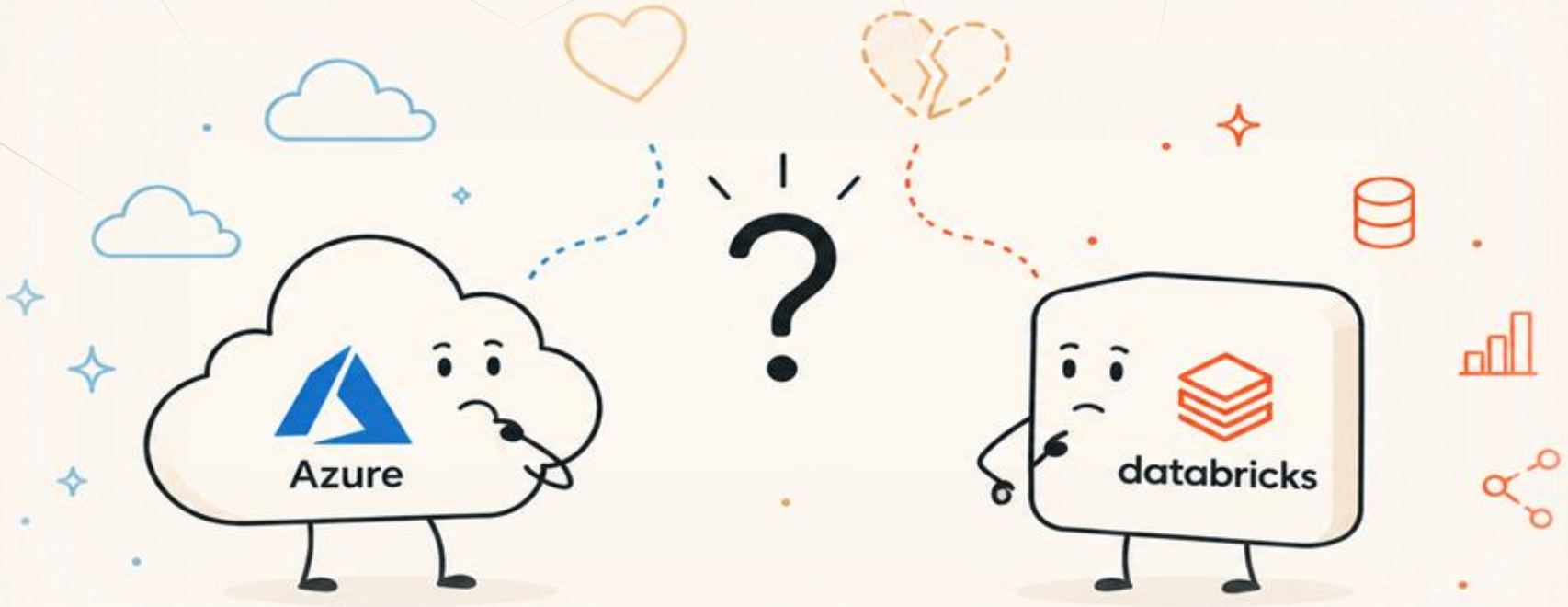
HOW DATABRICKS ACTUALLY INTEGRATES WITH YOUR AZURE ESTATE



JULIA FØRDE

- Senior Consultant, Databricks SME
- Databricks MVP
- Microsoft on-premises → Azure → Azure Databricks
- Focus: Lakehouse's & migrations
- Public speaking & LinkedIn

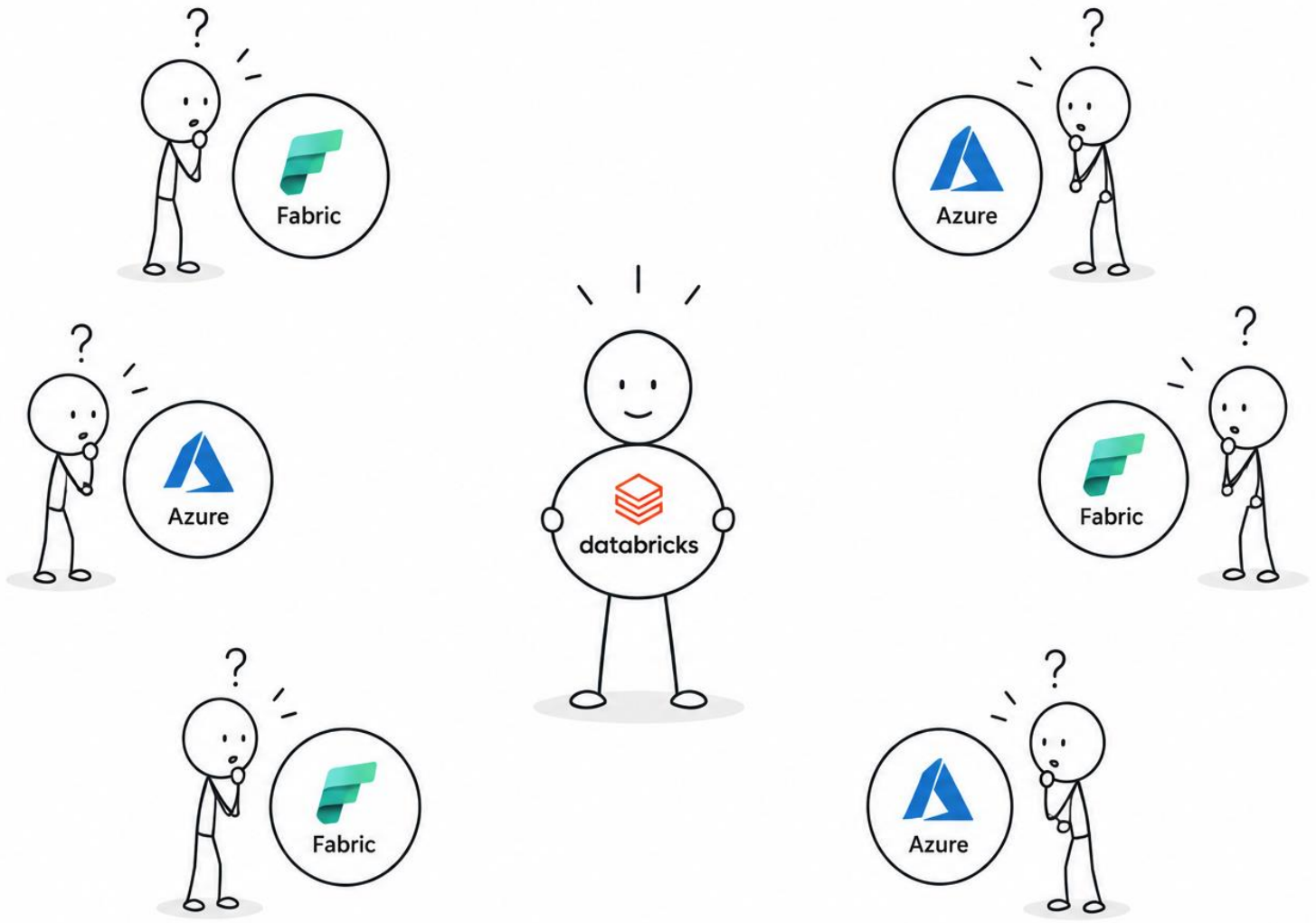


LET'S SET THE SCENE:

- Not worked with Databricks before?
- Mainly worked as a developer in Databricks?
- Has had administration responsibility of Databricks?



WHY THIS SESSION?



COMMON MISCONCEPTIONS

- Databricks is a standalone product in Azure
- Databricks integrates a lot worse with Azure than for example Fabric
- Databricks = setting subscription guardrails
- Sharing data is always hard
- Databricks does not fit with our Azure strategy



WHAT IS AZURE DATABRICKS

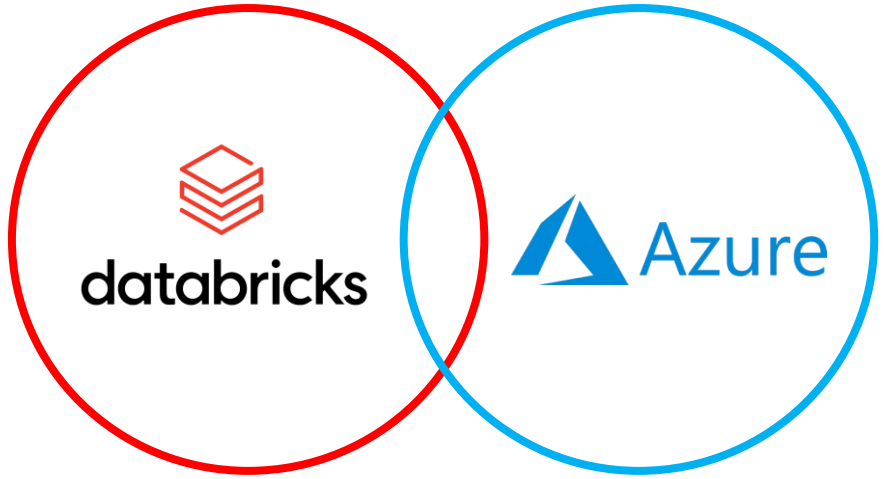
Independent Company
Databricks is a separate private equity company, founded 2014.

Cloud independent.
Runs on AWS, GCP & Azure

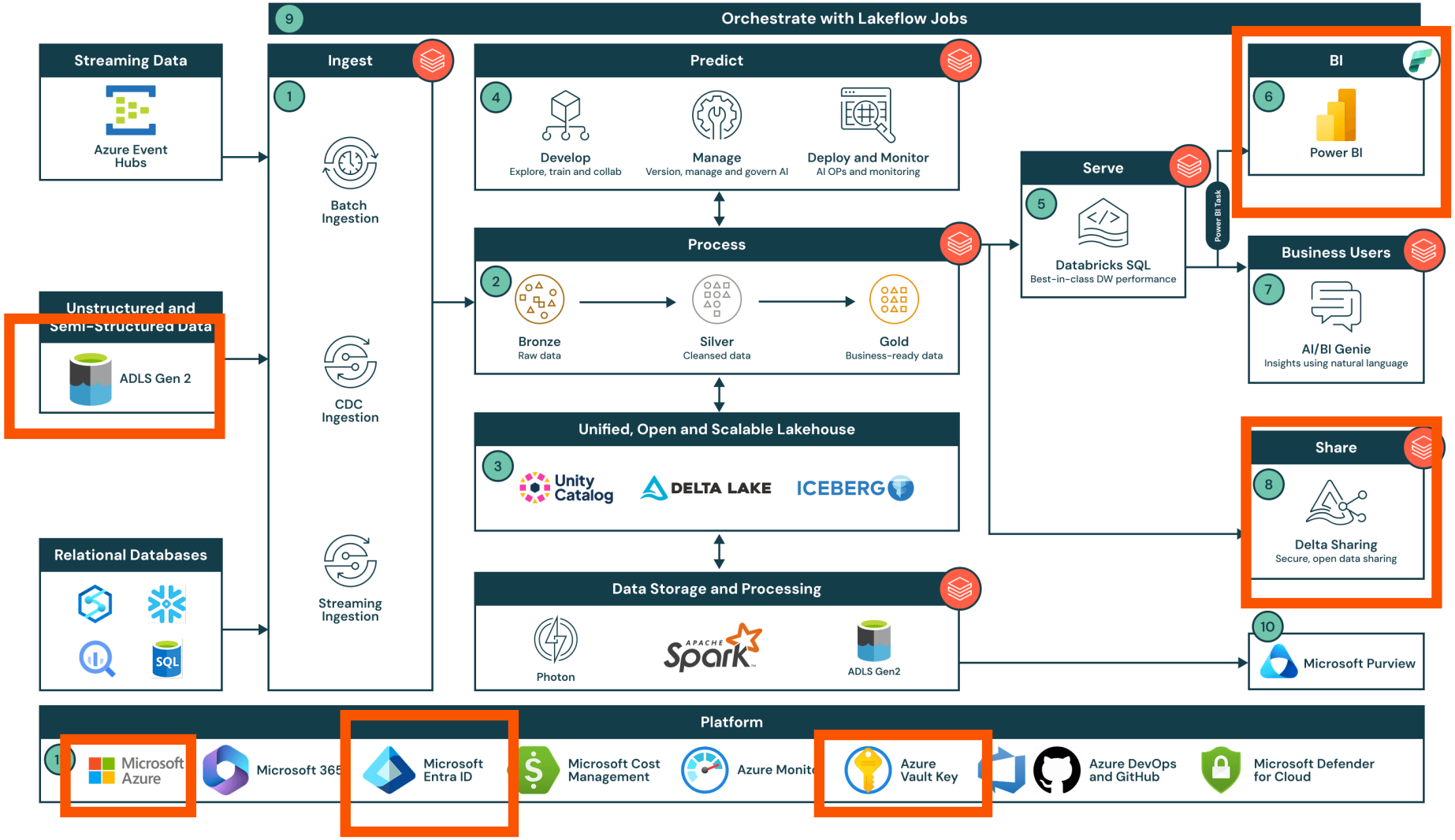
Mature Integration:
First party service since 2017

Core business:
Data warehousing, AI/ML, Delta sharing, Apps, BI

Vision:
Open-source is a clear priority

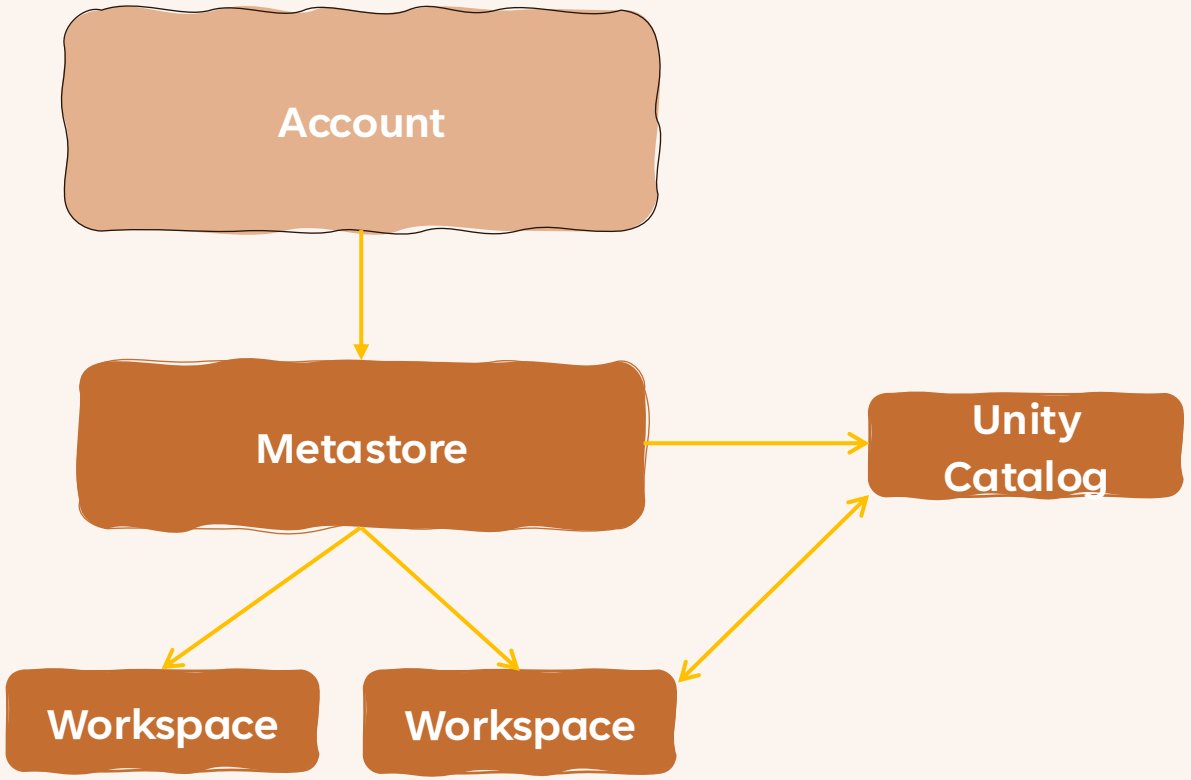


AZURE + DATABRICKS = TRUE



DATABRICKS ACCOUNT

- A workspace is assigned to one metastore
- The metastore contains catalogs
- A workspace can have read/ write access to 0 or many catalogs



ACCOUNT LEVEL

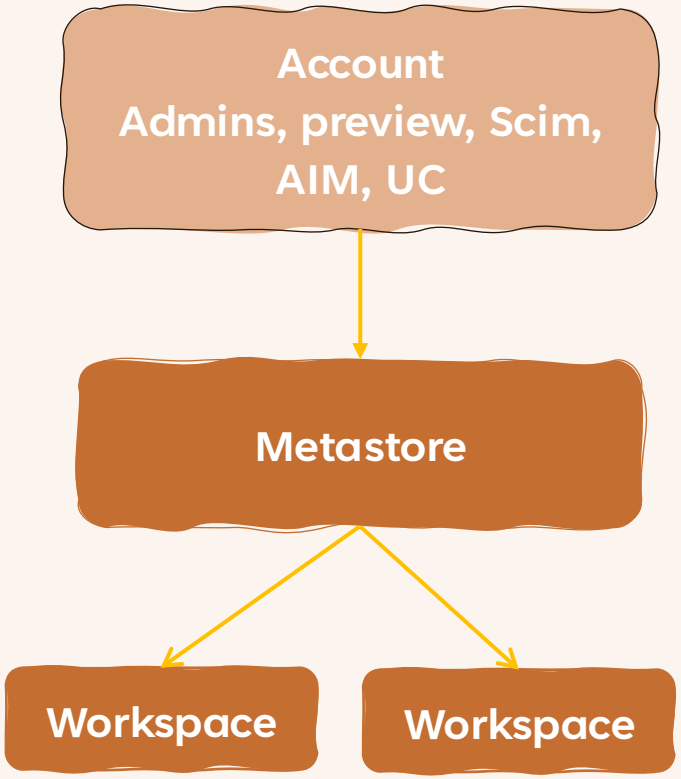
Purpose: Centralized control over billing, identity management (can be synced to Entra Id), global security settings and unity catalog enablement

Default admin: You global Azure Admin

Access: Account admins have visibility into **costs** and **usage** across all workspaces, enabling cross-team resource monitoring.

Risk: Misconfigurations here can lead to organization-wide data leaks.

Best Practice: Limit account admin access to trusted individuals that have extensive knowledge of Databricks



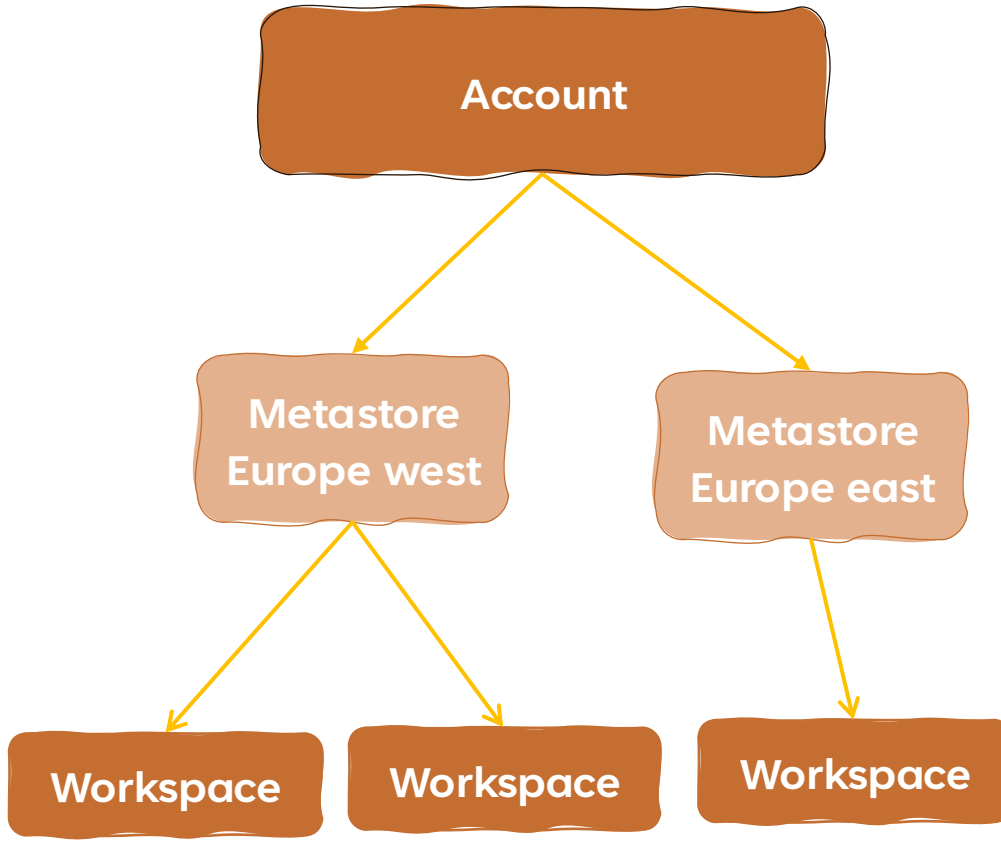
METASTORE LEVEL

Purpose: Handles where the metadata lives and is stored

Data Sharing: **Data sharing enablement** and permissions are controlled from here. The Metastore admin governs what workspaces are linked to each metastore.

Risk: Improper access control at the Metastore level can lead to **cross-team data leakage and give employees the possibility to share data externally.**

Best Practice: Assign dedicated **Metastore admins** to manage data sharing, access controls, and minimize cross-team exposure.



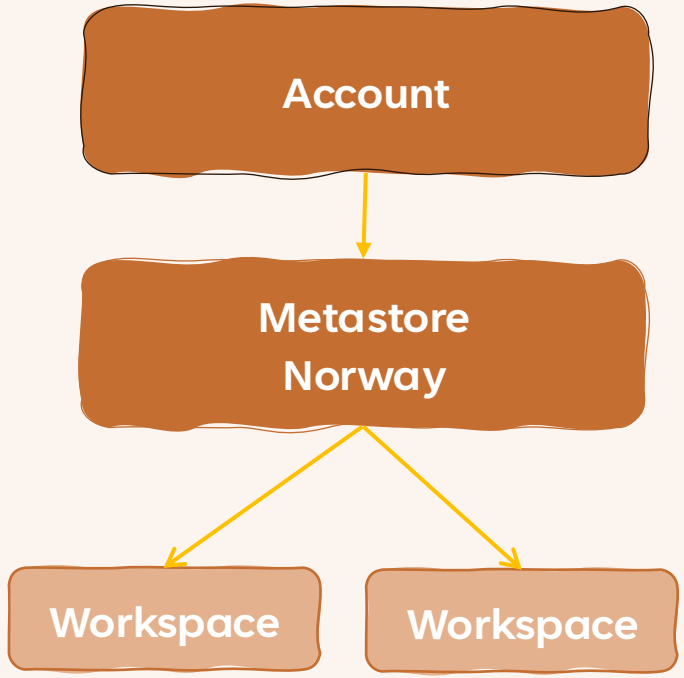
WORKSPACE LEVEL

- **Purpose: environment where** teams work on their data and models in isolation.

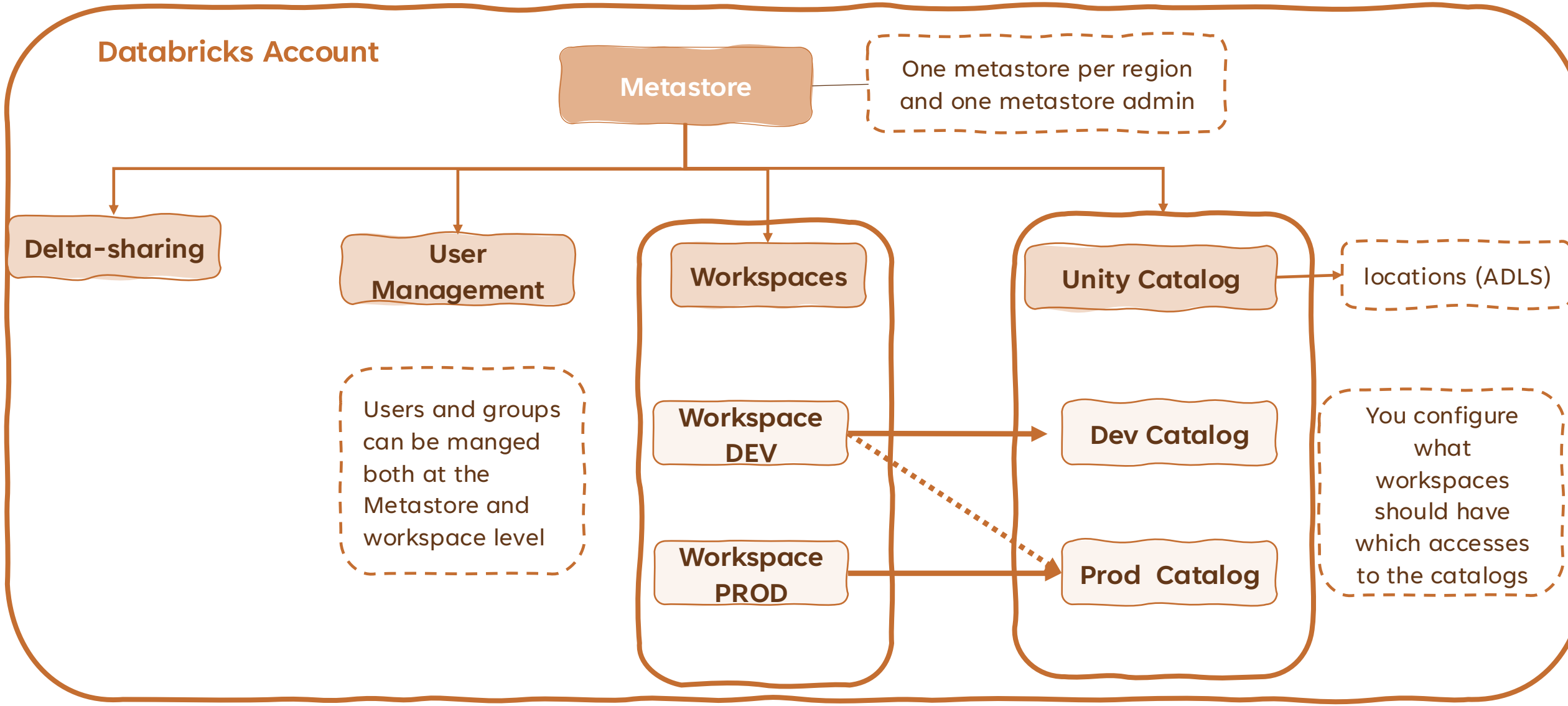
- **Risk**: **Internal data leaks** can occur if access to workspaces is not properly restricted. Teams may unintentionally expose data to unauthorized users

- **Best Practice**: Apply **RBAC** within each workspace to ensure users only have access to the necessary data and resources. Clearly define boundaries between teams' data and workspaces.

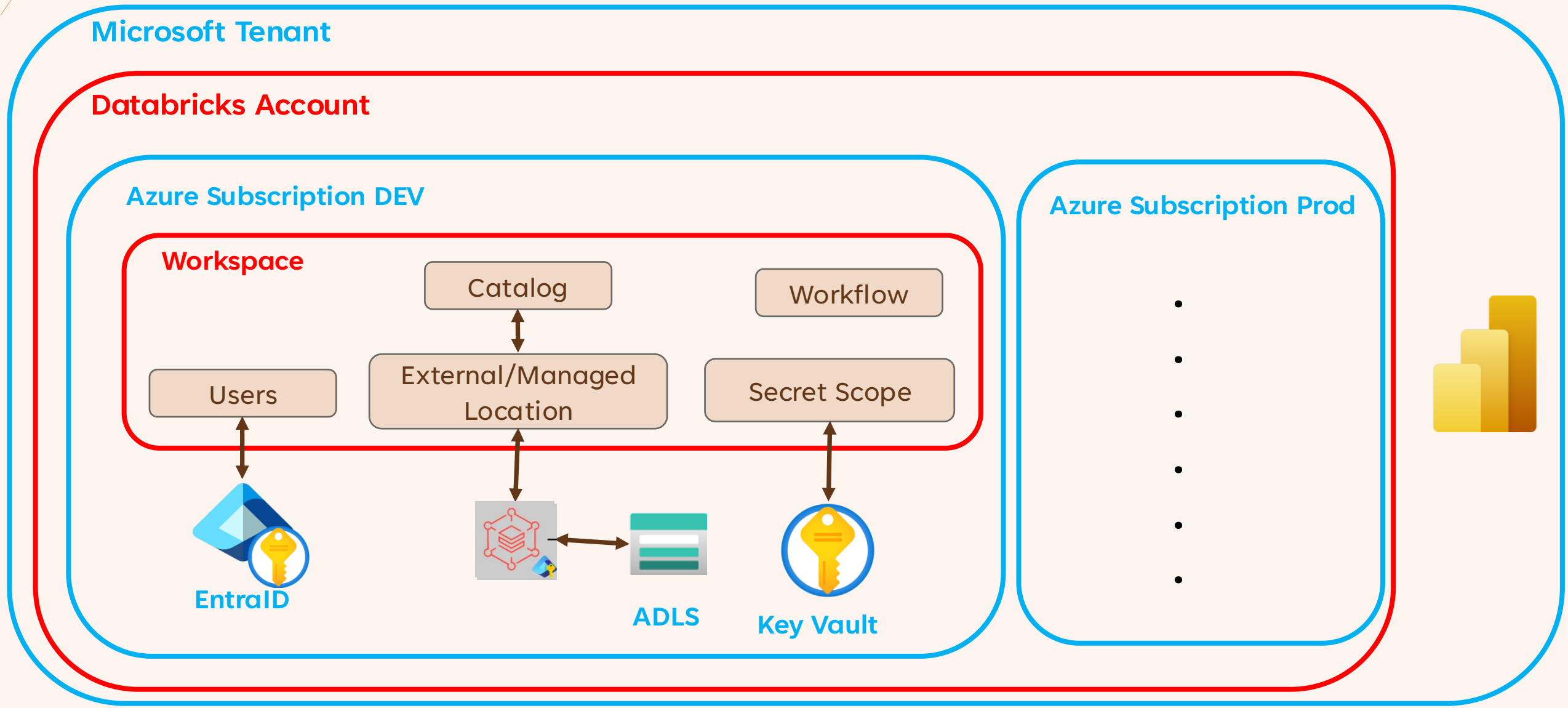
NB: Delta Sharing enabled = workspace admins can share data externally by default



TYPICAL ACCOUNT STRUCTURE



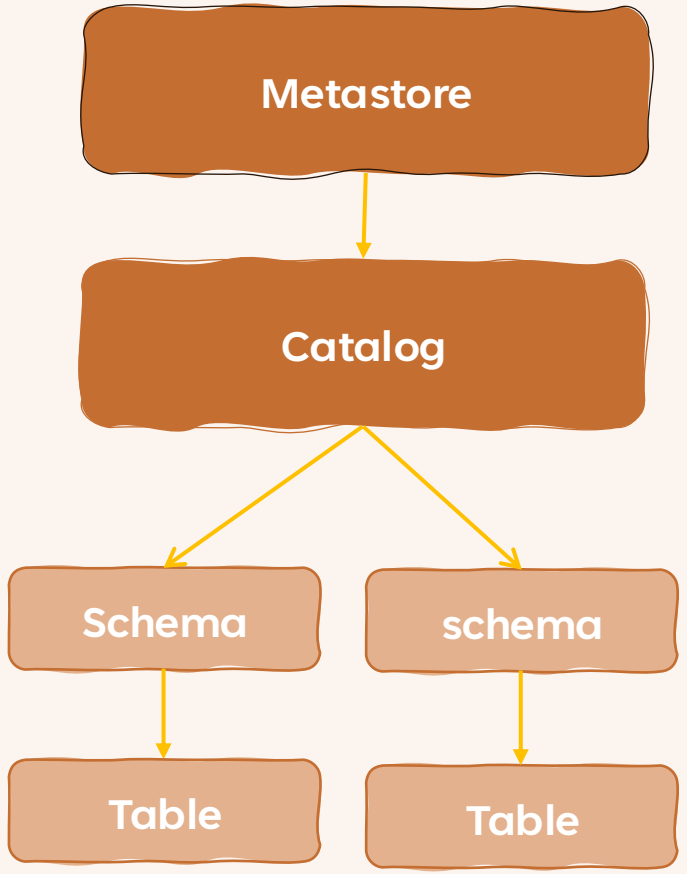
HOW DATABRICKS INTEGRATES WITH AZURE?



UNITY CATALOG – MORE THAN A CATALOG

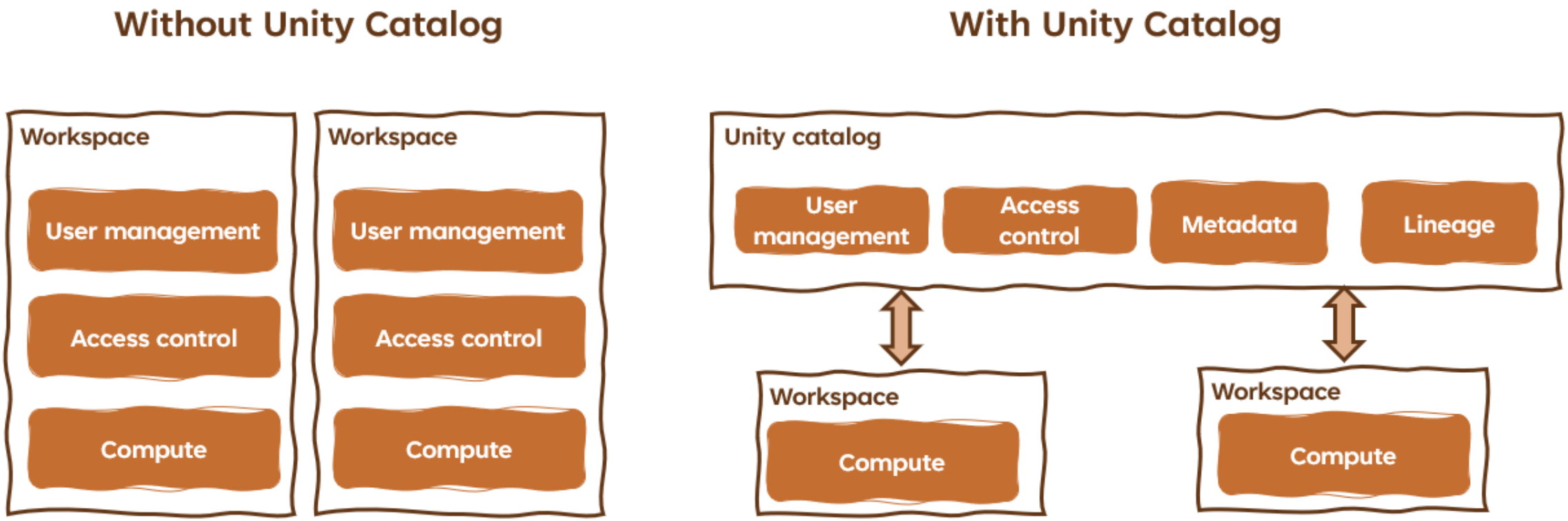
"A data catalog is a collection of metadata, combined with data management and search tools, that help data users find the data that they need."

- Column level lineage
- Metadata
- Documentation
- Sharing of data
- User management
- Masking
- Data Quality
- Metrics



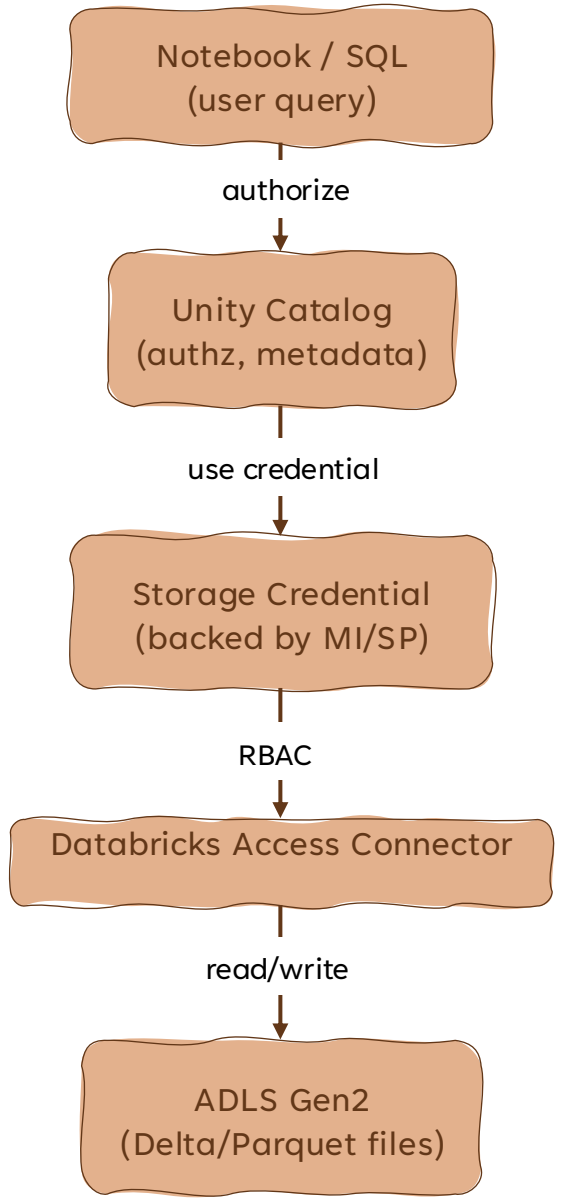
IF YOU HAVE HIVE, YOU SHOULD MIGRATE

- Hive = Governance per workspace → doesn´t scale
- No lineage, no centralized access control
- UC moves governance above the workspace layer



HOW UNITY CATALOG CONNECTS TO ADLS

- Your data is stored in ADLS Gen2
- UC is governance/metadata.
- UC reaches ADLS using the Databricks Access Connector (Managed Identity).
- That same identity is used for both:
  - Managed Locations (catalog/schema managed storage)
  - External Locations (path prefixes bound in UC)
- No secrets in notebooks → Store secrets in Azure Key Vault
- Use UC-enabled clusters only.



MANAGED vs EXTERNAL TABLES IN DATABRICKS

MANAGED
(Databricks decides the name of the path)

Unity Catalog
(Databricks)

You define the table name in Unity Catalog

- sales.gold.customers
- sales.silver.orders
- marketing.campaigns

Databricks translates to a managed location (GUID paths)

abfss://.../managed/

- 8f2a91ab-0c3e-4d99-8c1a-...
- c72d44ef-9b10-4b66-a2f3-...
- d1e9f6b2-7a11-4f0d-8e22-...
- ...

Physical storage path is system generated (GUIDs) and hard to navigate

DROP TABLE
Metadata + files are deleted

Metadata + Files →

EXTERNAL
(You decide the folder path)

You define the table name in Unity Catalog

- sales.gold.customers
- sales.silver.orders
- marketing.campaigns

You

You map the table to an external location (your path)

abfss://.../

- gold/customers/
- silver/orders/
- raw/campaigns/
- ...

You control the folder path and naming (easy to navigate)

DROP TABLE
Only metadata is removed

Metadata →

Files remain in ADLS

BOTH STORE DATA IN ADLS GEN2

ADLS Gen2 | Databricks Access Connector

Same storage. Same connector.

Same Unity Catalog

Same Databricks

Same storage

Difference is ownership

SERVICE PRINCIPAL VS MANAGED IDENTITY

Aspect	**Managed Identity (MI)**	**Service Principal (SP)**
What it is	Identity created & managed by Azure System or user assigned	App Registration in Entra ID, created manually with client ID/secret or cert.
Where it works	Valid in the same Azure AD tenant. Can be granted RBAC on storage in other subscriptions *within that tenant*. ❌ Not usable cross-tenant.	Recognized across all subscriptions in the tenant. ✓ Can also be federated/guest in other tenants.
Secrets	No secrets. Azure rotates credentials automatically.	Requires secret or cert, stored in Key Vault. You manage rotation.
Best fit	Default for most setups (workspace + ADLS in same tenant).	Needed for cross-tenant scenarios, or when org policy requires explicit app registrations.

We have seen how Unity Catalog connects to ADLS for data governance. But data governance isn’t just about storage, it’s also about **who gets access**.

IDENTITY & ACCESS – IF YOU DO NOTHING...

- Azure Contributor & Owner roles → automatically become workspace admins on first login

- Admins can:
  - Create/delete clusters (cost risk 💸).
  - Grant themselves catalog access (governance risk 🔒).
  - Bypass intended security boundaries.

- Catalog permissions must be **manually managed per user**, which doesn't scale.

IDENTITY & ACCESS – BEST PRACTICE

- SCIM (legacy):Needs Enterprise App in Entra ID. Admin must manually add groups → only those sync. Every 40 minutes
- AIM (since Sept):Direct Microsoft Graph integration. No Enterprise App or manual group selection. Real-time.
- Workspace admins assign workspace entitlements to synced Entra ID groups.
- Catalog, schema, and table permissions are assigned inside the workspace via Unity Catalog.
- Keep the number of workspace admins minimal.
- Regularly audit Entra ID groups and Databricks entitlements.

Controlling compute access

Entitlements:

Who has access to what?

- ***Consumer*** *- Dashboard, genie, apps*
- ***SQL*** *– SQL Warehouse, queries, dashboards*
- ***Workspace*** *– notebook, jobs, pipelines*
- ***Cluster Creation*** *– Admin by default*

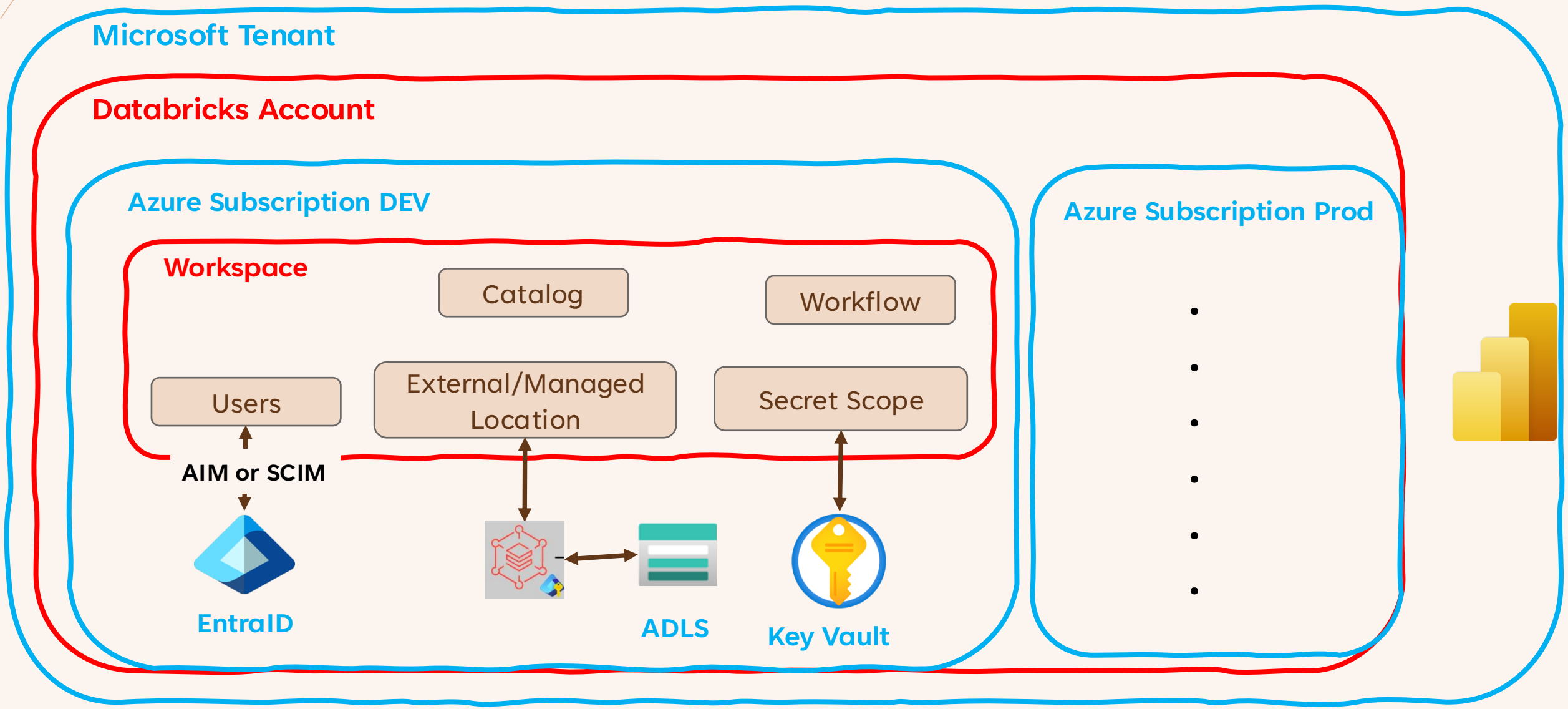
Cluster policies:

What can be created – classical compute?

- *Fix or restric node types*
- *Set Max DBUs per hour*
- *Enforce Autoscaling*
- *Enforce tags for chargeback*

For Serverless: Cluster policies don´t apply, to alert and tag serverless spending use budget policies

RECAP OF HOW DOES DATABRICKS INTEGRATE WITH AZURE?



But wait — if workspaces can span subscriptions, where is the actual data boundary?

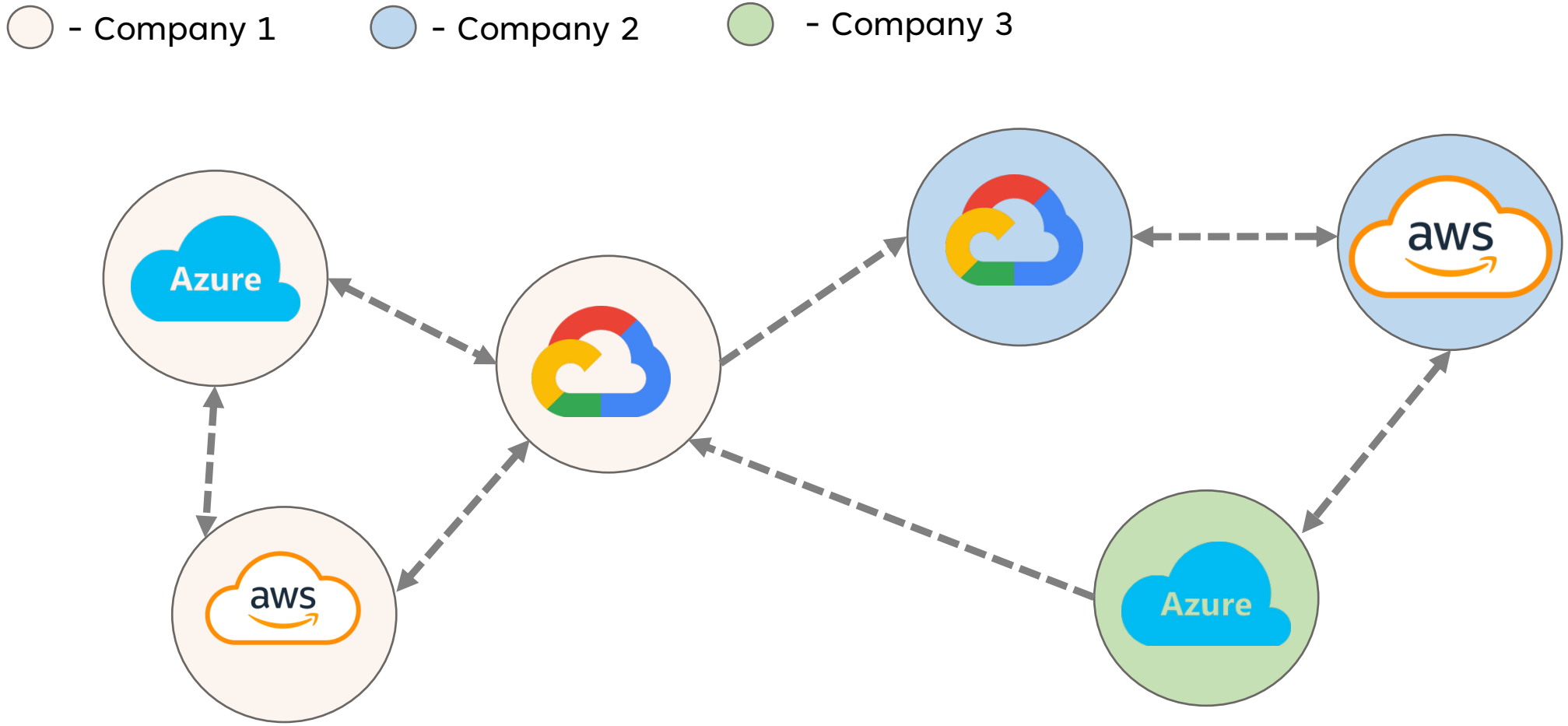
WHERE IS THE REAL BOUNDARY?

- **Catalog** = unit of governance inside a metastore.
- **Metastore = true technical boundary** for sharing.
- Subscriptions are not a boundary for accessing data inside a metastore.
- Workspaces in the same Metastore can directly access shared catalogs (even across subscriptions).
- Different metastores (even in the same tenant) → require Delta Sharing.

So what if I want to move my data out of my Metastore?

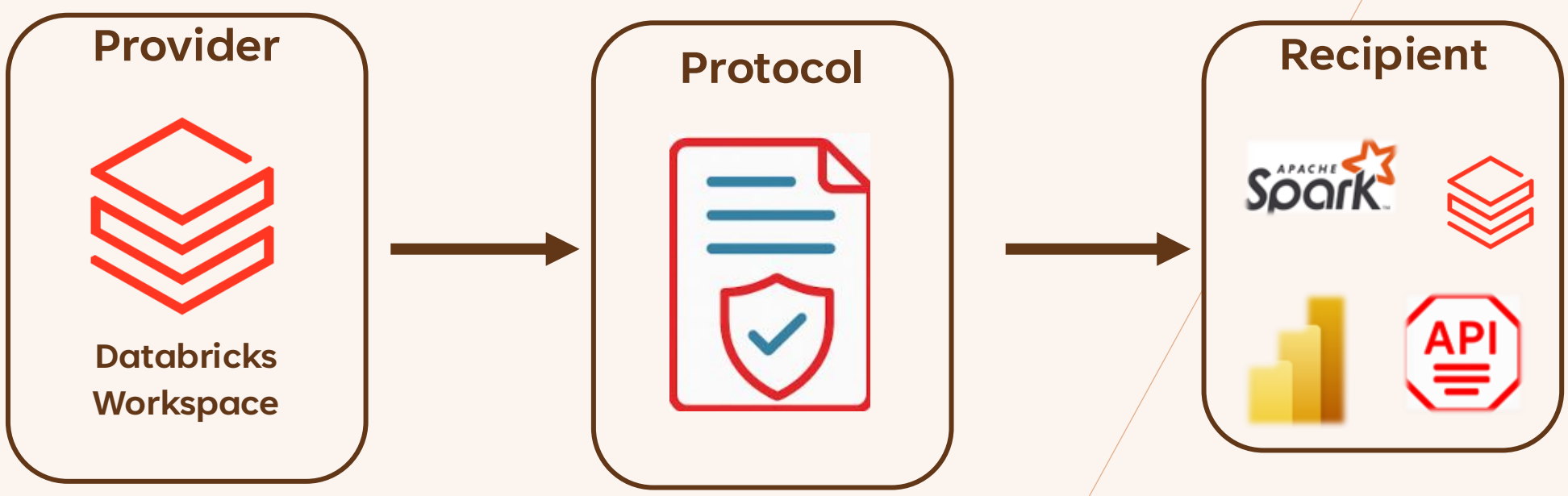
MOVING DATA IS HARD..

“Data is fragmented—across clouds, systems, and partners. Sharing it is hard.



WHAT IS DELTA SHARING?

Delta Sharing *is an open protocol that lets you securely share live data from Delta Lake with others, without moving or copying it.*

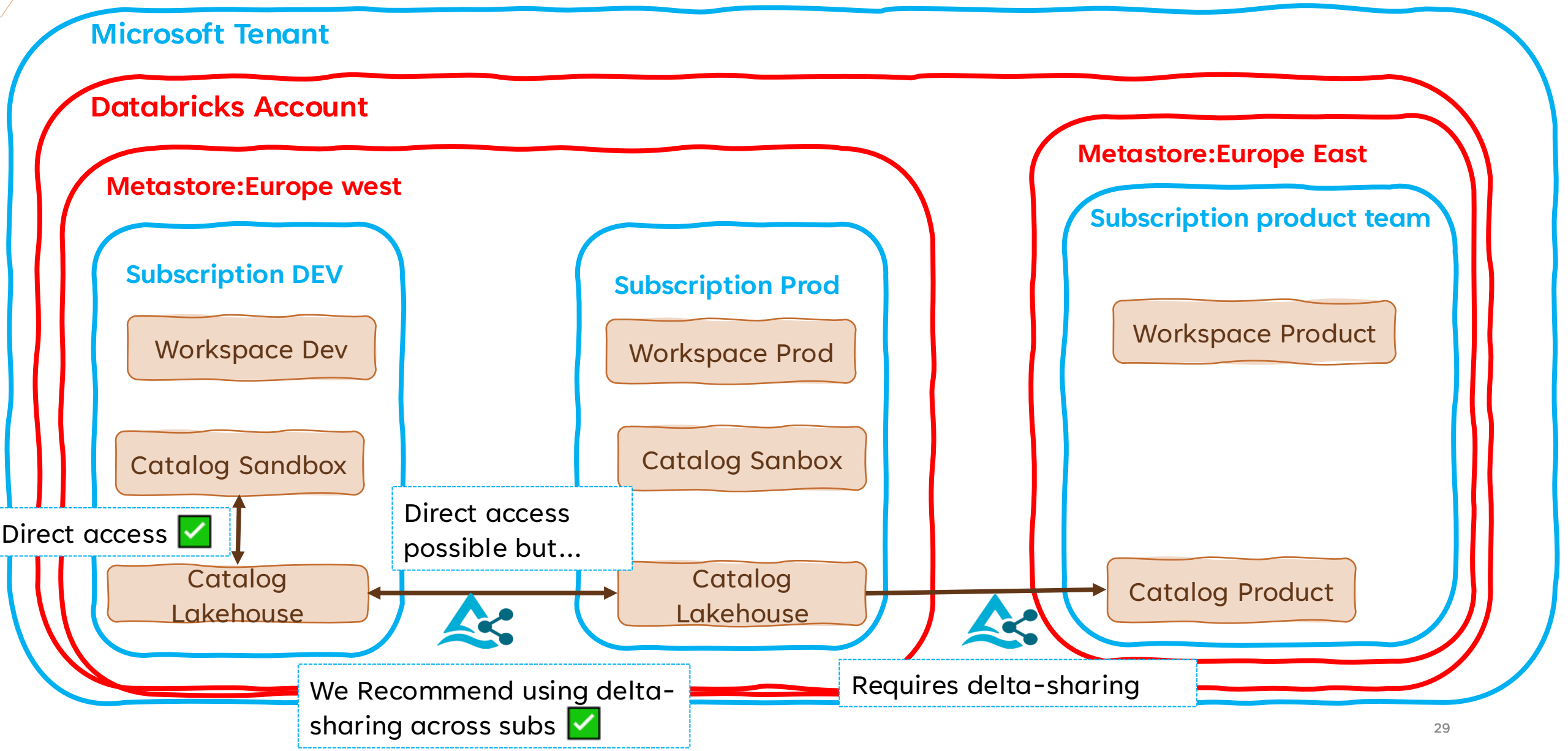


Works **across workspaces, accounts, tenants, clouds.**

DELTA SHARING OPTIONS IN DATABRICKS

Option	What it's for
🧑💻 **Notebooks & UI**	Internal sharing & collaboration
🧊 **Clean Rooms**	Private, secure external collaboration
🧩 **Federation**	Query live-data across sources
🛒 **Marketplace**	Share or monetize data products
🔌 **APIs & Ext. Tables**	Integrate with external tools
💬 **SQL & Spark**	Analyze shared data at scale

BEST PRACTICE SHARING DATA



We looked at goverance and sharing what about BI & Power BI ?

DATABRICKS ← → POWER BI

- **Direct Lakehouse (SQL Endpoints) → live query from Power BI**
  - DirectQuery → near real-time, query hits Databricks SQL
  - Import → better performance, good for curated gold data

- **Delta Sharing Connector** → secure, zero-copy access to UC tables.

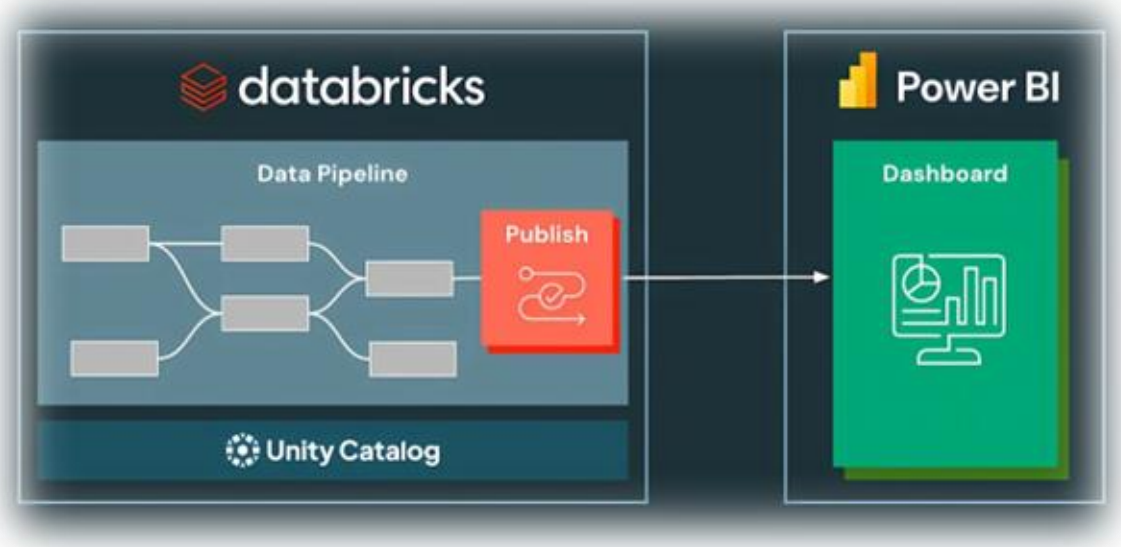
- Mirror Unity Catalog tables to Onelake, then connect to Power BI

- Common pattern: Databricks bronze/silver → Fabric gold → Power BI
- **Import vs Direct Query:**
  - *Import* → performance, good for curated gold data.
  - *Direct Query* → near real-time.

✅**Best practice:** Use Delta Sharing for cross-workspace/tenant scenarios.

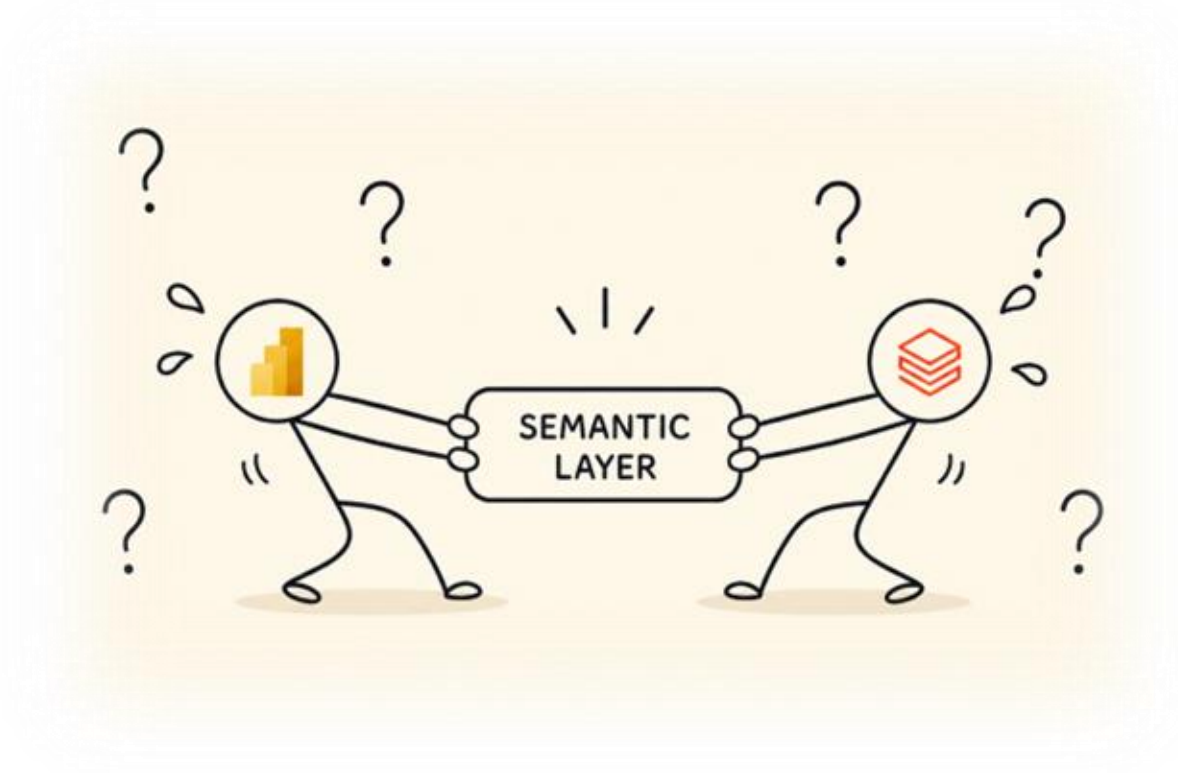
REFRESHING POWER BI

- **Two refresh patterns:**
  - *Scheduled refresh* in Power BI Service (simple, time-based)
  - *Triggered refresh* from Databricks jobs (after pipeline finishes)
- **Triggered refresh ensures data freshness & avoids delays**
- Use a **Service Principal** with Power BI REST API for triggering
- Store SP secret in **Key Vault → Databricks secret scope**.



THE WAR OVER THE SEMANTIC LAYER

- Databricks has entered the semantic layer market
- Metric View: Used across your Databricks Estate
- Not same level as PBI, but wider use case
- Both fight over ”holding” the semantic layer



THE CURRENT ”DRAMA”...

- Many requests a way to connect metric views to Power BI
- Databrick & Microsoft launched a connector, **YAY!**
- The first version was clunky
- DEPRECATED </3

SO WHAT TO DO?

- No native connector
- Testing Metric Views + keeping Power BI? → **Tabular Editor**

  Semantic Bridge
- Is is in early version, test properly

GETTING IT RIGHT FROM THE START

1. **Account & Metastore** — one per region, pick admins carefully
2. **Identity** — AIM + Entra ID groups, never individuals
3. **Storage** — Access Connector → ADLS, external / managed?
4. **Workspaces** — dev/prod split, cluster policies before access
5. **Sharing** — same metastore = direct, cross-metastore = Delta Sharing

KEY TAKEAWAYS

- Make sure you choose your Account & Metastore admin wisely they have permission across workspaces and subscriptions
- Identity matters → AIM (GA) is now the recommended way to sync Entra ID groups
- Storage access → prefer Managed Identity (Access Connector), use SPs only for cross-tenant
- Sharing data → Delta Sharing across metastores/tenants
- Governance first → assign permissions to groups, not individuals &minimize workspace admins
- Use the Databricks Acces connector to read and write to your ADLS

THANK YOU !

