DATA+AI SUMMIT BY databricks

HOW DLT STRETCHED CDC CAPABILITIES & KEPT ETL LIMBER AT HINGE HEALTH

Veera Mukkanagoudar, Sr. Engineering Manager, Hinge Health

Alex Owen, Sr. Solutions Architect, Databricks

SPEAKERS

Veera Mukkanagoudar

Sr. Engineering Manager

Hinge Health

Alex Owen

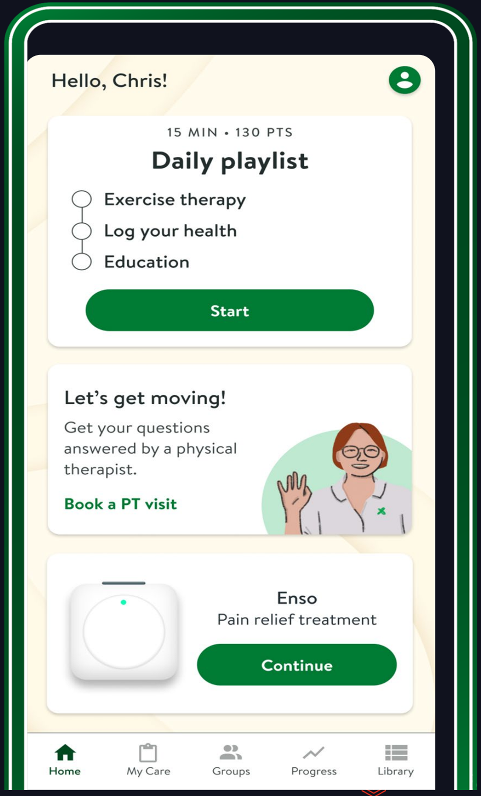
Sr. Solutions Architect

Databricks

Transforming Pain Treatment

Our digital clinic reduces unnecessary surgeries and opioid use

- Provide musculoskeletal (MSK) care
- Our digital clinic for joint and muscle pain gets people moving and keeps them moving to reduce unnecessary surgeries and opioid use.



The market leading MSK solution

4 in 5 employers with a digital MSK solution choose Hinge Health

45+ health plans and Pharmacy benefit management choose Hinge Health

2.4x ROI validated by multiple 3rd parties

1 Million members treated

"If you are looking for something that is **effective, will resonate with your employees and is easy to implement,** this is what you need."

Associate Director, Systems Benefit Admin

50 testimonials shared per day | 4.9 ★★★★★

STRETCH DEMO!

What problem are we solving?

Our Journey to an optimized CDC Architecture

Data Engineering Mission	The Challenge	The Solution
Enhance Hinge Health's capabilities with data intelligence to ensure the delivery of high-quality, timely, and cost-effective care across all products and services.	Build a efficient [low cost and low latency] data platform to transform MSK data at scale.	Mirror source databases in a Lakehouse target by collecting and writing change logs from Aurora to serve data in Delta for AI and non-AI use cases using DLT.

Transforming Pain Treatment

Some stats for context

- Data sources: 70+
- Postgres data sources: 35+
- Postgres tables: 4000+
- Velocity: 6.5 mbps & 1.6k messages/sec

Hinge Health™

Building Blocks

Foundational Concepts

Change Data Capture

- Replicate data between Systems
- Real-time tracking of Changes
- Faster time to Insights

Multiplexing

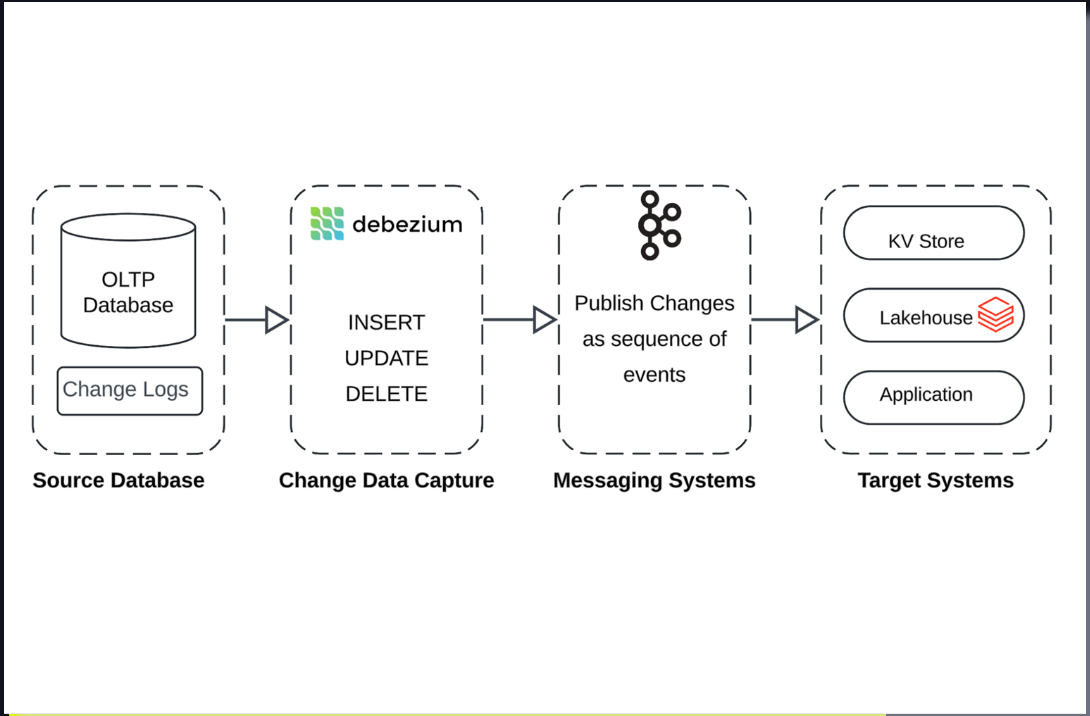
- Multiple data streams from single source
- Ingest at Scale
- Simplify Management

Extract, Load, Transform (ELT)

- Load target system from a Source
- Flexibility to transform on demand
- Improved Scalability

Change Data Capture - CDC

Easily identify & sync changes between systems



- Change Logs
- Debezium
- Kafka
- Incremental loading
- Slow Changing Dimensions

Change Data Capture

Slow Changing Dimensions: Type 1: Current

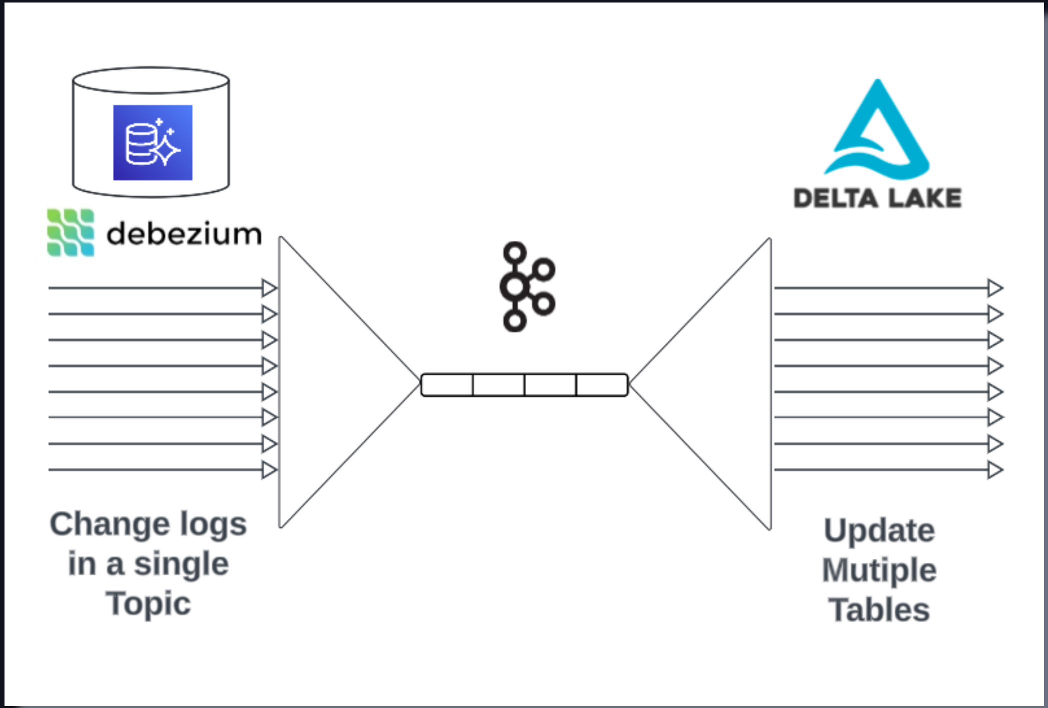
customer_id	customer_name	customer_city
123	Bob	Los Angeles, CA
456	Jane	San Francisco, CA
789	Cindy	Springfield, MO

Slow Changing Dimensions: Type 2: History

customer_id	customer_name	customer_city	start_at	end_at
789	Cindy	New York, NY	Monday	Wednesday
789	Cindy	Springfield, MO	Wednesday	null
123	Bob	Los Angeles, CA	Sunday	null
456	Jane	San Francisco, CA	Saturday	null

Multiplexing Architecture

Send multiple data streams over a shared medium

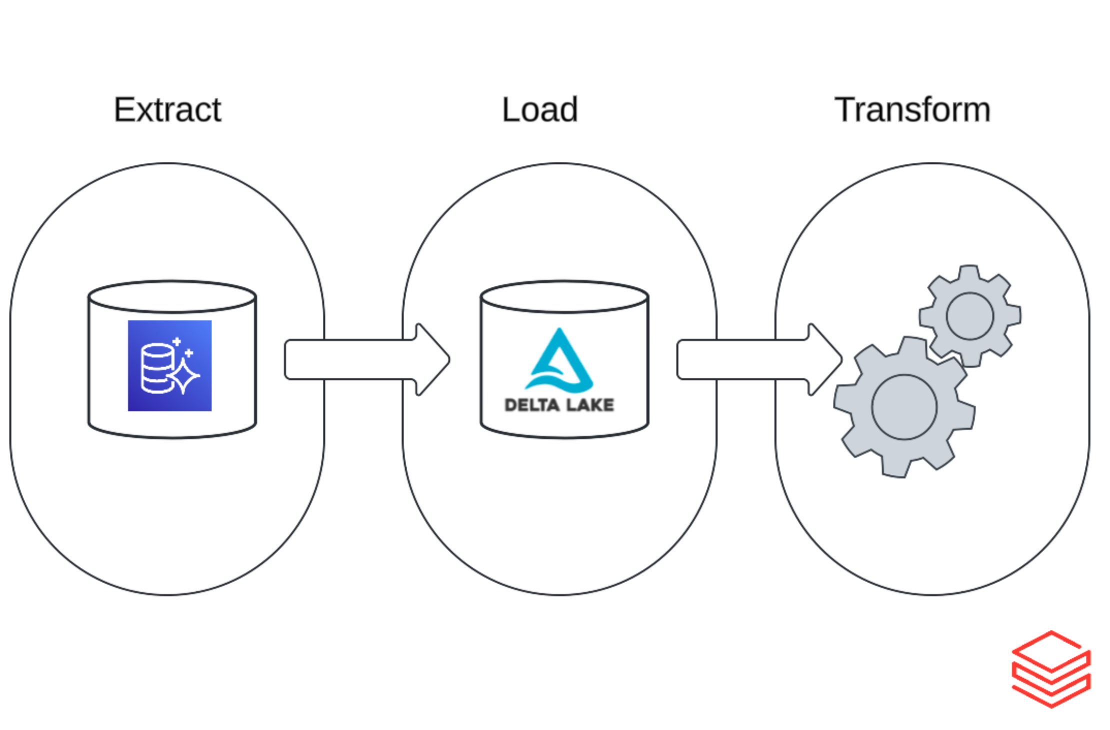


- Simple
- Incremental/ Change Data
- Resource Utilization
- Data Onboarding
- n tables : 1 Kafka topic

Extract, Load, Transform - ELT

Data transformed in the target System

- Not ETL!
- Loaded in a raw form
- Scalability
- Transform on demand
- Cost Optimized



We evaluated different solutions

Our Journey to an optimized CDC Architecture

What we evaluated

- AWS Solutions: EMR/ Redshift/ Glue
- Managed data warehouse with proprietary file format
- Databricks - Lakehouse

What was better for all of our use cases

- Single platform to support ELT, Data Warehousing and ML workloads.
- Support batch and streaming ELT
- Parameterized pipelines in Python and SQL

Delta Live Tables - DLT

Building reliable, maintainable & performant data pipelines

Source

```
/* Create a temp view on the accounts table */
CREATE STREAMING VIEW account_raw AS
SELECT * FROM cloud_files(“/data”, “csv”);
```

Bronze

```
/* Stage 1: Bronze Table drop invalid rows */
CREATE STREAMING TABLE account_bronze AS
COMMENT "Bronze table with valid account ids"
SELECT * FROM account_raw ...
```

Silver

```
/* Stage 2:Send rows to Silver, run validation rules */
CREATE STREAMING TABLE account_silver AS
COMMENT "Silver Accounts table with validation checks"
SELECT * FROM account_bronze ...
```

Gold

Accelerate ETL development

Declare **SQL or Python** and DLT automatically orchestrates the DAG, handles retries, changing data

Automatically manage your infrastructure

Automates complex tedious activities like **recovery, auto-scaling, and performance optimization**

Ensure high data quality

Deliver reliable data with built-in **quality controls, testing, monitoring, and enforcement**

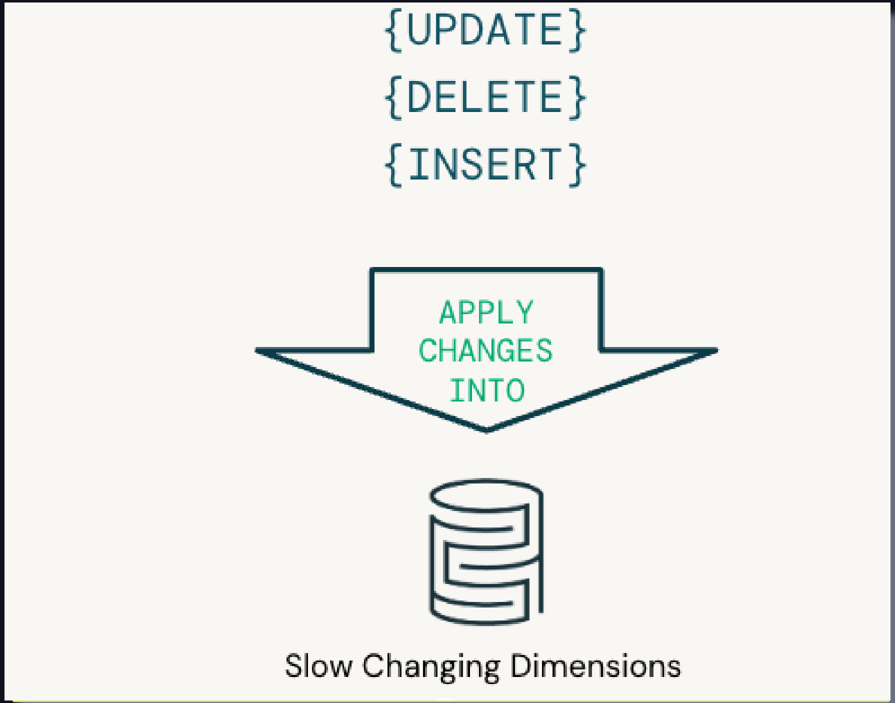
Unify batch and streaming

Get the simplicity of SQL with freshness of streaming with one **unified API**

DLT - Process CDC

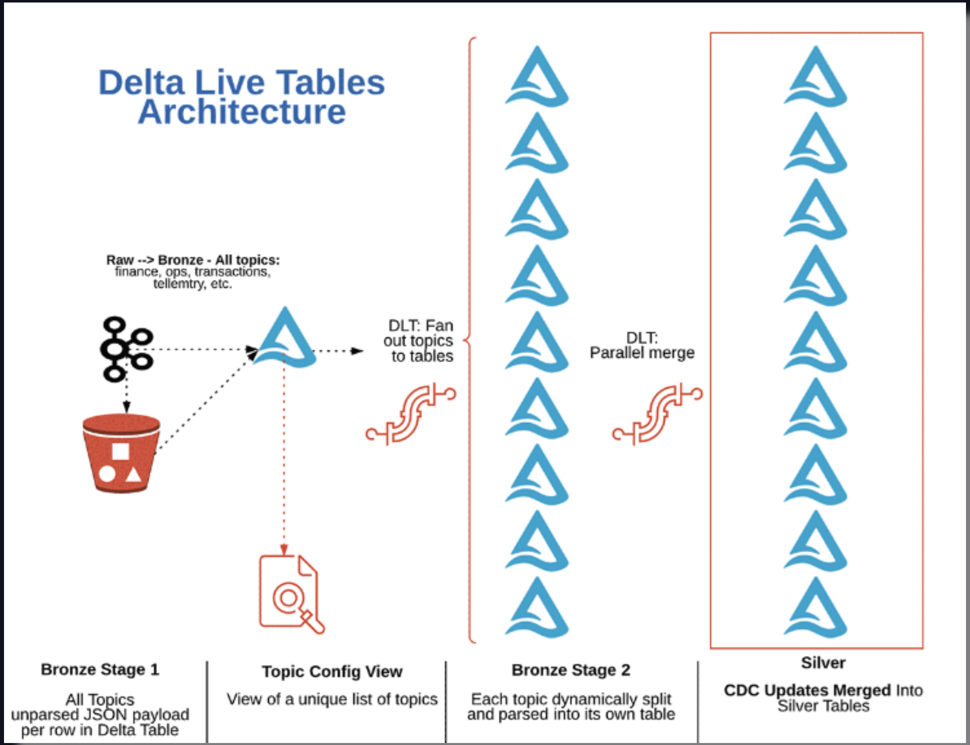
Maintain an up-to-date replica of a table stored elsewhere

- APPLY CHANGES INTO
- SCD type 1 and 2
- Handles out-of-order events
- Maintenance Jobs
    - Small File Problem
    - File rewrite problem/ Deletion Vector



DLT - Multiplexing

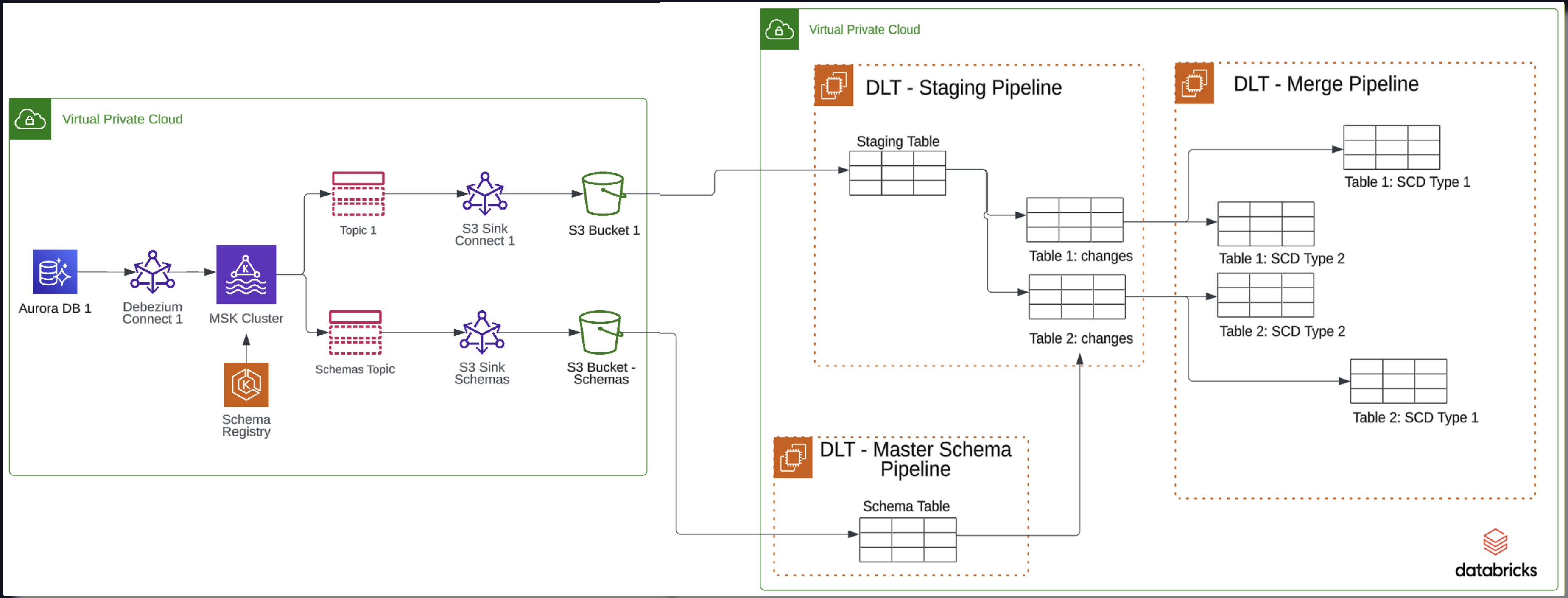
Automatically discovering and process new tables



- Raw Staging Table
- Unpack Distinct Tables
- Collect Changes
- Merge Updates (Apply Changes)
- Declarative and Dynamic

First Iteration

Our Journey to an optimized CDC Architecture

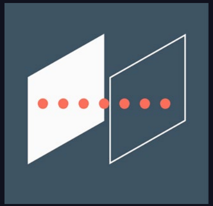


Challenges with First Iteration

Our Journey to an optimized CDC Architecture

Challenges

- Architectural complexity
- Reliability
- Cluster utilization
- Cost



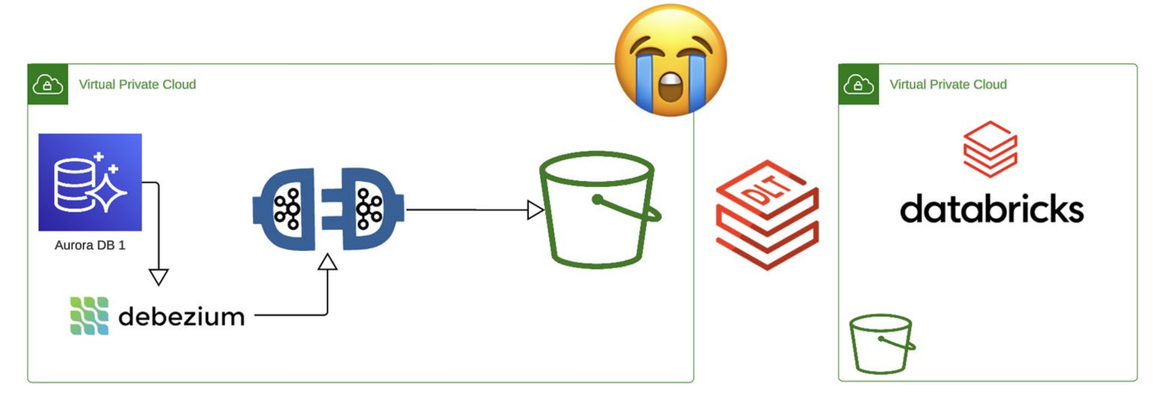
Solution

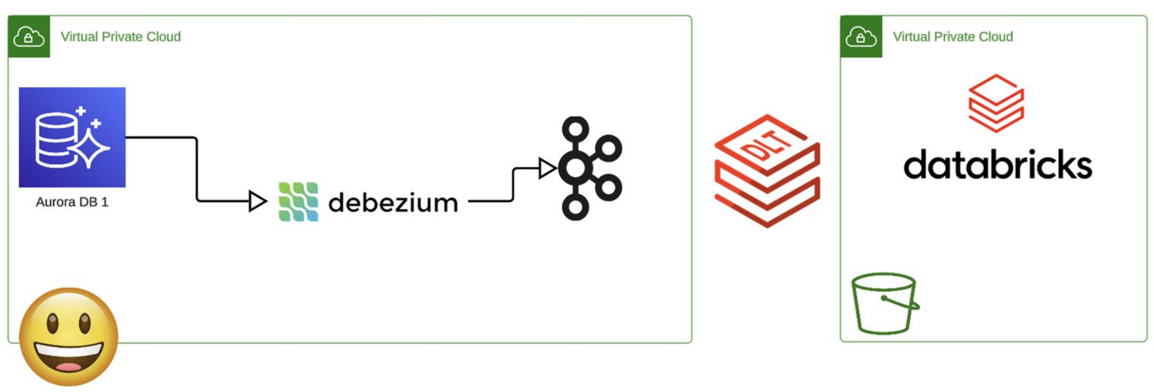
- Moved to Kafka
- Improved Staging Table Design
- Right sized DLT pipelines
- Improved data onboarding process

Simplified Ingestion

Moving to Kafka reads from s3 ingestion

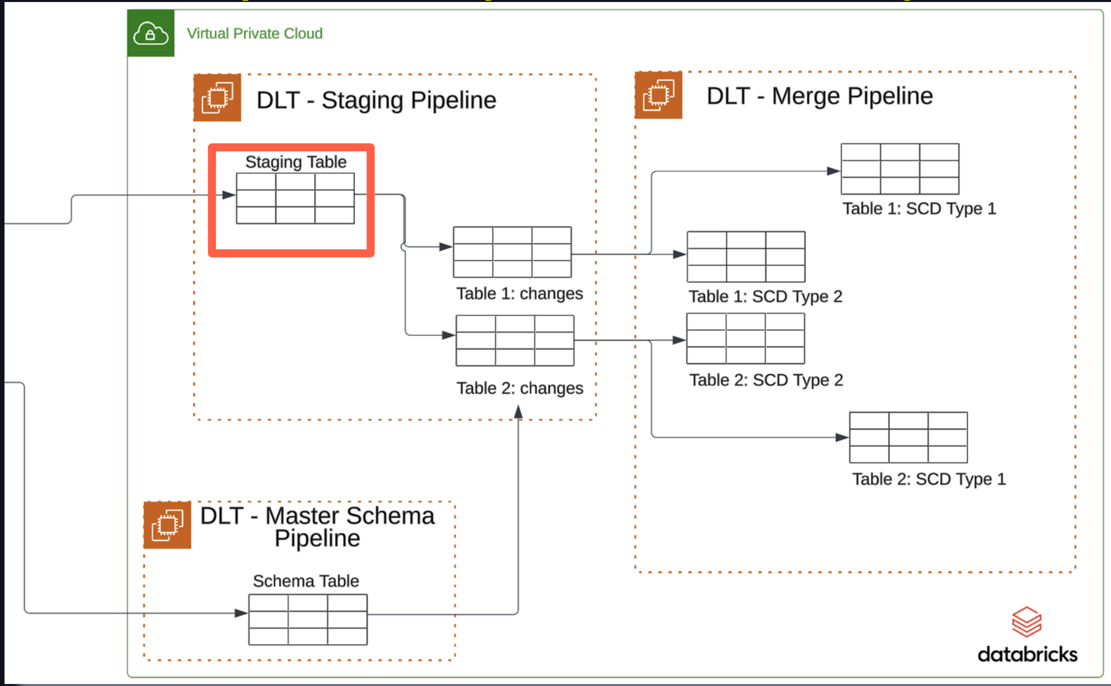
- Kafka s3 Sink connector issues
- Reduced Complexity
- Improved Reliability
- Reduced Costs

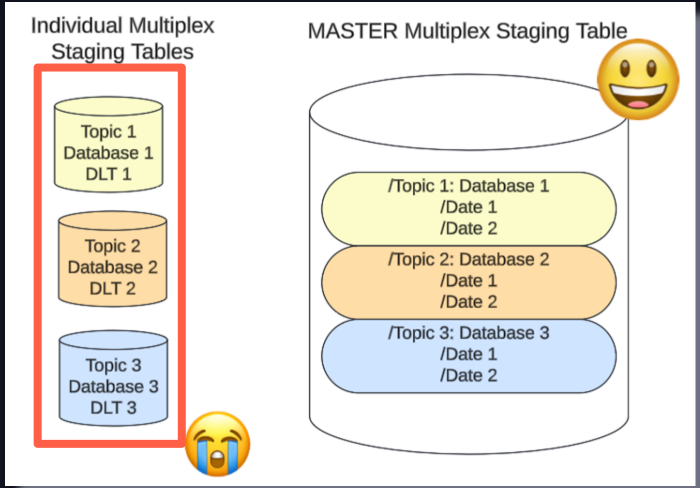




Improved Staging Table Design

Master topics table provided flexibility





Dynamic Table to Pipeline Mapping

Improved Cluster Utilization

- Source Data
- Destination Data
- Table Assignment
- Dynamic
- Declarative Framework

Schema Table

db	tbl	schema
db_a	t1	{id: int}
db_a	t2	{id: int]
db_a	**t3**	**{id: int}**

Pipelines Table

tbl	pipe	count
t1	p1	2
t2	p1	2

Identify new data source

tbl	pipe	count
t1	p1	2
t2	p1	2
new t3	**null**	**null**

Assign to Pipeline

tbl	pipe	count
t1	p1	3
t2	p1	3
new t3	**p1**	**3**

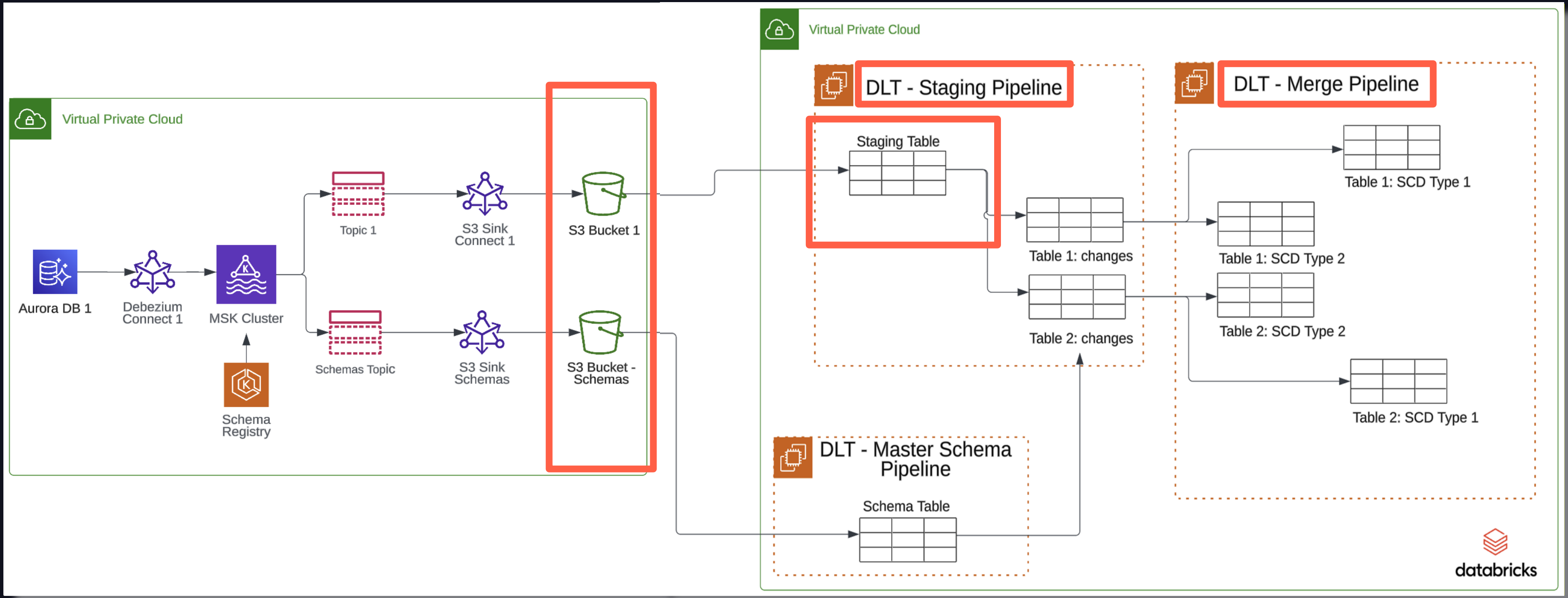
["t1", "t2", **"new t3"**]

```python
def create_table(name):
  @dlt.table(name=name)
  def t():
    return spark.read.table(name)

tables = ["t1", "t2"]
for t in tables:
  create_table(t)
```

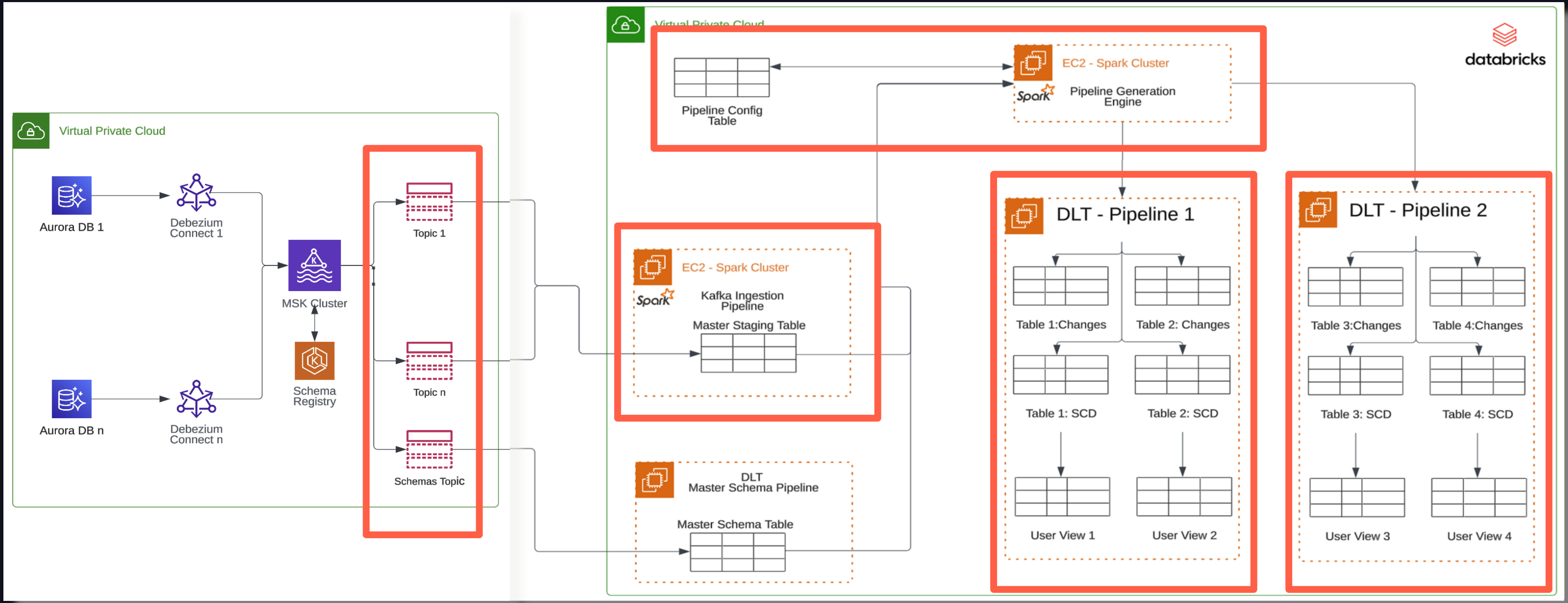
First Iteration Revisited

Our Journey to an optimized CDC Architecture



Improved State Architecture

Our Journey to an optimized CDC Architecture



SUMMARY

Summary

Our Journey to an optimized CDC Architecture

We built......

- Change Data Capture Pipelines
- Kafka as the source
- A Master Staging table for all DBs
- Table Assignment Engine
- DLT Pipelines

And achieved ...

- Latency reduction - By 80%
- Cost reduction – By at least 50%
- Improved Cluster Utilization
- Increased Reliability - Near 0 support tickets
- Manage less Infrastructure

Most of all happy stakeholders !!

DATA+AI SUMMIT