



Fabric before bed

*A slow walk through Microsoft Fabric,
OneLake and the IQ on top*

BY DIEGO · OCTOBER 2026 · 20 MINUTES

BEFORE YOU START

How to read this

Written for the end of a long day

One idea per chapter. Short lines. No code.

Every chapter ends with one line to remember. If you're too tired for the chapter, read only that line. Tomorrow you'll still know something.

Technical words get explained the moment they show up.

Facts are checked against Microsoft Learn as of October 2026.
Fabric ships something new every month, so some of this will age.
That's the deal with this product.

It's still ok.

Three tools in a trench coat

What Fabric actually is

For years, doing analytics on Azure meant buying separate things. Data Factory to move data. Synapse to crunch it. Power BI to show it. Each had its own storage, its own security and its own bill. Gluing them together was a job title.

Microsoft Fabric is the merge. It went generally available in November 2023.

Fabric is one analytics platform with one lake, one capacity to pay for and many engines on top.

It's SaaS (software as a service). You don't create virtual machines, storage accounts or clusters. You open a browser at app.fabric.microsoft.com, make a workspace and start. You don't even need an Azure subscription to use it, although most companies buy it through one.

If you know Power BI, you already know the front door. Fabric grew out of the Power BI service. Same portal, same workspaces, a lot more inside.

REMEMBER

One lake, one capacity, many engines.

CHAPTER 2

Tenant, capacity, workspace, item

Four words that explain the whole structure

Fabric is nested like boxes in boxes.

Tenant your organization, one per company

Capacity the engine room you pay for, in one region

Workspace a team's folder, sits on a capacity

Item a lakehouse, a report, a notebook, a pipeline

Boxes in boxes. Domains group workspaces by business area.

The **tenant** is your organization in Microsoft Entra, the same identity system behind your Office login. One company, one tenant.

A **capacity** is a pool of compute you buy. It lives in one Azure region. That region is also where the data of its workspaces is stored, which matters a lot in Belgium. More on money in chapter nine.

A **workspace** is a team's shared folder. It sits on one capacity and holds items. Access is given per workspace through four roles: Admin, Member, Contributor and Viewer.

An **item** is the actual thing you build. A lakehouse. A warehouse. A notebook. A pipeline. A report. Everything in Fabric is an item in a workspace.

On top of workspaces there are **domains**. A domain groups workspaces by business area, like Finance or Supply Chain, each with its own admins. That's how Fabric does data mesh, the idea that each business area owns its own data instead of one central team owning everything.

REMEMBER

Tenant holds capacities. Capacities hold workspaces. Workspaces hold items.

OneLake, OneDrive for data

The part that makes everything else work

Every Fabric tenant gets exactly one OneLake. You can't make a second one. You can't delete it. There's nothing to set up.

Microsoft calls it OneDrive for data. That's a decent picture. One place, organized in folders, where everybody's data lives, with permissions on top.

Underneath it's Azure Data Lake Storage. Tables are stored as **Delta Parquet** or **Iceberg**. Both are open file formats for tables. Parquet stores data by column so reads are fast. Delta and Iceberg add a transaction log on top, so you get versions, safe updates and time travel. Open means any tool can read them, inside or outside Fabric.

Every engine in Fabric reads and writes the same copy of the data in OneLake.

That's the big idea. Spark writes a table. The SQL engine queries it. Power BI reports on it. Nobody exports, copies or loads anything in between. One copy.

Before this, every engine kept its own copy. Five copies of the sales table, all slightly different, each with its own refresh schedule. You know the meeting where two dashboards show different revenue. That meeting.

The **OneLake catalog** is where you find things. You search, filter by domain or owner, see who endorsed a table and where it came from. Owners get a governance view too: what's labeled, what isn't, what needs attention.

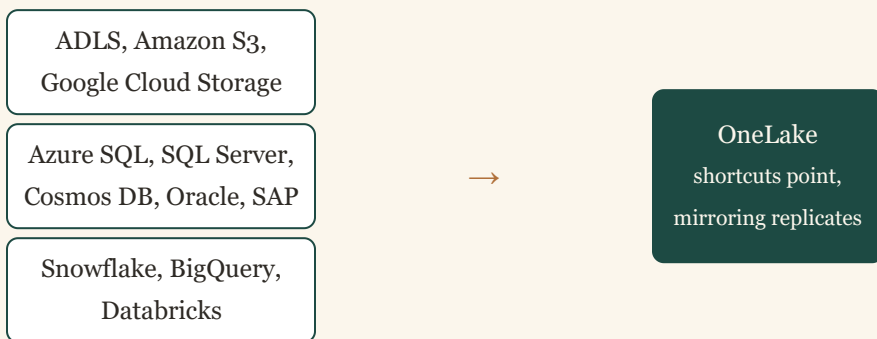
REMEMBER

One lake per tenant, open formats, one copy for every engine.

Getting data in without copying it

Shortcuts and mirroring

Your data isn't in Fabric yet. It's in an Azure data lake, an S3 bucket, a SQL database, Snowflake, Databricks, SAP. Two features bring it close without the usual pain.



Two ways in. Neither needs you to build a pipeline.

A **shortcut** is a pointer. It looks like a folder in OneLake but the data stays where it is. Think of a desktop shortcut to a file on a network drive. Delete the shortcut and the source is untouched. Shortcuts can point to other places in OneLake, to Azure Data Lake Storage, to Amazon S3, to Google Cloud Storage and more. You create them in a lakehouse or a KQL database.

Mirroring keeps a live copy of a database in OneLake. You connect a source and Fabric replicates the tables into Delta format, continuously. Changes can land as fast as every 15 seconds. Sources include Azure SQL Database, SQL Server, Azure

Cosmos DB, PostgreSQL, Oracle, SAP, Snowflake and Google BigQuery. Each mirrored database also gets a SQL endpoint you can query right away.

For Azure Databricks, mirroring only copies the catalog structure and uses shortcuts for the data. Fabric calls that metadata mirroring. The data stays in Databricks. Fabric just knows where it is.

The rule of thumb: **shortcut** when the data already sits in a lake in an open format. **Mirror** when it lives in a database. Build a pipeline only when you have to transform on the way in.

One caveat. A shortcut to another cloud still reads across clouds. The other cloud may charge you for data leaving it. Free in Fabric isn't always free everywhere.

REMEMBER

Shortcut points to data. Mirroring keeps a live copy of a database.

The engines, one by one

Fabric calls them workloads

A workload is a family of items for one kind of job. Here they are, in the order data usually travels.

Data Factory. Moving and shaping data. Pipelines orchestrate steps. Dataflow Gen2 uses Power Query, the same low-code editor as in Excel and Power BI. More than 200 connectors.

Data Engineering. Apache Spark, the open-source engine for processing big data across many machines. You write notebooks in Python or SQL. The main item is the lakehouse.

Data Warehouse. Full T-SQL, the SQL dialect of SQL Server. Tables, views, stored procedures, transactions. Stored as Delta in OneLake like everything else.

Databases. SQL database in Fabric, an operational database based on Azure SQL Database. Good for apps and for writing data back. It mirrors itself into OneLake automatically.

Real-Time Intelligence. Data in motion: sensors, logs, clicks. Chapter eight.

Data Science. Machine learning models with experiment tracking. Predictions can land in tables for reports.

Power BI. Reports, dashboards and semantic models. Chapter seven.

IQ. The newest one. Business meaning on top of data, for people and agents. Chapter eleven.

You don't need all of them. Most customers start with Data Factory, a lakehouse and Power BI. That's already a complete platform.

REMEMBER

Different engines for different jobs, all on the same lake.

Lakehouse, warehouse or database?

The question every first workshop gets stuck on

They all store tables in OneLake. The difference is who works on them and how.

A **lakehouse** holds files and tables together. You write to it with Spark, in notebooks. It has two folders. Files holds anything. Tables holds Delta tables. Every lakehouse comes with a **SQL analytics endpoint**, a read-only SQL door so analysts can query the tables without touching Spark. Data engineers love it.

A **warehouse** is for people who think in SQL. You read and write with T-SQL, with transactions across tables. SQL developers and BI teams love it.

A **SQL database** is for apps. Many small reads and writes, every second. It's the operational side, the one that runs a process, not the one that analyzes it.

An **eventhouse** is for time-stamped streams: logs, telemetry, sensor readings. You query it in KQL. Chapter eight again.

Pick by the skills of the team, not by the size of the data.

All four can feed each other and Power BI. Mixing is normal. A lakehouse for raw data and a warehouse for the final business tables is a very common pattern.

The common layout is called **medallion**. Bronze holds raw data as it arrived. Silver holds cleaned data. Gold holds business-ready tables for reports. Three layers, each one tidier than the last.

Materialized lake views help here. You write the silver or gold step as a Spark SQL statement. Fabric stores the result as a table, refreshes it in the right order and checks data quality rules on the way. Less custom code for the same pipeline.

REMEMBER

Spark people: lakehouse. SQL people: warehouse. Apps: SQL database. Streams: eventhouse.

Power BI and Direct Lake

The trick that made Fabric worth it for BI teams

A Power BI report reads from a **semantic model**. That's the layer with tables, relationships and measures, the business formulas written in DAX like Revenue or Margin. Reports are only the drawings on top.

A semantic model can get its data in three ways.

Import. Copy all the data into the model's memory. Very fast to query. Slow and heavy to refresh. And it's yet another copy.

DirectQuery. Send every click as a query to the source. Always fresh. Often slow.

Direct Lake. New with Fabric. The model reads the Delta tables in OneLake straight into memory when needed. Import speed, without the copy.

With Direct Lake, a refresh doesn't copy data. It only updates which version of the files to read. Fabric calls it framing. It takes seconds instead of hours.

Direct Lake gives you import speed without the import copy.

The catch: the gold tables must be in good shape. Direct Lake is only as quick as the Delta tables underneath. And depending on

how the model is built, it can fall back to DirectQuery when it hits limits, which is slower. Check that before a big demo.

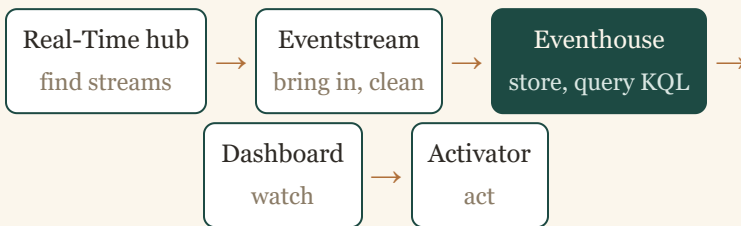
REMEMBER

Semantic model is the brain. Direct Lake reads OneLake directly, no copy.

Real-Time Intelligence

For data that never stops arriving

Some data arrives all the time. Machine sensors on a factory line. App logs. Payment events. Waiting for tonight's batch is too late. Real-Time Intelligence handles that.



Find, ingest, store, watch, act

The **Real-Time hub** is the catalog of streams in your organization. Find a stream, subscribe, done.

An **eventstream** brings data in without code. Kafka, Azure Event Hubs, IoT Hub, database change feeds, even Amazon Kinesis and Google Pub/Sub. It can filter, clean and aggregate on the way.

An **eventhouse** stores the events, indexed by time. You query it in **KQL**, the Kusto Query Language. KQL reads like a pipeline: take this table, filter, then summarize. Fast on billions of rows. It has built-in anomaly detection too.

Real-Time dashboards show it live. **Activator** acts on it. You set a rule like "if the temperature stays above 80 for five minutes"

and Activator sends a Teams message, an email or starts a pipeline.

Real-time means reacting when it happens, not on a schedule.

Real-time doesn't have to mean huge. A few events per minute is fine. The point is the reaction.

REMEMBER

Eventstream in, eventhouse stores, Activator acts.

Money, or how capacity works

The chapter customers read twice

You don't pay per query. You pay for a **capacity**, a fixed amount of compute measured in **capacity units**, CUs.

Capacities come in F sizes: F2, F4, F8 and doubling all the way up to F2048 and beyond. The number is the CUs. An F64 has 64.

You buy F capacities in Azure. Billing is per second with a one-minute minimum. You can pause a capacity when nobody uses it, or reserve it for a year to pay less. The old Power BI Premium P capacities are being retired in favor of F.

Two license rules cause most surprises.

Viewing reports. On F64 or bigger, people with a free license can view Power BI content. Below F64, every viewer needs a Pro or Premium Per User license.

Building reports. Creating Power BI content in a shared workspace still needs a Pro license, whatever the capacity size.

Now the clever part. Jobs can **burst**, using more compute than your capacity has, so they finish fast. Fabric then **smooths** the cost over time. Interactive work, like clicking in a report, is spread over at least five minutes. Background work, like a nightly refresh, is spread over 24 hours. So a big job doesn't block the morning.

When you overspend for long, Fabric **throttles**, in steps. Up to 10 minutes of future capacity used up: nothing happens. Past 10 minutes: new interactive work waits 20 seconds. Past an hour: new interactive work is rejected. Past 24 hours: everything is rejected until the debt is paid off.

A capacity is a budget you borrow against. Borrow too much and the bank says no.

There's a free 60-day trial with the power of an F64. And the Capacity Metrics app shows who used what. Install it on day one.

REMEMBER

Pay for a capacity. Bursting and smoothing absorb spikes. Too much for too long means throttling.

Who can see what

Security and governance without a headache

Three levels, from coarse to fine.

Workspace roles decide who can open a workspace and what they can do: Admin, Member, Contributor or Viewer.

Item sharing gives someone one item without the whole workspace. One report, one lakehouse.

OneLake security goes down to folders, tables, rows and columns. Share the sales table but hide the Cost column. Show a partner only the rows where Region is Belgium. The rule is stored once in OneLake and applies everywhere: in SQL, in Spark, in Power BI.

Define the rule once in OneLake and every engine obeys it.

Before Fabric, you set the same rule in each engine and hoped they matched. They didn't.

Sensitivity labels from Microsoft Purview, like Confidential, travel with the data. Label a table and the report built on it inherits the label, even when exported to Excel. Purview also adds data loss prevention and audit for supported Fabric items.

For other companies there's **external data sharing**. You share live data with another tenant and they read it in place. No copy, no file sent around by email.

REMEMBER

Workspace roles for teams, OneLake security for rows and columns, Purview labels that travel.

AI on top: Copilot, data agents, Fabric IQ

Where Fabric meets the agents from the Foundry book

Copilot in Fabric helps you build. It writes SQL from a sentence, suggests notebook code, explains a failing Spark job, drafts a report page. It's on by default for paid capacities F2 and up. It uses your capacity, so watch the metrics.

A **data agent** helps people ask. You point it at a lakehouse, a warehouse, a semantic model, a KQL database or an ontology. You add instructions and example questions. People then ask in plain language and get answers from governed data. Data agents are generally available. You can publish them to Microsoft 365 Copilot, Copilot Studio and Foundry. So a Foundry agent can call a Fabric data agent as its data expert.

Fabric IQ is the newest layer, mostly in preview. The idea: tables and columns mean nothing to an agent. Business concepts do. Customer. Shipment. Machine. Fabric IQ describes your business in those terms.

Ontology (preview). A shared dictionary of your business: entity types, their properties, how they relate and the rules. Bound to real tables. You can generate one from an existing semantic model.

Graph. For questions about chains of relations. Which orders were affected by the sensor that failed in Ghent?

Planning. Budgets, forecasts and what-if scenarios, written back to a SQL database. No more planning in a separate tool and a spreadsheet.

Operations agent. Watches live data, spots anomalies and proposes actions.

Fabric IQ is one part of Microsoft IQ. Work IQ knows how people work. Foundry IQ knows documents and policies. Web IQ knows the web. Fabric IQ knows the business data.

Agents are only as good as the meaning you give the data.

REMEMBER

Copilot helps builders. Data agents answer business users. Fabric IQ gives data business meaning.

From dev to prod

The grown-up chapter

Clicking things together in one workspace is fine for a pilot. In production, you want versions, reviews and safe releases.

Git integration connects a workspace to a Git repository in Azure DevOps or GitHub. Items become files you can review, branch and roll back. Every change has a history and an author.

Deployment pipelines move content between stages, like Development, Test and Production, each its own workspace. You compare stages, then promote with a click.

Variable libraries hold the values that change per stage, like a connection string or a folder name. Same pipeline, different values in Test and Production.

Without this, people fix production directly at 11 at night. Everybody has done it once. Don't make it a habit.

REMEMBER

Git for history, deployment pipelines for promotion, variable libraries for settings.

One morning, start to finish

Everything so far in one story

A manufacturer with a plant near Ghent. Here's what Fabric does before the plant manager's first coffee.

1. All night, the ERP database is **mirrored** into OneLake. Orders and stock arrive as Delta tables.
2. Sensor history from an old Azure data lake is reached through a **shortcut**. Nobody copied it.
3. A **notebook** in the lakehouse cleans the raw data. **Materialized lake views** build the gold tables.
4. At 06:00 a **Direct Lake** semantic model frames the new data in seconds. The production report is fresh.
5. Meanwhile, live sensor readings flow through an **eventstream** into an **eventhouse**.
6. At 06:40 a press runs hot. **Activator** posts in the maintenance team's Teams channel.
7. The plant manager asks the **data agent** in Copilot: "Which orders depend on press 4 this week?" The answer comes from the **ontology**, which knows that orders use machines.
8. **OneLake security** hides the cost columns from the manager's view. The **Confidential** label follows the data into Excel.

9. The nightly jobs were heavy. **Smoothing** spread them over 24 hours, so nobody noticed.

One copy of the data. One bill. Nine features. Zero exports.

REMEMBER

Mirror, shortcut, transform, Direct Lake, stream, act, ask. All on one lake.

Things I watched people trip on

Including me

Region. The capacity's region is where the data lives. Pick it before anyone uploads anything. Moving later is work.

Too small to share. Below F64, every viewer needs a Pro license. Do the license math before picking the size.

One heavy job, everyone throttled. Everything on one capacity shares one budget. Separate dev from prod, or heavy engineering from BI.

No metrics. Without the Capacity Metrics app you're guessing. Install it on day one.

Copies by habit. Teams used to copying data keep copying. Ask "can this be a shortcut?" first.

Preview in production. Ontology and much of Fabric IQ are in preview. Fine for pilots. Check the status before a production contract.

Cross-cloud reads. Shortcuts to S3 or Google Cloud are convenient. The other cloud may still bill you for data leaving it.

Direct Lake on messy tables. Thousands of tiny files make everything slow. Keep the gold layer tidy.

The pocket glossary

For when you wake up and forgot half of it

Tenant. Your organization in Microsoft Entra.

Capacity. The compute you pay for, in one region. Sized in CUs.

CU. Capacity unit. The unit of compute. An F64 has 64.

Workspace. A team's folder of items, on one capacity.

Item. Anything you build: lakehouse, report, notebook.

Domain. A group of workspaces for one business area.

OneLake. The single lake of the tenant. OneDrive for data.

Delta / Iceberg. Open table formats on Parquet files.

Shortcut. A pointer to data that stays where it is.

Mirroring. A live replica of a database in OneLake.

Lakehouse. Files and tables, written with Spark.

SQL analytics endpoint. Read-only SQL on a lakehouse.

Warehouse. Tables written and read with T-SQL.

SQL database. Operational database for apps.

Medallion. Bronze raw, silver clean, gold business-ready.

Materialized lake view. A table defined by a query, refreshed by Fabric.

Semantic model. Tables, relationships and measures behind a report.

Direct Lake. Semantic model reading OneLake directly. No copy.

Eventstream. No-code stream ingestion.

Eventhouse / KQL. Store for events, queried with the Kusto Query Language.

Activator. Rules that act when data matches a condition.

Smoothing. Spreading a job's cost over time.

Throttling. Delays and rejections when the capacity is overspent.

OneLake security. Row, column and folder rules for every engine.

Data agent. Plain-language Q&A over governed data.

Ontology. Your business described as entities and relations.

Fabric IQ. The business-meaning layer for people and agents.

LIGHTS OUT

Before you close it

Foundry is where agents think.

Fabric is where the data they need lives.

Go to sleep.

TL;DR

Fabric is one SaaS analytics platform. Tenant, capacity, workspace, item.

OneLake is the single lake. Open formats. Every engine uses the same copy.

Shortcuts point to data elsewhere. Mirroring keeps databases in sync.

Lakehouse for Spark, warehouse for SQL, SQL database for apps, eventhouse for streams.

Direct Lake gives Power BI import speed without the copy.

You pay for capacity. Smoothing absorbs spikes. F64 and up lets free users view reports.

OneLake security sets the rule once. Purview labels travel with the data.

Copilot helps builders, data agents help everyone else, Fabric IQ gives data meaning.

Next time: a Fabric data agent and a Foundry agent working together, with the bill attached.

Sources: Microsoft Learn, Microsoft Fabric documentation (overview, OneLake, shortcuts, mirroring, licenses, throttling, Direct Lake, Real-Time Intelligence, Copilot in Fabric, data agents, Fabric IQ, planning, materialized lake views), pages updated between March and October 2026. Written in personal capacity. Features change often, check before quoting.