Building an End-to-End ELT+CDC Data Lakehouse prototype with Iceberg.

5 min read · Nov 4, 2024

Oleg Ustimenko Follow

Introduction.

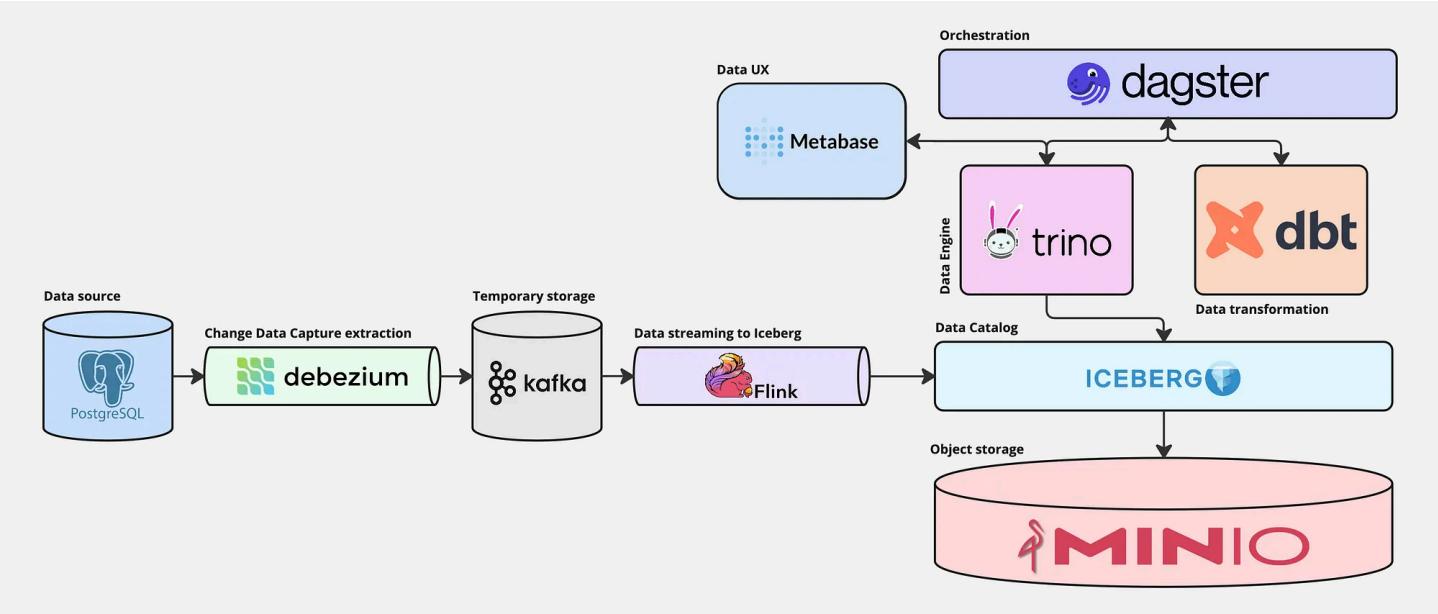
A few months ago, I decided to dive into Apache Iceberg, intrigued by its capabilities for building a cloud-agnostic data lakehouse. After researching, prototyping, and refining, I'm excited to share my experience creating an end-to-end data streaming lakehouse prototype!

This project demonstrates a complete pipeline — from data extraction in a source system to storage and analysis in a lakehouse format — designed as a flexible alternative to traditional data warehouses.

Check out the GitHub page to explore and run the project yourself.

Architecture Overview.

Here's a quick look at the architecture:



Core concepts of the solution:

- **ELT (Extract, Load, Transform) pipeline.** In this setup, raw data flows directly to the destination before transformations are applied, forming the foundation for flexible transformations and future analysis. The ELT approach allows us to keep an untouched “raw” layer in our data model, letting us experiment with different transformation methods or analyses as needed.
- **CDC (Change Data Capture).** Data changes are extracted in real-time from the PostgreSQL database’s WAL (Write-Ahead Log) by Debezium and sent to Kafka. This method provides a full history of changes to the data.
- **Headless data architecture.** With Apache Iceberg as the data backbone, this architecture enables the use of various processing engines thanks to **the decoupling of data storage from data processing.** The data resides in an S3-compatible object storage (MinIO), and Iceberg provides a unified structure accessible by different engines. In this project, we use Flink for real-time data streaming and Trino for SQL-based analytics and transformation.

Let’s break down each component:

- **PostgreSQL:** Serving as our source database, PostgreSQL holds the data we want to replicate and analyse in real time.

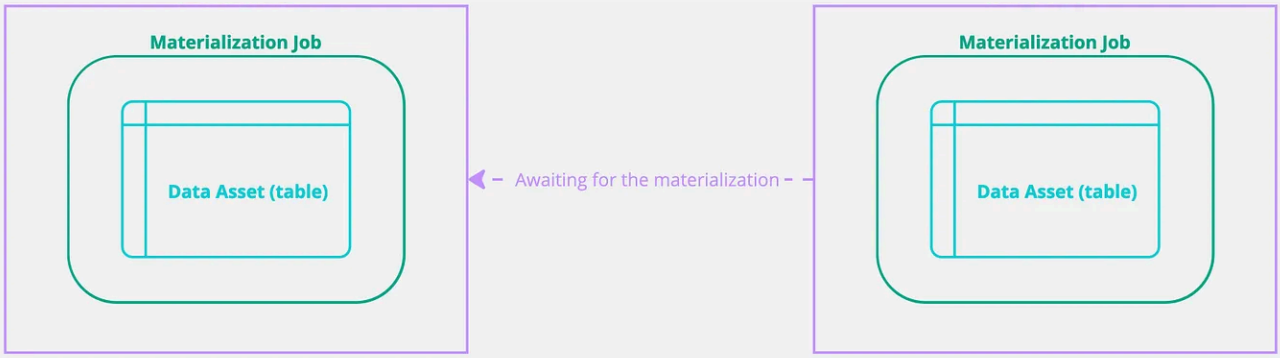
- **Debezium Connector:** A change data capture (CDC) tool, Debezium extracts real-time changes from PostgreSQL and streams them into Kafka.
- **Kafka:** Acting as a temporary storage layer, Kafka holds the streamed CDC data from Debezium before further processing.
- **Flink:** As a streaming engine, Flink ingests data from Kafka, performs any required transformations, and loads the data into Iceberg tables.
- **Iceberg:** Iceberg provides a high-performance table format for big data analytics. In this project, I used Iceberg's REST API catalog, but there are other alternatives like Nessie, Hive, AWS Glue etc.
- **MinIO:** As a local S3-compatible object storage solution, MinIO hosts Iceberg data files, simulating the cloud environment.
- **Trino:** Trino enables SQL-based queries on Iceberg data, making it easy to perform complex analytical queries across large datasets.
- **DBT:** For data transformations within Iceberg, DBT provides a straightforward, SQL-first workflow that's ideal for managing transformations.
- **Dagster:** Dagster orchestrates the pipeline, coordinating the DBT transformations as well as running data optimization tasks on schedule basis.
- **Metabase:** Finally, Metabase offers an intuitive user interface for visualising and analysing the data, helping bring the story behind the data to life.

Automating Data Transformations with Dagster

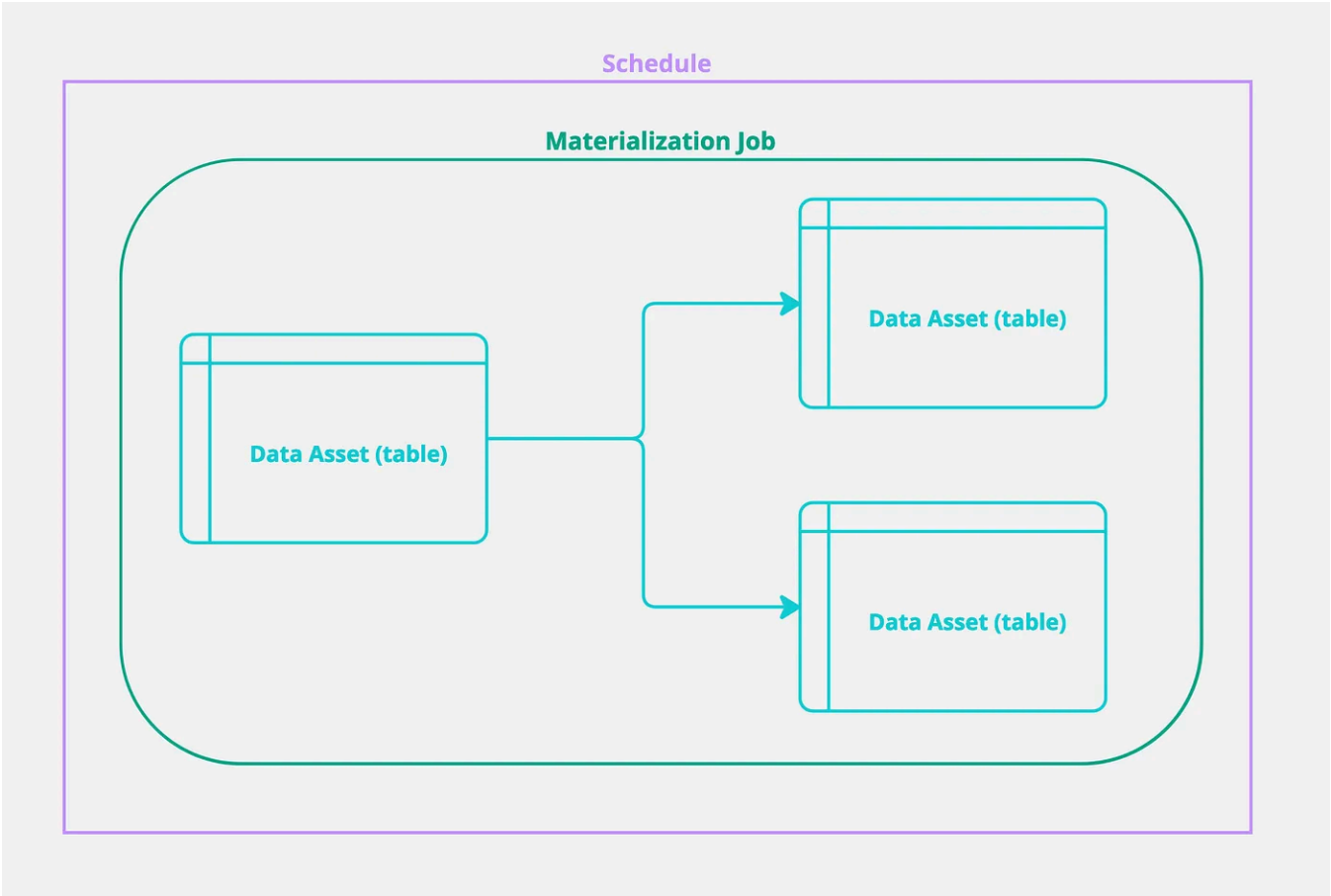
Leveraging the Factory Pattern in Dagster

In this architecture, Dagster orchestrates each dbt model as a **data asset.** Each data asset has a dedicated **job** for materialization and its own **sensor** to monitor upstream asset statuses. These sensors play a vital role, watching for changes in upstream tables and triggering new jobs automatically when necessary. This setup makes it easy to declaratively add new data transformation pipelines for each new table in the raw data layer without restructuring the entire codebase.

By using the Factory design pattern, assets, jobs, and sensors are dynamically



While each data asset in this setup has its own dedicated materialization job and sensor, it's also possible to group multiple data assets under a single job and use scheduled runs instead of sensors. The choice between individual jobs and grouped assets depends on factors like resource utilization and the timing requirements for data availability — whether updates are needed every 10 minutes or just once a day, for instance.



Data Storage Optimization in Iceberg

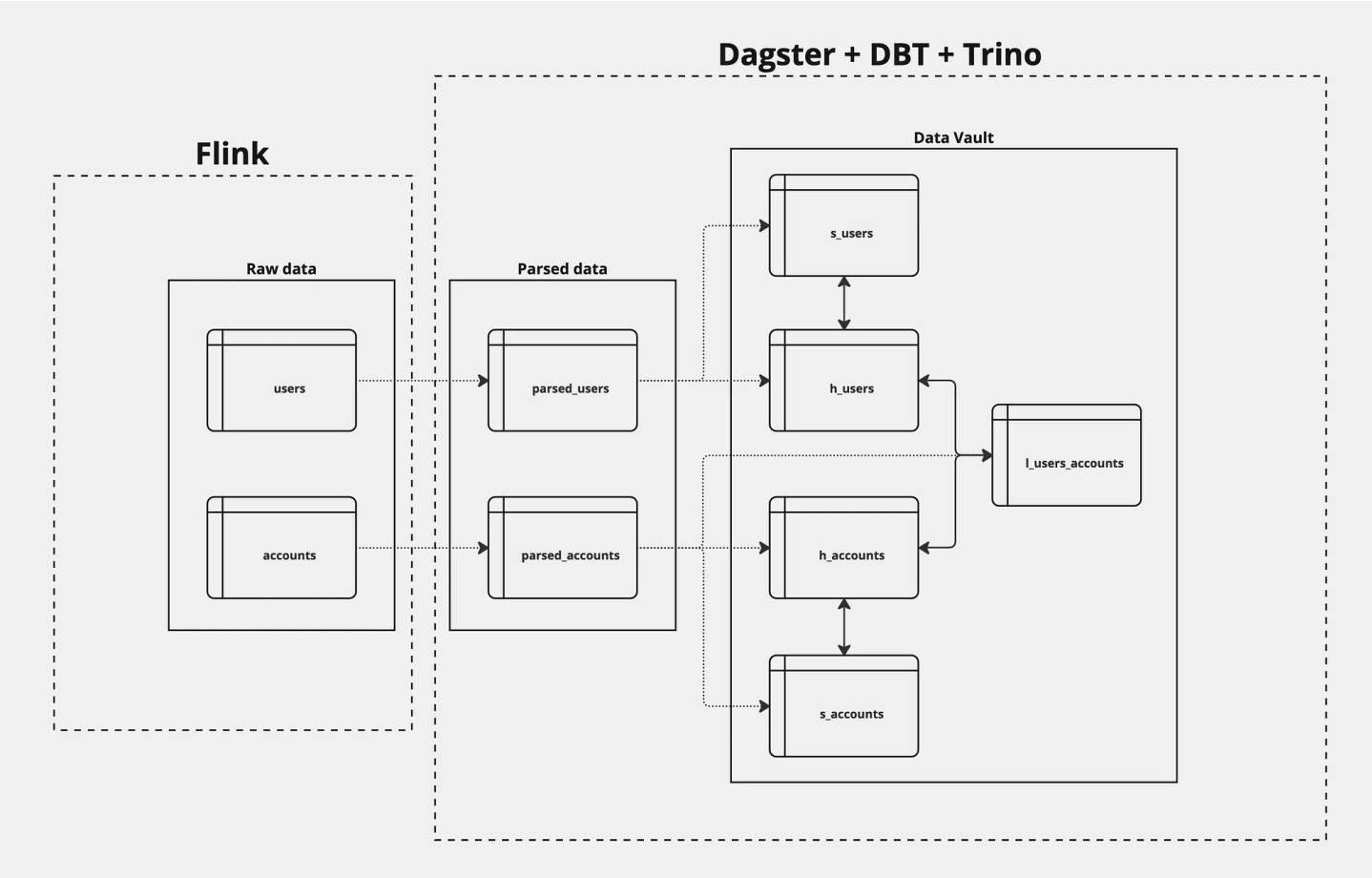
To keep the storage layer efficient and manageable, I've implemented several **data storage optimization jobs** in Dagster. These jobs run on a scheduled basis and

handle essential maintenance tasks, ensuring that the Iceberg tables remain optimized for performance and storage. Here's an overview of these optimizations:

- **Data Compaction**: Consolidates small files into larger ones, improving query performance by reducing the number of files read.
- **Snapshot Expiration**: Removes outdated snapshots to control metadata growth and reduce storage costs.
- **Orphan File Removal**: Cleans up files that are no longer referenced by the Iceberg metadata, keeping the storage layer organized.
- **Dropping Extended Stats**: Purges extended statistics periodically to avoid accumulation that can slow down operations.
- **Cleaning Temporary dbt Views**: Removes temporary views created during dbt transformations from the object storage to keep the environment clean and efficient.

Data model

The data model in this project is structured in three distinct layers to manage and organize data effectively, allowing for flexible transformations and historical analysis:



- **Raw Data:** This is an append-only layer where data is ingested from Apache Flink. Here, data arrives as-is, creating a complete and unaltered record of all events for downstream processing.
- **Parsed Data:** Using dbt incremental models, data in this layer is parsed into a flat, structured table. Additional metadata is added, which is valuable for further analysis and transformations. This layer serves as a prepared base for more refined data modeling.
- **Data Vault:** In this layer, dbt incremental models are used to create hubs, satellites, and links. This model keeps a complete history of all changes, preserving the data lineage and historical records in the satellites, ensuring that changes in the source data are tracked over time.

Conclusion

This project aims to provide a working, end-to-end prototype that combines several powerful technologies and concepts. While it's not production-ready and has ample room for optimization, tuning, and refactoring, it serves as a solid foundation for understanding how these components work together. As a local, easily deployable

playground, this setup offers an accessible way to experiment with modern data architecture patterns and explore the capabilities of a data lakehouse in action.

Check out the GitHub project to dive in and see how these tools and approaches can be adapted, expanded, and refined for further use cases.

Lakehouse Architecture | Apache Iceberg | Dagster | Data Streaming | Data Vault

Follow

Written by Oleg Ustimenko

11 followers · 2 following

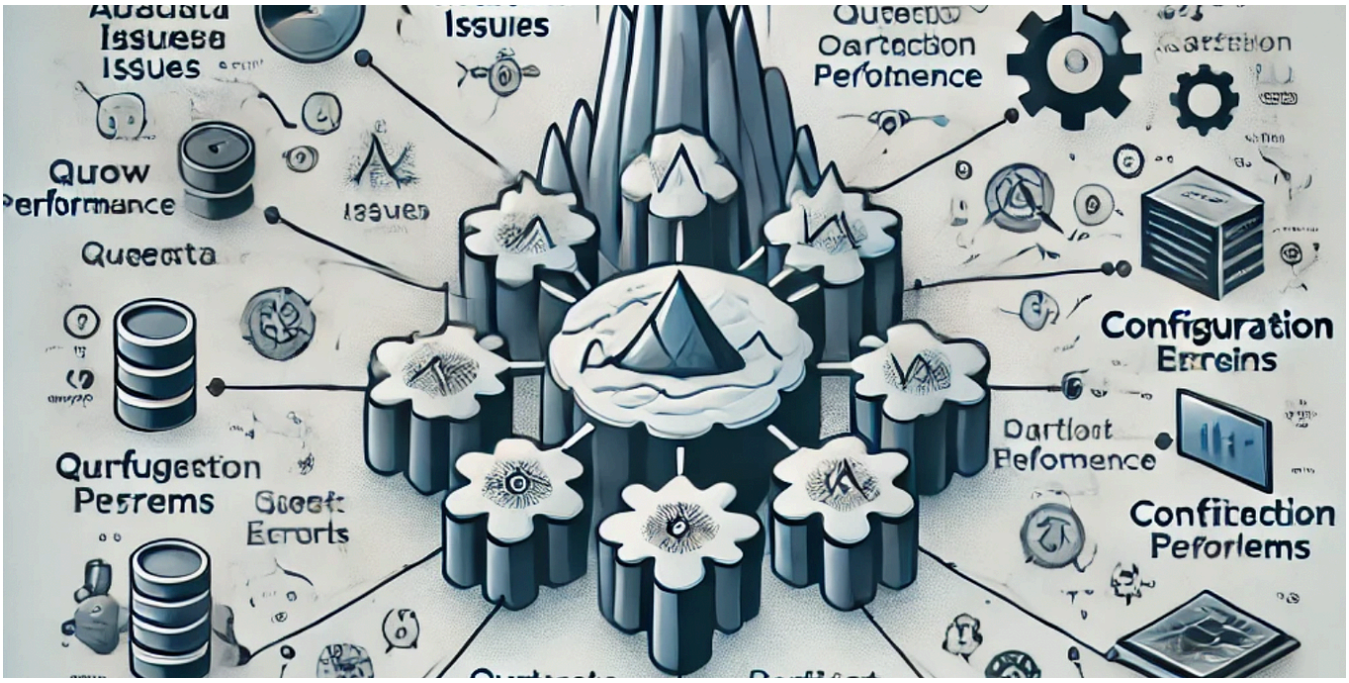
Hi! I am a Data Engineer.

No responses yet

Dat Hoang

What are your thoughts?

Recommended from Medium



Vaibhav Srivastava

Apache Iceberg: Troubleshooting

Table of Contents

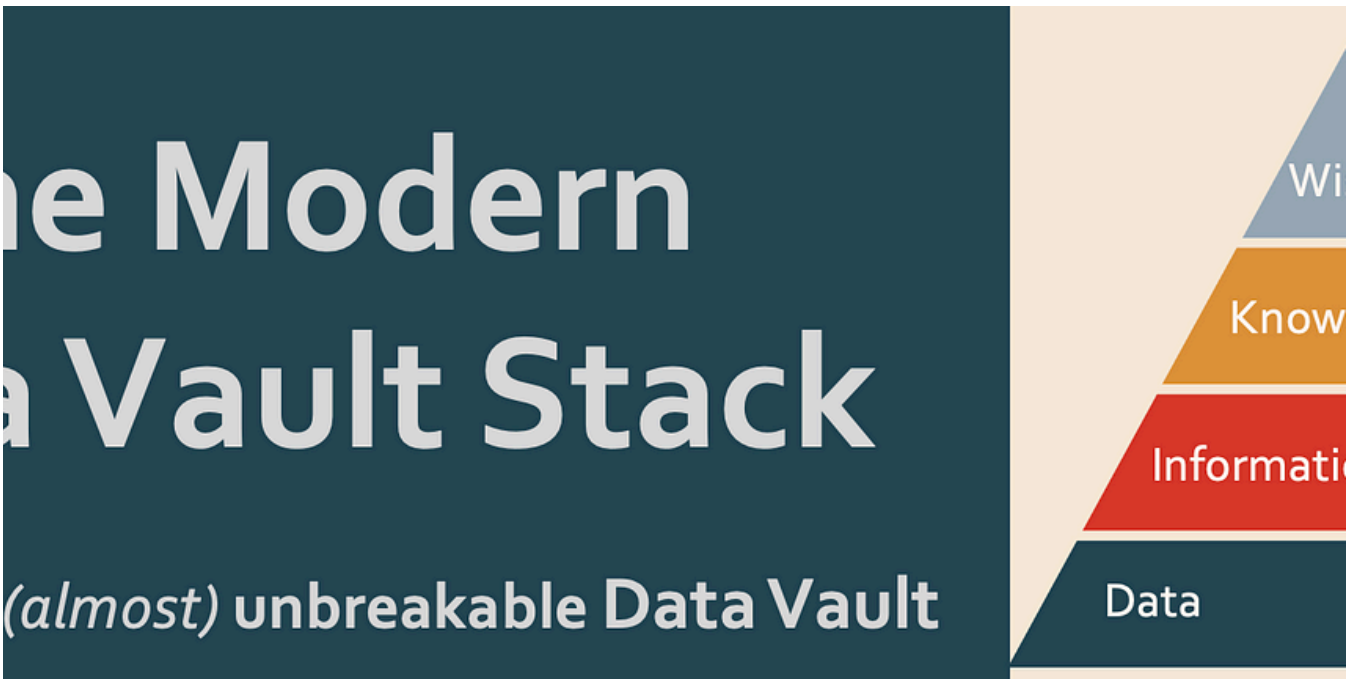
Dec 21, 2024 1

Soumil Shah

Learn How to Use DuckLake with DuckDB to Build Datalakes on S3

DuckLake makes building and managing lakehouses as easy as writing SQL. By combining DuckLake with DuckDB, you can create, update, and...

May 28 31



In The Modern Scientist by Patrick Cuba

the Modern Data Vault Stack

For decades data analytics has been delivered in a standard pattern of zones and layers, each with a special purpose and governed by...

Jan 15 29 2

In Towards Data Engineering by Douglas Souza

Engineer: MinIO Storage

Continuing my articles about building my project and study architecture, I will show how I added MinIO and synchronized it with Spark...

Feb 3 31

Sheikh Wasiu Al Hasib

Building a Modern Data Lakehouse Architecture with Apache Iceberg, Spark, Flink, ClickHouse, and...

In the world of big data, traditional architectures are being replaced by modular, real-time systems that unify batch processing, machine...

Apr 19 1

kubernetes

Sohail Saifi

Kubernetes Is Dead: Why Tech Giants Are Secretly Moving to These 5 Orchestration Alternatives

I still remember that strange silence in the meeting room. Our CTO had just announced we were moving away from Kubernetes after two years...

6d ago 783 51

See more recommendations