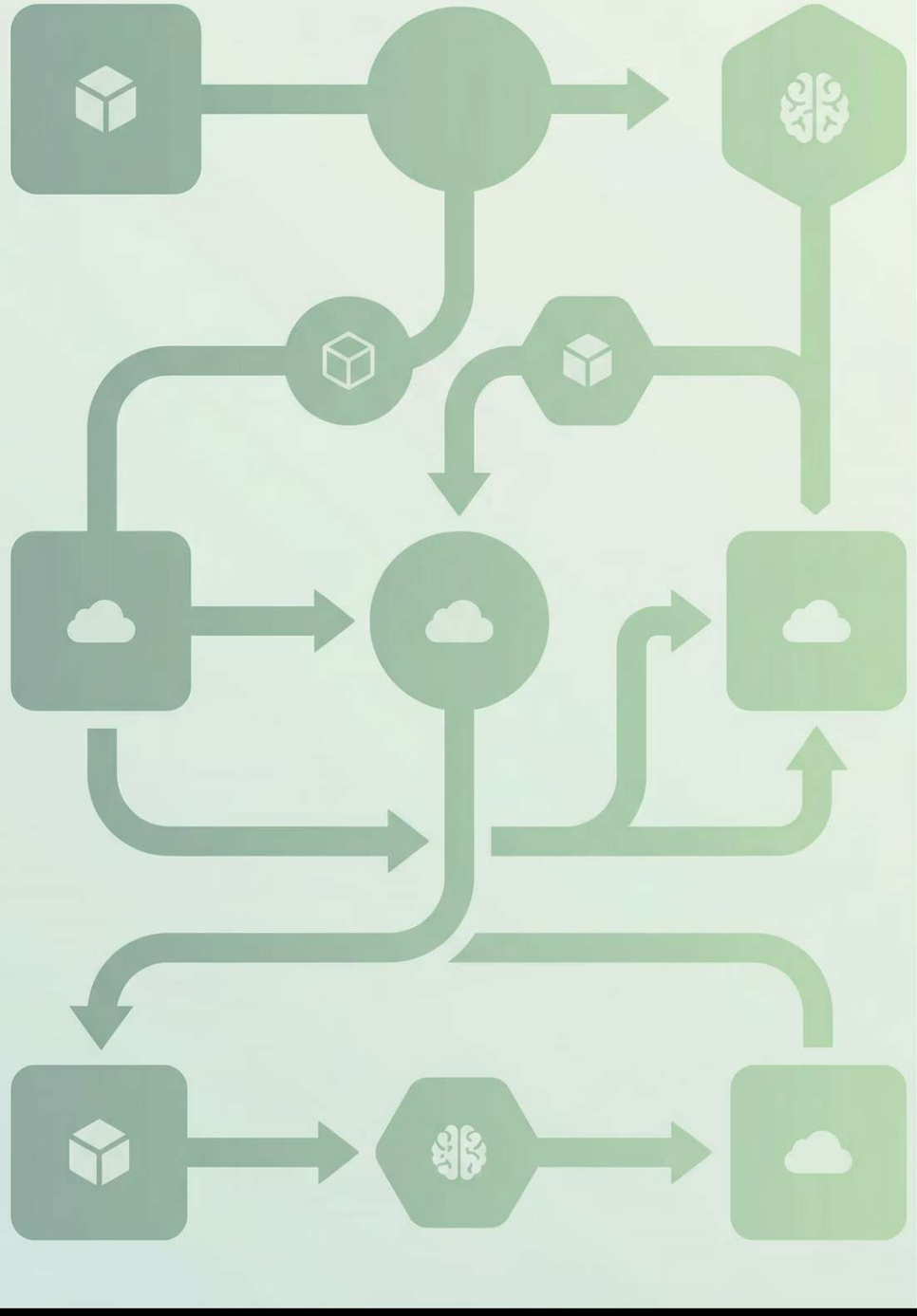


Simplifying CDC: Migrating from Debezium to ClickPipes

A journey from self-managed Debezium to Serverless ClickPipes

Meet the Speaker & The Mission

Abhash Solanki | DevOps Engineer @ Spyne

spyne

Who is Spyne:

Spyne powers the future of automotive retail with **Studio AI** and **Vini AI**, built to help dealerships sell faster and engage smarter.

- **Studio AI** transforms raw vehicle photos into studio-quality images, spins, and videos.
- **Vini AI** acts as an always-on AI sales assistant, answering calls and chats, qualifying leads, and booking appointments 24/7.

About the Speaker:

As a DevOps Engineer, I specialize in AWS, building and operating high-throughput infrastructure that supports Spyne's AI workloads. I've been managing data warehousing at Spyne, working extensively with CDC-based data flows and pipelines. With 1.5 years of experience, I design and manage infrastructure at scale.

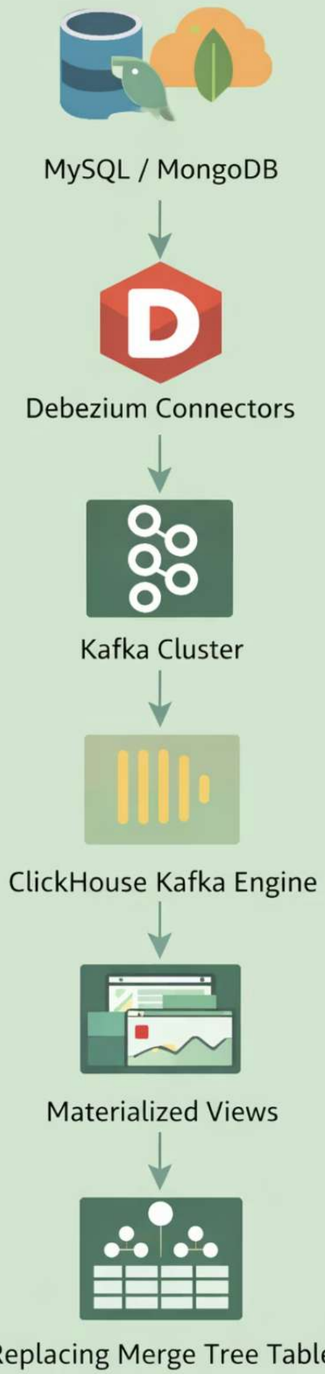
The Context:

With my focus on eliminating "infrastructure toil." I recently led the migration of our massive CDC data pipelines from self-managed Debezium to serverless ClickPipes to solve critical scaling challenges.

The "Before" Architecture: Navigating the Labyrinth

Before our migration, our Change Data Capture (CDC) pipeline was a complex beast, an intricate web of open-source tools and custom configurations. This architecture, while functional, demanded significant operational overhead and constant attention.

> Maintaining Debezium connectors and Kafka offsets was a source of constant operational toil, consuming valuable engineering hours.



The Integration Bottleneck: Manual Onboarding

Why adding a new table used to take hours:

The "Type Mismatch" Nightmare

Mapping flexible MongoDB BSON or MySQL types to strict ClickHouse DDL was a manual, error-prone process. A single mismatched data type could cause silent failures or data rejection.

The "New Table" Ritual

Every new table triggered a DevOps ticket:

- Manually writing `CREATE TABLE` scripts for ClickHouse.
- Updating Debezium connector configurations (JSON).
- Restarting pipelines, risking downtime.

Slow Velocity

Data engineering became a bottleneck for product releases. We spent more time on DDLs than analyzing data.

The Main Pain Point: Schema Drift Nightmares

Our most significant bottleneck was the notorious challenge of schema evolution. In a dynamic development environment, `ALTER TABLE ADD COLUMN` operations in MySQL were common, but our existing pipeline struggled to adapt.

The Silent Killer

When developers added new columns in MySQL, our Kafka Engine Tables would dutifully consume messages with the updated schema. However, the downstream Materialized Views would silently ignore this new data, leading to data inconsistencies and a lack of visibility into critical new features.

Manual Intervention

Each schema change necessitated manual intervention: we had to pause pipelines, manually update view definitions, and often restart entire CDC processes. This introduced significant delays and increased the risk of errors, making every release a high-stakes operation.

The Migration: Embracing Simplicity with ClickPipes

Our search for a more robust and manageable solution led us to ClickPipes within ClickHouse Cloud. The promise of a fully managed service, designed for seamless CDC, was compelling. The shift represented a strategic move away from self-managed infrastructure towards a more streamlined, cloud-native approach.

> The setup time for new data pipelines dropped from meticulous Debezium configuration and Schema mapping to mere minutes within the ClickHouse Cloud UI. This drastic reduction in initial setup effort was a monumental win for our team.

The Big Win: Automated Schema Evolution

One of the most transformative benefits of ClickPipes was its inherent capability to handle schema evolution automatically. This feature directly addressed our most pressing pain point, eliminating the manual toil and associated risks.

Real-time Detection

ClickPipes automatically detects upstream schema changes in sources like MongoDB and MySQL. This includes operations such as adding new columns, which previously caused significant disruption.

Instant Application

Upon detection, ClickPipes applies these schema modifications to the corresponding ClickHouse tables immediately, without requiring any manual intervention or pipeline restarts.

Boosted Velocity

This zero-touch maintenance for schema changes drastically improved our velocity, allowing features to be deployed and data to be analysed with unprecedented speed and reliability.

The Reality Check: Managed Services Aren't Magic

While ClickPipes provided significant advantages, the transition to a "managed" service didn't equate to "zero engineering." We quickly learned that even with advanced platforms, a deep understanding of underlying mechanisms and thoughtful adaptation of operational patterns were still critical.

> We encountered specific architectural nuances and behaviours that required us to evolve our SQL patterns and operational awareness to fully leverage the benefits of ClickHouse Cloud. This was less about fixing flaws and more about optimising our approach for the new paradigm.

Challenge 1: The "Snapshot" Block

The first operational risk we uncovered was the behaviour of ClickPipes when adding new tables, particularly large historical ones. Understanding this interaction was crucial for maintaining continuous data flow.

Pipe Pauses During Snapshot:

Adding a new large table to an active ClickPipe causes the entire pipe to pause its Change Data Capture (CDC) streaming. The pipe enters a "snapshotting" phase for the new table, during which no real-time data is processed for any tables within that pipe.

Mitigation Strategy:

Our mitigation involved isolating large historical tables into separate, dedicated ClickPipes. This ensured that our primary, real-time data streams for frequently updated tables remained unaffected and continued to flow without interruption.

Challenge 2: The Consistency Trap with ReplacingMergeTree

ClickPipes leverages ClickHouse's `ReplacingMergeTree` table engine for efficient data ingestion and deduplication. While powerful, it introduces a concept of eventual consistency that caught us off guard initially.

Eventually Consistent Data:

> When querying directly with `SELECT *`, we would frequently observe duplicate rows, representing old and new versions of the same record before the merge process completed. This was unexpected behaviour for immediate consumption.

> Our solution was to implement `SELECT ... FINAL` in all downstream queries. This forces ClickHouse to perform the deduplication during the query execution, guaranteeing only the latest version of each row is returned.

> The trade-off was increased CPU usage on the ClickHouse cluster due to the on-demand deduplication. We had to carefully balance between data accuracy and query performance, optimising our queries for this new reality.

Challenge 3: The "All-or-Nothing" Recovery Risk

Handling Outages: Partial Snapshots vs. Full Resyncs

> **The Nightmare Scenario: The Binlog Gap**
> A source database outage (like an AWS RDS failover) can cause a loss of binary logs before the CDC consumer can catch up. This breaks the CDC chain, invalidating the replication slot and halting data flow.

1

The Old Way (Debezium): Surgical Recovery

With self-hosted Debezium, we maintained granular control over our data recovery process. This allowed for targeted, efficient solutions:

- We could trigger "Partial Snapshots" to specifically repair missing data chunks.
- Through "Ad-hoc Blocking," we could isolate and fix issues without affecting the integrity or flow of other pipeline segments.

This meant we only re-processed what was absolutely necessary, keeping costs and latency minimal.

2

The New Way (ClickPipes): The "Nuclear" Option

Managed ClickPipes, while powerful, currently lack these granular recovery controls. This presents a significant challenge during critical incidents:

- If a stream breaks due to log loss, the standard recovery path often necessitates a **Full Resync** of the entire table (or even the whole pipe).
- This forces us to re-ingest Terabytes of historical data, incurring massive ingress costs and leading to high latency while the snapshot rebuilds.

The lack of surgical repair options means a small gap can trigger a very expensive and time-consuming "all-or-nothing" recovery.

Conclusion & Key Takeaways

Our journey from a self-managed Debezium/Kafka CDC pipeline to ClickPipes on ClickHouse Cloud was a significant engineering undertaking. While not without its challenges, the strategic trade-offs have yielded substantial benefits for Spyne.

1

Eliminated Operational Toil

We successfully removed the burden of managing debezium configurations, kafka and complex schema mappings.

2

Solved Schema Drift

Automated schema evolution with ClickPipes proved to be a game-changer, drastically improving our velocity and data consistency.

3

Adapted Engineering

We learned to optimise our queries for `ReplacingMergeTree` and strategically manage costs under ClickHouse Cloud's consumption pricing model.

> The trade-offs were undeniably worth it for the enhanced stability and significant speed gains we achieved in our data pipelines at Spyne.