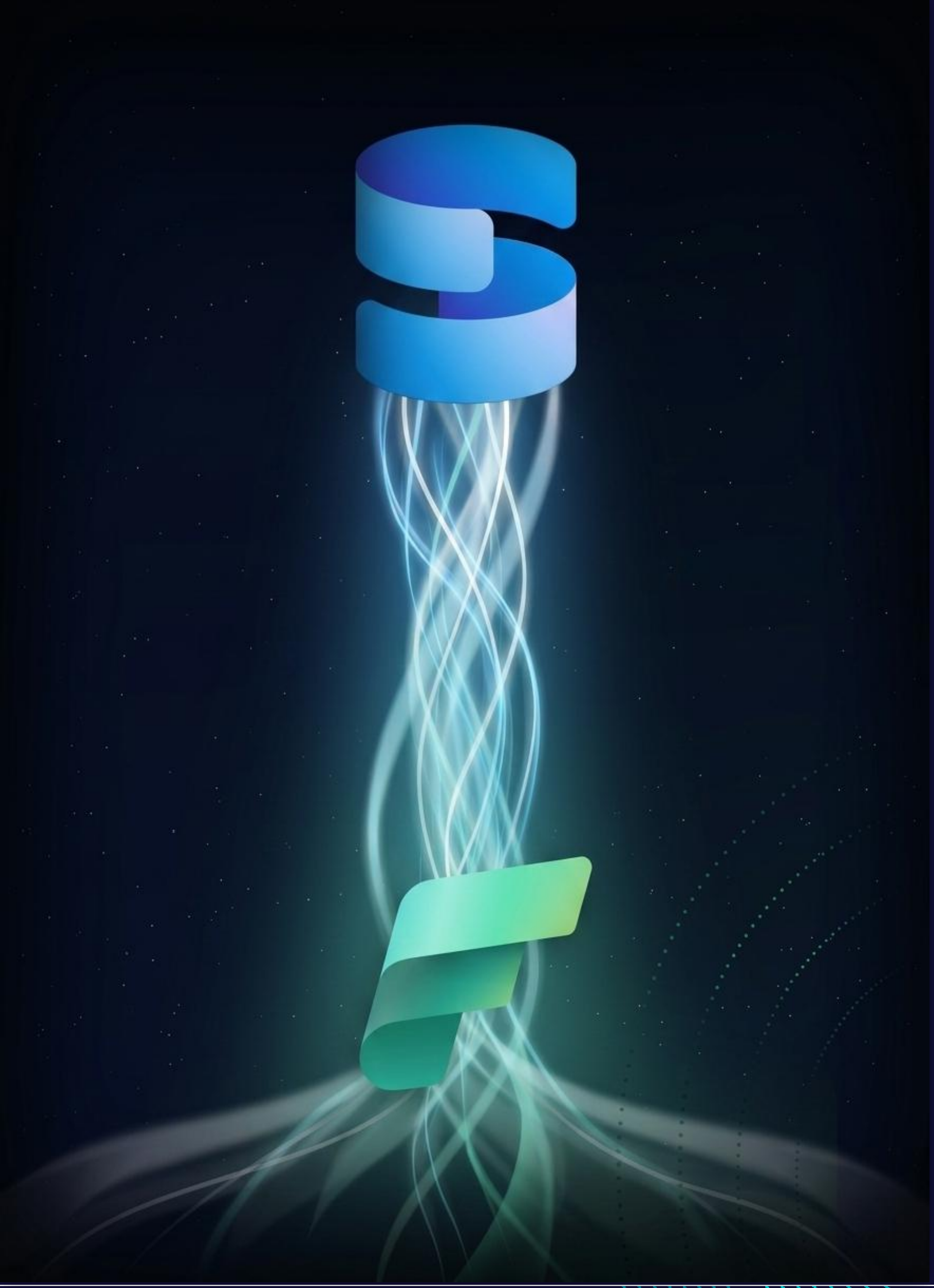
#FABCONSQLCON2026

# FABCON | SQLCON

Microsoft Fabric COMMUNITY CONFERENCE | Microsoft SQL COMMUNITY CONFERENCE

**ATLANTA** MARCH 16 - 20, 2026



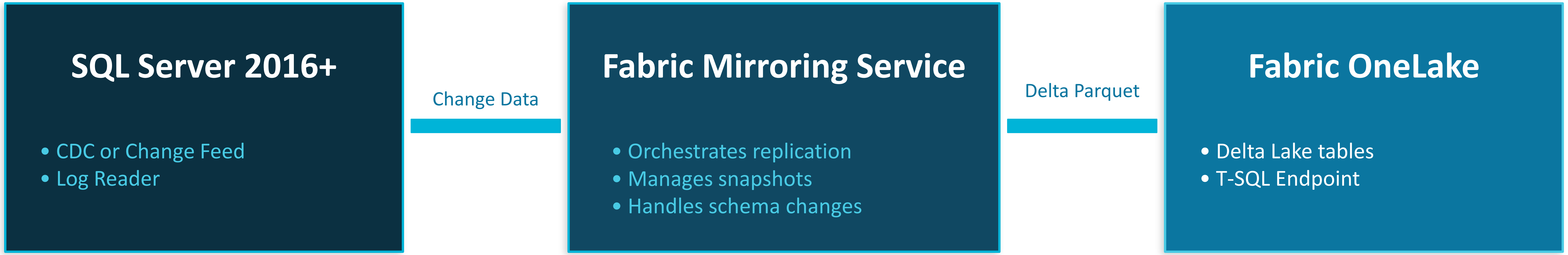
# Mirroring for SQL Server in Fabric: Inside the Replication Process

Meagan Longoria
Justin Cunningham

ProcureSQL

# What We'll Cover

# What is Fabric Mirroring?



### Near Real-Time

Typical latency of seconds to minutes; not a batch process

### No ETL Pipeline Needed

Managed service — no ADF or Spark jobs required to replicate

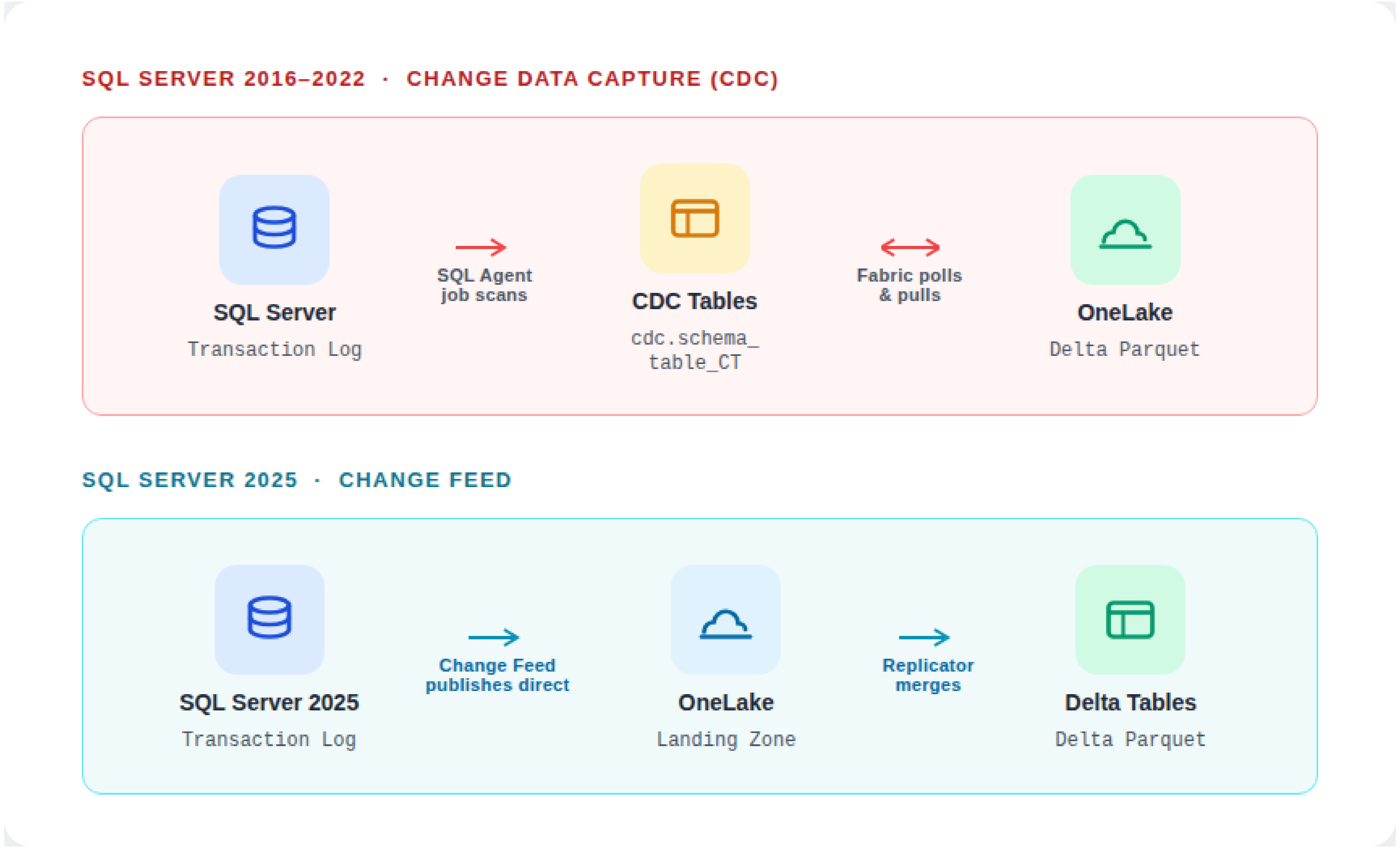
### Incremental by Default

After initial snapshot, only changed rows are replicated**

### Read-Only in Fabric

Mirrored tables are read-only; T-SQL endpoint allows adding views, procs, functions, RLS, OLS.

# A Big Change for Mirroring with SQL Server 2025



# The Technology Behind the Scenes

## SQL Server 2016-2022: CDC

**Change Data Capture**

- sys.sp_cdc_enable_db enables CDC on the database
- Per-table capture instances created in the cdc schema
- SQL Agent job cdc.{db}_capture reads the transaction log
- SQL Agent job cdc.{db}_cleanup purges old change entries
- SQL Agent must be running — if stopped, replication lag accumulates

## SQL Server 2025 and Azure SQL: Change Feed

**Change Feed**

- Change Feed is built into the database engine — no SQL Agent required
- sys.dm_change_feed_log_scan_sessions for native monitoring
- No cdc schema, no capture/cleanup job management
- Azure SQL: enabled per-database via Fabric mirroring configuration

Both paths: initial full snapshot always runs first before incremental changes begin

# SQL Server 2025+ Mirroring: Requirements

**SQL Server 2025 (on-premises)**
Not supported on Azure VM or Linux instances

**CDC must NOT be enabled on the source database**
Change Feed replaces CDC; the two conflict if both enabled

**Replication must NOT be enabled**
SQL Server Replication conflicts with Change Feed mirroring

**Primary database of AG only; no FCI**
Failover cluster instances not supported

**ALTER ANY EXTERNAL MIRROR, VIEW DATABASE PERFORMANCE STATE, VIEW DATABASE SECURITY STATE permissions**
Must be granted to the Fabric login

**Delayed transaction durability must be disabled**
Database cannot be mirrored if delayed durability is enabled

# SQL Server 2025+ Mirroring: Limitations

**Max 1000 tables per mirrored database**
"Mirror all data" picks first 1000 alphabetically; remaining tables skipped

**No column filtering**
All columns replicate; use views on the Fabric side to limit exposure

**Unsupported data types**
json, vector, FILESTREAM block entire table; PK on geometry/geography/hierarchyid/UDT/sql_variant blocks table; computed columns skipped

**DDL changes trigger full table reseed**
ALTER/DROP column; partition switch blocked while mirroring

**Table features that block replication**
Temporal/ledger history, Always Encrypted, in-memory, graph, external tables

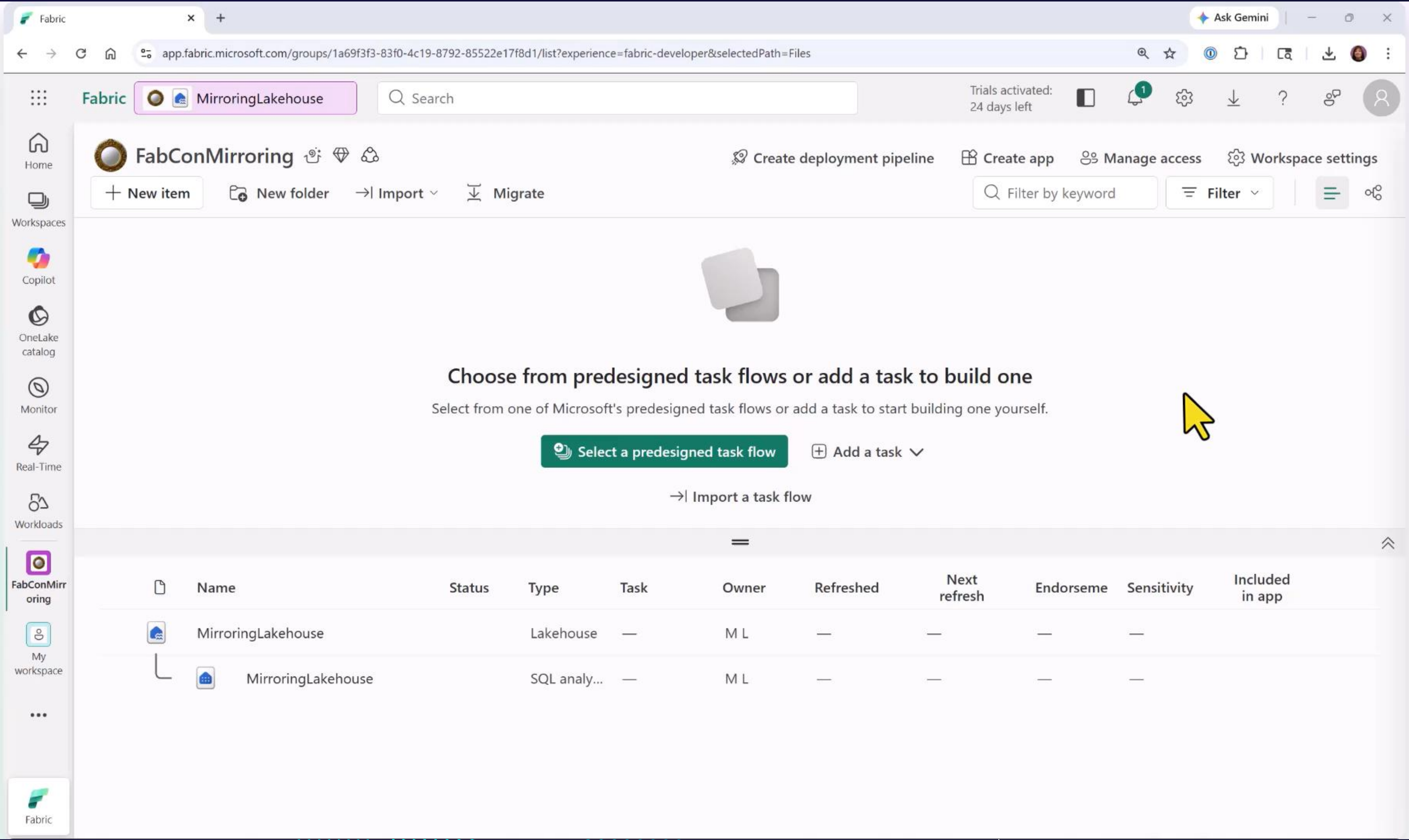
**Security not propagated to Fabric**
RLS, column permissions, dynamic data masking not reflected in OneLake

# Configuring Replicated Objects

| ✓ Supported | ✗ Not Supported |
|---|---|
| **Table Selection** Choose up to 1000 individual tables; all user tables available by default | **Column Filtering** All columns always replicated; use Fabric views to limit exposure |
| **Tables Without Primary Keys** No PK required since April 2025; existing tables need a stop/start to pick up | **Tables w/ PK/index on unsupported types** geometry, geography, hierarchyid, sql_variant, UDTs, datetime2(7) |
| **ADD COLUMN (DDL)** Schema additions replicate automatically to Fabric | **Unsupported Data Types** json, vector block the entire table from mirroring |
| **Most Standard Data Types** int, bigint, varchar, nvarchar, datetime, decimal, bit, and more | **Views and Computed Columns** Only base tables replicate; no view or computed column support |

# DEMO

## Configuring Mirroring



Choose from predesigned task flows or add a task to build one

Select from one of Microsoft's predesigned task flows or add a task to start building one yourself.

| Name | Status | Type | Task | Owner | Refreshed | Next refresh | Endorseme | Sensitivity | Included in app |
|---|---|---|---|---|---|---|---|---|---|
| MirroringLakehouse | | Lakehouse | — | M L | — | — | — | — | |
| MirroringLakehouse | | SQL analy... | — | M L | — | — | — | — | |

# Data Retention Policies

## Source: Change Feed Retention (SQL Server)

| | |
|---|---|
| **Controlled by** | Internal change feed process |
| **History visibility** | Not configurable — depends on availability of log data |
| **Deleted row handling** | Deletes available in the change feed |

## Destination: OneLake Delta Retention

| | |
|---|---|
| **Controlled by** | Delta table versioning |
| **History visibility** | Default 1-day (new); 7-day (legacy); configurable |
| **Deleted row handling** | Soft deletes tracked in Delta log until outside of retention window |

⚠ If Change Feed falls too far behind due to log truncation or an outage, a gap can occur — requiring a full restart of replication.

## What Mirroring Enables in SQL Server 2025

**Change Feed Enabled on Database**

Enabled via Fabric portal through Azure Arc — no T-SQL sp_enable call

**Change Feed on Each Table**

Per-table tracking; no cdc schema — Change Feed uses SQL 2025 internal infrastructure

**sys.dm_change_feed_log_scan_sessions**

Monitor Change Feed scan sessions and latency

**No SQL Agent Required**

Change Feed uses a built-in internal process — no SQL Agent capture job needed

**No Cleanup Job Needed**

Unlike CDC, Change Feed has no user-managed retention or cleanup jobs

# How Data Lands in OneLake

## Phase 1: Initial Snapshot

- Full table read via bulk export
- Written as Parquet files to OneLake staging
- Delta Log initialized with Add operations
- Tables become queryable after snapshot completes
- Large tables may take minutes to hours

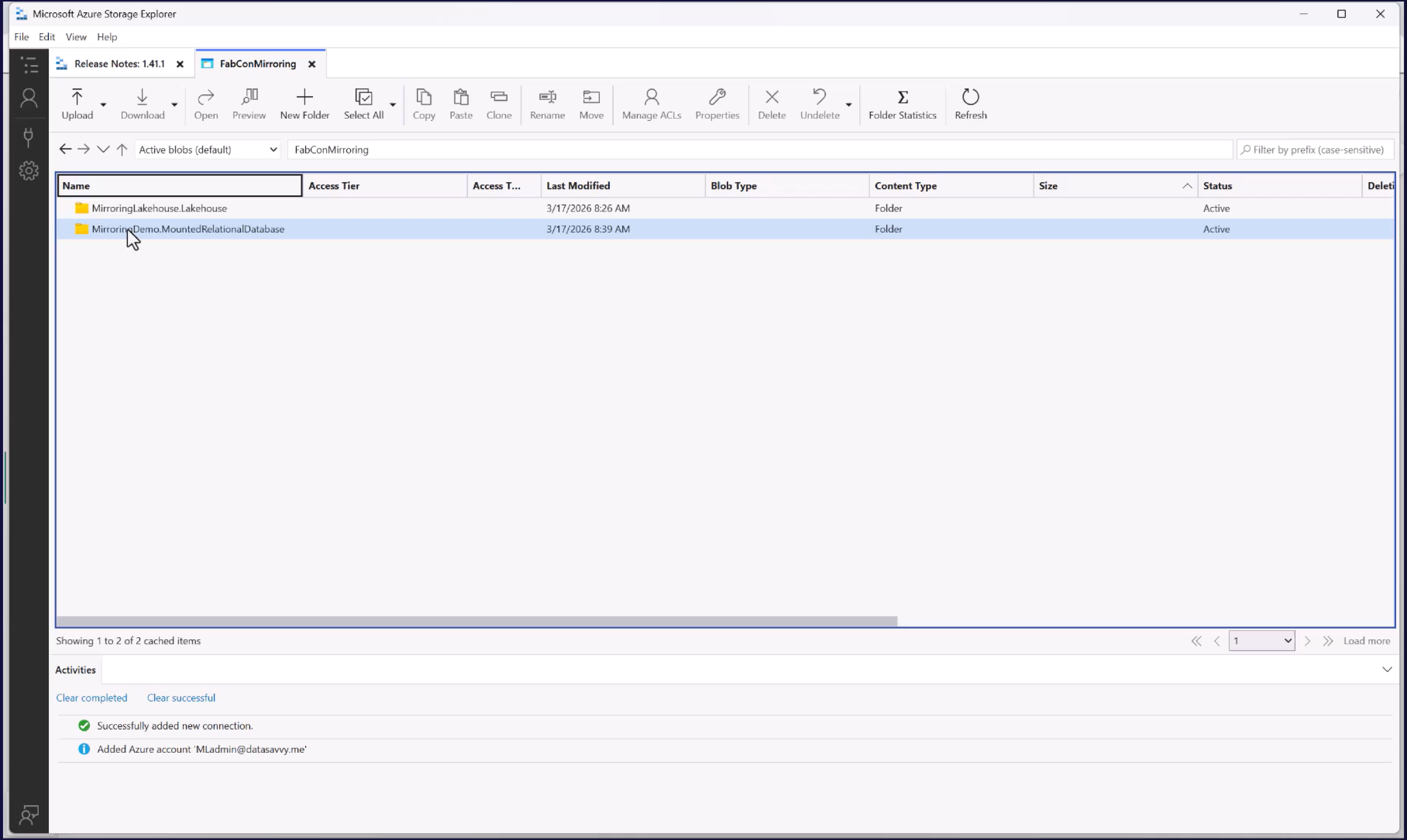
## Phase 2: Incremental (Change Feed)

- SQL 2025+ Change Feed streams row changes in near real-time
- Inserts → ADD to Delta log
- Updates = two CDF rows (pre-image + post-image)
- Deletes → DELETE entry in Delta log
- Vacuum/OPTIMIZE run periodically by Fabric

```
OneLake Path: Files/Mirrored/{WorkspaceId}/{MirroredDatabaseId}/{SchemaName}/{TableName}/
```

# DEMO

## Data in OneLake



| Name | Access Tier | Access T... | Last Modified | Blob Type | Content Type | Size | Status | Deleti |
|---|---|---|---|---|---|---|---|---|
| MirroringLakehouse.Lakehouse | | | 3/17/2026 8:26 AM | | Folder | | Active | |
| MirroringDemo.MountedRelationalDatabase | | | 3/17/2026 8:39 AM | | Folder | | Active | |

Showing 1 to 2 of 2 cached items

Load more

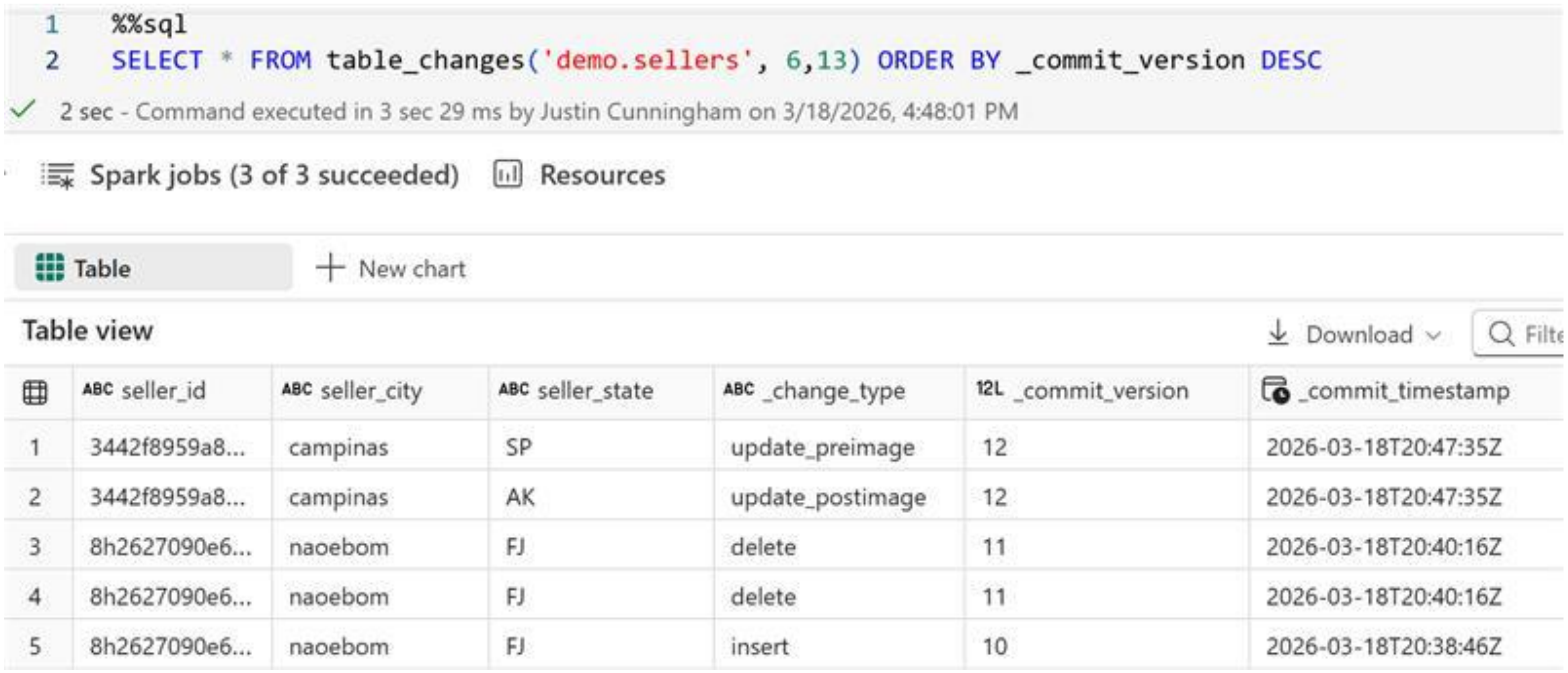
Activities

Clear completed    Clear successful

Successfully added new connection.

Added Azure account 'MLadmin@datasavvy.me'

# Mirroring Change Data Feed



```
%%sql
SELECT * FROM table_changes('demo.sellers', 6,13) ORDER BY _commit_version DESC
```

✓ 2 sec - Command executed in 3 sec 29 ms by Justin Cunningham on 3/18/2026, 4:48:01 PM

| | seller_id | seller_city | seller_state | _change_type | _commit_version | _commit_timestamp |
|---|---|---|---|---|---|---|
| 1 | 3442f8959a8... | campinas | SP | update_preimage | 12 | 2026-03-18T20:47:35Z |
| 2 | 3442f8959a8... | campinas | AK | update_postimage | 12 | 2026-03-18T20:47:35Z |
| 3 | 8h2627090e6... | naoebom | FJ | delete | 11 | 2026-03-18T20:40:16Z |
| 4 | 8h2627090e6... | naoebom | FJ | delete | 11 | 2026-03-18T20:40:16Z |
| 5 | 8h2627090e6... | naoebom | FJ | insert | 10 | 2026-03-18T20:38:46Z |

## Performance & Storage Considerations

**Transaction Log Size**

Change Feed holds log truncation until changes are replicated. Monitor log size and ensure long-running transactions don't cause the log to fill.

**Minor Change Feed Overhead**

The internal Change Feed process adds minimal CPU/IO. No polling interval or maxtrans tuning needed.

**Change Feed Internal Storage**

Change Feed publishes changes directly to Fabric landing zone — no cdc schema or change tables created on source

**Low Source Overhead**

Change Feed reads directly from the transaction log with minimal CPU/IO impact. No external job process to monitor.

# Monitoring & Logging

## On the Source (SQL Server)

- sys.dm_change_feed_log_scan_sessions — latency and scan sessions
- sys.dm_change_feed_errors — Change Feed errors with error codes
- EXEC sp_help_change_feed — table state (4=healthy) and full config
- sys.databases (log_reuse_wait_desc) — check if log is growing due to mirroring delays
- No SQL Agent jobs — Change Feed is managed internally by SQL Server 2025

## In the Fabric Portal

- Mirroring status: Running / Stopped / Error
- Per-table replication status (rows, last update)
- Error messages surfaced in mirroring properties
- Check OneLake file timestamps for staleness

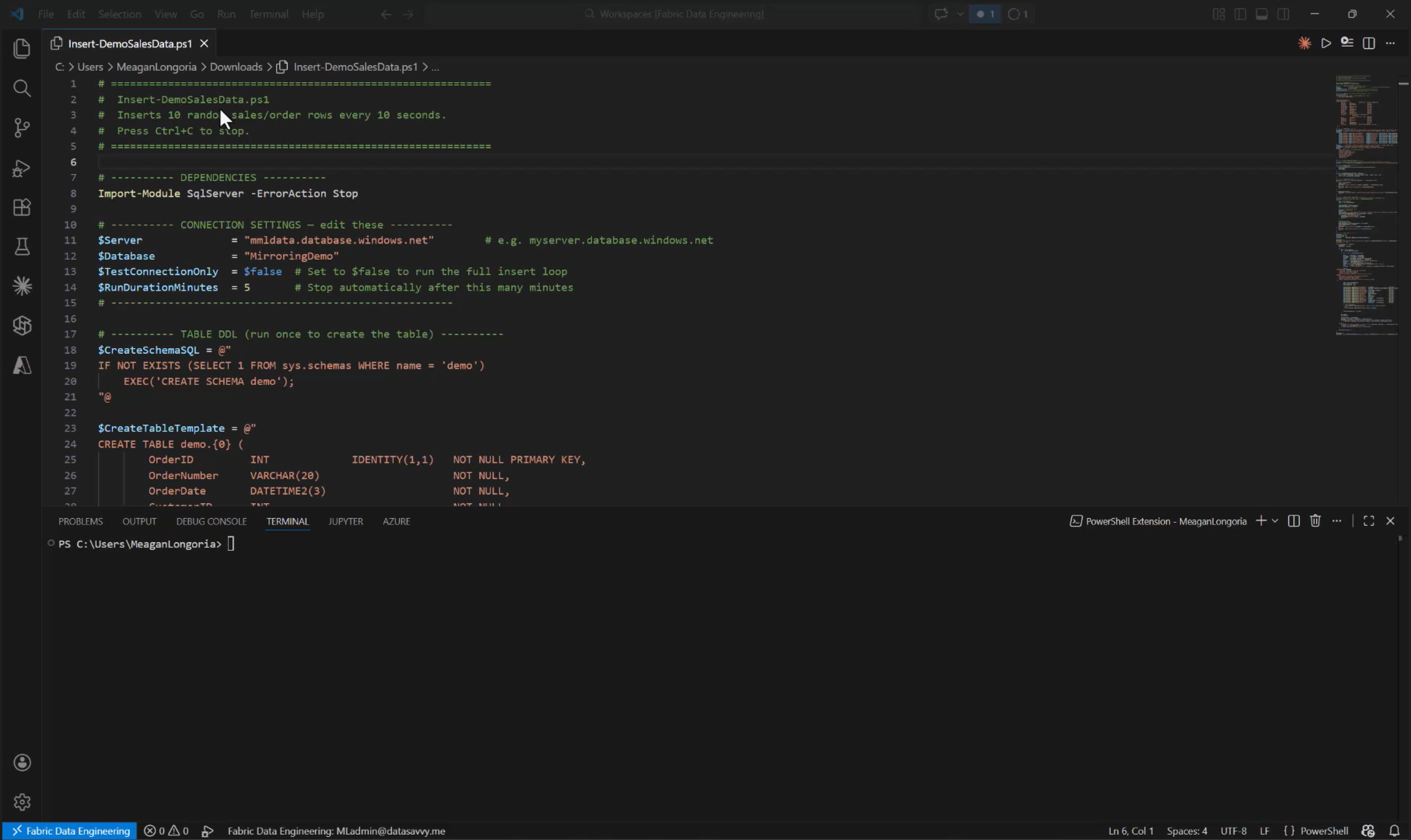
## Programmatic / Advanced

- Fabric REST API — GET mirrored database status
- .NET SDK (Microsoft.Fabric.Api) for C# automation
- PowerShell / AZ CLI — calls Fabric REST API for start/stop/status
- Custom alerts via Azure Monitor / Log Analytics
- Custom queries / alerts on workspace monitoring data

**Replication lag > 5 minutes** may indicate Change Feed scan delays, log truncation issues, or network problems between source and Fabric.

## DEMO

## Monitor the SQL Server Change Feed


File Edit Selection View Go Run Terminal Help
Workspaces [Fabric Data Engineering]
Insert-DemoSalesData.ps1
C: > Users > MeaganLongoria > Downloads > Insert-DemoSalesData.ps1 > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER AZURE
PowerShell Extension - MeaganLongoria
Fabric Data Engineering
Fabric Data Engineering: MLadmin@datasavvy.me
Ln 6, Col 1 Spaces: 4 UTF-8 LF PowerShell

```powershell
# ============================================================
#  Insert-DemoSalesData.ps1
#  Inserts 10 random sales/order rows every 10 seconds.
#  Press Ctrl+C to stop.
# ============================================================

# ---------- DEPENDENCIES ----------
Import-Module SqlServer -ErrorAction Stop

# ---------- CONNECTION SETTINGS — edit these ----------
$Server             = "mmldata.database.windows.net"       # e.g. myserver.database.windows.net
$Database           = "MirroringDemo"
$TestConnectionOnly  = $false  # Set to $false to run the full insert loop
$RunDurationMinutes  = 5       # Stop automatically after this many minutes
# ------------------------------------------------------

# ---------- TABLE DDL (run once to create the table) ----------
$CreateSchemaSQL = @"
IF NOT EXISTS (SELECT 1 FROM sys.schemas WHERE name = 'demo')
    EXEC('CREATE SCHEMA demo');
"@

$CreateTableTemplate = @"
CREATE TABLE demo.{0} (
        OrderID         INT             IDENTITY(1,1)   NOT NULL PRIMARY KEY,
        OrderNumber     VARCHAR(20)                     NOT NULL,
        OrderDate       DATETIME2(3)                    NOT NULL,
```

```
PS C:\Users\MeaganLongoria>
```

## When Replication Restart May Be Required

### Requires Full Restart (re-snapshot)

**Table removed & re-added**

Must re-initialize from scratch for that table

**Source DB restore**

LSN chain breaks; Change Feed history is reset

**Schema change (DROP/ALTER column)**

Breaking DDL not supported by incremental Change Feed

**Change Feed gap on source**

Transaction log truncated before Change Feed could read it

### Self-Healing / Pause & Resume OK

**Network blip / timeout**

Fabric auto-resumes from last known LSN

**Adding new tables**

New snapshot only for new tables; existing continue

**Adding a new column**

Column added to mirrored table; unsupported types skipped

**Credential / password expiry**

Update connection credentials; mirroring resumes

**Minor column rename (via sp)**

Tracked via Change Feed metadata reset

# Restarting Replication Safely

**Steps to restart mirroring in Fabric**

1. Stop mirroring from the Fabric portal (or via REST API)
2. Verify or fix the root cause on the source SQL Server
3. Confirm Change Feed is active on the source database
4. Optionally remove and re-add problem tables from the mirror config
5. Start mirroring — initial snapshot will re-run
6. Monitor replication status until tables show 'Running'

**Pre-Restart Checklist**

- ✓ Fabric login credentials and permissions verified
- ✓ No blocking long transactions on source
- ✓ Network connectivity Fabric → Source OK
- ✓ Transaction log space available
- ✓ Downstream consumers aware of re-snapshot
- ✓ Snapshot window planned for off-peak hours

## Key Takeaways

**01 Change Feed replaces CDC**

Fabric Mirroring for SQL 2025+ depends on Change Feed — ensure CDC and Replication are disabled on the source.

**02 Retention matters on both ends**

SQL side: log truncation before Fabric reads Change Feed = restart risk. Fabric side: Delta table retention controls how long history is queryable.

**03 Monitor proactively**

Use DMVs, the Fabric portal, and custom alerts to catch lag before it becomes a gap.

**04 Not all DDL is safe**

Breaking schema changes (DROP/ALTER) can require a full re-snapshot.

**05 Restarts mean re-snapshots**

Plan restart windows carefully — large tables can take significant time to reload.

**06 Two ways to query mirrored data**

Use the SQL analytics endpoint for T-SQL queries, or access OneLake Delta files directly from Spark, pipelines, and other Fabric workloads.

## Mirroring questions?

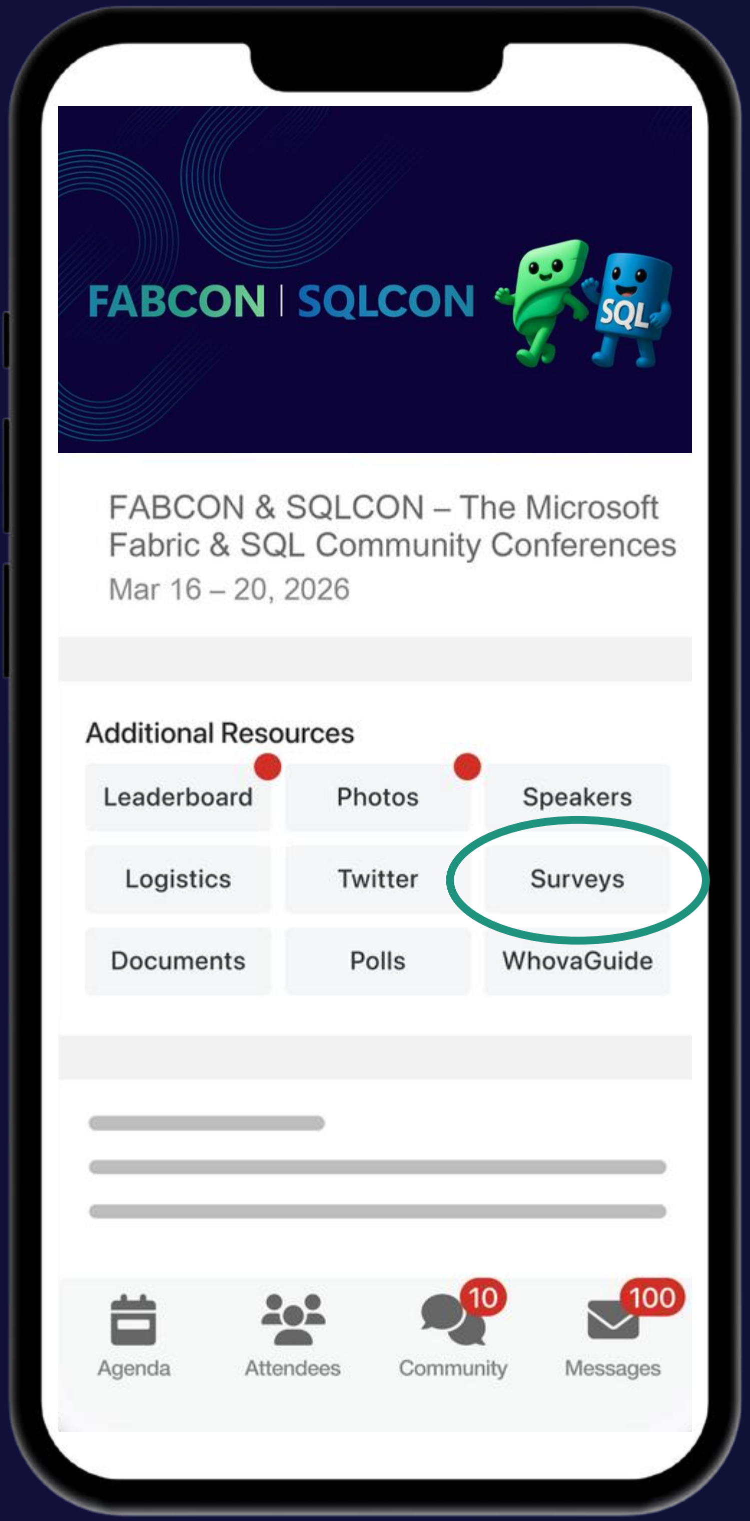
*Let’s talk about it.*

**Meagan Longoria**
**Justin Cunningham**

ProcureSQL

## How was the session?

Complete Session Surveys in Whova for your chance to WIN PRIZES!



## Get Two Fabric Certifications for FREE

Attendees of FABCON can take the Fabric Analytics Engineer or Fabric Data Engineer exam for free. Be part of the 2 fastest growing role-based certifications in Microsoft history.

**Request your voucher by March 23, 2026.**

https://aka.ms/fabcon/cert100