05-02 00:00:00.000000	05/02/2000	2	2nd	Tuesday	3	1
389	20000503	2000-05-03 00:00:00.000000	05/03/2000	3	3rd	Wednesday	4	1
427	20000504	2000-05-04 00:00:00.000000	05/04/2000	4	4th	Thursday	5	1
685	20000505	2000-05-05 00:00:00.000000	05/05/2000	5	5th	Friday	6	1
877	20000506	2000-05-06 00:00:00.000000	05/06/2000	6	6th	Saturday	7	1
993	20000529	2000-05-29 00:00:00.000000	05/29/2000	29	29th	Monday	2	5

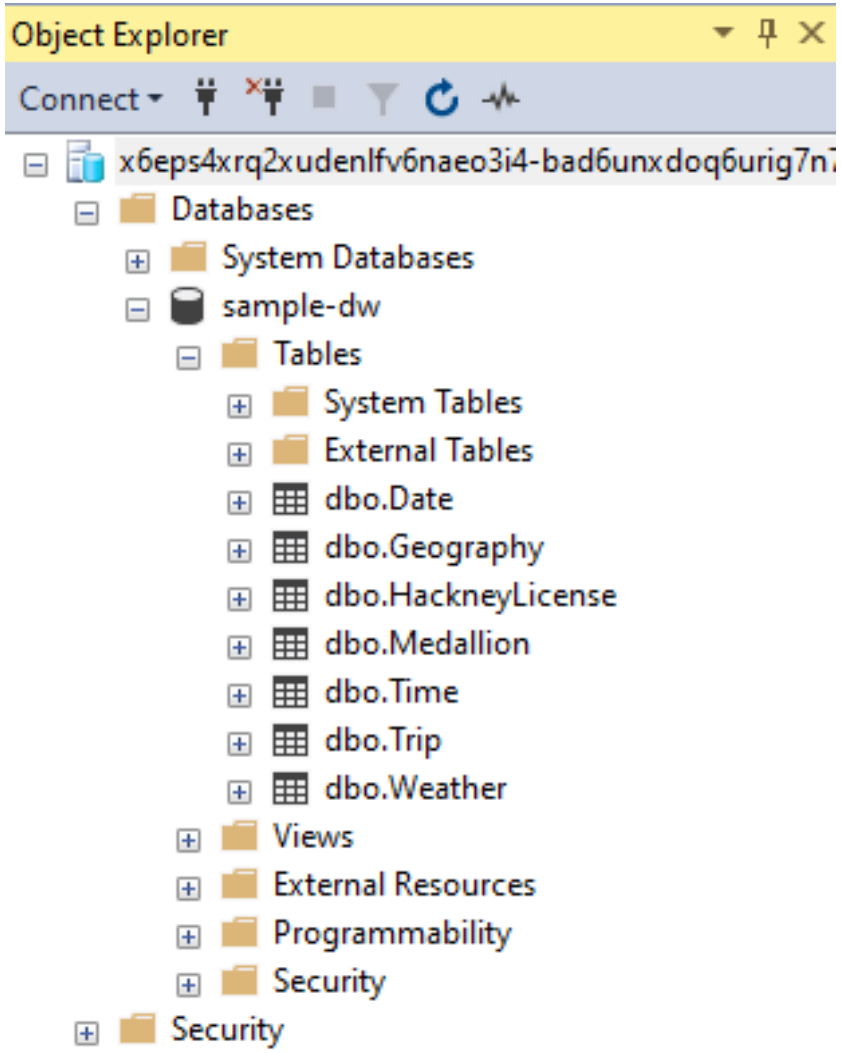
Connect with SQL clients

SQL connection string

Copy this string and use it to connect externally to the item from Power BI desktop or client tools.

x6eps4xrq2xudenlfv6naeo3i4-tonn3enr4vau3chrhyyvxxm7py.msit-datawarehouse.fabric.microsoft.cor

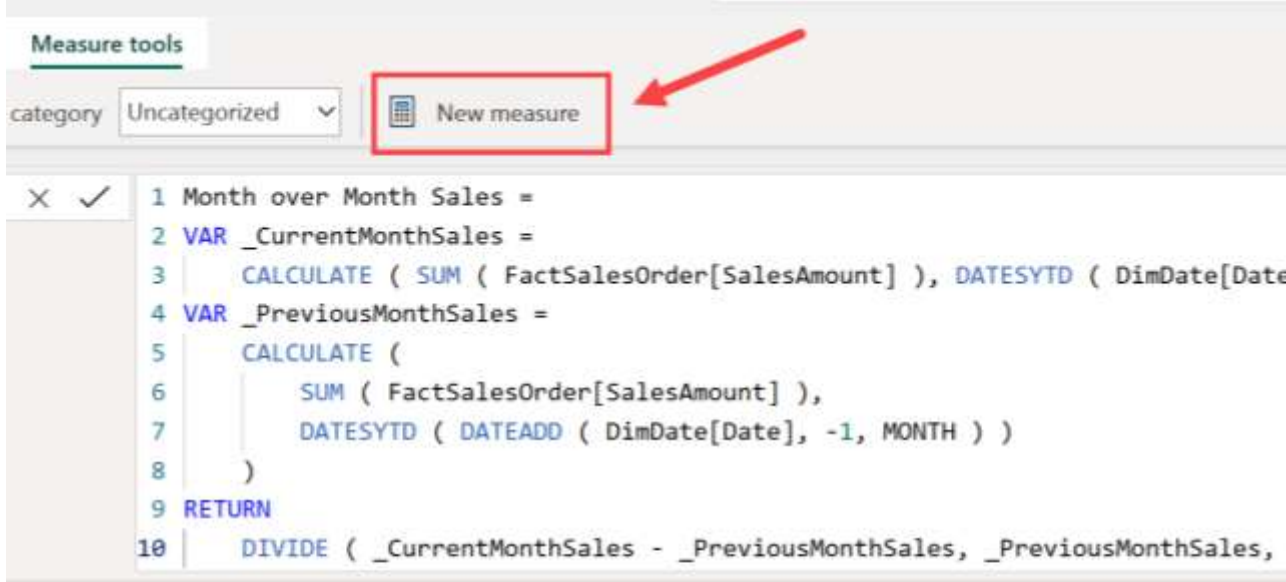


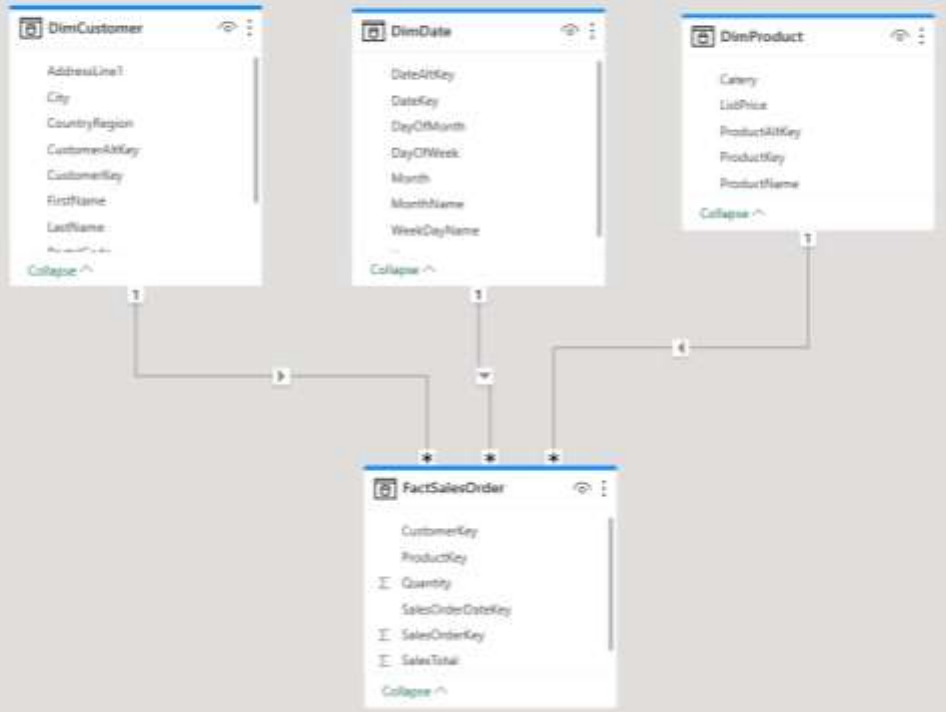


DAX calculations

Use DAX to create measures

- Create same calculations as with T-SQL
- Hundreds of functions
- Contextual calculations
- As complex as needed





Aggregate measures

Use T-SQL to create measures

Traditional SQL Data Warehouses relied on T-SQL to create measures.

Common functions are:

- JOIN
- GROUP BY

```
-- Sales amounts by year and quarter
SELECT  dates.CalendarYear,
        dates.CalendarQuarter,
        SUM(sales.SalesAmount) AS TotalSales
FROM dbo.FactSales AS sales
JOIN dbo.DimDate AS dates
   ON sales.OrderDateKey = dates.DateKey
GROUP BY dates.CalendarYear, dates.CalendarQuarter
ORDER BY dates.CalendarYear,
dates.CalendarQuarter;
```

Output – total sales by year and quarter

CalendarYear	CalendarQuarter	TotalSales
2020	1	25980.16
2020	2	27453.87
2020	3	28527.15
2020	4	31083.45
2021	1	34562.96
2021	2	36162.27

Extend measures

Building from last example

Further explores the **quarterly sales totals by city** based on the customer's address details.

```
-- Quarterly total sales by customer city
SELECT  dates.CalendarYear,
        dates.CalendarQuarter,
        custs.City,
        SUM(sales.SalesAmount) AS TotalSales
FROM dbo.FactSales AS sales
JOIN dbo.DimDate AS dates ON sales.OrderDateKey =
dates.DateKey
JOIN dbo.DimCustomer AS custs ON sales.CustomerKey
= custs.CustomerKey
GROUP BY dates.CalendarYear,
dates.CalendarQuarter, custs.City
ORDER BY dates.CalendarYear,
dates.CalendarQuarter, custs.City;
```

Join in a snowflake schema

Fact to extended dimension

This example joins **FactSales** and the **DimProduct** and **DimCategory** dimension tables, which requires two joins to complete.

```
-- items sold by product category
SELECT  cat.ProductCategory,
        SUM(sales.OrderQuantity) AS ItemsSold
FROM dbo.FactSales AS sales
JOIN dbo.DimProduct AS prod ON sales.ProductKey = prod.ProductKey
JOIN dbo.DimCategory AS cat ON prod.CategoryKey = cat.CategoryKey
GROUP BY cat.ProductCategory
ORDER BY cat.ProductCategory;
```

Output – items sold by product category

ProductCategory	ItemsSold
Accessories	28271
Bits and pieces	5368
...	...

Ranking functions

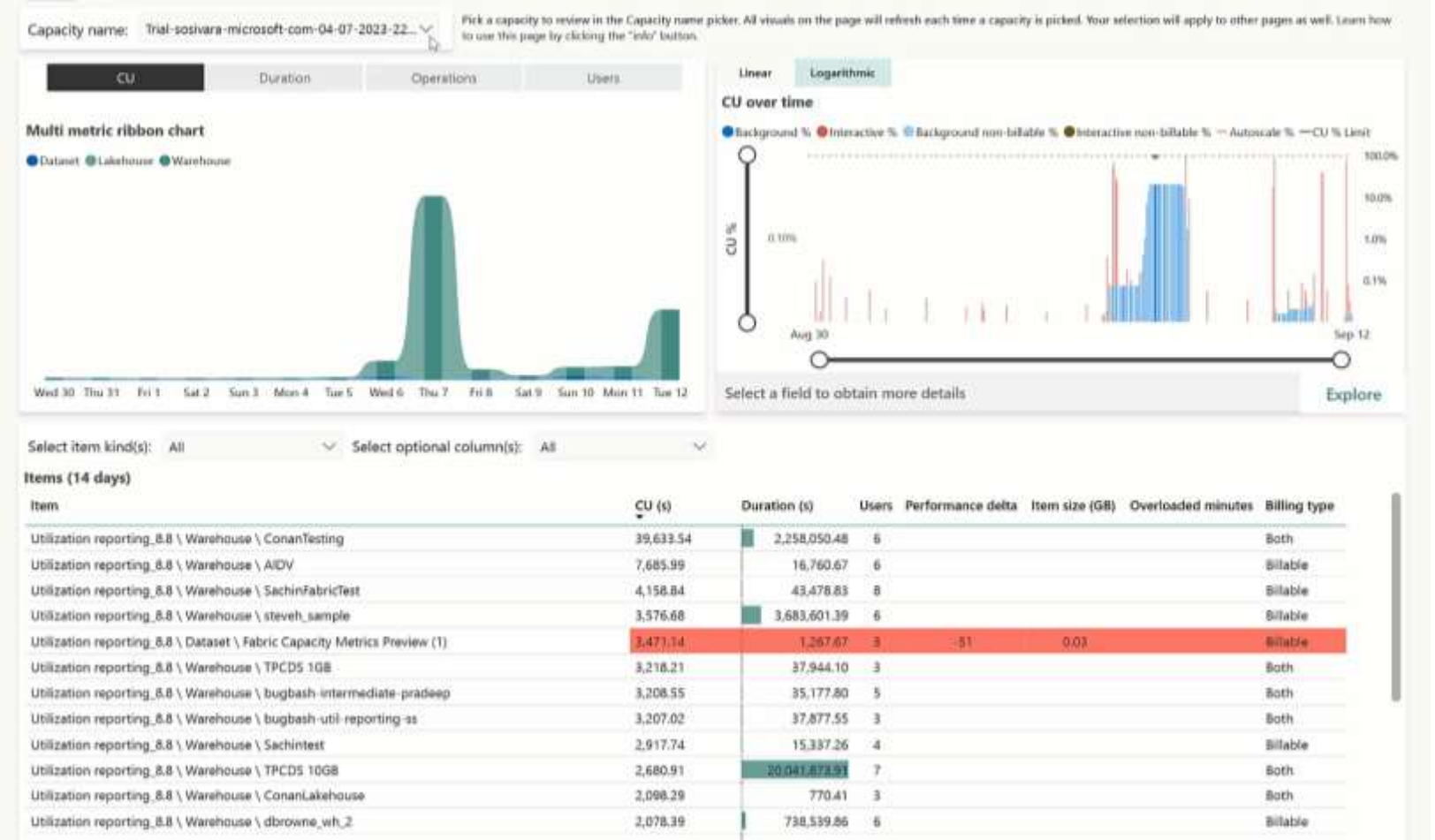
- ROW_NUMBER
- RANK
- DENSE_RANK
- NTILE

```
-- Group by category and relative rank
SELECT  ProductCategory,
        ProductName,
        ListPrice,
        ROW_NUMBER() OVER
            (PARTITION BY ProductCategory ORDER BY
ListPrice DESC) AS RowNumber,
        RANK() OVER
            (PARTITION BY ProductCategory ORDER BY
ListPrice DESC) AS Rank,
        DENSE_RANK() OVER
            (PARTITION BY ProductCategory ORDER BY
ListPrice DESC) AS DenseRank,
        NTILE(4) OVER
            (PARTITION BY ProductCategory ORDER BY
ListPrice DESC) AS Quartile
FROM dbo.DimProduct
ORDER BY ProductCategory;
```

Monitor capacity metrics

Fabric Capacity Metrics app

- Monitor and manage workload costs
- Requires admin install
- Filter by activity type
- Tracks Capacity Units (CUs)
  - Primarily read/writes



Monitor current activity

Dynamic Management Views (DMVs) with T-SQL:

- **sys.dm_exec_connections:** Connections between warehouse and engine
- **sys.dm_exec_sessions:** Sessions between item and engine
- **sys.dm_exec_requests:** Active requests in a session

```
--- Identify long-running queries
SELECT request_id, session_id, start_time, total_elapsed_time
    FROM sys.dm_exec_requests
    WHERE status = 'running'
    ORDER BY total_elapsed_time DESC;


--- Find session_id (user)
SELECT login_name
    FROM sys.dm_exec_sessions
    WHERE 'session_id' = 'SESSION_ID WITH LONG-RUNNING QUERY’;


--- Terminate session (ADMIN only)
KILL 'SESSION_ID WITH LONG-RUNNING QUERY';
```

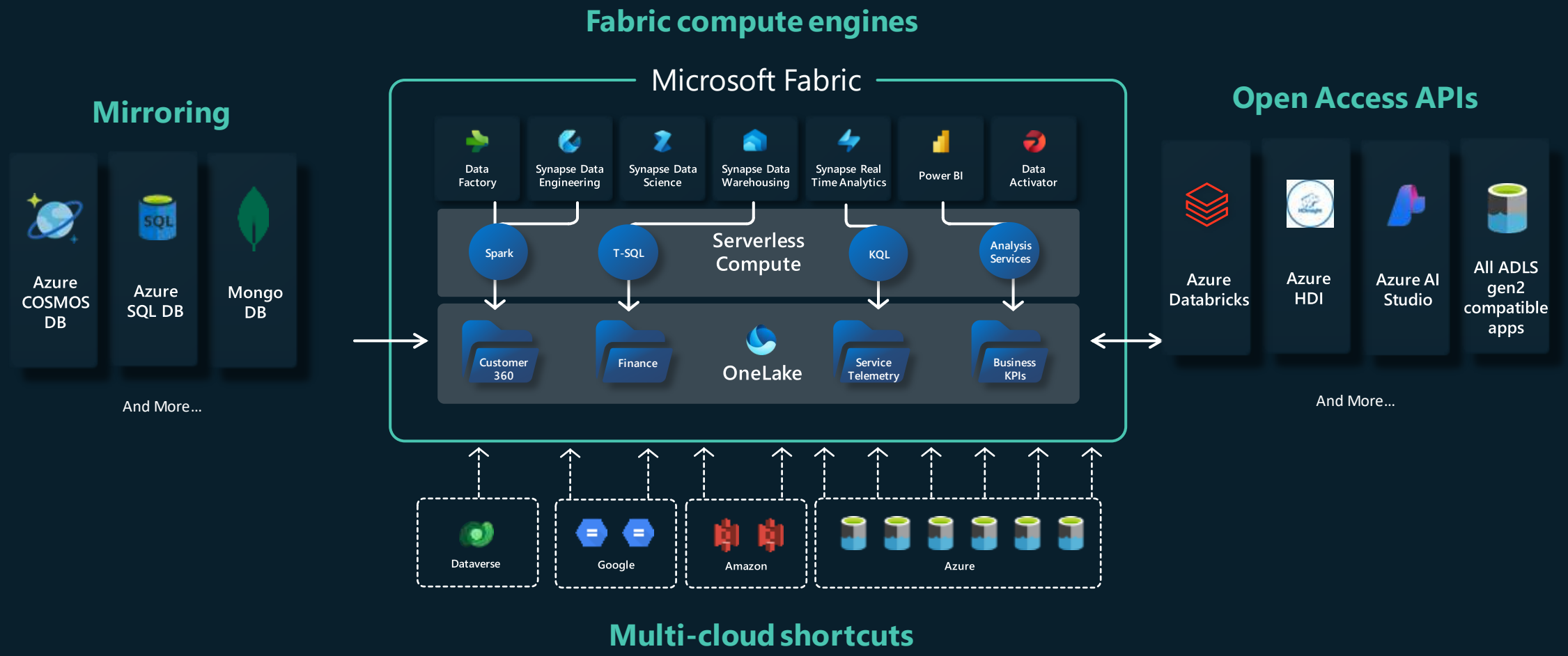
Query insights views

- **queryinsights.exec_requests_history**: Completed SQL queries
- **queryinsights.long_running_queries**: Query execution times
- **queryinsights.frequently_run_queries**: Frequently run queries

```
--- queries from previous hour
SELECT start_time, login_name, command
FROM queryinsights.exec_requests_history
WHERE start_time >= DATEADD(MINUTE, -60,
GETUTCDATE())


--- long-running queries, multiple runs
SELECT last_run_command, number_of_runs,
median_total_elapsed_time_ms,
last_run_start_time
FROM queryinsights.long_running_queries
WHERE number_of_runs > 1
ORDER BY median_total_elapsed_time_ms DESC;


-- Frequently run queries
SELECT last_run_command, number_of_runs,
number_of_successful_runs,
number_of_failed_runs
FROM queryinsights.frequently_run_queries
```

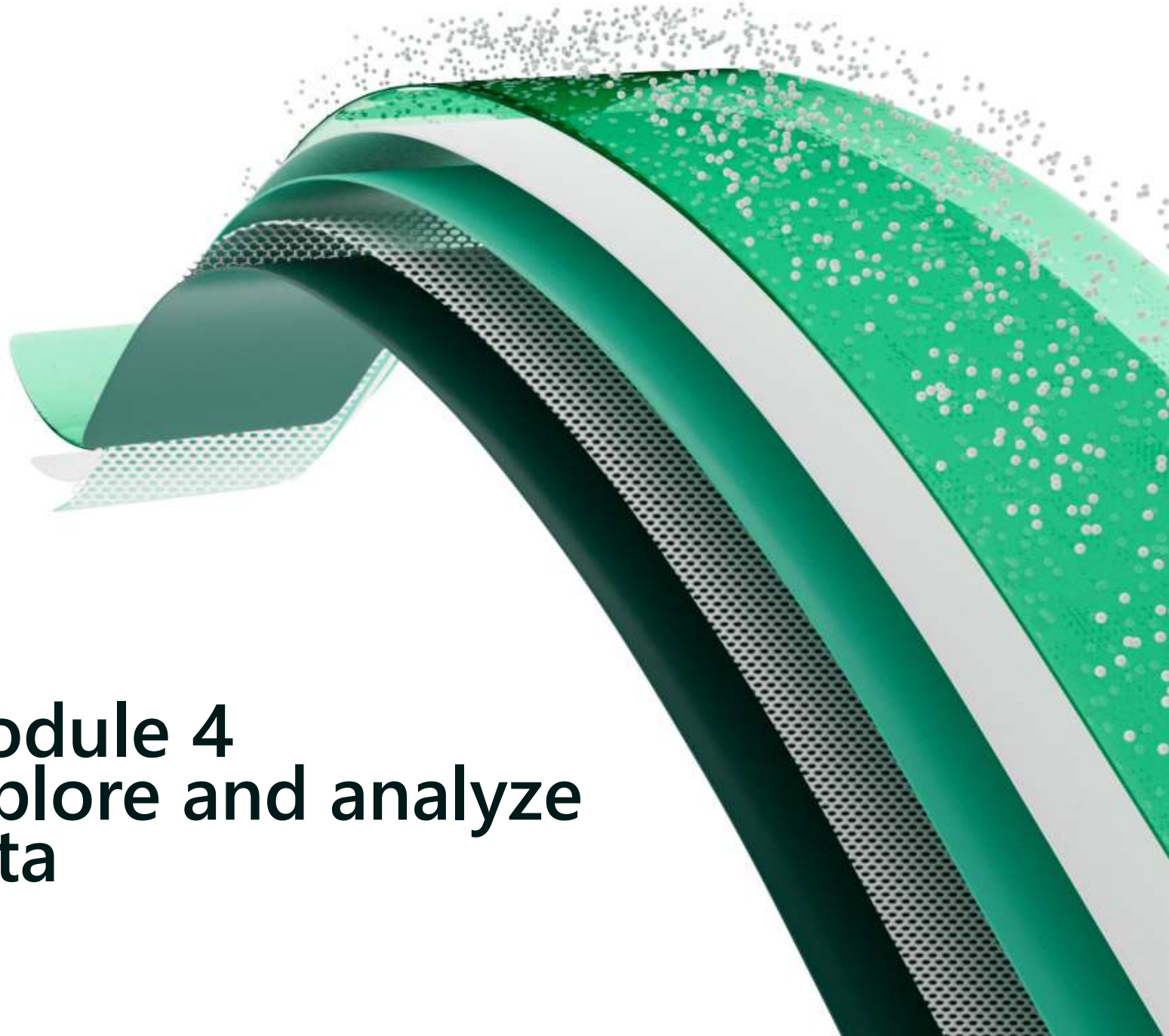
All roads lead to OneLake

Creating Data Gravity in OneLake



4

Module 4
Explore and analyze data



Explore and analyze data (20–25%)

Perform exploratory analytics

Implement descriptive and diagnostic analytics

Integrate prescriptive and predictive analytics into a visual or report

Profile data

Query data by using SQL

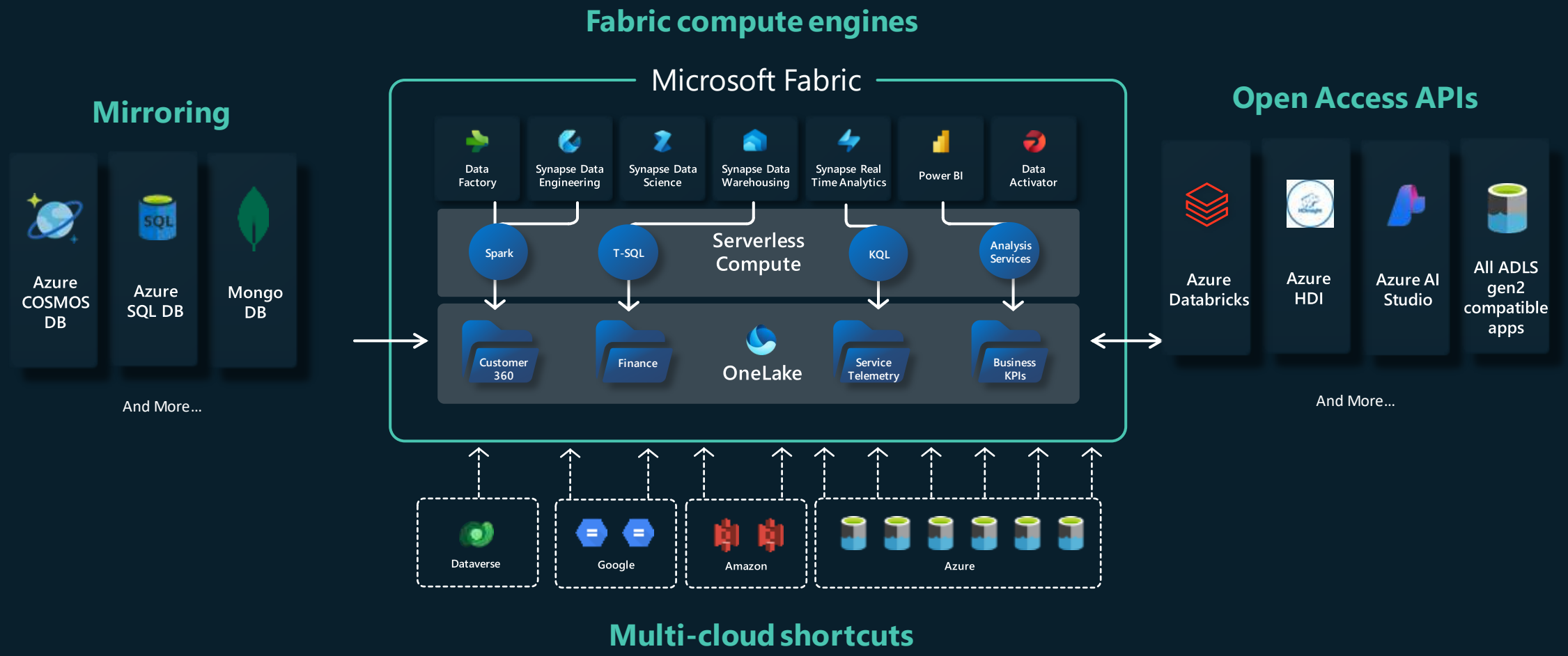
Query a lakehouse in Fabric by using SQL queries or the visual query editor

Query a warehouse in Fabric by using SQL queries or the visual query editor

Connect to and query datasets by using the XMLA endpoint

All roads lead to OneLake

Creating Data Gravity in OneLake



Warehouse and Data ingestion overview

Data Ingestion in Fabric Warehouse

From source data to insights in minutes, regardless of your level of expertise

Choose your desired experience:

- No code, or low code with intuitive UX
- Code rich using T-SQL

Data is instantly optimized for queries with minimal or no knobs required

Data distribution managed automatically

Advanced storage optimizations

Data formats and data type support

All data is stored in Parquet format

Exact numeric	Approximate numeric	Date and Time	Character strings	Binary strings	Other
bit bigint int smallint decimal(p,s) numeric(p,s)	float(n) real	date datetime2(n)* time(n)*	char(n) varchar(n)	varbinary(n)	uniqueidentifier

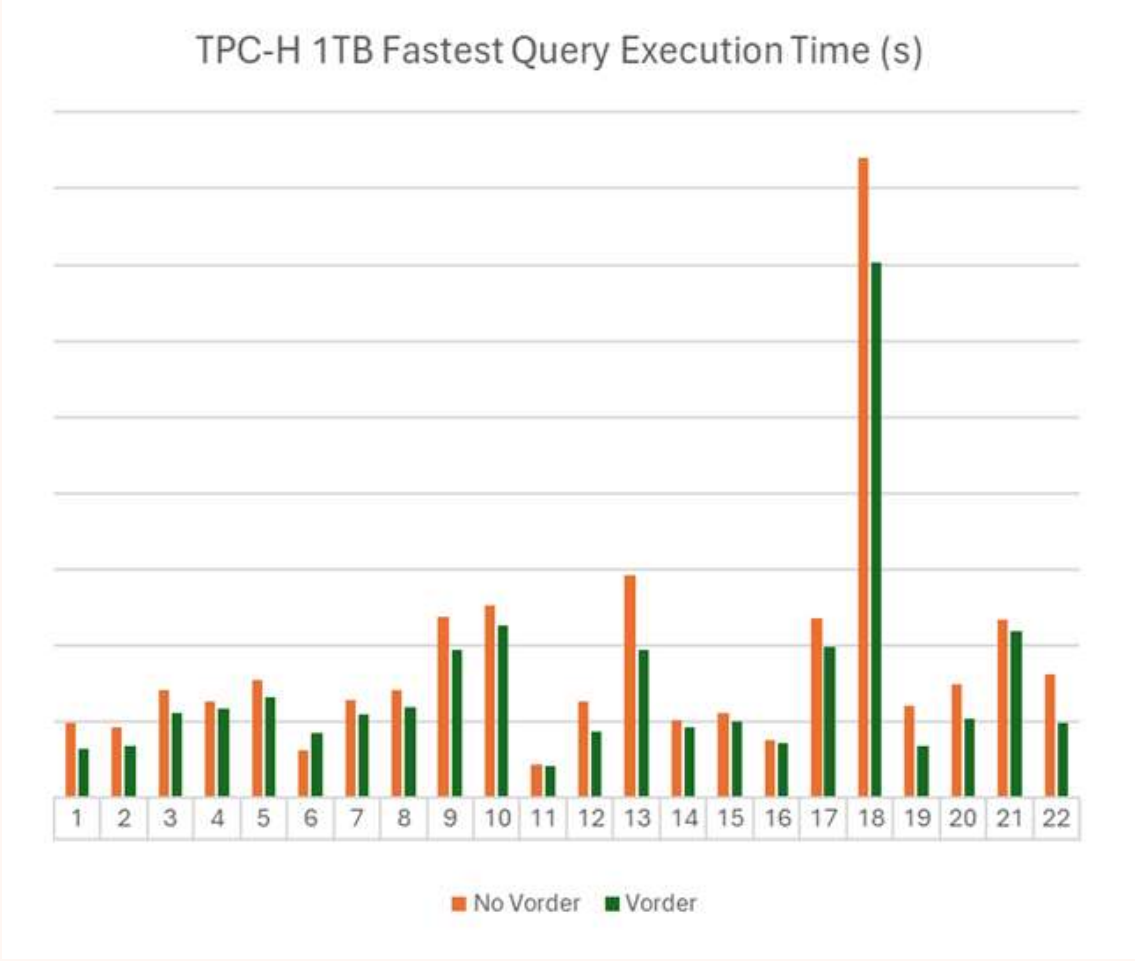
- For **datetime2** and **time**, precision is limited to 6 digits for fractions of seconds (SQL Server supports up to 7 digits).
- char/varchar can be used in favor of nchar/nvarchar without data loss, but char/varchar uses more storage to store some Unicode characters than nchar/nvarchar.
- LOB types in general (including **char/varchar(max)**) not yet available but coming **very** soon.

V-Order write optimization

V-Order is a write time optimization to the parquet file format that enables lightning-fast reads under the Microsoft Fabric compute engines, such as Power BI, SQL, Spark and others. It applies special sorting and compression to Parquet

Warehouse queries benefit from faster read times with v-order, still ensuring files are 100% compliant to Parquet's open-source specification

All warehouse data is written with v-order optimization at ingestion time



Data distribution

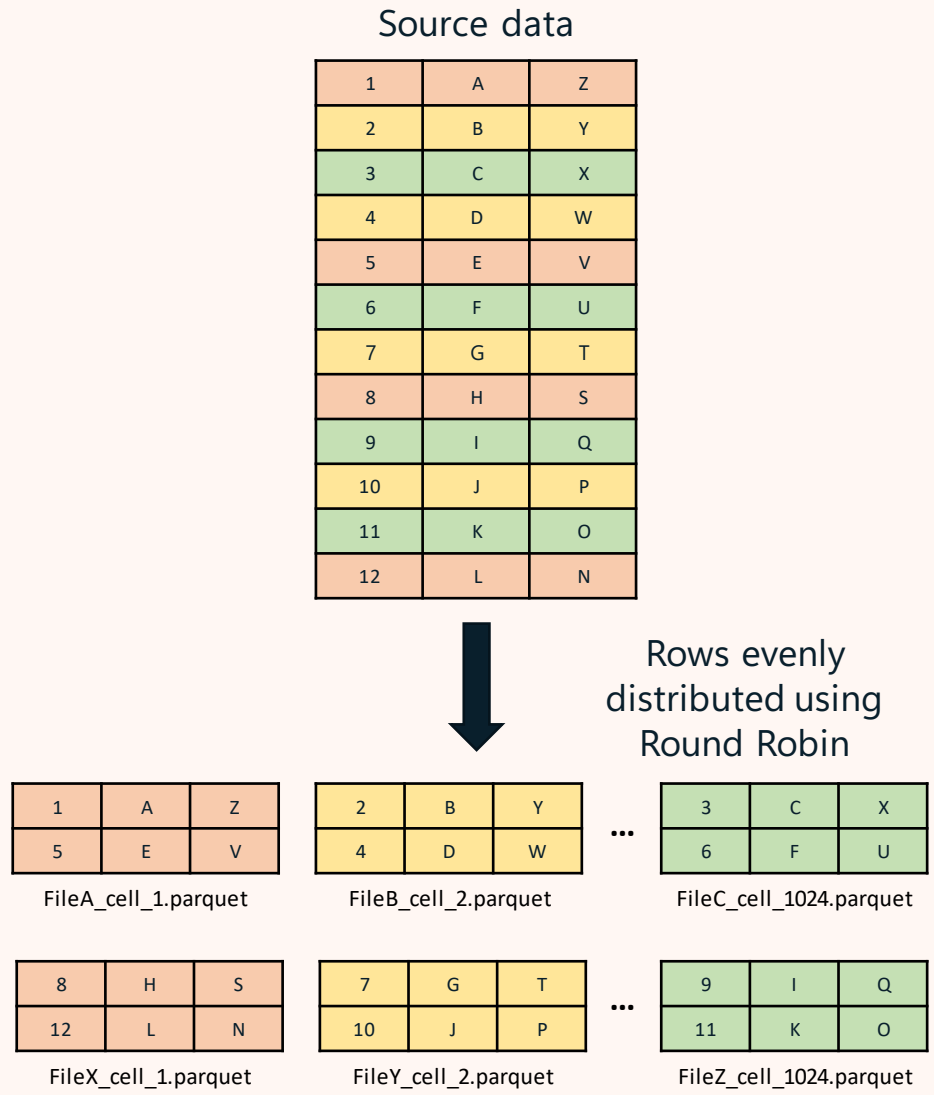
Data is stored across distributions for scalability

Fabric Warehouse supports Round Robin distribution

- No need to specify distribution column(s)
- No need to concern with data skew
- Source data is split into smaller data cells
- Each new parquet file is assigned with a cell ID between 0 and 1023. As a result, tables have a fixed number of cells: 1024
- As tables grow, we allocate cell IDs to new parquet files ensuring an evenly-distributed number of parquet files per cell ID

This allows:

- Uniform work distribution during query processing
- Scale-out ingestion operations with multiple nodes processing source data



Behind the scenes: how data distribution works

Source data

Source file 1		
1	A	Z
2	B	Y
3	C	X
...	...	...

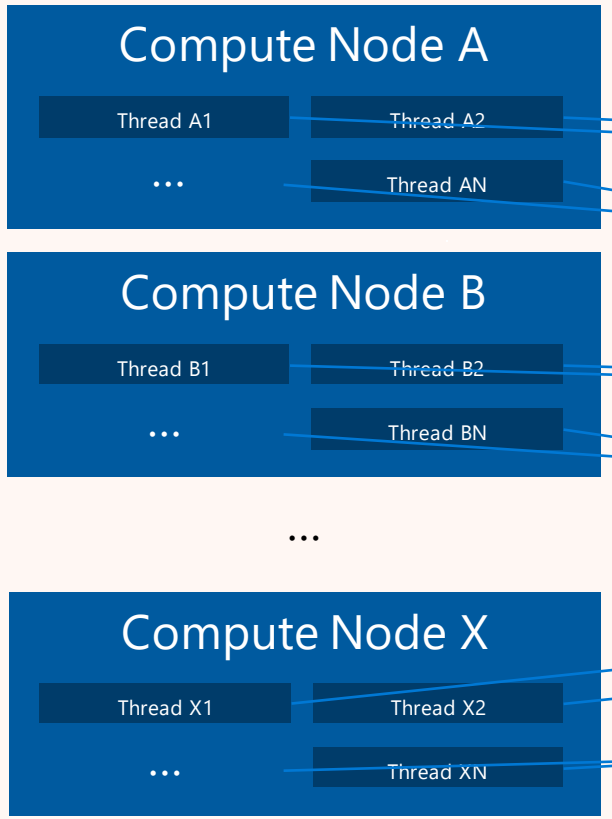
Source file 2		
2637	D	W
2638	E	V
2639	F	U
...	...	...

...

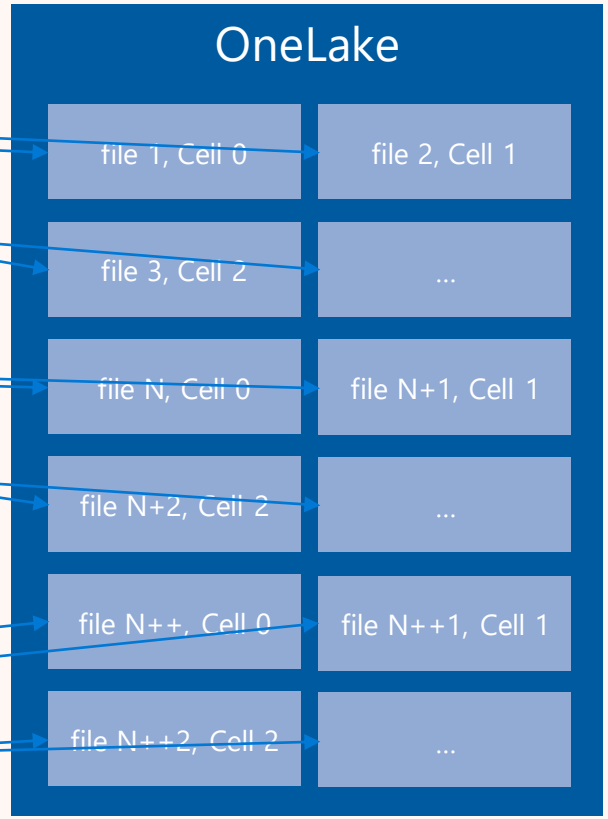
Source file N		
7890	G	T
7891	H	S
7892	I	R
...	...	...

UQO determines the number of compute nodes needed to process source data

Each compute node processes a subset of the source data



Each thread writes parquet files in OneLake in parallel. Each file is tagged with a cell ID

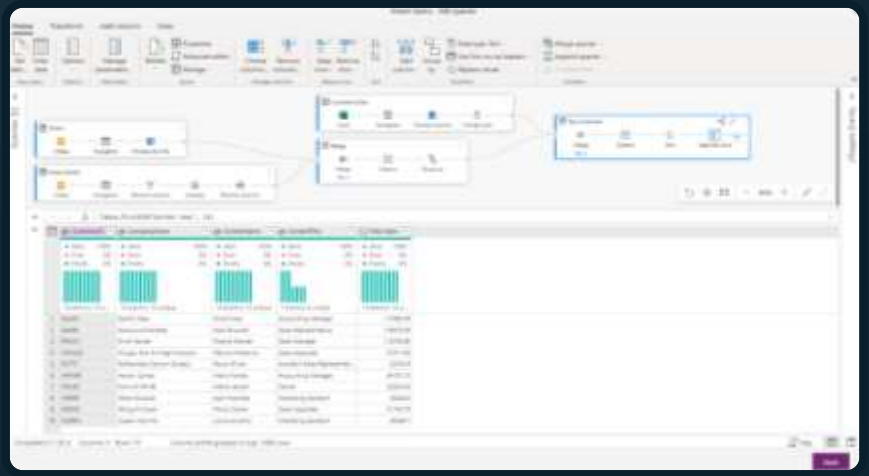


No code data prep with Dataflows gen 2

The worlds data

Through 170+ connectors

Familiar Power Query experiences



Through 400+ no code transformations

Warehouse

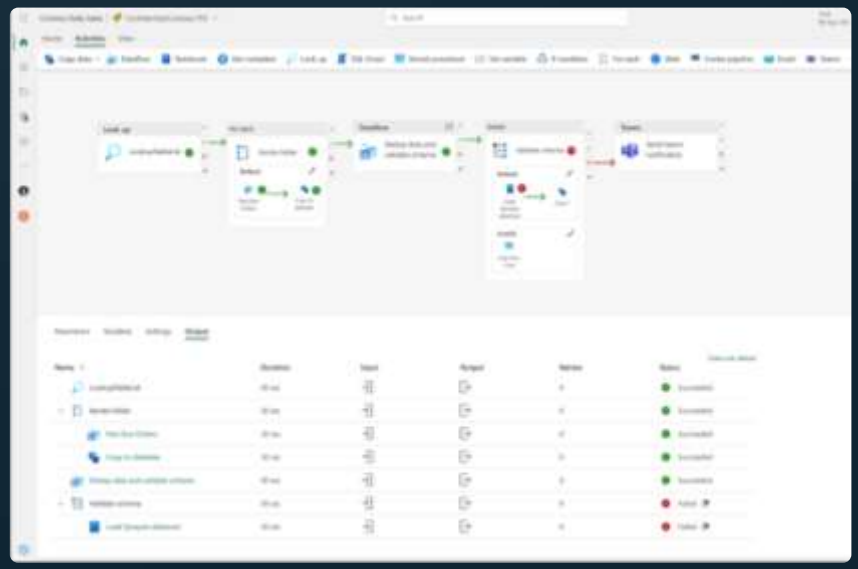
Ingest and transform data at scale

Data orchestration and data prep with Pipelines

The worlds data

Through 170+ connectors

ETL, CDC, or ELT data prep needs



Rich library of activities and pro code data prep experiences

Warehouse

Ingest and transform data at scale

...and of course, you can use SQL too!

SQL COPY INTO, transactions, and more

Support for:	COPY INTO
	CTAS
	INSERT … SELECT
	Common Table Expressions
	Stored Procedures
Transactions and locking:	Supports snapshot isolation
	INSERTS and SELECTS only require SCHEMA stability locks
	UPDATES and DELETES do not block tables

Simple, Familiar SQL Syntax

```
COPY INTO table_name.[(Column_list)]
FROM '<external_location>' [,...n]
WITH
 (
 [FILE_TYPE = {'CSV' | 'PARQUET' '} ]
 [,CREDENTIAL = (AZURE CREDENTIAL) ]
 [,COMPRESSION = { 'Gzip' | 'Snappy'}]
 [,FIELDQUOTE = 'string_delimiter']
 [,FIELDTERMINATOR = 'field_terminator']
 [,ROWTERMINATOR = 'row_terminator']
 [,FIRSTROW = first_row]
 [,ENCODING = {'UTF8'|'UTF16'}]
 );
```

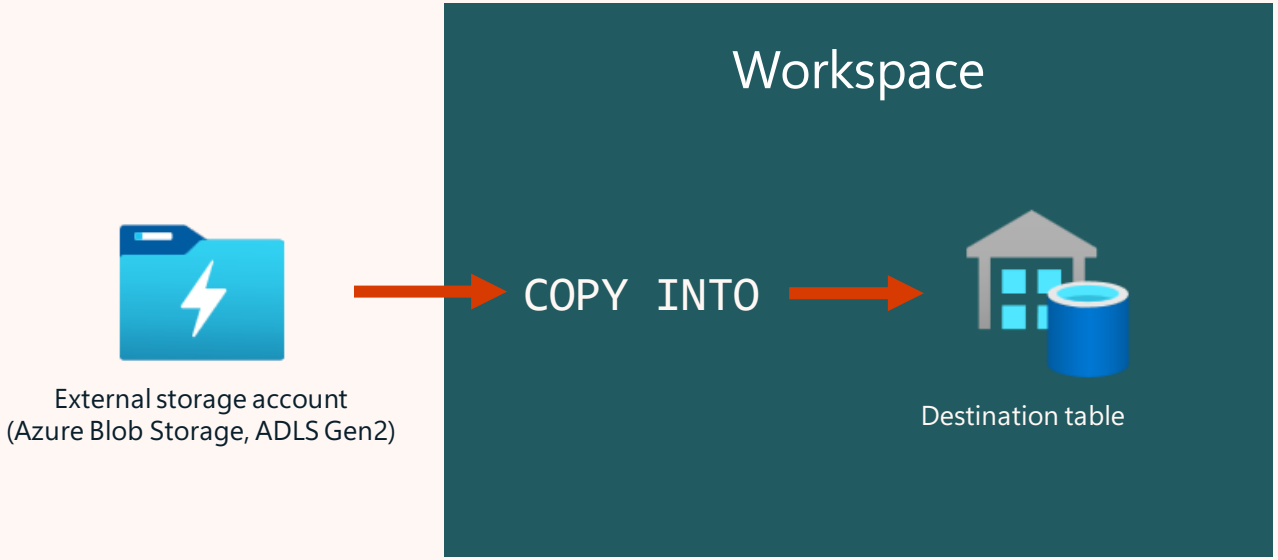
```
CREATE TABLE [dbo].[FactInternetSales_new]
AS
SELECT * FROM [dbo].[FactInternetSales];
```

```
INSERT INTO dbo.EmployeeSales
SELECT 'SELECT', sp.BusinessEntityID, c.LastName, sp.SalesYTD
FROM warehouseA.Sales.SalesPerson AS sp
INNER JOIN warehouseB.Person.Person AS c
    ON sp.BusinessEntityID = c.BusinessEntityID
WHERE sp.BusinessEntityID LIKE '2%'
ORDER BY sp.BusinessEntityID, c.LastName;
```

```
WITH sales_data AS (
     SELECT customer_id, SUM(total_sales) AS total_sales
FROM sales
GROUP BY customer_id
)
SELECT customer_id, total_sales
FROM sales_data
WHERE total_sales > 1000;
```

COPY INTO

- Enables flexible, high-throughput data ingestion from external Azure storage accounts
- Supports PARQUET and CSV formats
- No additional database objects required



Syntax:

```
COPY INTO table_name.[(Column_list)]
FROM '<external_location>' [,...n]
WITH
 (
 [FILE_TYPE = {'CSV' | 'PARQUET' '} ]
 [,CREDENTIAL = (AZURE CREDENTIAL) ]
 [,ERRORFILE = '[http(s)://account/container]/directory[/]]']
 [,ERRORFILE_CREDENTIAL = (AZURE CREDENTIAL) ]
 [,MAXERRORS = max_errors ]
 [,COMPRESSION = { 'Gzip' | 'Snappy'}]
 [,FIELDQUOTE = 'string_delimiter']
 [,FIELDTERMINATOR = 'field_terminator']
 [,ROWTERMINATOR = 'row_terminator']
 [,FIRSTROW = first_row]
 [,ENCODING = {'UTF8'|'UTF16'}]
 [,PARSER_VERSION = {'1.0'|'2.0'}]
 )
```

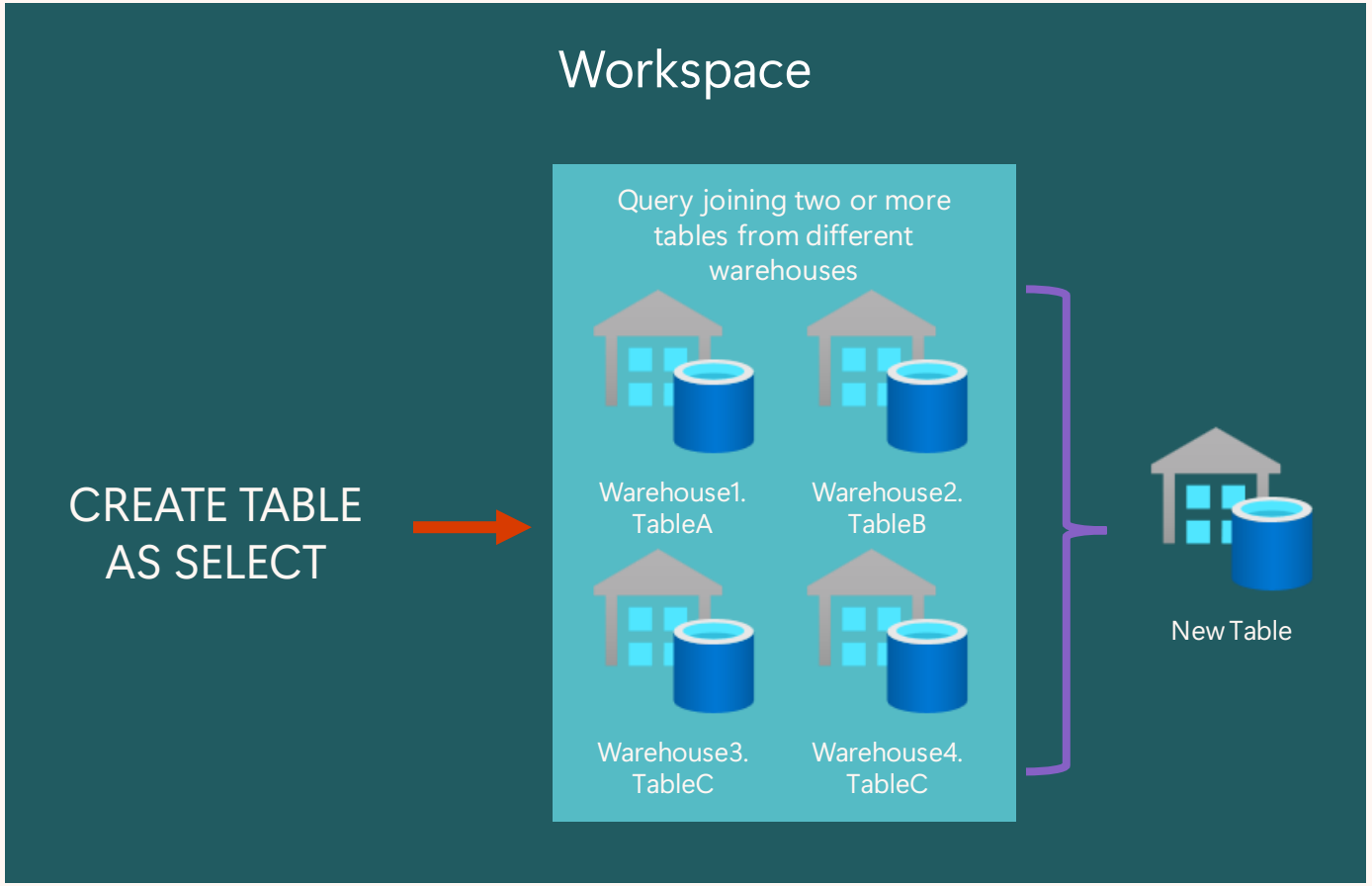
CREATE TABLE AS SELECT (CTAS)

A parallel operation that creates a new table based on the output of a SELECT statement. Cross-workspace queries are supported.

The simplest and fastest way to create and insert data into a table with a single command.

Example:

```
CREATE TABLE
[dbo].[FactInternetSales_new]
AS
SELECT * FROM [dbo].[FactInternetSales];
```



INSERT…SELECT

For existing tables, INSERT…SELECT allows you to insert data from other tables using complex, cross-warehouse queries

Example:

```
INSERT INTO dbo.EmployeeSales
    SELECT 'SELECT', sp.BusinessEntityID, c.LastName, sp.SalesYTD
    FROM warehouseA.Sales.SalesPerson AS sp
    INNER JOIN warehouseB.Person.Person AS c
        ON sp.BusinessEntityID = c.BusinessEntityID
    WHERE sp.BusinessEntityID LIKE '2%'
    ORDER BY sp.BusinessEntityID, c.LastName;
```

Decision guide

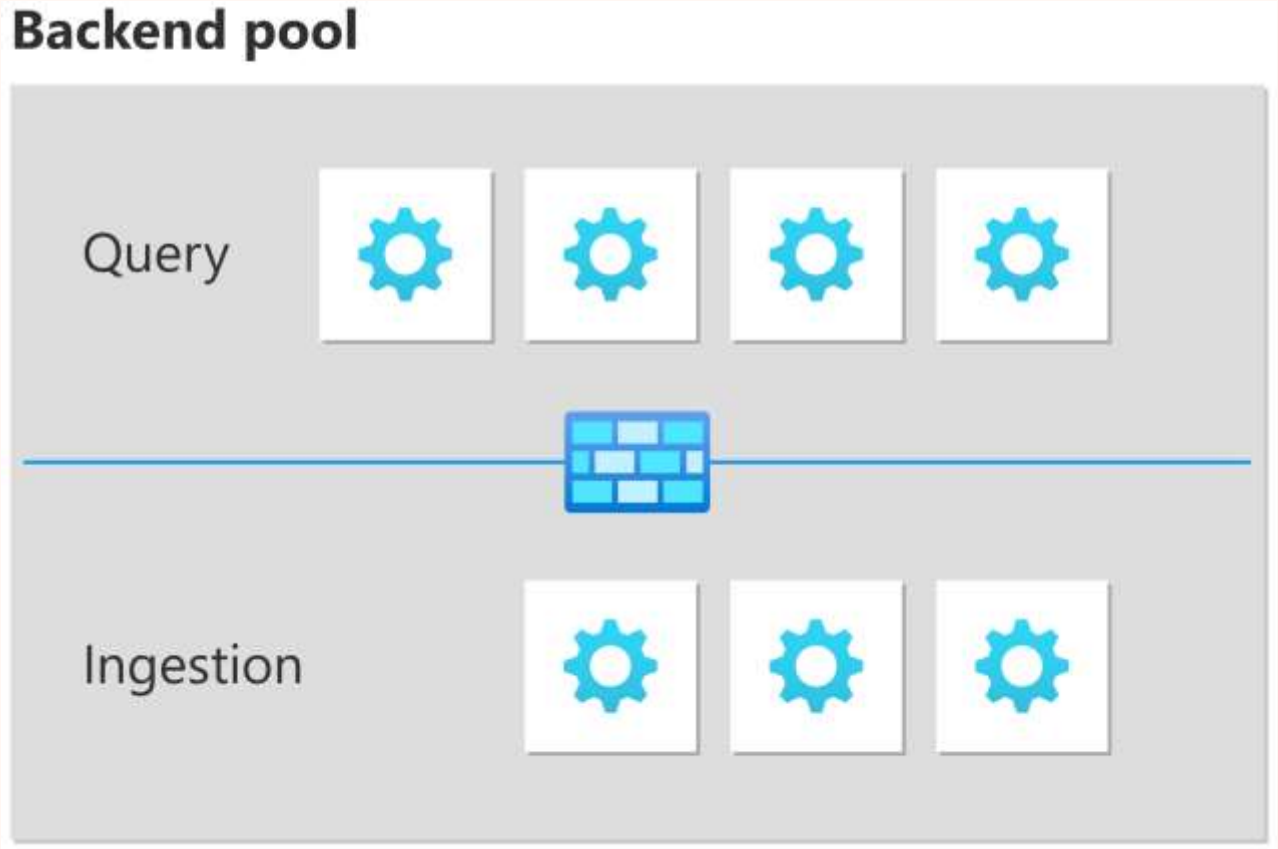
	Pipelines	Dataflow Gen 2	T-SQL
Use case	Data lake and data warehouse migration, Data ingestion Light-weight transformation	Data transformation, data wrangling, data profiling Data ingestion	Data transformation, data processing, data profiling Data ingestion
Primary developer persona	Data engineer, data integrator	Data engineer, data integrator, business analyst	Data engineer, data developer
Primary developer skillset	ETL, SQL, JSON	ETL, M, SQL	T-SQL
Code written	No code Low code	No code Low code	Code
Data volume	Low to High	Low to High	Any
Development interface	Wizard Canvas	Power Query	Fabric Editor
Sources	100+ connectors	150+ connectors	ADLS, warehouse tables, lakehouse tables
Destinations	30+ connectors	Lakehouse Azure SQL database Azure Data explorer Azure Synapse analytics	Warehouse
Transformation complexity	Low: Lightweight - type conversion, column mapping, merge/split files, flatten hierarchy	Low to high: 300+ transformation functions	Low to high: Anything supported with T-SQL

Performance considerations

Ingestion isolation

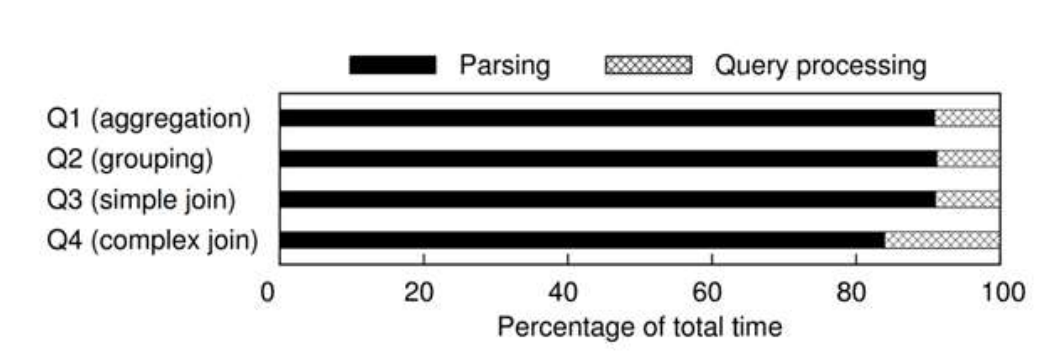
In the backend compute pool of Warehouse in Microsoft Fabric, loading activities are provided resource isolation from analytical workloads.

This improves performance and reliability, as ingestion jobs can run on dedicated nodes that are optimized for ETL and do not compete with other queries or applications for resources.



Leverage PARSER_VERSION 2.0 for CSV by avoiding fallback scenarios

Parsing non-binary data is a time-consuming operation



Source: Microsoft Research, "Mison: A Fast JSON Parser for Data Analytics"

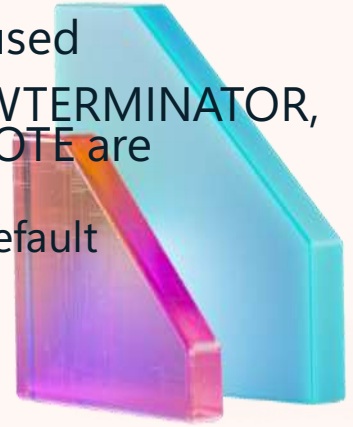
COPY INTO with PARSER_VERSION 2.0 improves ingestion performance for CSV files.

Example:

```
COPY INTO destination_table
FROM '[external account]'
WITH (
    FILE_TYPE = 'CSV',
        …
    PARSER_VERSION = '2.0' --this is the default
)
```

COPY INTO fallbacks to Parser 1.0 when:

- Compressed CSV files are used
- Files with UTF-16 encoding are used
- Multicharacter or multibyte ROWTERMINATOR, FIELDTERMINATOR, or FIELDQUOTE are provided.
  - However, '\r\n' is accepted as a default ROWTERMINATOR



Best practices

COPY INTO:

- Consider splitting large CSV or Parquet files, especially when the number of files is small
  - Using 1,000 files is 7x faster than using a single, large file with the same data volume
  - Keep file sizes to **at least** 4MB
- Parallel loads achieve higher throughput

All ingestion:

- Data ingestion scales with larger Fabric Capacities
  - If more throughput is needed, consider using a larger capacity and parallel loads
- Avoid singleton INSERT INTO statements (a.k.a.: trickle INSERTS) as they each work as a single ingestion operation and produce small Parquet files

Write T-SQL queries

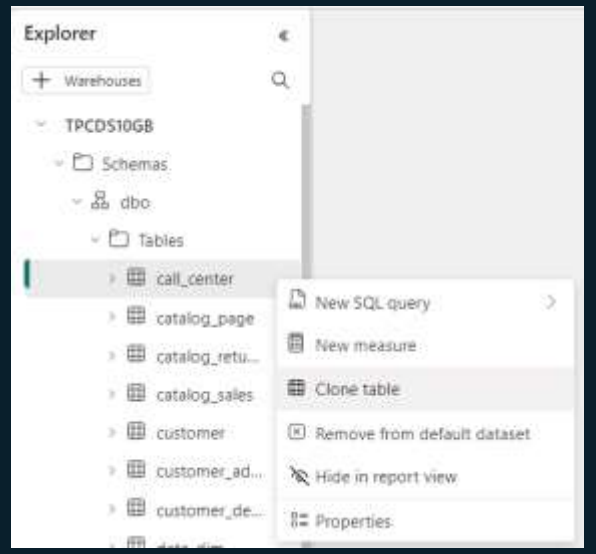
SQL Query Editor and Visual Query Editor

Object Explorer

S: 1 min

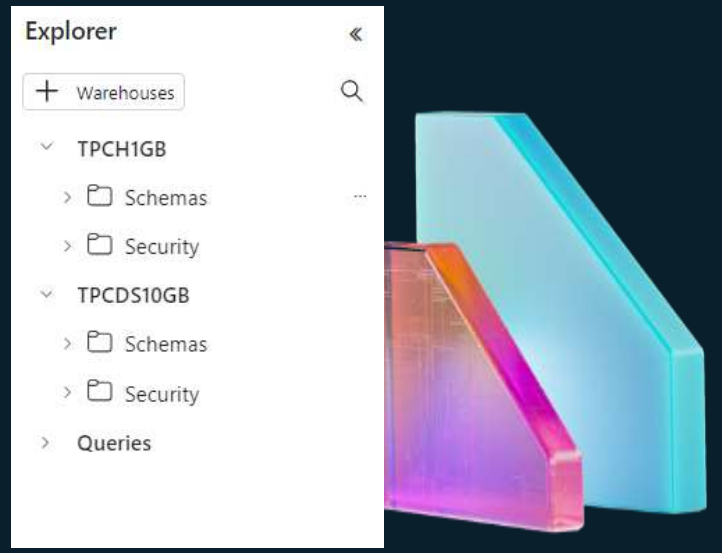
Templates based on existing objects

- Quickly create or drop existing objects from the Object Explorer using templates
- Preview Top 100 rows from tables and views with a click of a button
- Easy to clone tables in a warehouse using context menu option in Object explorer.



Add multiple warehouses

- Easy to add additional warehouses within current workspace to perform cross database querying.
- Can write a T-SQL query with three-part naming convention to refer to objects and join them across warehouses.

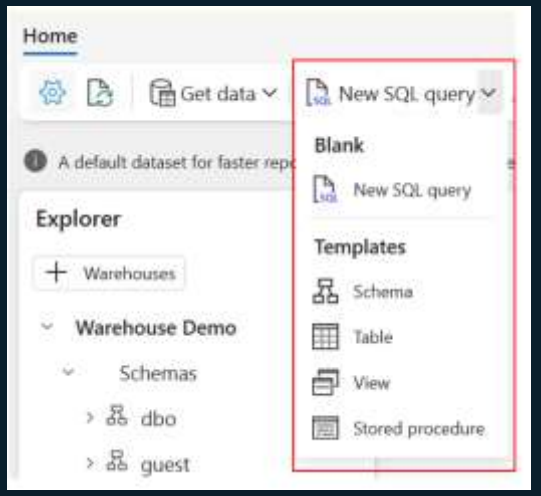


S: 1 min

Writing a SQL query

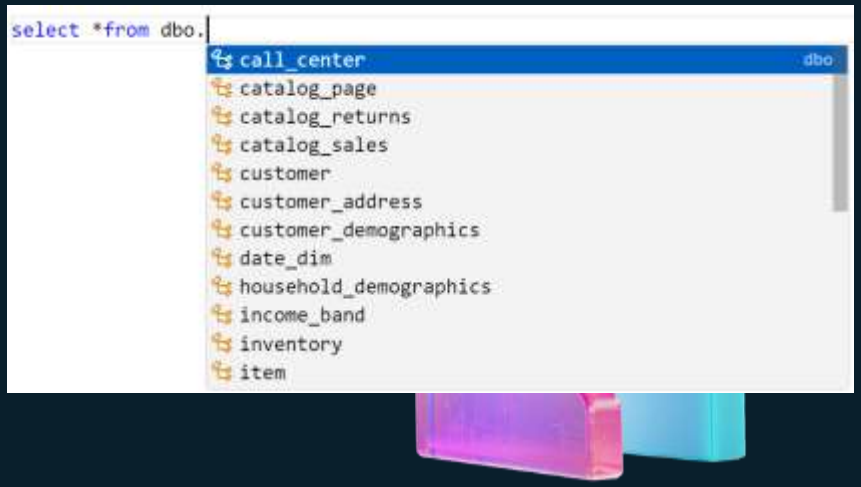
Templates to create new objects

- Easy to create a new query using New SQL query -> Templates.
- Create T-SQL objects with code templates that populate in SQL query window.



IntelliSense and Keyword highlights

- SQL query editor provides support for IntelliSense, code completion, syntax highlighting



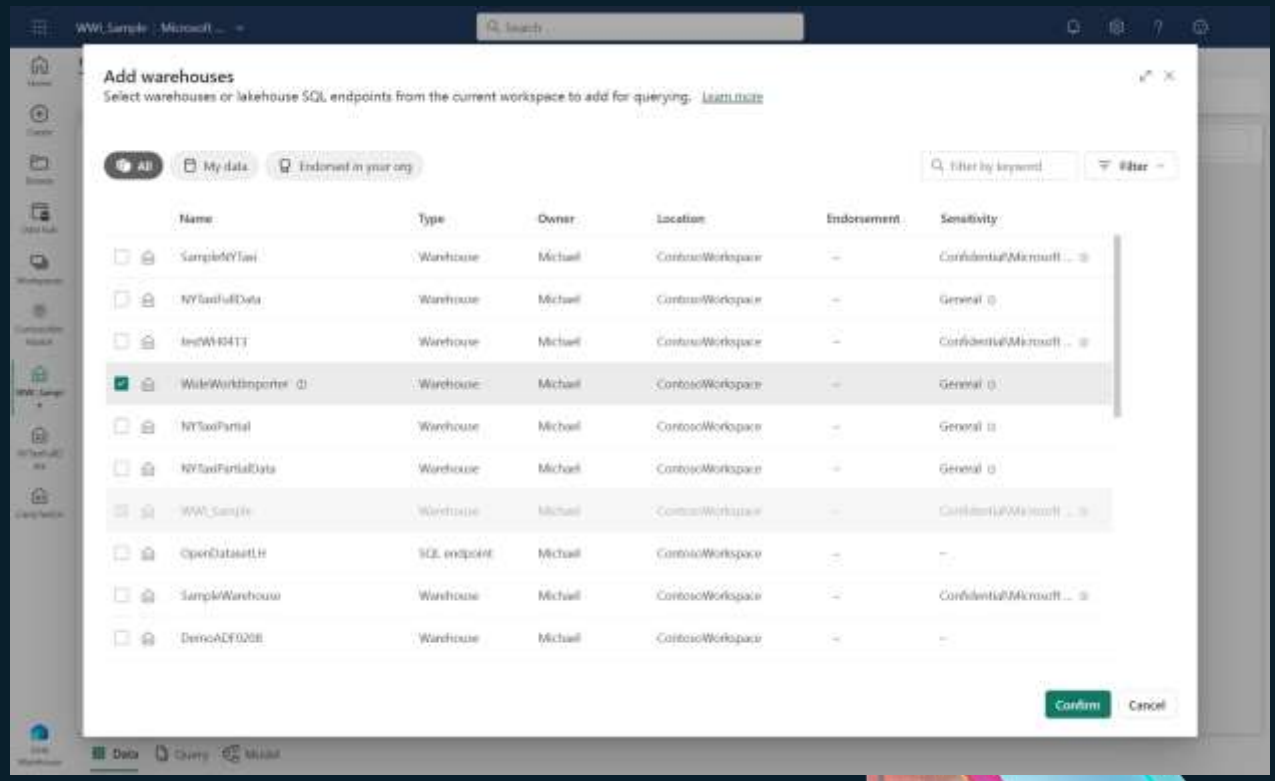
Keyboard Shortcuts

- Keyboard shortcuts provide a quick way to navigate and allow users to work more efficiently in SQL query editor. (Give a link). Different from SSMS.

Cross database querying

S: 1 min

- Easy to add additional warehouses within current workspace to perform cross database querying
- Create cross-warehouse queries via SQL query editor using 3-part naming syntax: [Warehouse].[Schema].[Table]
- Cross-warehouse queries can be used across SQL analytics endpoint and Warehouses in the current workspace

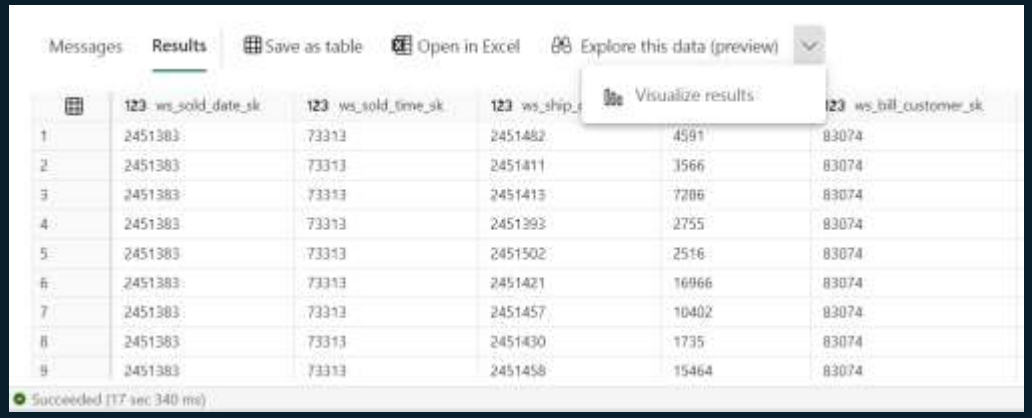


J: 1 min

Viewing and analyzing your query results

Data grid

- View your SQL query results in an interactive data grid right within the query editor



Open in Excel

- Analyze and refresh data from your SQL query directly inside Excel to ensure you are working with up-to-date data

Explore your data (preview)

- Perform ad-hoc exploration of your query results before building a report

Visualize Results

- Produce visual Power BI reports on top of your query results

Save as...

J: 1 min

Save as View

- Easy to create a view from selected SQL query text
- For the selected query, choose the desired schema and provide a name for the view

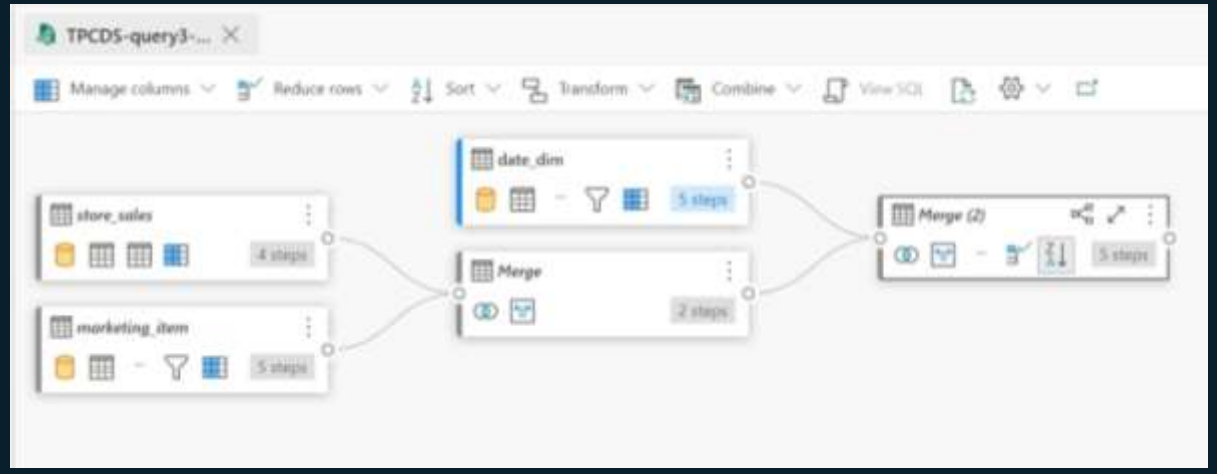
Save as table

- Quickly save results of your SQL query into the table (CTAS) with just a simple click
- For the selected query, specify the warehouse, schema and table name to save results into the table

J: 1 min

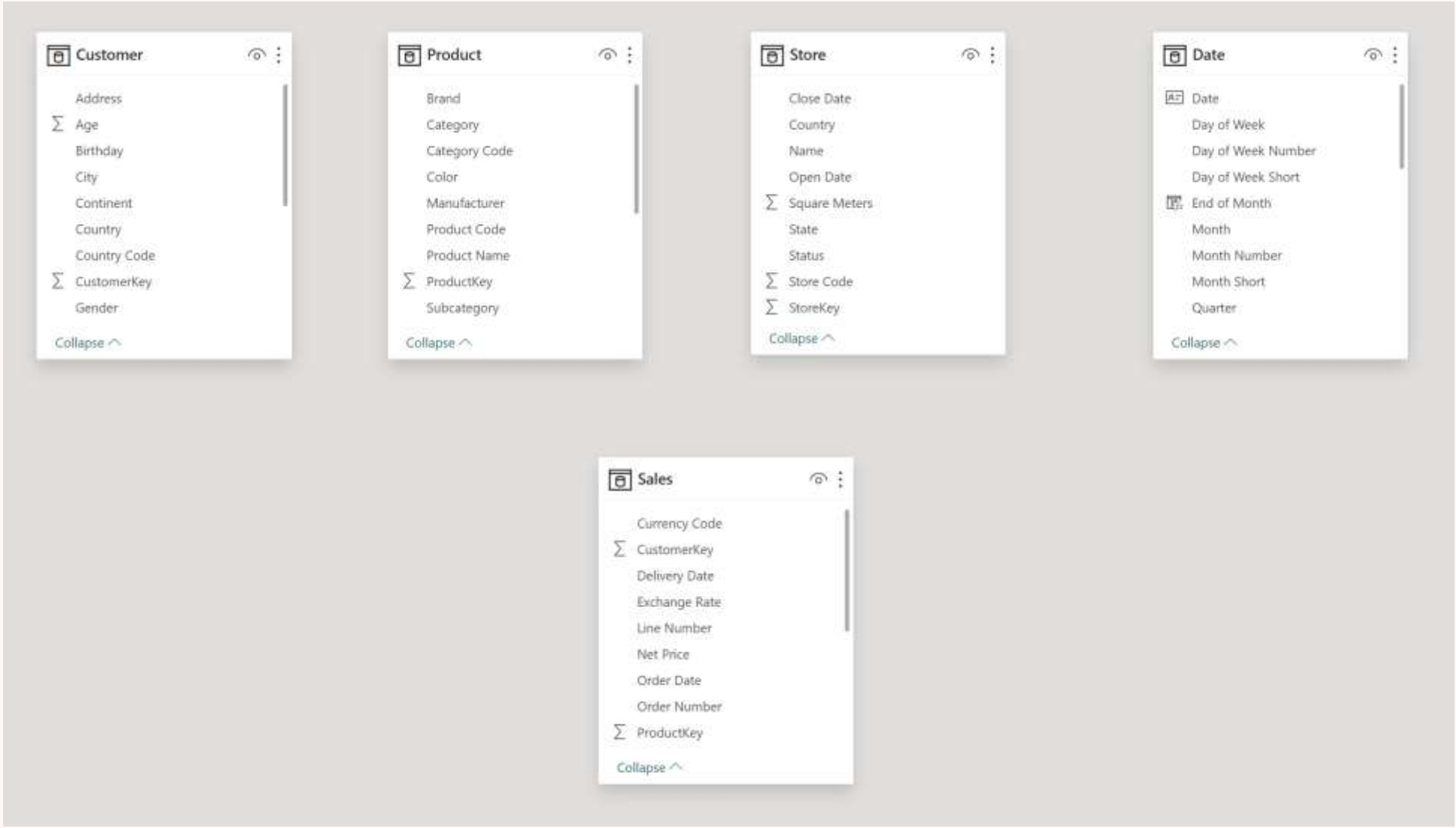
Visual query editor

- No-code experience for citizen developers to analyze the data (leverages the familiar Power Query Diagram view)
- Simply drag-drop tables into the Visual query editor and apply steps to get results for your analysis
- Create cross-warehouse queries through drag/drop of tables from respective Warehouse/ SQL analytics endpoint and apply merge condition

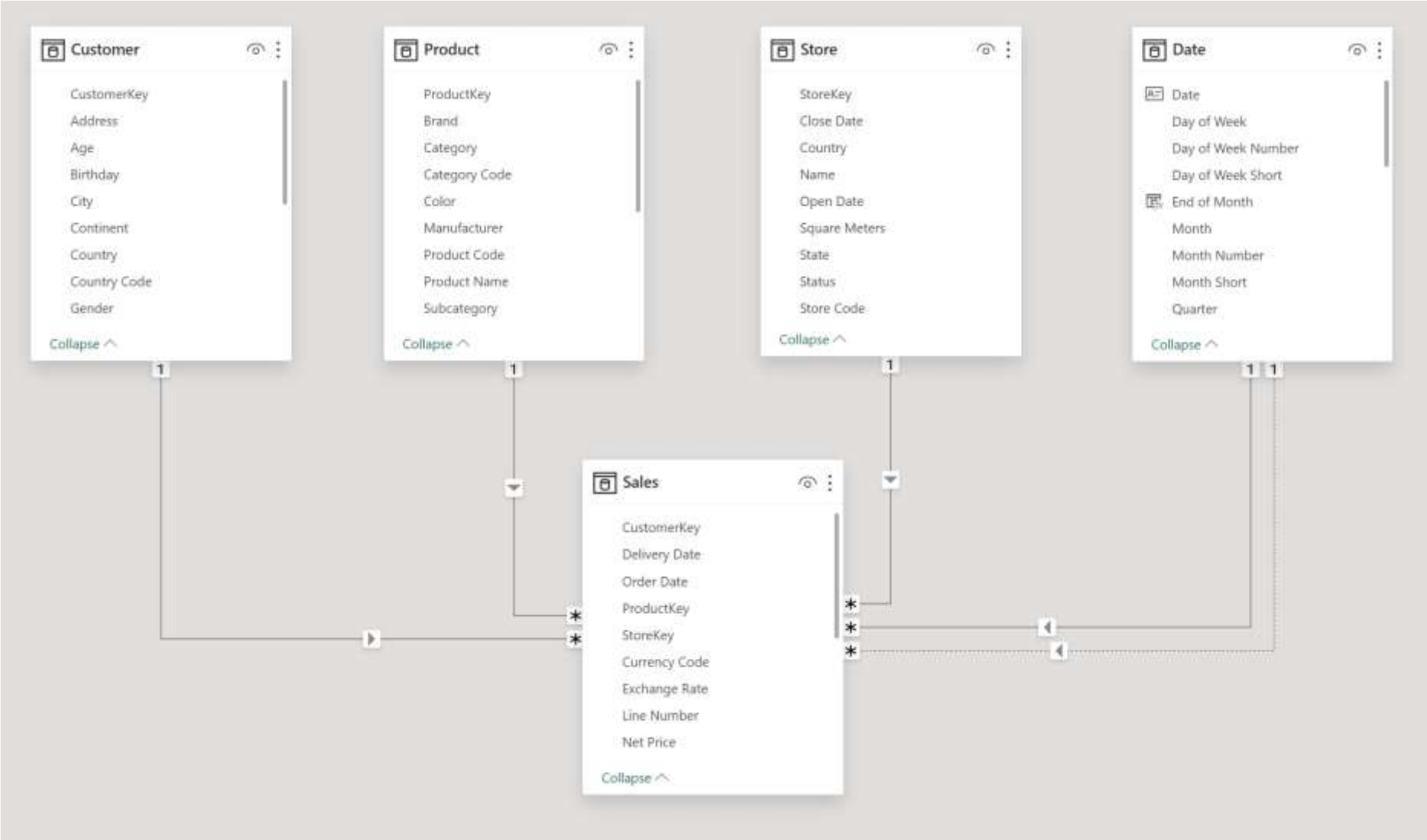


Semantic model, Not just a dataset

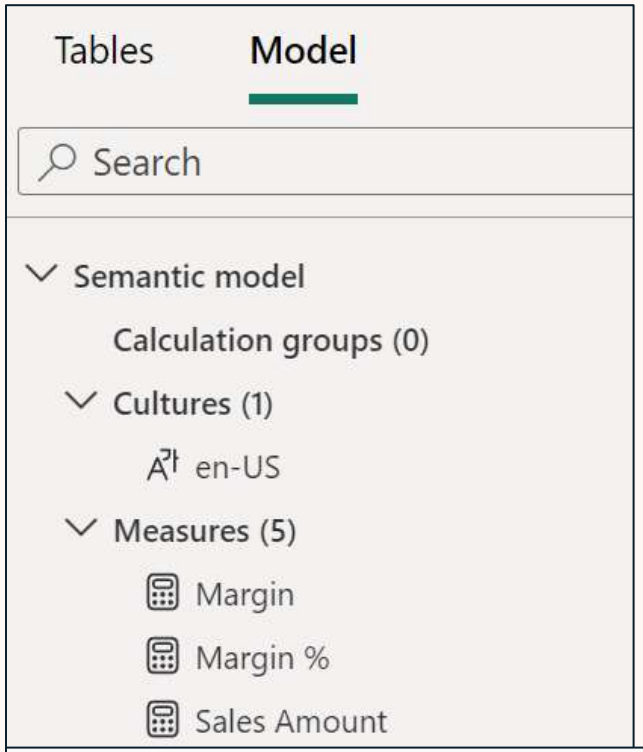
Data: only tables

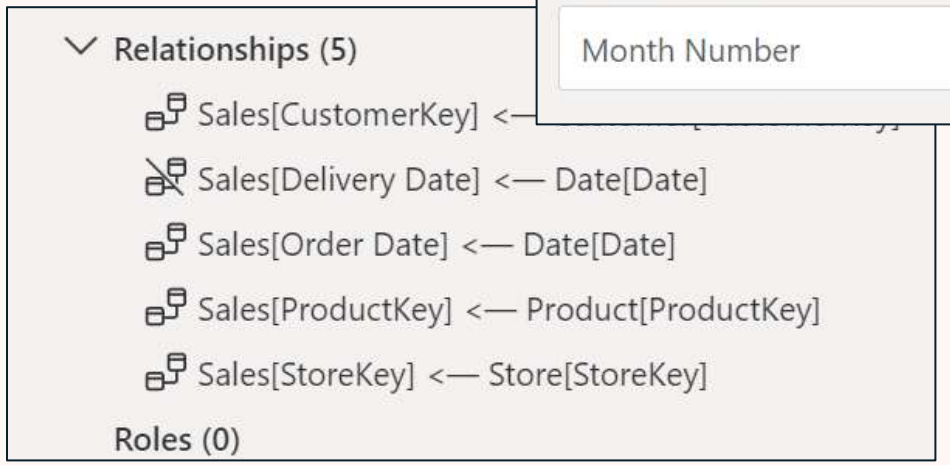


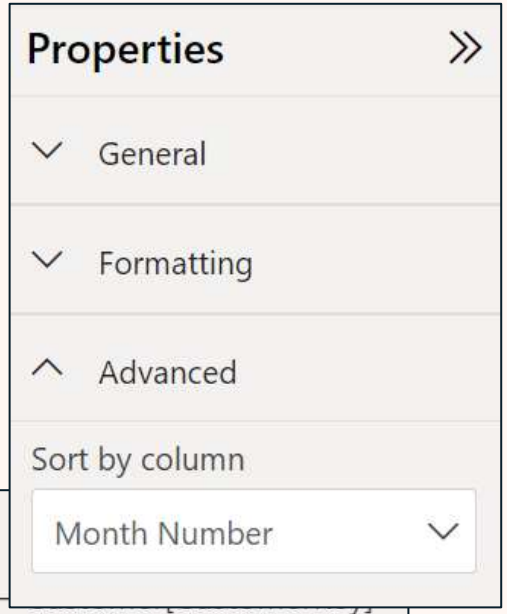
Dataset: tables + relationships + more...

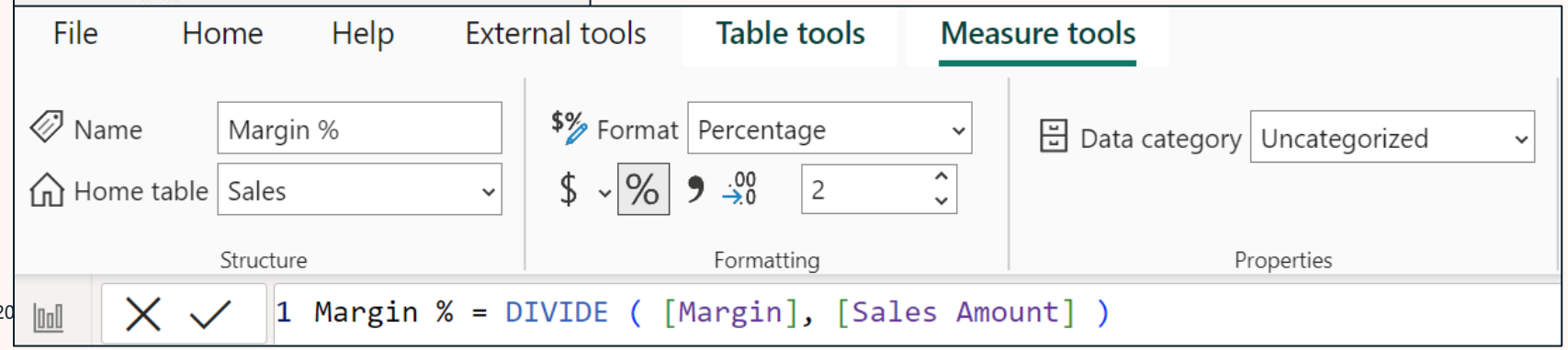


Semantic model









Dynamic calculations embedded in the model

- Model-level semantics:
  - Calculated columns
  - Calculated tables
- Report-level semantics:
  - Measures
  - Calculation groups
- Composable language: DAX

Layout features

Layout features

- Format string
  - Static on columns
  - Static or dynamic on measures
- Hierarchies
- Sort column by
  - Month name
  - Day name
  - Account name by section
- Folders
- KPI

SQL vs. DAX

Sales by Month (Order Date)

DAX

```
EVALUATE
SUMMARIZECOLUMNS (
    'Date'[Calendar Year Month],
    'Date'[Calendar Year Month Number],
    "Sales Amount", [Sales Amount],
    "Margin", [Margin],
    "Margin %", [Margin %]
)
ORDER BY
'Date'[Calendar Year Month Number]
```

SQL

```
SELECT
    d.[Year Month],
    d.[Year Month Number],
    SUM ( s.Quantity * s.[Net Price] ) AS SalesAmount,
    SUM ( s.Quantity * (s.[Net Price] - s.[Unit Cost] ) )
        AS Margin,
    SUM ( s.Quantity * (s.[Net Price] - s.[Unit Cost] ) )
        / SUM ( s.Quantity * s.[Net Price] ) AS [Margin %]
FROM Sales s
LEFT JOIN [Date] d
    ON d.[Date] = s.[Order Date]
GROUP BY d.[Year Month Number], d.[Year Month]
ORDER BY d.[Year Month Number]
```

Commission by Price Range

DAX

```
EVALUATE
SUMMARIZECOLUMNS (
    'Price Ranges'[Price Range],
    'Price Ranges'[PriceRangeKey],
    "Margin", [Margin],
    "Margin %", [Margin %],
    "% of Margin", [% of Margin]
)
ORDER BY 'Price Ranges'[PriceRangeKey]
```

SQL

```
),
CommissionByPriceRange AS (
    SELECT
        pr.PriceRangeKey,
        pr.PriceRange,
        SUM ( s.Quantity * s.[Net Price] ) AS SalesAmount,
        SUM ( s.Quantity * (s.[Net Price] - s.[Unit Cost] ) ) AS
    FROM Sales s
    LEFT JOIN PriceRange pr
        ON pr.[Min] <= s.[Net Price] AND pr.[Max] > s.[Net Price]
    GROUP BY pr.PriceRangeKey, pr.PriceRange
)
SELECT
    PriceRange,
    Margin,
    Margin / SalesAmount AS [Margin %],
    Margin / (SELECT SUM ( Margin ) FROM CommissionByPriceRange )
FROM CommissionByPriceRange
ORDER BY PriceRangeKey;
```

Security features

Row-level security (RLS)

- Static
  - All the role members have access to the same data
  - The security perimeter does not depend on the user, only on the role
- Dynamic
  - DAX expression based on connected user
  - Can be data-driven, based on specific configuration tables in the model

Object-level security (OLS)

- Define tables/columns that are not visible (restricted)
- Restrictions propagate to any calculation
- Example: hides sensitive data (SSN)

Modeling features

Different types of many-to-many relationships (M2M)

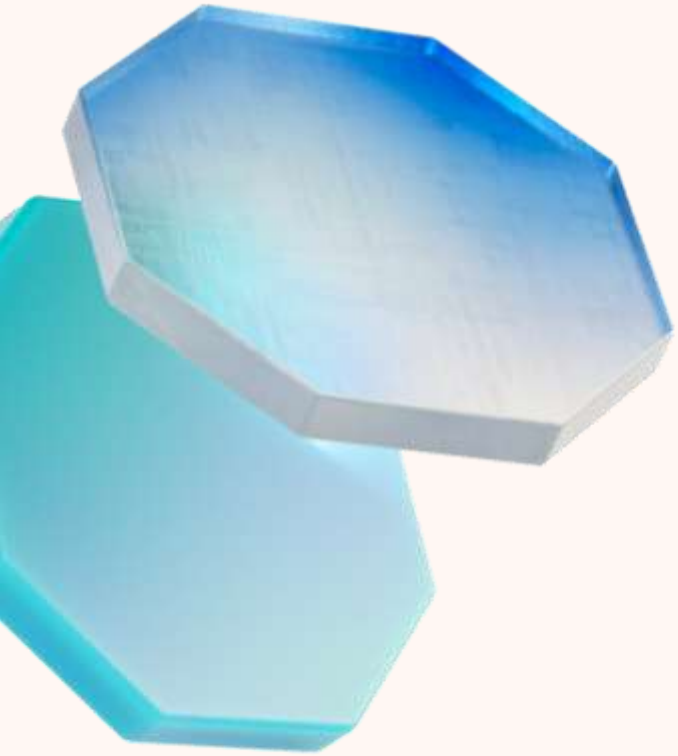
- M2M relationship between business entities (dimensions)
  - Use a bridge table from the data source
  - Apply a bidirectional filter on one relationship to enable non-additive calculations through M2M relationship
  - Example: bank accounts and customers who are account owners)
- M2M cardinality relationship
  - Propagate a relationship at a different granularity
  - Specify single direction for filter propagation (do not use bidirectional filters)
  - Example: budget by brand / country, dimensions products / customers

Composite models

Composite models

- Create a semantic model based on an existing one
  - No data duplication
  - Model artifacts are just a reference (proxy) to the remote model
- The local model can:
  - Add local tables
  - Add relationships, also between local and remote tables
  - Add measures, with automatic remote execution whenever possible
  - Add calculation groups
  - Rename existing objects (tables, columns, measures)
- The composite model can be published
  - Refresh needed for local tables only - including calculated tables

Conclusions

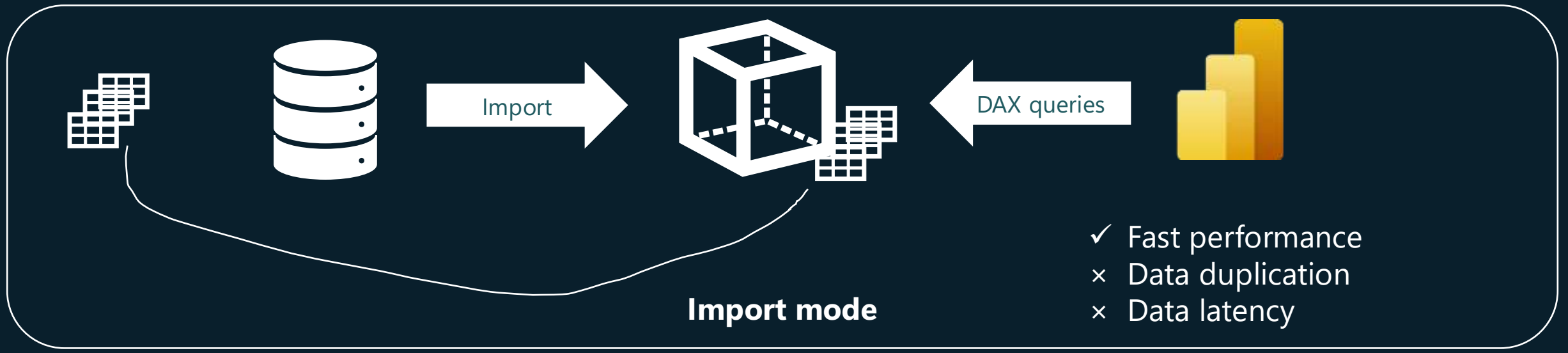


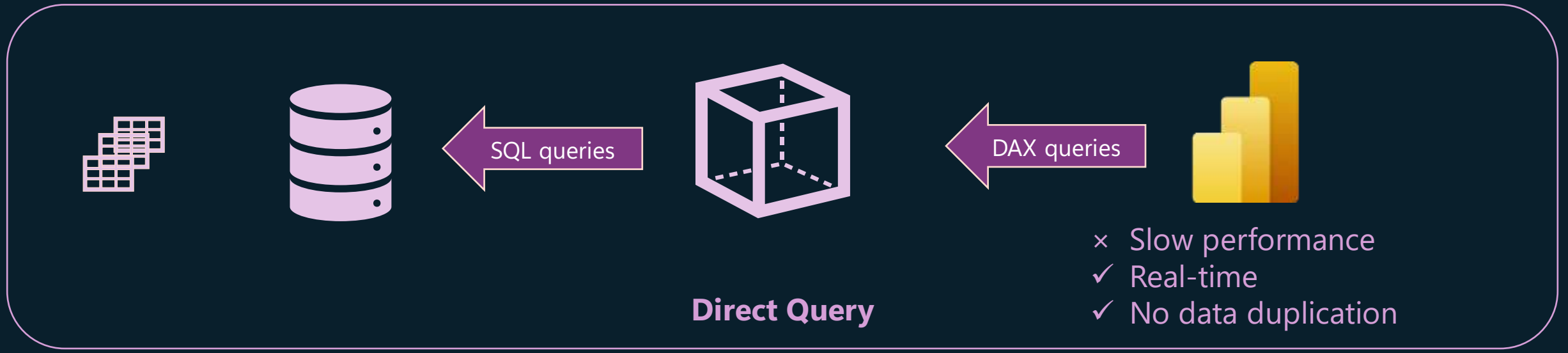
Adopt semantic models

- Control data quality
- Simplify data consumption
- Improve the productivity of report authors
- Extend certified models for task-specific specialized models
  - Leverage self-service BI by reusing corporate BI assets
- It was called dataset,
  but it has always been a semantic model

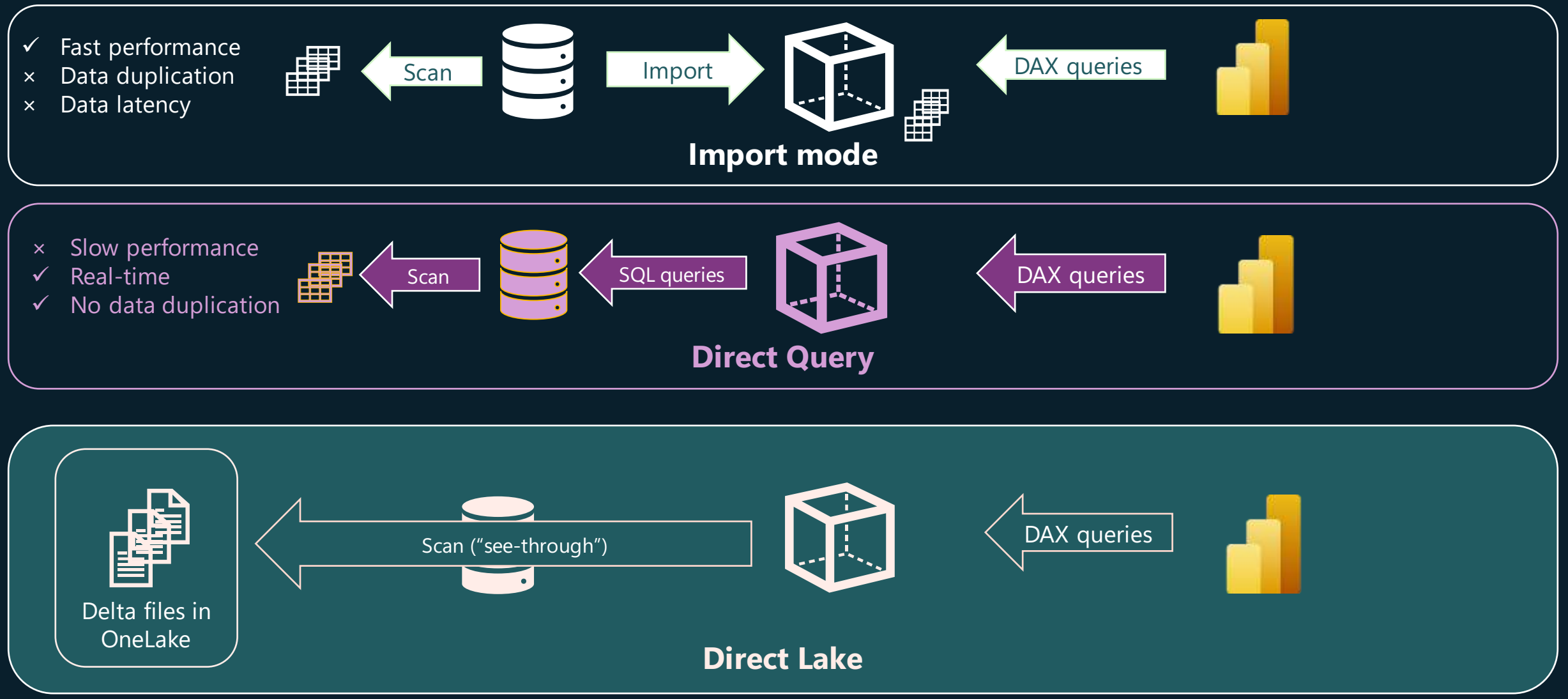
Power BI Semantic models: Storage modes

Power BI Architecture – Pre-Fabric





Power BI Architecture – Fabric



Direct Lake mode

Revolutionary feature!

✓ Prerequisites

- ✓ Fabric F Capacity/Power BI Premium
- ✓ Lakehouse + SQL Endpoint (for DQ fallback)/Warehouse
- ✓ Delta tables
- ✓ V-Ordering*

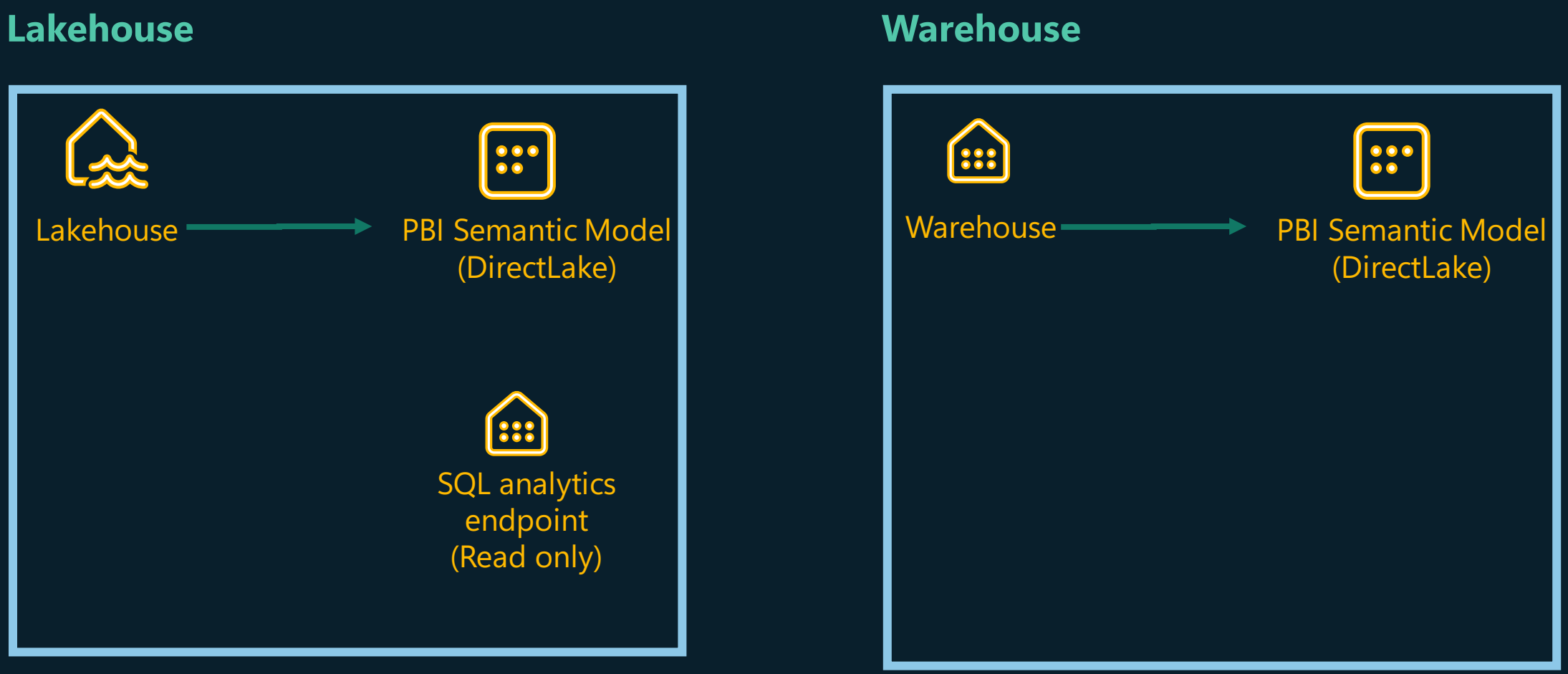
* V-Ordering
Fabric-specific way of additionally optimizing Parquet files when writing data

Diving directly into a lake!

All engines read/write Delta files!

Explore end-to-end analytics with Microsoft Fabric - Training | Microsoft Learn

Default Power BI semantic model in DirectLake!



How does this magic work?

- Semantic model is initially empty
- Loads data into memory on-demand quickly
- Parquet is very similar to how Power BI stores its data in import mode (in VertiPaq)
- Power BI engine (Analysis Services) reads from Delta tables directly
- Refreshes just the parts it needs on-demand
- Direct Lake uses the SQL endpoint to discover schema and security information

Paging

- Use DMVs to see data paging / temperature:

```
select * from $SYSTEM.DISCOVER_STORAGE_TABLE_COLUMNS
Order by dictionary_Last_Accessed Desc
```

```
select * from $SYSTEM.DISCOVER_STORAGE_TABLE_COLUMN_SEGMENTS
Order by Last_Accessed Desc
```

TODAY(): Only works against the custom semantic model!

Semantic Model Refresh

Refreshing the data model

- Framing!
- Reads the Delta Log
- Metadata only

```
1  %%sql
2
3  DESCRIBE HISTORY cms_provider_drug_costs
```

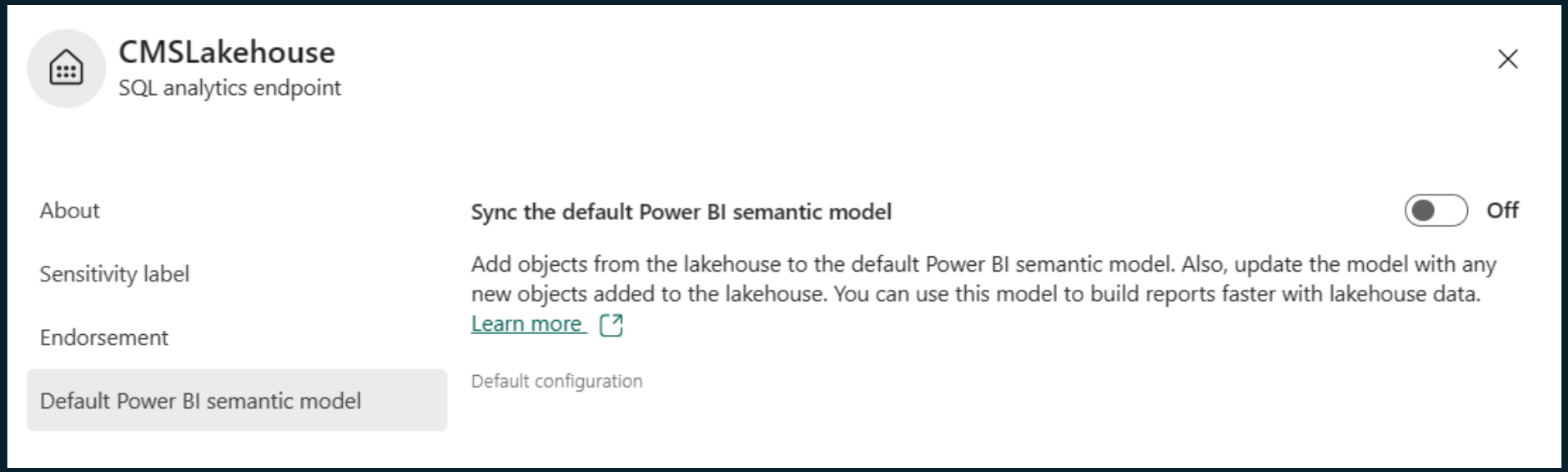
2 sec -Command executed in 2 sec 372 ms by shabnam watson on 7:18:58 PM, 3/09/24 Spark S

Spark jobs (1 of 1 succeeded) Log

Table Chart Search

	12L version	timestamp	ABC userId	ABC userName	ABC operation
1	8	2024-03-10T00:11:01Z	NULL	NULL	WRITE
2	7	2024-03-10T00:08:03Z	NULL	NULL	WRITE
3	6	2024-03-10T00:05:03Z	NULL	NULL	WRITE
4	5	2024-03-10T00:02:01Z	NULL	NULL	WRITE
5	4	2024-03-09T23:59:00Z	NULL	NULL	WRITE
6	3	2024-03-09T23:56:02Z	NULL	NULL	WRITE
7	2	2024-03-09T23:53:06Z	NULL	NULL	WRITE
8	1	2024-03-09T23:50:13Z	NULL	NULL	WRITE
9	0	2024-03-09T23:47:25Z	NULL	NULL	WRITE

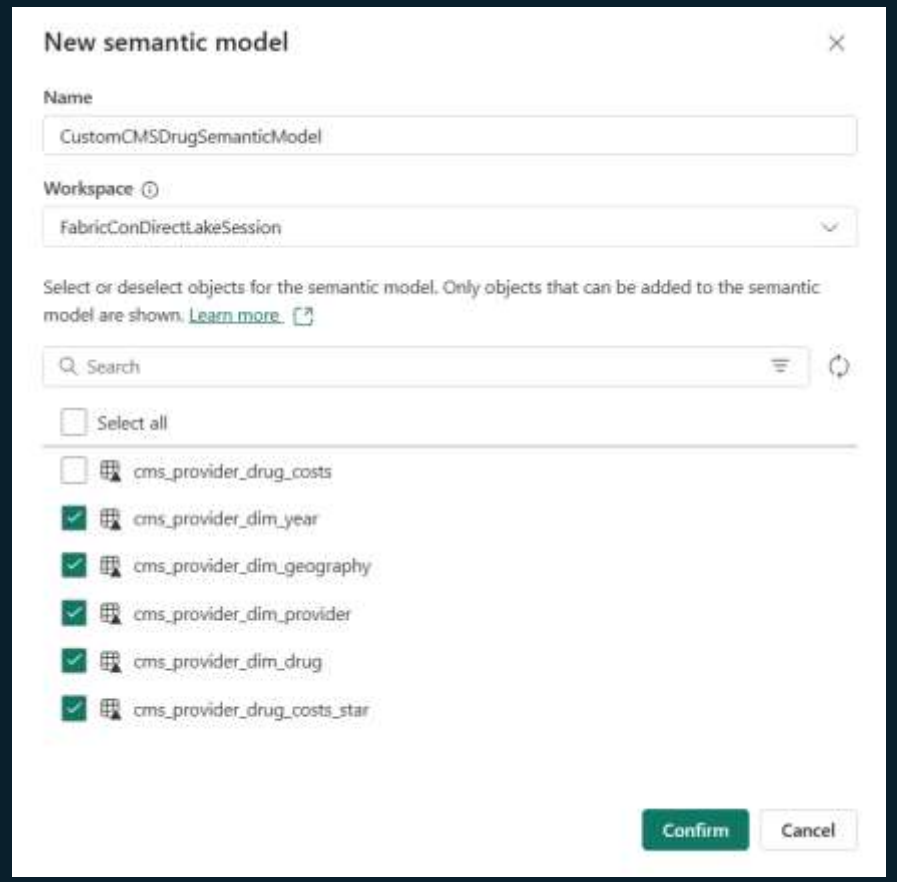
Syncing from lakehouse for default semantic model



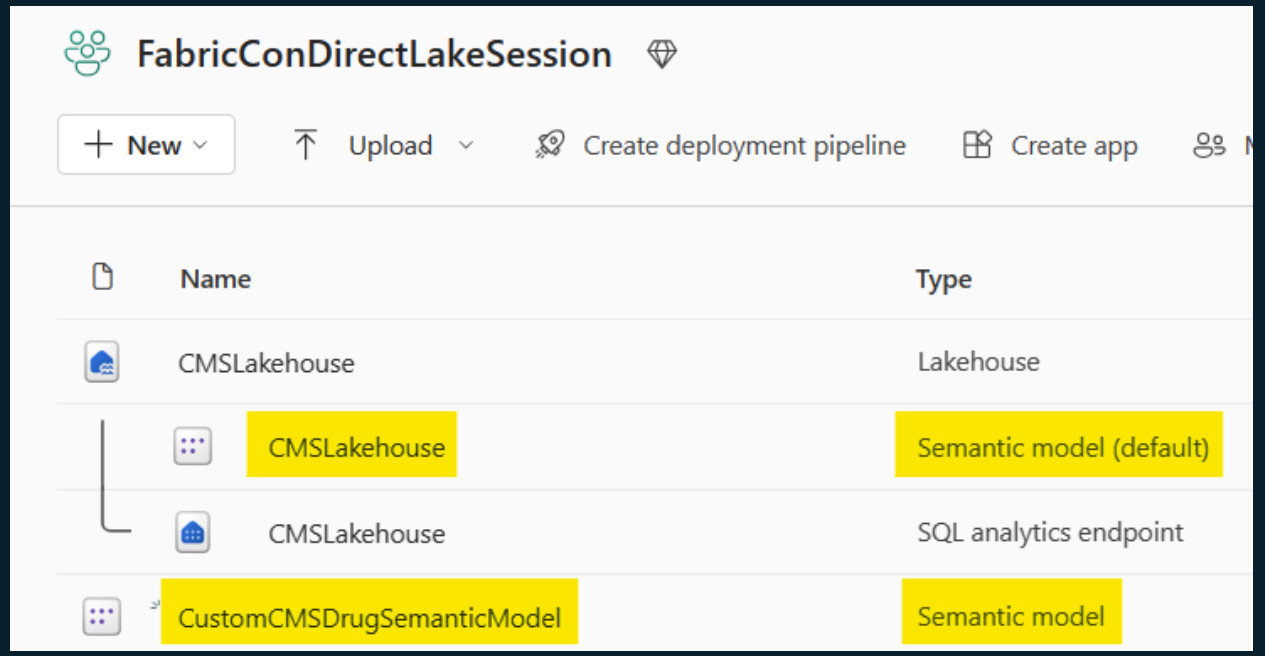
- If set to off (default), you need a refresh to add new tables.

Custom Semantic Model

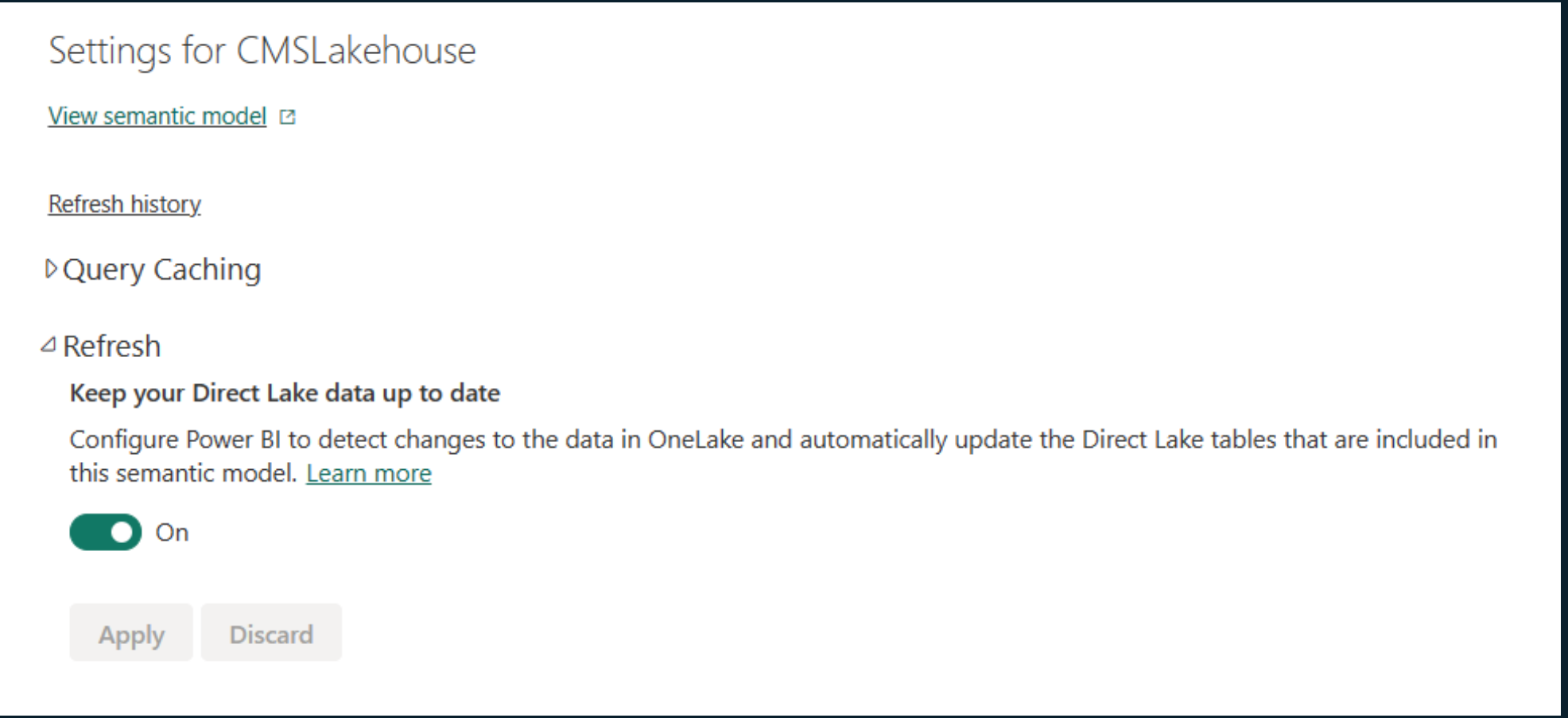
Custom Semantic Model



Not based on relationships in the SQL Endpoint



Refresh Options for default semantic model



Custom Semantic Model

- Refresh can be scheduled
- Schema changes are not updated automatically
- Spark notebooks (SemPy) or Dataflows Gen2

Fallback to Direct Query

Fallback to DirectQuery

RLS/OLS in Warehouse

Views

Learn about Direct Lake in Power BI and Microsoft Fabric - Power BI | Microsoft Learn

DirectLake Guardrails

Fabric/Power BI SKUs	Parquet files per table	Row groups per table	Rows per table (millions)	Max model size on disk/OneLake[1] (GB)	Max memory (GB)
F2	1,000	1,000	300	10	3
F4	1,000	1,000	300	10	3
F8	1,000	1,000	300	10	3
F16	1,000	1,000	300	20	5
F32	1,000	1,000	300	40	10
F64/FT1/P1	5,000	5,000	1,500	Unlimited	25
F128/P2	5,000	5,000	3,000	Unlimited	50
F256/P3	5,000	5,000	6,000	Unlimited	100
F512/P4	10,000	10,000	12,000	Unlimited	200
F1024/P5	10,000	10,000	24,000	Unlimited	400
F2048	10,000	10,000	24,000	Unlimited	400

Learn about Direct Lake in Power BI and Microsoft Fabric - Power BI | Microsoft Learn

Max Memory = memory resource limit for how much data can be paged in for each query

Max model size on disk/Onelake = limit beyond which all queries fall back to DirectQuery

DMV for DirectQuery fallback reason

- $SYSTEM.TMSCHEMA_DELTA_TABLE_METADATA_STORAGES
- Zero is good, anything else means fallback to DirestQuery

```
SELECT * FROM $SYSTEM.TMSCHEMA_DELTA_TABLE_METADATA_STORAGES
```

100

Log **Results** History

otLocation	CurrentVersion	TableObjectID	DatamartObjectID	FramedSchemaName	FallbackReason
97f85e-dbff-4070-a2c5-a0083e2dce29/9fc14bda-c43a-448f-ae33-2f73857a9f88/Tables/cms_provider_dim_year/	0	06ce4911-751f-4fe5-add7-d4d9ac3b4228	e541e52f-ef04-4637-8902-b1f3d1d7d005	dbo	0
97f85e-dbff-4070-a2c5-a0083e2dce29/9fc14bda-c43a-448f-ae33-2f73857a9f88/Tables/cms_provider_dim_geography/	0	6c322592-39dc-4fdd-b729-dc8a59e7af3f	e541e52f-ef04-4637-8902-b1f3d1d7d005	dbo	0
97f85e-dbff-4070-a2c5-a0083e2dce29/9fc14bda-c43a-448f-ae33-2f73857a9f88/Tables/cms_provider_dim_provider/	0	e2e600c3-dfdb-4cbd-938b-49c02c43af87	e541e52f-ef04-4637-8902-b1f3d1d7d005	dbo	0
97f85e-dbff-4070-a2c5-a0083e2dce29/9fc14bda-c43a-448f-ae33-2f73857a9f88/Tables/cms_provider_dim_drug/	0	1ecadce2-26d5-4a9f-9e6e-c07a84387aa7	e541e52f-ef04-4637-8902-b1f3d1d7d005	dbo	0
97f85e-dbff-4070-a2c5-a0083e2dce29/9fc14bda-c43a-448f-ae33-2f73857a9f88/Tables/cms_provider_drug_costs_star/	0	7abceec4-017e-449f-8329-6eef161e0114	e541e52f-ef04-4637-8902-b1f3d1d7d005	dbo	0

DirectLakeBehavior property

Can be configured by using Tabular Object Model (TOM) or Tabular Model Scripting Language (TMSL)

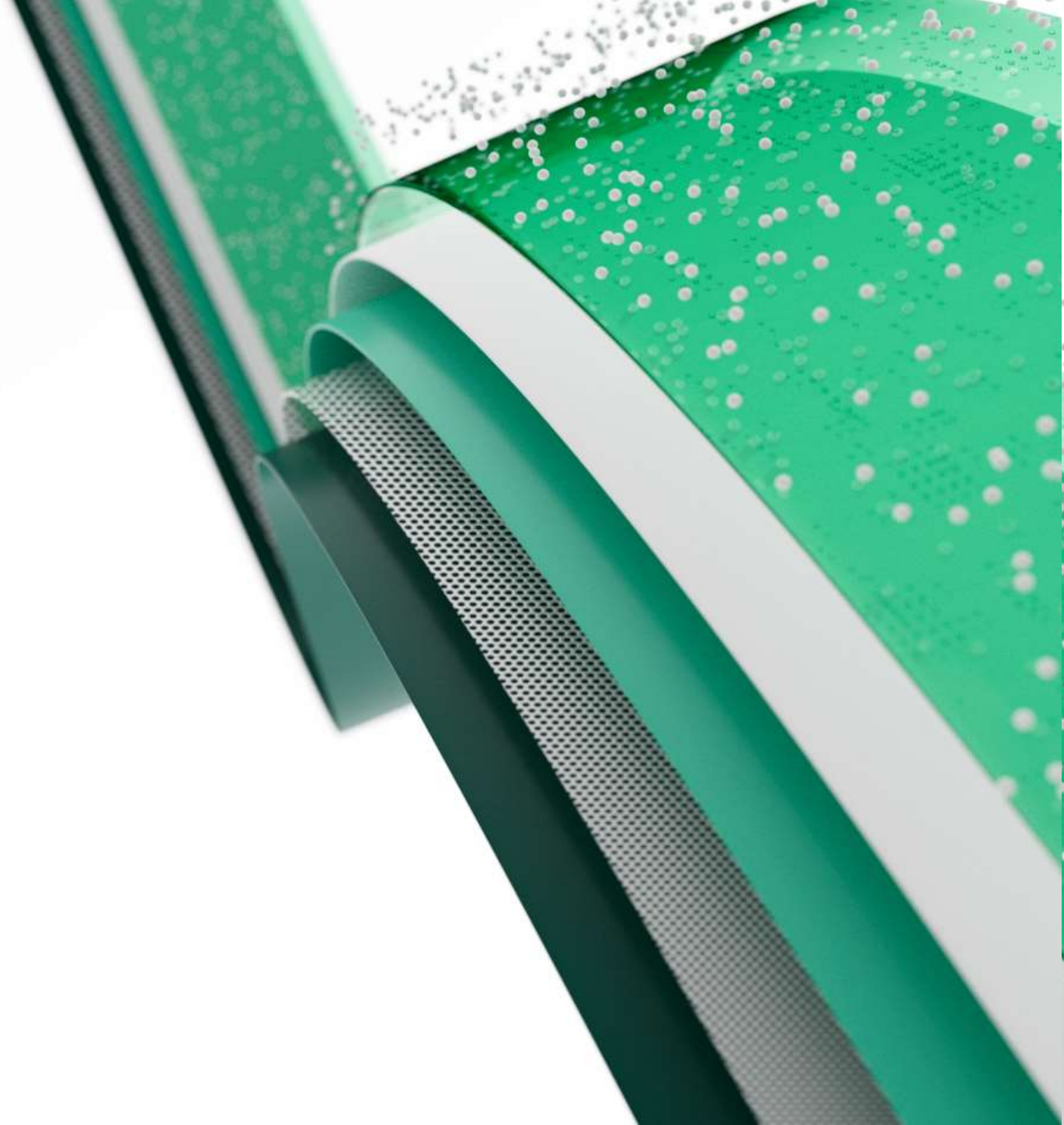
Is a visual using DirectLake or DirectQuery?

Performance analyzer in Power BI Desktop, SQL Server Profiler

If you see DirectQuery in Performance Analyzer

Microsoft

Practice test



Live Practice test

Access Wifi

SSID:

Event Code:

Access the test

Aka.ms/fabric-cert-day

Microsoft

Prepare for exam day

What to expect?

40-60 questions

Plan for 180 minutes

Question types include more than multiple choice

Case studies require complex decision making

Tip: You can check your technical setup in advance

Checkin

Plan for 20-30 minutes for this step

Have an official ID nearby

Nothing on your desk

Unplug all second monitors

Have your smartphone available for checkin

Choose a calm place

No glassed windows if possible

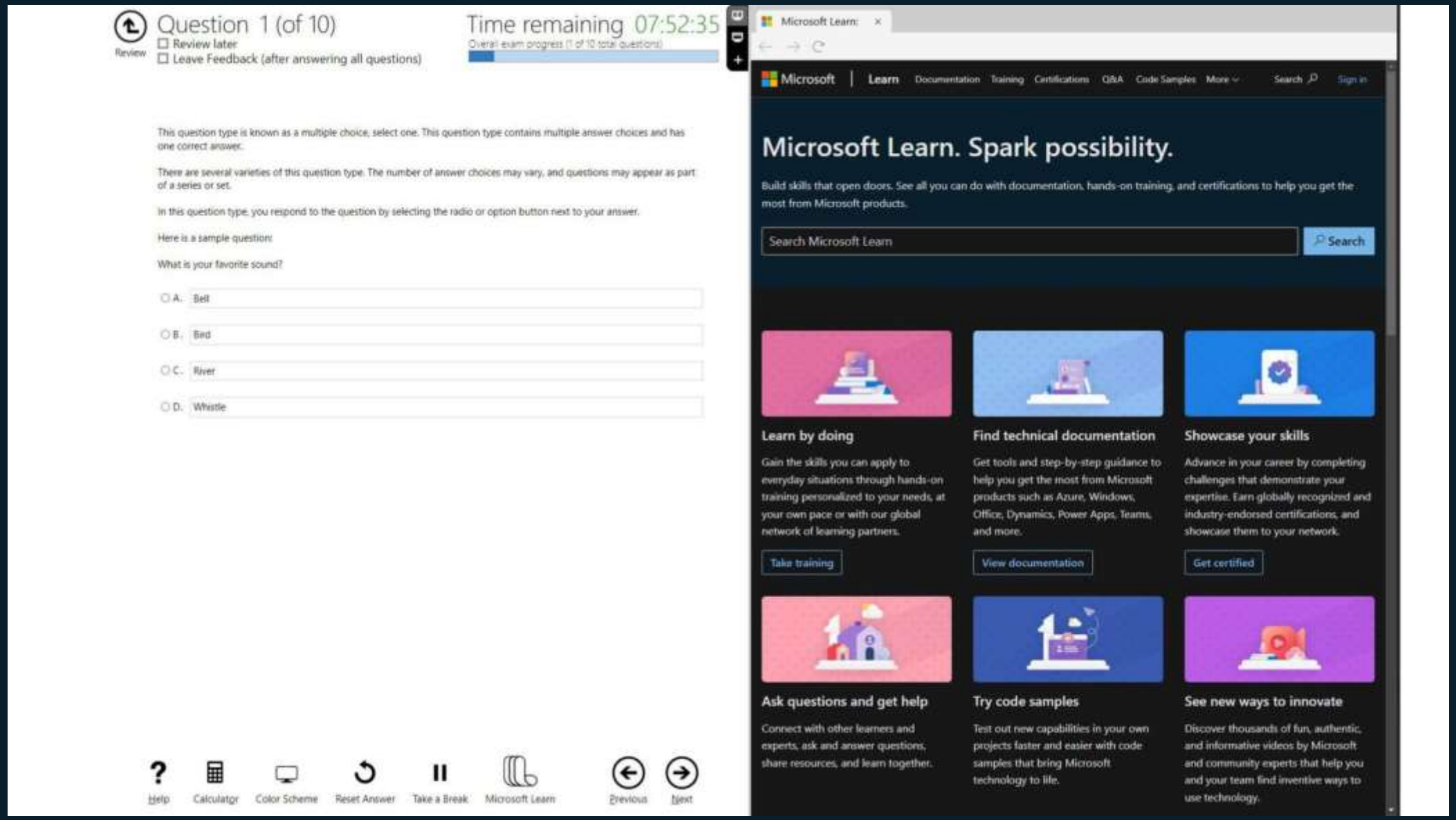
Let’s take an exam

Exam scoring

Exam cut store (700 on a scale of 0-1000). **!= 70%**

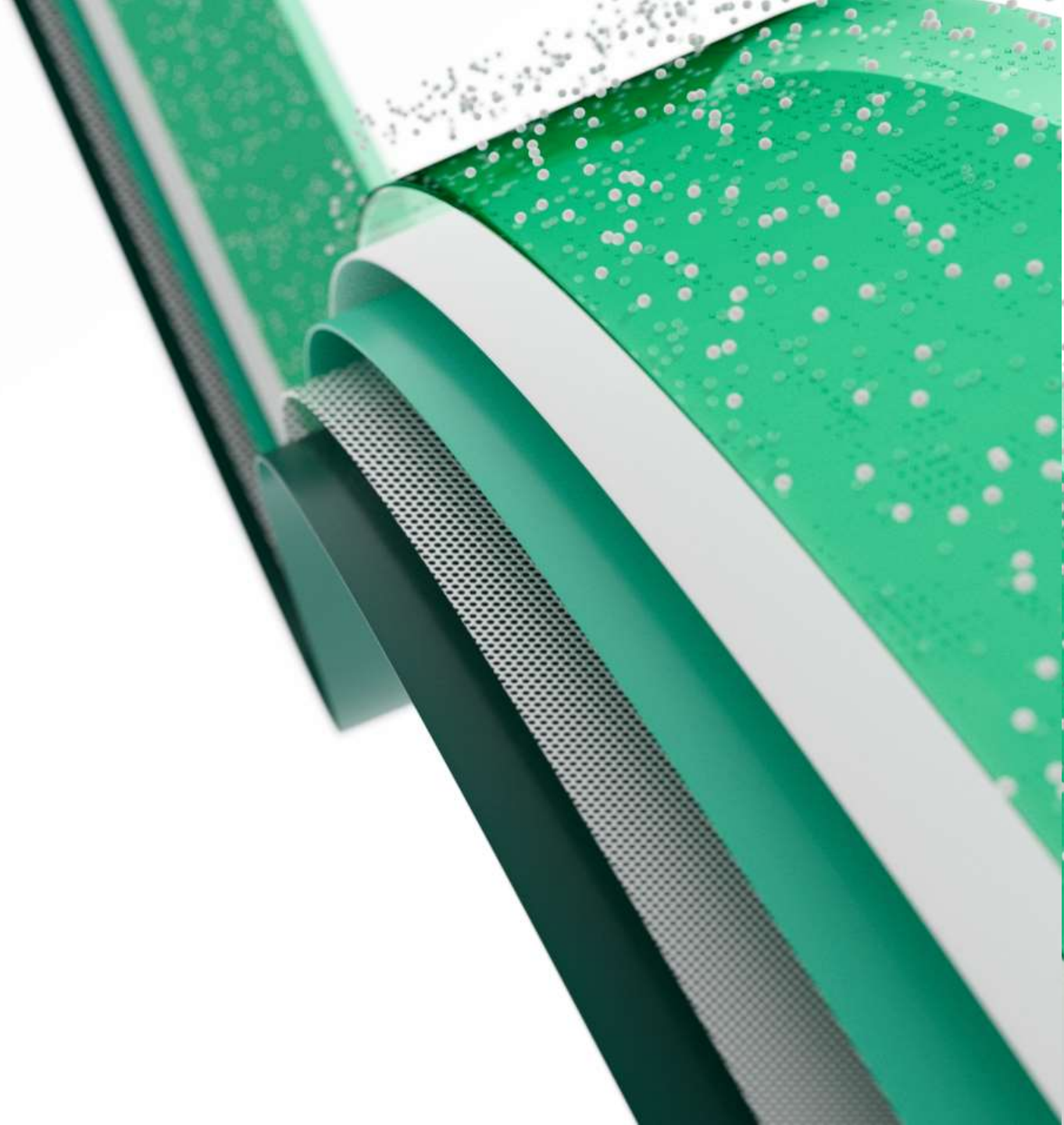
No points deducted for wrong answers
Mark questions for review
Don’t overthink it
Some questions are worth more than one point
Answer every question!

Use Microsoft Learn during exam



Microsoft

Last words

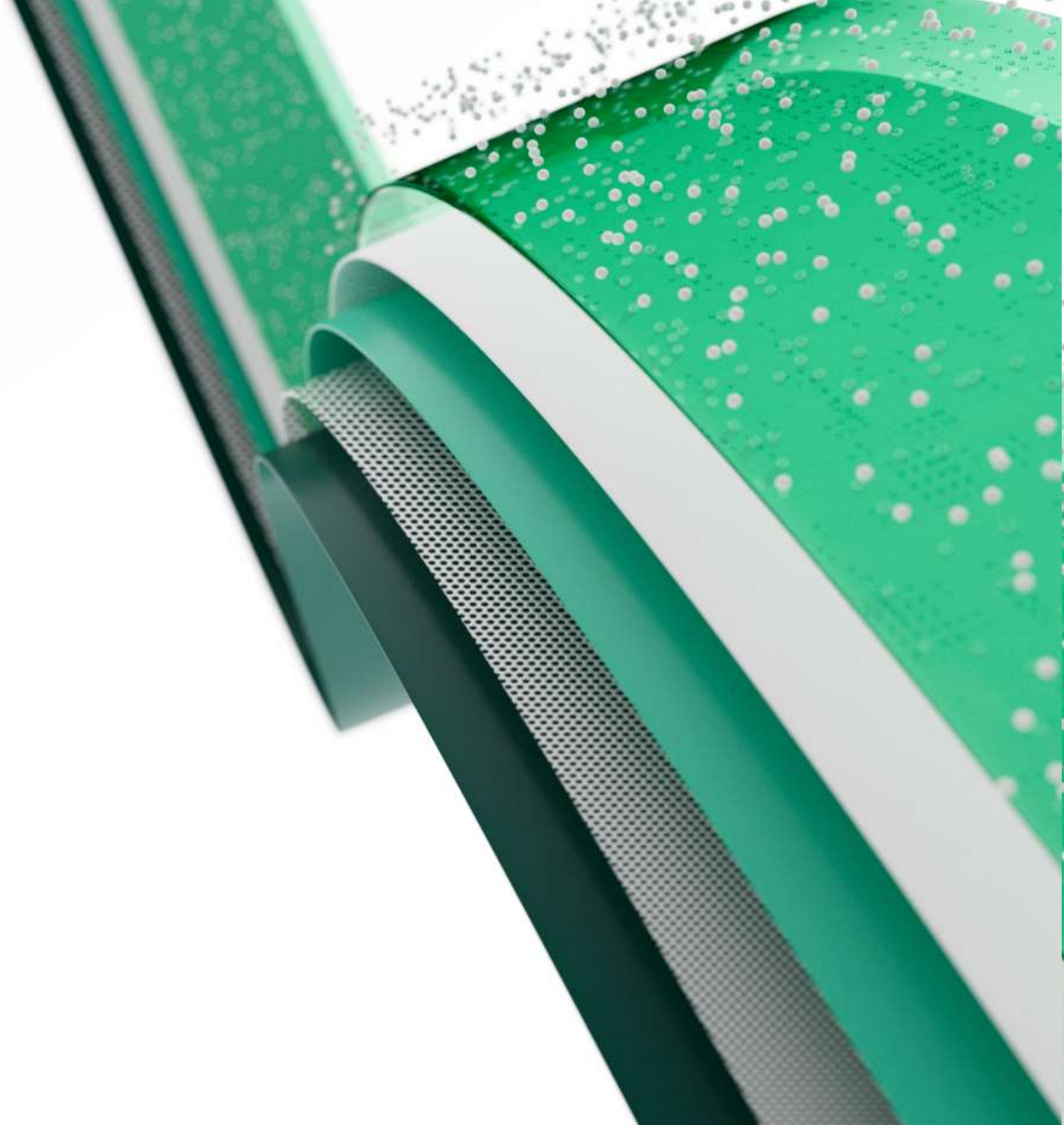


If you pass the exam

Do not hesitate to share your success on LinkedIn and tag us (Frédéric Gisbert and Christopher Maneu) ☺

Microsoft

Q&A



Microsoft

Thank you

<Presenter
Company name>