CDAO Canada

Building a Data-Driven Platform

A Case Study in Meeting Real Business Needs

Milind Joshi, CTO, Foxquilt

When the Status Quo was “Buy Whenever Possible”

- Features too hard to build
- Some teams tried - gave up
- Fragile infrastructure
- Bought the slick marketing
- Upfront + annual SaaS fees
- Limited to the vendor tools
- Vendor decides supported integrations
- Business works around data platform
- Vendor jacks up the prices
- Too late - **stuck**

What is your business? Wrangling Data?

- Data platform constraints drives Business Ops
- Using a relational database? Want to pivot?
  - Schema Change!
- Using a document database? Need analytics?
  - ETL/ELT!
- Data platforms were always a constraint
- Data engineering was super expensive
- Everything took too long and cost too much money
- Distraction from the main business

The New "Build vs. Buy" Equation

The "Buy" side traditionally won on speed → Often results in vendor lock-in and feature bloat

AI-driven development (Copilots, automated CI/CD, and Infrastructure-as-Code) shifted equilibrium

The Old Way

Building required hiring a team of 10 for 12 months, while buying required a 3-month implementation.

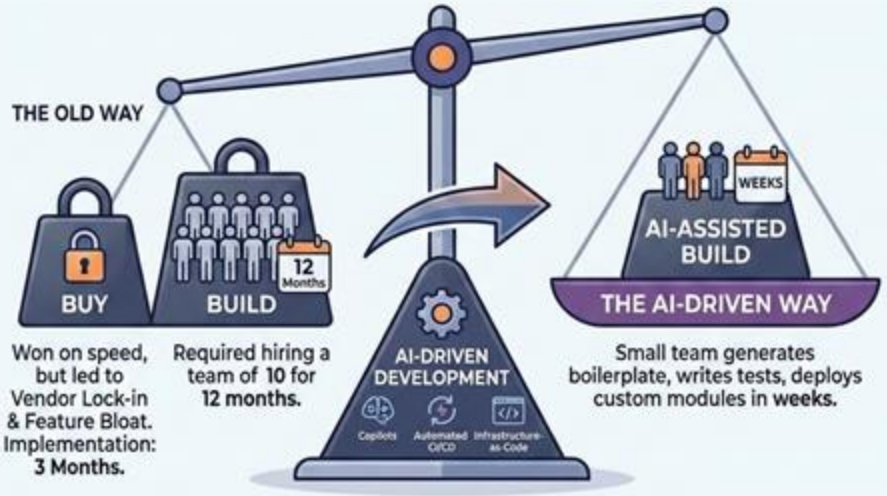
The AI-Driven Way

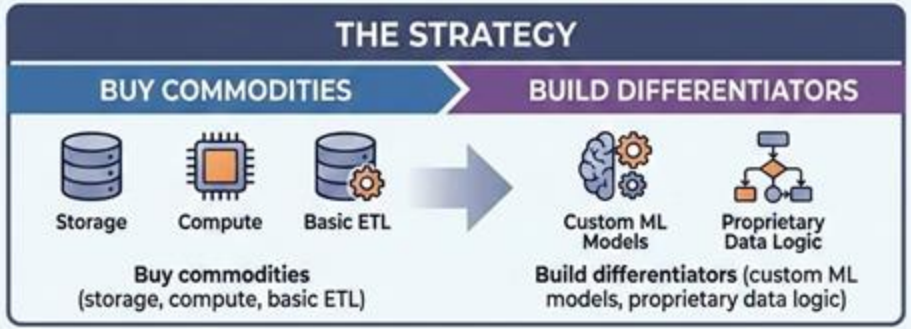
AI tools enable a small team to generate boilerplate, write tests, and deploy custom modules in weeks.

The Strategy

Buy commodities (storage, compute, basic ETL) and build differentiators (custom ML models, proprietary data logic).

The New "Build vs. Buy" Equation





Fitting the Platform to the Business

Don't let vendor-dictated, opinionated architecture control your business processes; your data platform should reflect your organizational DNA.

Business-First Design

Map your value stream before choosing a database. Avoid forcing real-time needs onto cheaper, batch-processing platforms.

Rigidity Trap

Off-the-shelf platforms often force data normalization that loses industry-specific context.

Modular Architecture

Use a composable stack so components can be swapped out without rebuilding the whole ecosystem.

Architecture for Real-World Problems

Designing a platform isn't about the "coolest" tech. It's about closing the gap between raw data and a business decision.

Core Architectural Principles:

Decoupling:

Separate storage from compute to scale independently.

Idempotency:

Ensure that running a data process multiple times yields the same result (crucial for trust).

Data Contracts:

Define clear "promises" between data producers and consumers to prevent broken dashboards when upstream schemas change.

Enabling Day-to-Day Decisions

Data is useless if it sits in a lake.

From Dashboards to Actions

Move beyond "passive" dashboards. Modern tools now allow for "reverse ETL," where data insights are pushed back into operational tools so employees can act on them immediately.

It needs to be packaged into **data products**.

Self-Service Analytics

Use AI natural-language interfaces (text-to-SQL) to allow non-technical managers to query the data platform without waiting for a data analyst.

The KPI Library

Centralize definitions so "revenue" means the same thing to Finance as it does to Sales.

Governance, Trust, and Sharing

Automation has turned governance from a "bottleneck" into an "accelerant."

- **Automated Governance:** Use AI to automatically tag PII (Personally Identifiable Information) and apply maskings, rather than relying on manual audits
- **The Trust Framework:**
  - **Lineage:** "Where did this number come from?"
  - **Quality:** "Is this data fresh and accurate?"
  - **Access:** "Should this user see this?"
- **Secure Sharing:** Consider utilizing "zero-copy" data sharing to collaborate with partners without actually moving or duplicating sensitive data.
- **Copy Easy:** But if you actually have to make a copy - don't be nervous, it is not as difficult or expensive as it was before.

Case Study | The foxquilt Velocity Stack

MongoDB + Python + AWS automation accelerated our build speed. To match product speed, we chose a flexible, ergonomic stack over rigid planning.

MongoDB (Schema-on-Read)

- Avoided "stop-the-world" SQL migrations.
- Document model handled evolving data without constant DDL changes.
- Impact: New features launched and stored same day; no DBA needed.

Python (Universal Language)

- Seamlessly integrated (FastAPI, Pandas/Polars) as the single source of logic (ingestion to transformation).
- AI assistants rapidly generated complex scripts and unit tests.

Serverless Automation (AWS)

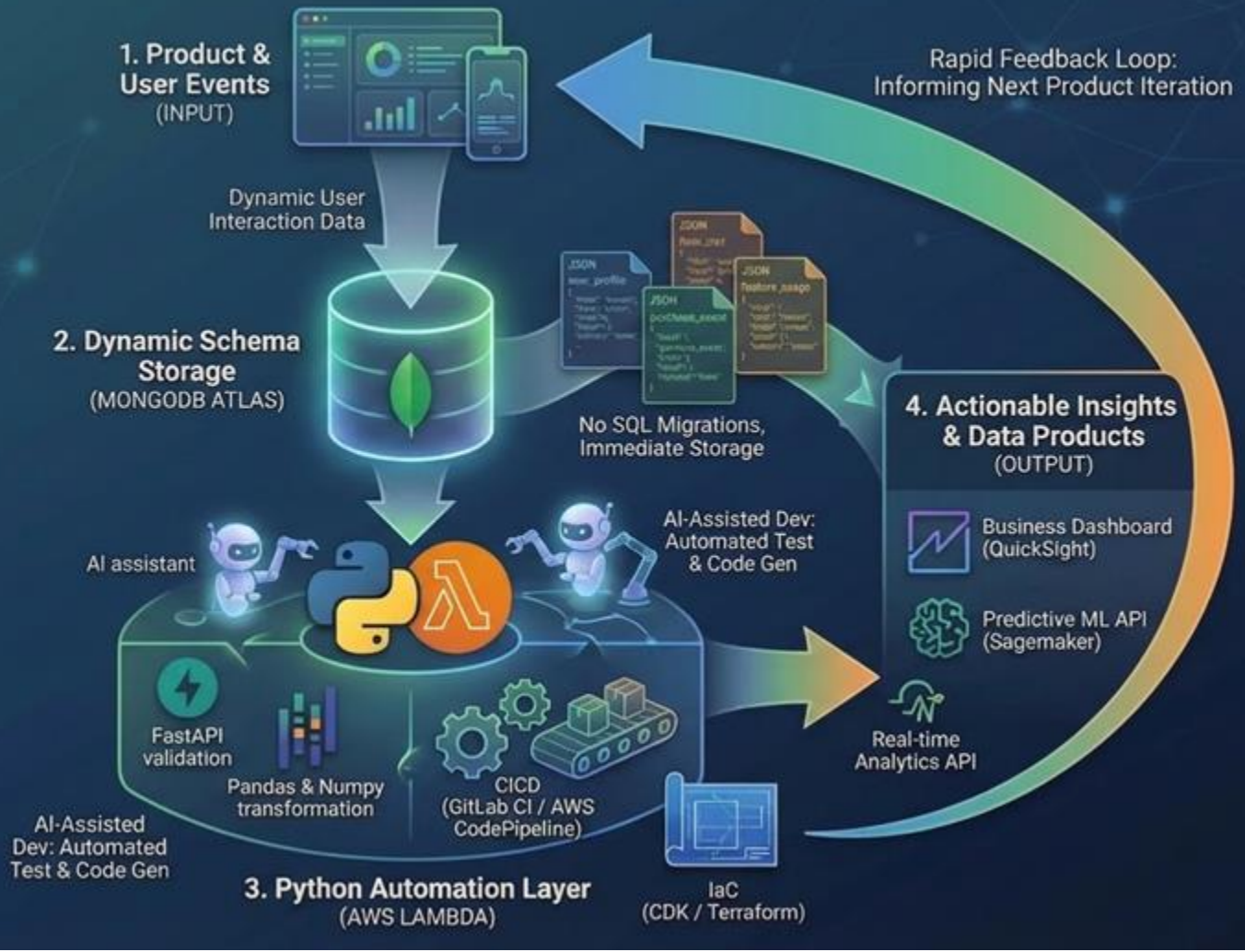
- Shifted from scheduled jobs to real-time, event-driven pipelines (Lambda, EventBridge).
- IaC (CDK/Terraform) defined the environment, allowing new staging environments in <10 minutes.

Active Data Flywheel

Velocity Booster
AI-ASSISTED DEVELOPMENT

Velocity Booster
AWS SERVERLESS AUTOMATION

Velocity Booster
FLEXIBLE PYTHON PIPELINES



The "Flexibility Tax" | Bridging the Gap

Nested JSON vs. relational analysis: The bottleneck

The Relational Problem: BI tools (Tableau, PowerBI) expect rows/columns and can't handle nested data/dynamic schemas.

The Skills Gap: Analysts prefer SQL. Forcing them to learn MongoDB's Aggregation Framework ($unwind, $lookup) a major hurdle.

The Reality: Without automated "flattening," fast development is negated by slow analyst work.

MongoDB + Python speeds dev but slows down data analysis.

Solution | The Translation Layer

To maintain velocity without blinding the business, we must build bridges:

Easiest: Automated flattening (Python/AWS Lambda to materialize nested docs into flat tables).

Harder: Semantic mapping (Define metrics once over fluid schemas).

Hardest: SQL proxies (Drivers for standard SELECT queries against NoSQL collections).

foxquilt | We Are in Commercial Insurance

- Operating in Canada & US
- Highly regulated, with stiff penalties and fines for non-compliance
- Highly data driven business
- Need to move fast to compete against the incumbents
- Able to launch first iteration of platform within 6 months
- Continuously iterated for over 5 years
- Monthly releasing new features
- Average Cost / Transaction < 50c
- Average Cost / Dashboard update < $1
- Fast moving team, even if relatively small

Conclusion | The Modern Data Platform Development Equation

Balancing velocity with visibility: The shift to "Build/Hybrid" is about owning competitive advantages, not just technology.

Core Takeaways:

- **AI the Great Equalizer:** Automation and AI reduce cost of building perfect custom data products.
- **The Velocity Stack:** Rapid iteration (MongoDB, Python, AWS) needs a business "Translation Layer."
- **Build for Differentiation:** Buy infrastructure (Cloud, Storage); build unique business logic.
- **Trust through Governance:** Automated governance and data contracts maintain integrity at speed.

The Path Forward

Identify high-value gaps where purchased platforms compromise the business.

Automate "boring" parts (AI, IaC) so teams focus on data products.

Bridge the gap early: Build automated flattening and SQL-friendly views into sprints.

Final Thought: Successful companies evolve data platforms at the speed of their customers.