#FABCONSQLCON2026

# FABCON
## Microsoft Fabric
COMMUNITY CONFERENCE

# SQLCON
## Microsoft SQL
COMMUNITY CONFERENCE

**ATLANTA** MARCH 16 - 20, 2026

100+ workspaces.

16 per subject area.

$90K to rebuild views for a migration they didn't need.

*— r/MicrosoftFabric, r/PowerBI*

# Proven Fabric Architecture Patterns to Minimize Refactoring

Matthias Falland

# About Me

Matthias Falland

Microsoft Data Platform MVP

@TheTrustedAdvisor

# Agenda

1 Workspace Topology
Why topology decisions in Week 1 become permanent infrastructure

2 Medallion Layers
Which implementation choice creates an invisible performance cliff

3 Cross-Workspace Sharing
Why 'zero-copy' doesn't mean 'zero-cost'

+ Hidden Traps · Decision Framework · Resources

# Design Decisions, Not Button Clicks

This session is about architecture patterns, not product demos.

Understanding WHY prevents the mistakes that cause expensive refactoring.

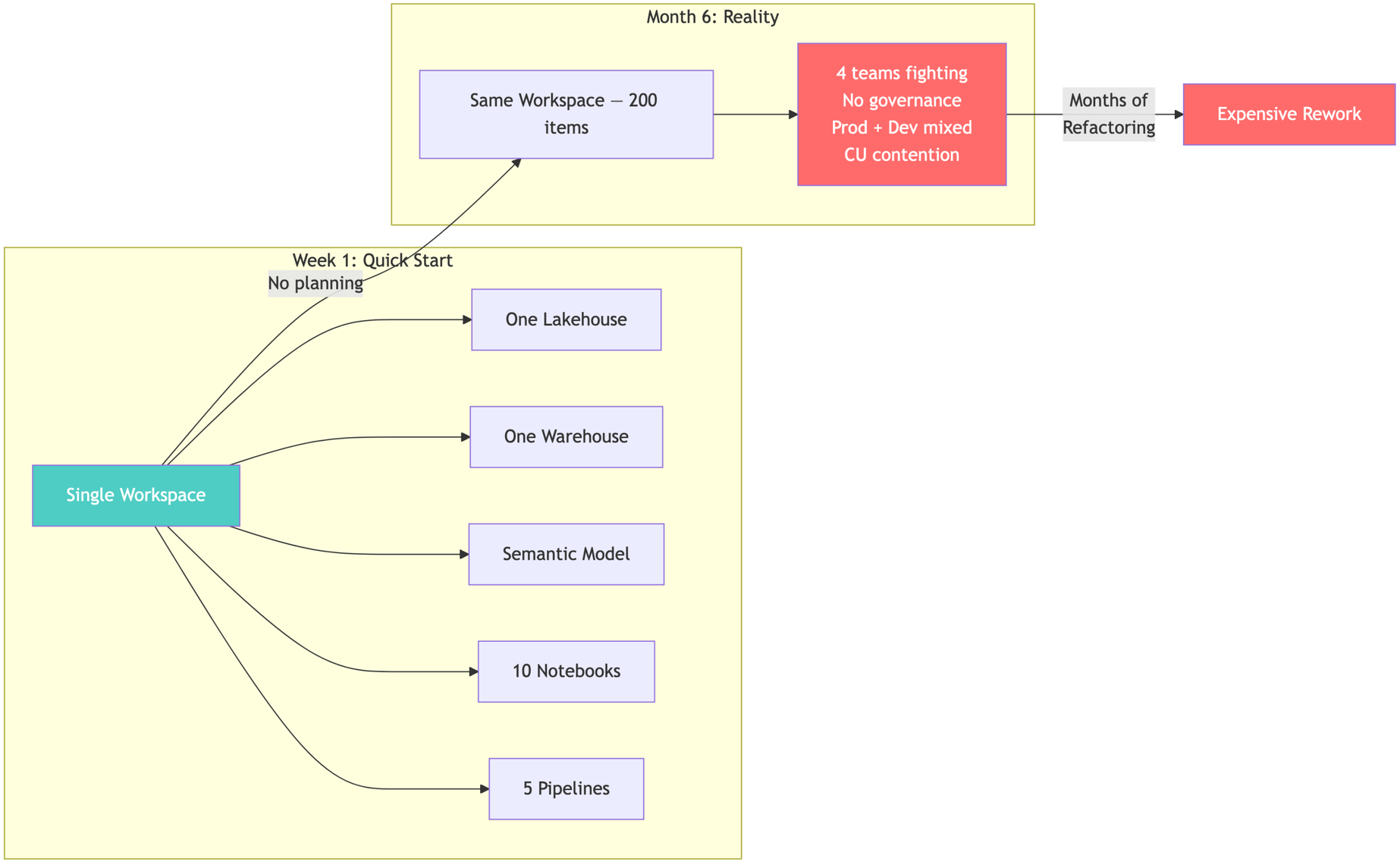
We're here for the WHY, not the HOW.

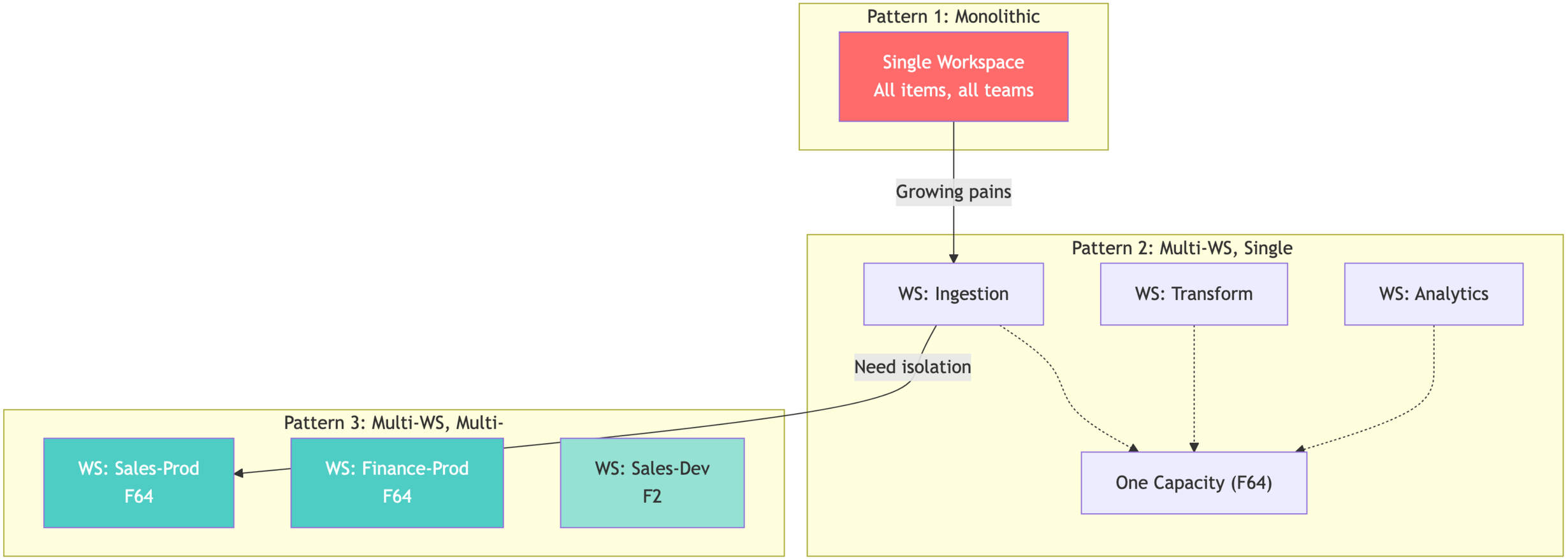
# The Problem: Week 1 vs Month 6

WEEK 1:
Clean workspace. 5 items. 2 team members. Everything works.

MONTH 6:
200+ items. 5 teams sharing one workspace. No isolation.
Junior dev can delete production pipelines.

Result: Months of expensive rework.
Workspaces are the PRIMARY boundary — security, capacity, governance, deployment.


Month 6: Reality
Same Workspace — 200 items
4 teams fighting
No governance
Prod + Dev mixed
CU contention
Months of Refactoring
Expensive Rework
Week 1: Quick Start
No planning
Single Workspace
One Lakehouse
One Warehouse
Semantic Model
10 Notebooks
5 Pipelines


Pattern 1: Monolithic
Single Workspace
All items, all teams
Growing pains
Pattern 2: Multi-WS, Single
WS: Ingestion
WS: Transform
WS: Analytics
Need isolation
One Capacity (F64)
Pattern 3: Multi-WS, Multi-
WS: Sales-Prod
F64
WS: Finance-Prod
F64
WS: Sales-Dev
F2

# The Compounding Problem

Workspaces = Immutable Boundaries
*Workspace moves break shortcuts, permissions, pipelines, lineage*

Shortcuts = Permanent Dependencies
*Shortcuts create architectural contracts between workspaces*

Capacity = Architectural, Not Operational
*Capacity allocation determines who gets throttled when*

In Fabric, architecture decisions compound.

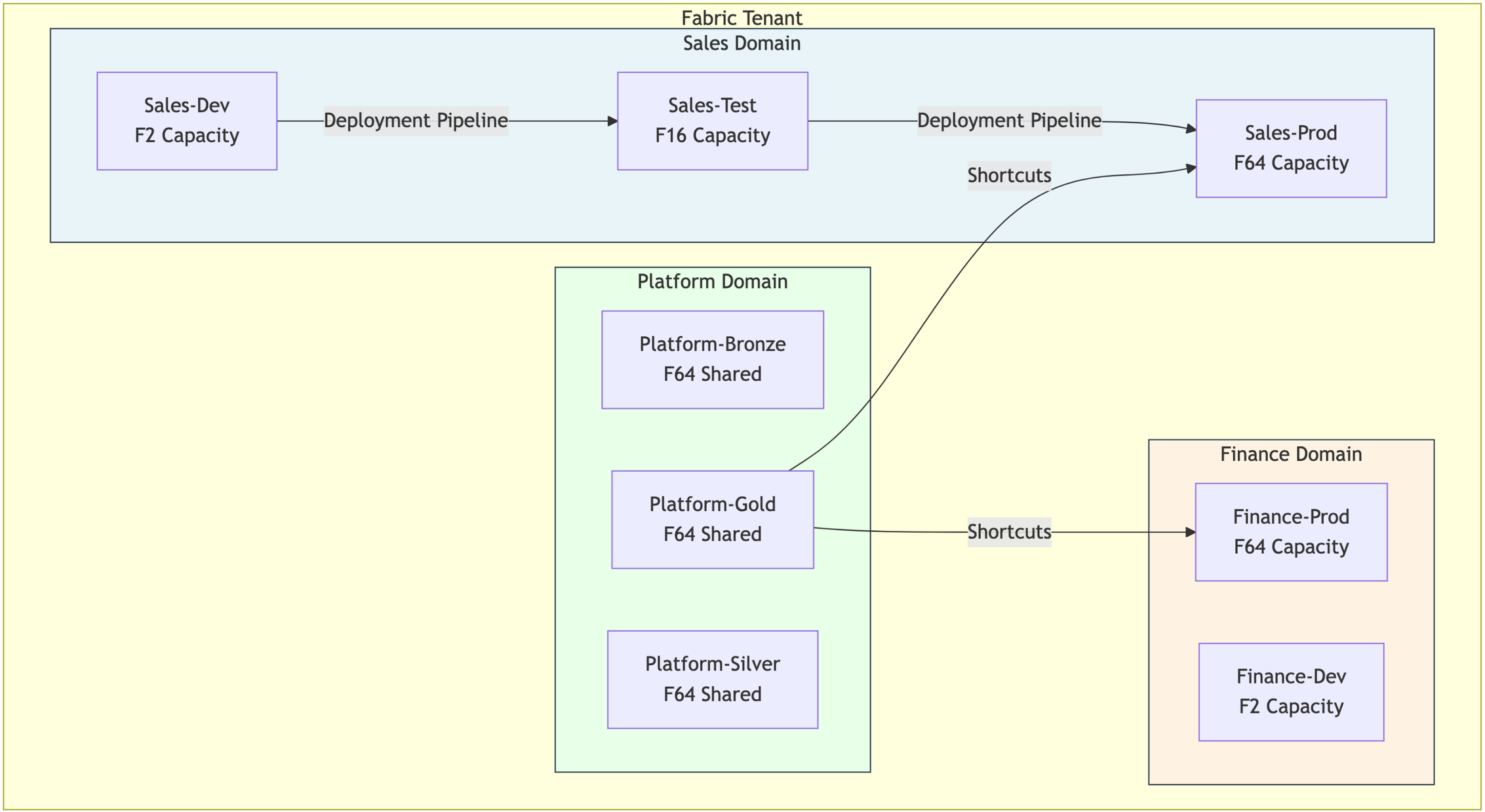
*A workspace choice affects capacity isolation, which affects Direct Lake performance, which affects whether your dashboard works on Monday morning.*

# Four Deployment Patterns: Combined, Not Chosen

Pattern 1: Single Workspace (POC)
Pattern 2: Multi-Workspace, Single Capacity (Development)
Pattern 3: Multi-Workspace, Multi-Capacity (Production)
Pattern 4: Multi-WS, Multi-Cap + Domains (Enterprise)

The insight: you COMBINE them, not choose one.

POC: Pattern 1 → Dev: Pattern 2 → Prod: Pattern 3 → Enterprise: Pattern 4

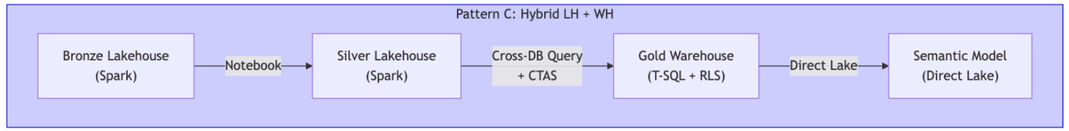

Fabric Tenant
Sales Domain
Sales-Dev
F2 Capacity
Deployment Pipeline
Sales-Test
F16 Capacity
Deployment Pipeline
Sales-Prod
F64 Capacity
Shortcuts
Platform Domain
Platform-Bronze
F64 Shared
Platform-Gold
F64 Shared
Shortcuts
Platform-Silver
F64 Shared
Finance Domain
Finance-Prod
F64 Capacity
Finance-Dev
F2 Capacity

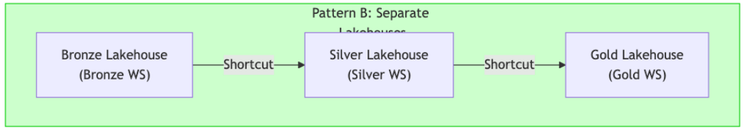
# Enterprise Pattern: Domain + Environment Matrix

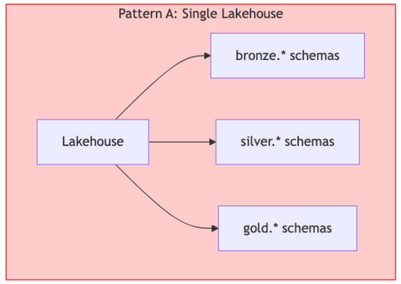
Sales Domain: Dev → Test → Prod
Finance Domain: Dev → Prod
Platform Domain: Bronze → Silver → Gold

Deployment pipeline arrows between environments
Shortcut arrows from Platform-Gold to domain workspaces

Domain + Environment = the pattern that works at every scale
From 5-team startups to 50-office governments


Pattern C: Hybrid LH + WH
Bronze Lakehouse (Spark)
Notebook
Silver Lakehouse (Spark)
Cross-DB Query + CTAS
Gold Warehouse (T-SQL + RLS)
Direct Lake
Semantic Model (Direct Lake)


Pattern B: Separate
Bronze Lakehouse (Bronze WS)
Shortcut
Silver Lakehouse (Silver WS)
Shortcut
Gold Lakehouse (Gold WS)


Pattern A: Single Lakehouse
Lakehouse
bronze.* schemas
silver.* schemas
gold.* schemas

# Naming Convention

{Domain}-{Environment}-{Purpose}

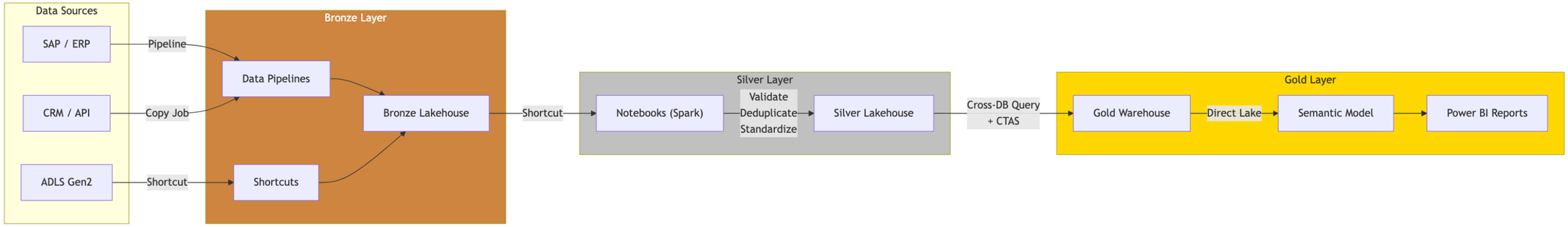
Examples:
Sales-Prod-Analytics / Finance-Dev-Ingestion / Platform-Prod-MasterData

Three questions naming answers:
1. Which workspace do I deploy to?
2. Which workspace did this 3 AM alert come from?
3. Which workspaces can I clean up?

A naming convention prevents more refactoring than any other single architectural decision.

# Workspace Topology Decision Matrix

| Teams | Data Sensitivity | Recommended Pattern |
|---|---|---|
| 1–3 | Low | Multi-WS, Single Capacity |
| 3–10 | Department-level | Multi-WS, Multi-Capacity |
| 10+ | Regulated / High | Multi-WS, Multi-Cap + Domains |
| SaaS / Multi-tenant | Customer isolation | Tenant-per-WS within Domains |


Data Sources
SAP / ERP
CRM / API
ADLS Gen2
Pipeline
Copy Job
Shortcut
Bronze Layer
Data Pipelines
Shortcuts
Bronze Lakehouse
Shortcut
Silver Layer
Notebooks (Spark)
Validate
Deduplicate
Standardize
Silver Lakehouse
Cross-DB Query
+ CTAS
Gold Layer
Gold Warehouse
Direct Lake
Semantic Model
Power BI Reports

**You can always split later.**
**You cannot easily merge.**

The management overhead of extra workspaces is far less than the refactoring cost of splitting a monolithic workspace.

# The Problem: Folders ≠ Architecture

One Lakehouse with /bronze, /silver, /gold folders
This is NOT medallion architecture.

No security boundaries between layers
Bronze steals capacity from Gold queries
Can't share Gold without exposing Bronze

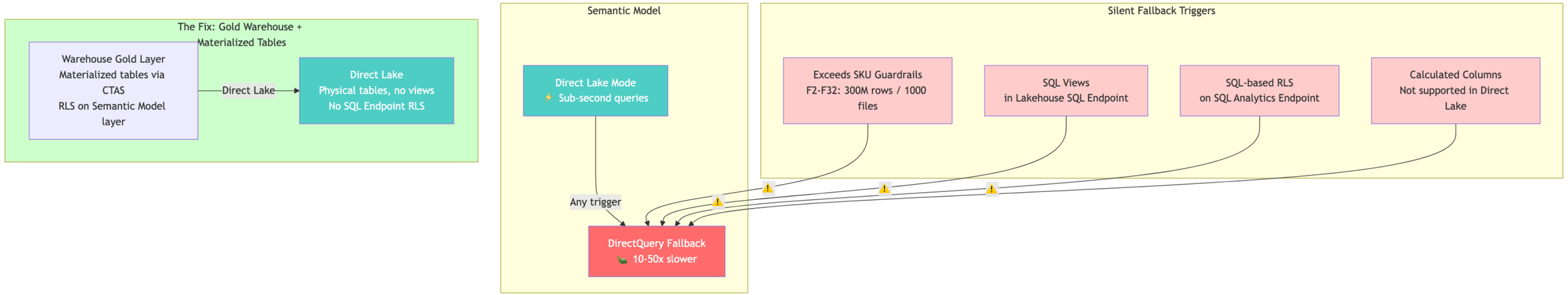
The folders were just … folders.

# The Implementation Choice That Costs You Later

WHERE do you put each layer? This one decision determines:

Security boundaries - Who can access what?
Performance isolation - Who competes for capacity?
Direct Lake behavior - Does it read Delta files … or silently fall back to DirectQuery?

Does your semantic model read directly from Delta files… or does it silently fall back to DirectQuery because of a design choice you made three months ago?


The Fix: Gold Warehouse + Materialized Tables
Warehouse Gold Layer
Materialized tables via CTAS
RLS on Semantic Model layer
Direct Lake
Direct Lake
Physical tables, no views
No SQL Endpoint RLS
Semantic Model
Direct Lake Mode
⚡ Sub-second queries
Any trigger
DirectQuery Fallback
🐌 10-50x slower
Silent Fallback Triggers
Exceeds SKU Guardrails
F2-F32: 300M rows / 1000 files
SQL Views
in Lakehouse SQL Endpoint
SQL-based RLS
on SQL Analytics Endpoint
Calculated Columns
Not supported in Direct Lake
⚠️
⚠️
⚠️
⚠️

# Three Medallion Implementation Patterns

Pattern A: Single Lakehouse + Schemas
Fastest setup, but no isolation between layers

Pattern B: Separate Lakehouses Per Layer
MS Learn recommended — independent security and capacity

Pattern C: Hybrid Lakehouse + Warehouse
Materialized Gold tables via CTAS, avoids Direct Lake fallback

# Pattern A: Single Lakehouse with Schemas

Single Lakehouse containing bronze.*, silver.*, gold.* schemas

Pros:
- Fastest setup, single workspace, one item

Cons:
- One set of permissions for all layers
- No capacity isolation
- Bronze contention impacts Gold queries

When to use: POC, single team, small data volumes only

# Pattern B: Separate Lakehouses Per Layer

Three separate Lakehouses in separate workspaces, connected by shortcuts

MS Learn: "We recommend that you create each lakehouse in its own, separate workspace."

- Independent security per layer
- Independent capacity allocation
- Share Gold without exposing Bronze
- Scale layers independently

Recommended for production workloads

# Pattern C: Hybrid LH + WH (The Direct Lake Trap)

Bronze LH → Silver LH → cross-database query + CTAS → Gold Warehouse
→ Direct Lake → Semantic Model (RLS here)

Direct Lake has SKU guardrails. Exceed them and it SILENTLY falls back to DirectQuery. 10-50x slower. No error. No warning.

F2–F32: 300M rows / 1,000 files max
F64+: 1.5B rows / unlimited

Fallback Triggers: SQL views | SQL-based RLS | Calculated columns | Exceeding guardrails

The Fix: Warehouse materializes Gold via CTAS. RLS on semantic model layer.
Zero fallback triggers

1 Layer separation at the workspace level — not for governance alone, but for capacity isolation and preventing the Direct Lake fallback cascade.

2 Silver is your most important layer. It's your schema contract, your blast radius boundary, your published API. If you reorganize Silver, every Gold workspace breaks.

# The Problem: Five Copies of Customer Data

Marketing needs Sales' customer data → CSV export → import
Finance needs same → another copy

5 copies of customer data — none of them are in sync
Days or weeks of staleness

Root cause: they didn't know about OneLake shortcuts

Every copy is a governance liability.
Every copy is storage cost.
Every copy is a GDPR risk.

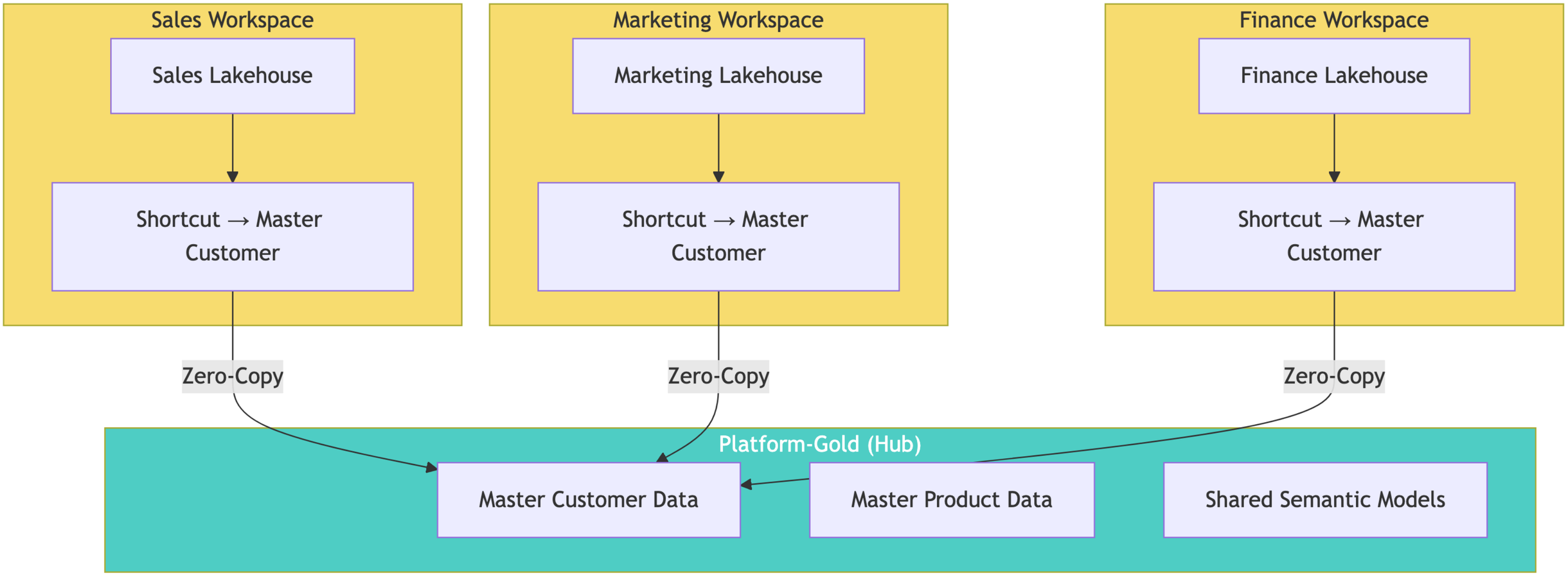
# The Cost Model Nobody Talks About

Who pays the CUs when using shortcuts?

The CONSUMER pays (Workspace B), not the source (Workspace A)

Hub-and-spoke: the hub barely registers reads — each spoke pays
Paused source capacity: data still accessible via shortcuts
OneLake storage persists independently of compute

"Zero-copy" doesn't mean "zero-cost".
It means the CONSUMER always pays.

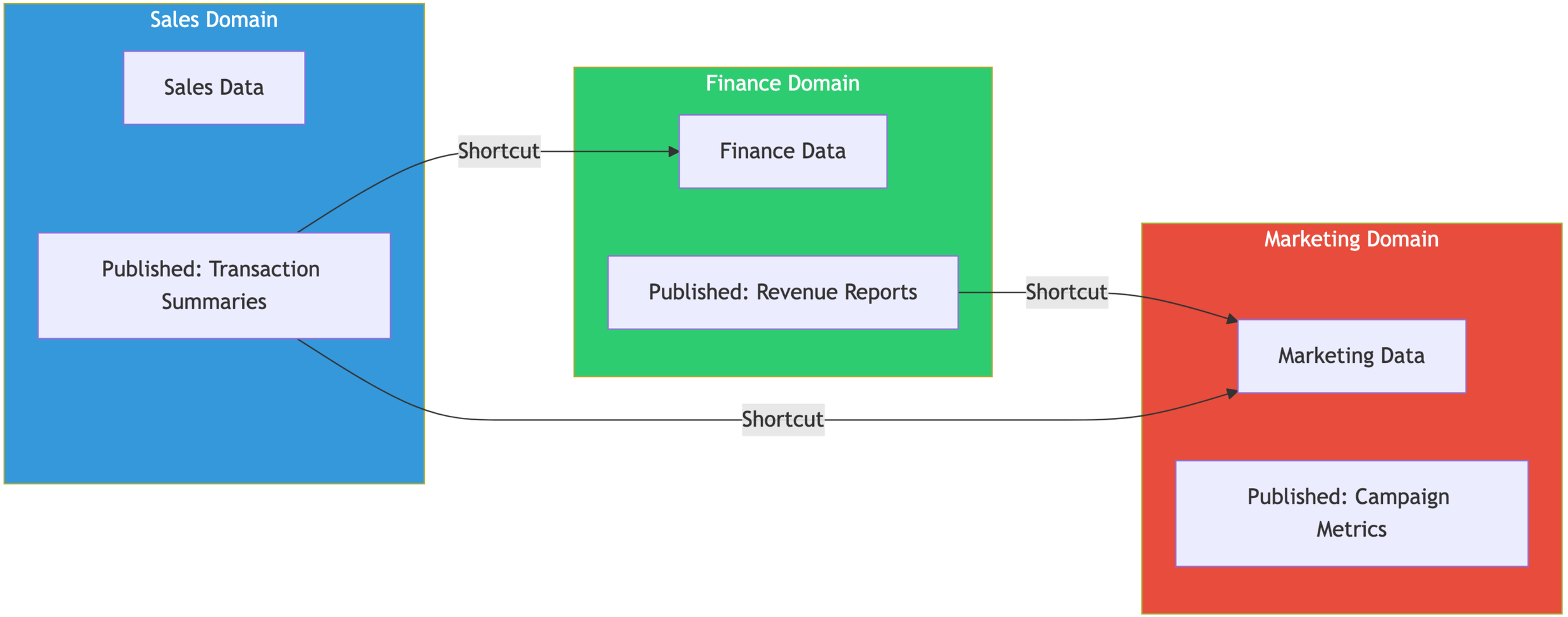

Sales Workspace
Sales Lakehouse
Shortcut → Master Customer
Marketing Workspace
Marketing Lakehouse
Shortcut → Master Customer
Finance Workspace
Finance Lakehouse
Shortcut → Master Customer
Zero-Copy
Zero-Copy
Zero-Copy
Platform-Gold (Hub)
Master Customer Data
Master Product Data
Shared Semantic Models

# Sharing Pattern 1: Hub-and-Spoke

Central "Platform-Gold" hub with master data
Spoke workspaces: Sales, Marketing, Finance
Each spoke creates shortcuts to the hub

One central workspace holds master data.
Every other workspace creates shortcuts to it.
One source of truth, zero copies.

When to use: Master data management, shared dimensional models, centralized reference data

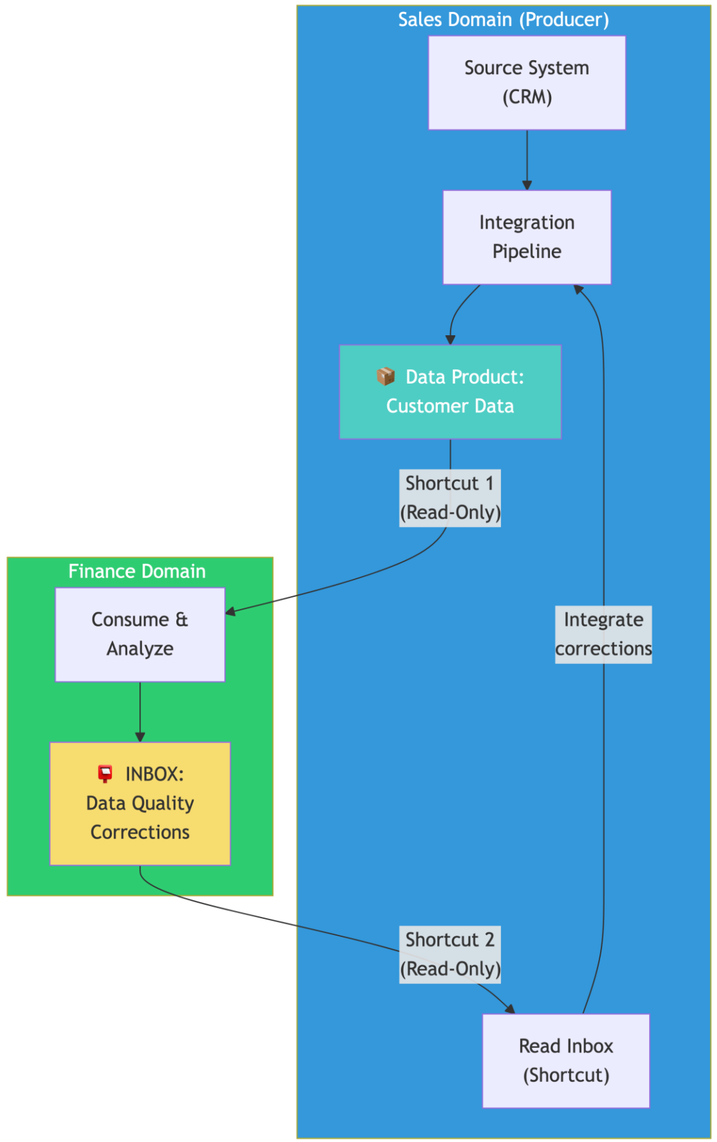

Sales Domain
Sales Data
Published: Transaction Summaries
Shortcut
Finance Domain
Finance Data
Published: Revenue Reports
Shortcut
Marketing Domain
Marketing Data
Published: Campaign Metrics
Shortcut

# Sharing Pattern 2: Data Mesh + The Inbox Pattern

Data Mesh: No central hub. Each domain publishes data products.
Others create shortcuts to consume.

The Inbox Pattern (Reverse Data Product):
Finance finds quality issues in Sales' data.
How do they send corrections BACK without write access?

Finance publishes corrections as THEIR OWN data product.
Sales reads via read-only shortcut.
Both shortcuts are read-only. Neither domain writes to the other's storage.
The governance chain is unbroken.


Sales Domain (Producer)
Source System (CRM)
Integration Pipeline
Data Product: Customer Data
Shortcut 1 (Read-Only)
Finance Domain
Consume & Analyze
INBOX: Data Quality Corrections
Integrate corrections
Shortcut 2 (Read-Only)
Read Inbox (Shortcut)

# Shortcut Security + Limits

Chained identity = your biggest governance asset

Traditional: copy data → original owner loses control
With shortcuts: data never moves → source permissions always apply
Revoke at source → every consumer loses access immediately

Key Limits:
- 100K shortcuts per Lakehouse
- 5 shortcut-to-shortcut chains max
- 10 path depth
- No non-Latin characters in paths

Tradeoff: more upfront permission management, zero risk of orphaned grants.

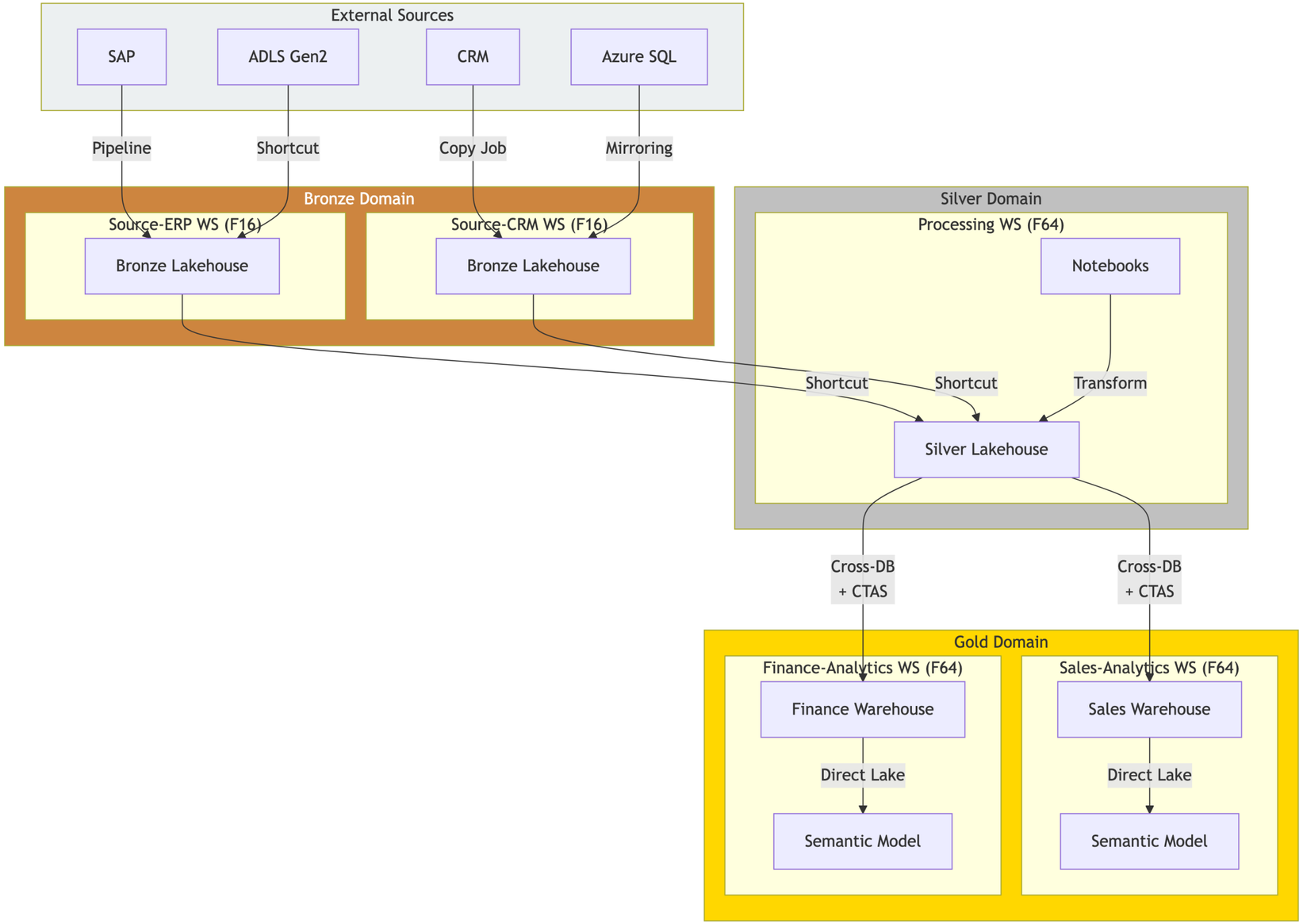
# Design your data flow as a Directed Acyclic Graph.

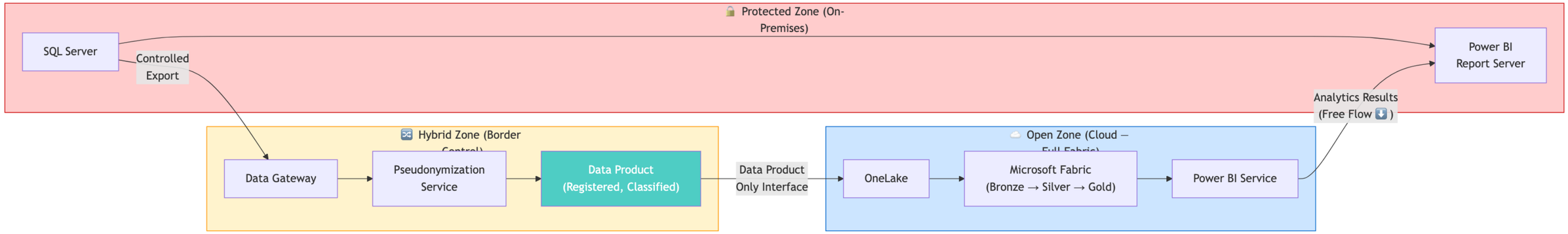
Data flows in one direction. No circular shortcuts.
Every item has a clear owner.

# Full Enterprise Reference Architecture

External Sources → Bronze Domain (per-source workspaces)
→ Silver Domain (processing workspace)
→ Gold Domain (per-business-unit analytics workspaces)
→ Semantic Models → Reports

Capacity sized per domain
Shortcuts connect all layers
Deployment pipelines per domain


External Sources
SAP
ADLS Gen2
CRM
Azure SQL
Pipeline
Shortcut
Copy Job
Mirroring
Bronze Domain
Source-ERP WS (F16)
Bronze Lakehouse
Source-CRM WS (F16)
Bronze Lakehouse
Silver Domain
Processing WS (F64)
Notebooks
Shortcut
Shortcut
Transform
Silver Lakehouse
Cross-DB + CTAS
Cross-DB + CTAS
Gold Domain
Finance-Analytics WS (F64)
Finance Warehouse
Direct Lake
Semantic Model
Sales-Analytics WS (F64)
Sales Warehouse
Direct Lake
Semantic Model

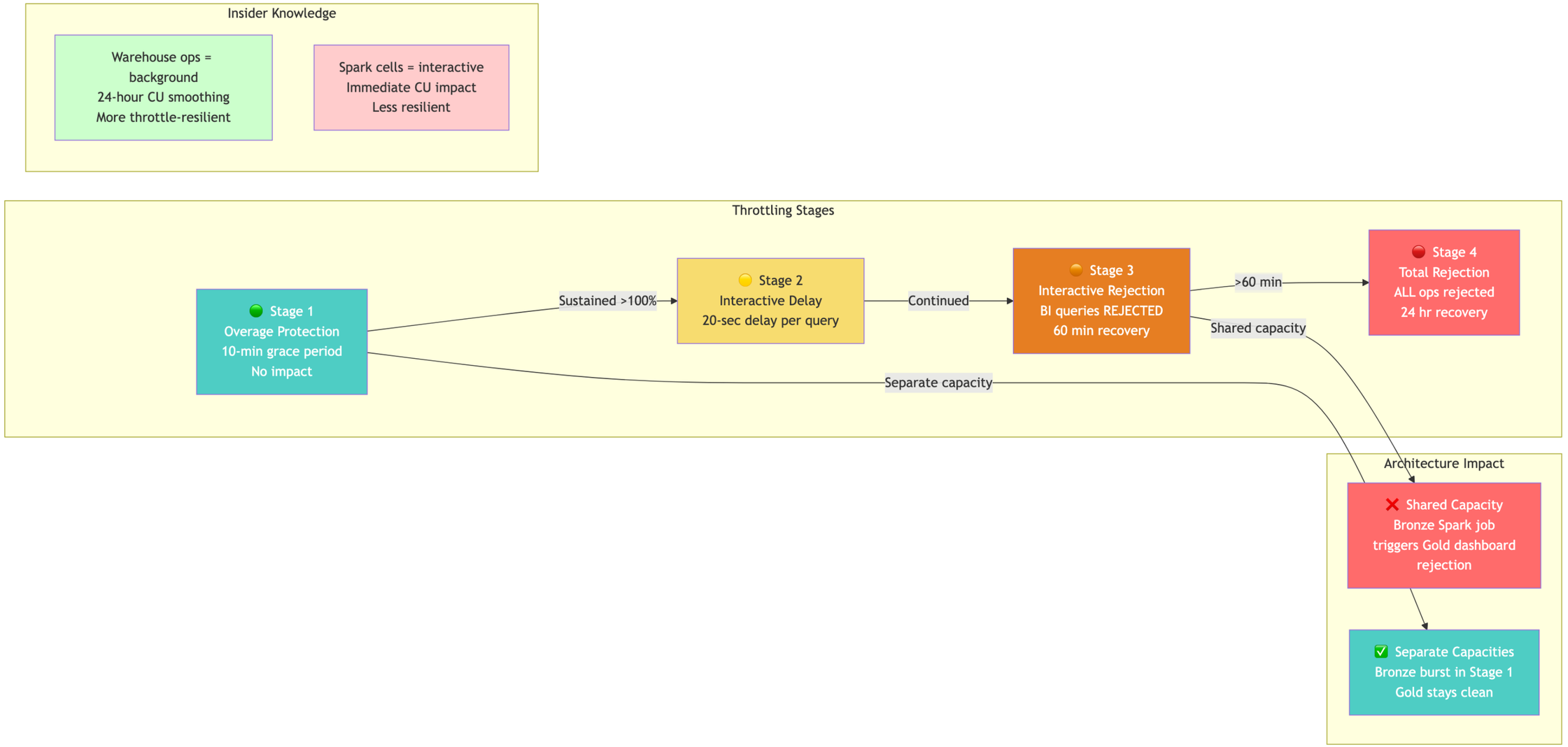

Protected Zone (On-Premises)
SQL Server
Controlled Export
Power BI Report Server
Analytics Results (Free Flow ⬇️ )
Hybrid Zone (Border
Data Gateway
Pseudonymization Service
Data Product (Registered, Classified)
Data Product Only Interface
Open Zone (Cloud —
OneLake
Microsoft Fabric (Bronze → Silver → Gold)
Power BI Service

# Why This Architecture + Throttling Insight

Enterprise principle: "Check upward, flow downward freely"
Data INTO the cloud requires governance checkpoints
Data flowing back (analytics, ML) flows freely

Capacity Throttling Cascade:
Stage 1: 10-min overage → Free burst (no impact)
Stage 2: Sustained → Requests delayed
Stage 3: Heavy → Background jobs rejected
Stage 4: Extreme → Interactive queries rejected

Separate capacities: Bronze burst stays in Stage 1 — Gold untouched
Insider trick: schedule Spark jobs within 10-minute burst windows


Insider Knowledge
Warehouse ops = background
24-hour CU smoothing
More throttle-resilient
Spark cells = interactive
Immediate CU impact
Less resilient
Throttling Stages
Stage 1
Overage Protection
10-min grace period
No impact
Sustained >100%
Stage 2
Interactive Delay
20-sec delay per query
Continued
Stage 3
Interactive Rejection
BI queries REJECTED
60 min recovery
>60 min
Stage 4
Total Rejection
ALL ops rejected
24 hr recovery
Shared capacity
Separate capacity
Architecture Impact
Shared Capacity
Bronze Spark job
triggers Gold dashboard
rejection
Separate Capacities
Bronze burst in Stage 1
Gold stays clean

# The Three Hidden Traps

1  Direct Lake Fallback
10-50x slower, no error message, no warning
Triggered by: SQL views, SQL-based RLS, calculated columns, exceeding guardrails

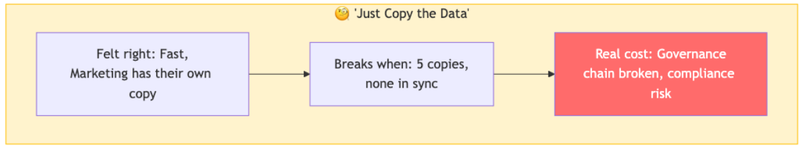
2  Deployment Pipeline Binding
Deployed semantic model still points to source Lakehouse
Must manually create datasource rules

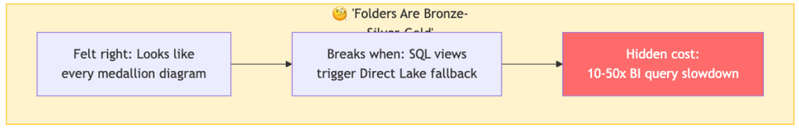
3  Capacity Throttling Cascade
Shared capacity = shared fate
Bronze Spark job → 4 throttling stages → Gold dashboards rejected

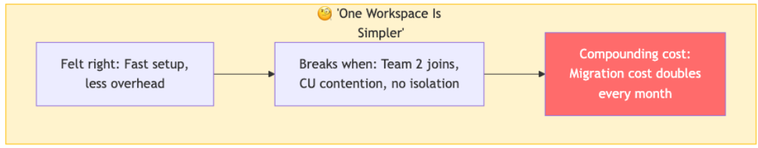
These traps ONLY surface at scale

## Decisions That Felt Right.

Every one of these made sense at the time.


'Just Copy the Data'
Felt right: Fast, Marketing has their own copy
Breaks when: 5 copies, none in sync
Real cost: Governance chain broken, compliance risk


'Folders Are Bronze-
Felt right: Looks like every medallion diagram
Breaks when: SQL views trigger Direct Lake fallback
Hidden cost: 10-50x BI query slowdown


'One Workspace Is Simpler'
Felt right: Fast setup, less overhead
Breaks when: Team 2 joins, CU contention, no isolation
Compounding cost: Migration cost doubles every month

# "One Workspace Is Simpler"

Felt Right:
One workspace for everything — less overhead, faster to set up

Breaks When:
Junior dev deletes production pipeline. 200 items, no isolation.
Cross-team overwrites.

Compounding Cost:
Every new item makes the blast radius larger.
Month 6: migration means recreating every workspace item, every shortcut, every permission.

# "Folders Are Good Enough"

Felt Right:
Folders in one Lakehouse — I can see everything, simple navigation

Breaks When:
No security boundaries. Bronze schema change cascades to Gold.
No independent capacity allocation.

Compounding Cost:
Splitting later = rewriting every notebook path, every shortcut reference, every pipeline.

# "Just Copy the Data"

Felt Right:
Copy the data — it's fast, it works, no permission complexity

Breaks When:
5 copies of customer data. None in sync. Storage cost 5x.
GDPR deletion request: which copies?

Compounding Cost:
Every new consumer = another copy.
Month 6: orphaned copies nobody knows about.

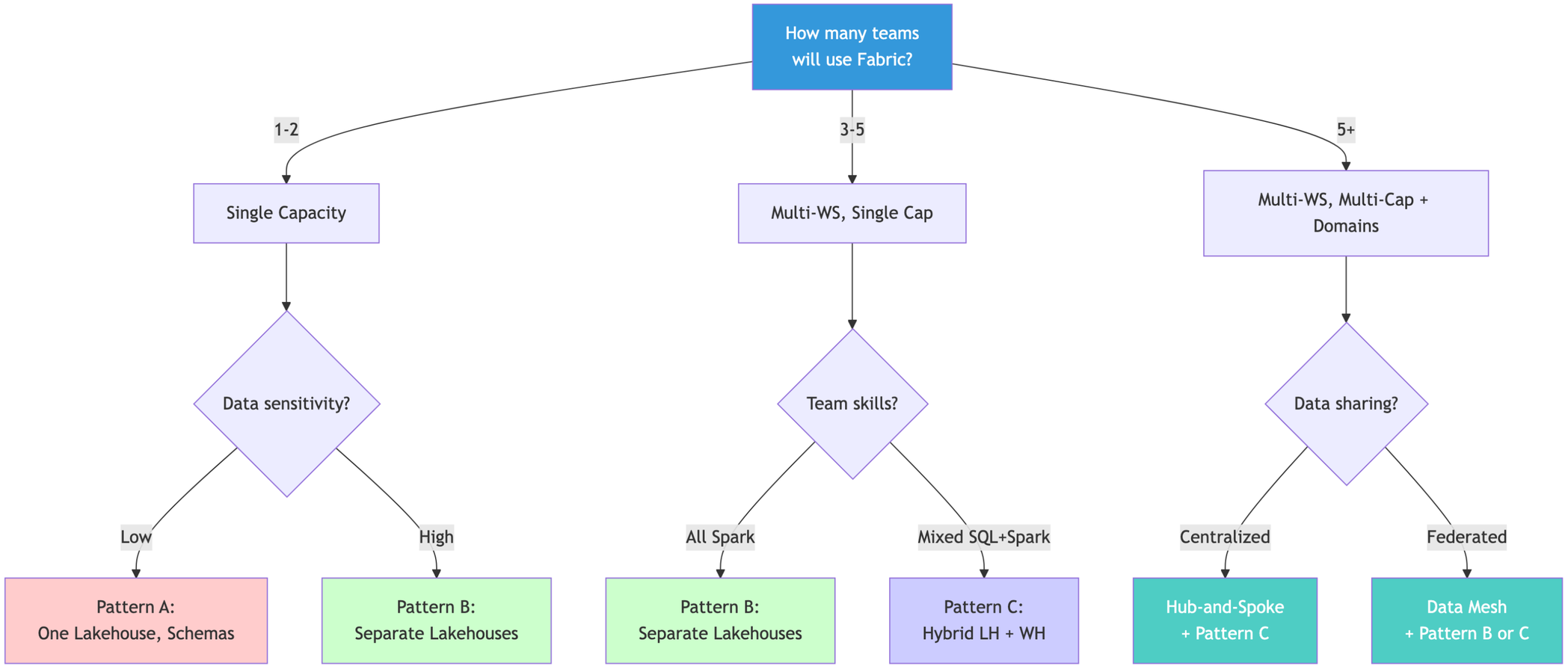
Notice: none of these were BAD decisions.
They were REASONABLE decisions that compound.
That's why architecture patterns exist

# Five Questions I Ask Every Customer in the First Hour

Answer these before loading a single row of data.

1 How many teams PRODUCE data?
2 What breaks if the wrong person sees the wrong data?
3 Does Gold need SQL features beyond SELECT?
4 Do consuming teams produce data too, or only consume?
5 Do you have data that can't leave the building?

I can predict 80% of the architecture from these five questions.

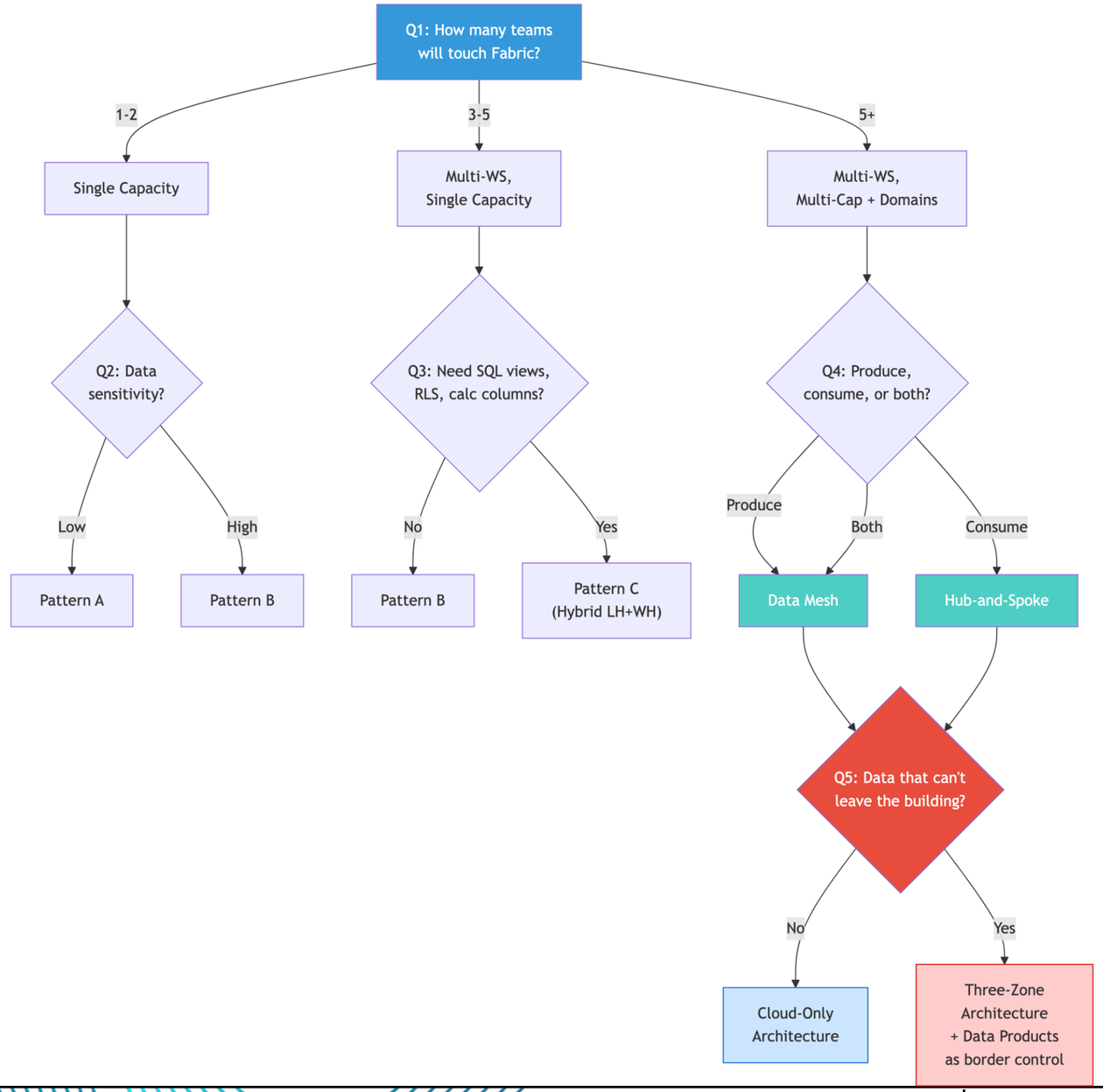

How many teams will use Fabric?
1-2
3-5
5+
Single Capacity
Multi-WS, Single Cap
Multi-WS, Multi-Cap + Domains
Data sensitivity?
Team skills?
Data sharing?
Low
High
All Spark
Mixed SQL+Spark
Centralized
Federated
Pattern A: One Lakehouse, Schemas
Pattern B: Separate Lakehouses
Pattern B: Separate Lakehouses
Pattern C: Hybrid LH + WH
Hub-and-Spoke + Pattern C
Data Mesh + Pattern B or C

# Decision Framework Flowchart

[DIAGRAM: Decision Framework — Five Questions Flowchart]

Q1: How many teams? (1-2 / 3-5 / 5+) → Workspace complexity
Q2: What breaks with wrong access? → Isolation level
Q3: Gold needs SQL beyond SELECT? → Lakehouse vs Warehouse
Q4: Consumers also produce? → Hub-and-Spoke vs Mesh
Q5: Data can't leave the building? → Cloud-only vs Three-Zone Architecture

Each question narrows to a specific pattern recommendation.
This is the slide the audience will photograph.


Q1: How many teams will touch Fabric?
1-2
3-5
5+
Single Capacity
Multi-WS, Single Capacity
Multi-WS, Multi-Cap + Domains
Q2: Data sensitivity?
Q3: Need SQL views, RLS, calc columns?
Q4: Produce, consume, or both?
Low
High
No
Yes
Produce
Both
Consume
Pattern A
Pattern B
Pattern B
Pattern C (Hybrid LH+WH)
Data Mesh
Hub-and-Spoke
Q5: Data that can't leave the building?
No
Yes
Cloud-Only Architecture
Three-Zone Architecture + Data Products as border control

# Framework Summary Table

| Scenario | Workspace | Medallion | Sharing |
|---|---|---|---|
| Small team, low sensitivity | Multi-WS, Single Cap | Pattern A (schemas) | N/A |
| Medium org, compliance | Multi-WS, Multi-Cap | Pattern B (sep. LH) | Hub-and-Spoke |
| Enterprise, SQL in Gold | Multi-WS + Domains | Pattern C (LH + WH) | Hub-and-Spoke |
| Federated, produce+consume | Multi-WS + Domains | Pattern B or C | Peer-to-Peer Mesh |

# Three Key Takeaways

1 Architecture decisions compound. What feels simple in Week 1 becomes your most expensive refactoring in Month 6.

2 The traps that matter most are invisible: Direct Lake fallback has no error, deployment binding has no warning, throttling cascades have no alert. Test at production scale or don't test at all.

3 Shortcuts aren't a storage optimization — they're your biggest governance asset. The data owner always retains control. If you're copying data between teams, you're creating governance debt.

*If your architecture can't answer "who owns this and what happens when it breaks" — it's not an architecture. It's a demo.*

# Resources

All MS Learn links from this session
Decision Framework (PDF)
Fabric Friday YouTube Series
Architecture Training

Scan the QR code or find me after the session.

[QR CODE PLACEHOLDER]

# Sound off.
# The mic is all yours.
# Influence the product roadmap.

Join the Fabric User Panel

Share your feedback directly with our Fabric product group and researchers.

https://aka.ms/JoinFabricUserPanel

Join the SQL User Panel

Influence our SQL roadmap and ensure it meets your real-life needs

https://aka.ms/JoinSQLUserPanel

# Sound off.
# The mic is all yours.
# Influence the product roadmap.

Join the Fabric User Panel

Share your feedback directly with our Fabric product group and researchers.

https://aka.ms/JoinFabricUserPanel

Join the SQL User Panel

Influence our SQL roadmap and ensure it meets your real-life needs

https://aka.ms/JoinSQLUserPanel