SPARK+AI SUMMIT 2020

Organized by databricks

Building Better Data Pipelines for Apache Spark with Delta Lake

Vini Jaiswal

Customer Success Engineer

Awesome TAs

- Emma Freeman (Curriculum Engineer)
- Spencer McGhin (Sr. Customer Success Engineer)
- Joe Widen (Lead Solutions Architect)
- Jean-Marc Spaggiari (Sr. RSA)

Agenda

Time	Topic
9:00 - 9:20	Introduction, Setup, Class Logistics
9:20 - 9:50	What is Delta Lake?
9:50 - 10:00	Break
10:00 - 10:35	Lab 1: Building a Delta Lake Data Pipeline
10:35 - 10:50	Review Lab, Q&A
10:50 - 11:00	Break
11:00 - 11:15	Delta Lake Features
11:15 - 11:35	Lab 2: Exploring Delta Lake Features
11:35 - 11:55	Summary, Q&A

Getting started:

- Login to the workspace
- Familiarize yourself with using chat to communicate with TAs
- Download materials from the Resources tab

By the end of this class you will be able to:

- Explain how Delta Lake works for building data pipelines
- Build a complete Delta Lake pipeline to stream data from raw ingestion to gold tables
- Understand how Delta Lake unifies batch and streaming data in a single pipeline
- Understand how Delta Lake schema enforcement and evolution impacts data writes and updates
- Understand how to perform data deletions to comply with data privacy laws and regulations

DELTA LAKE

What is Delta Lake?

SPARK+AI SUMMIT

Delta Lake is a technology for **building robust data lakes**. It is an **open source storage layer** specifically **designed to work with Apache Spark**.

A data lake built using Delta Lake is **ACID compliant**.

Delta Lake offers:

- ACID transactions on Spark
- Scalable metadata handling
- Streaming and batch unification
- Schema enforcement
- Time travel
- Upserts and deletes
- Fully configurable/optimizable
- Structured Streaming support

DELTA LAKE

Components of Delta Lake

SPARK+AI SUMMIT

Delta Lake is comprised of:

DELTA LAKE

- Delta tables
- The Delta optimization engine
- The Delta Lake storage layer

Delta Tables

Data files

- Parquet format
- Kept in cloud storage

Table registered in the Metastore

- Contains the data schema and metadata

Transaction log

- Kept in cloud storage
- Changes are stored as ordered, atomic commits
- Records every transaction that occurs
- Allows for Time Travel
- Single source of truth

The Delta Optimization Engine

DELTA LAKE

- Thanks to Apache Spark!
- File management optimizations
  - Compaction, data skipping, localized data storage
- Auto-optimized writes and file compaction
- Performance optimization via Delta caching

The Delta Lake Storage Layer

DELTA LAKE

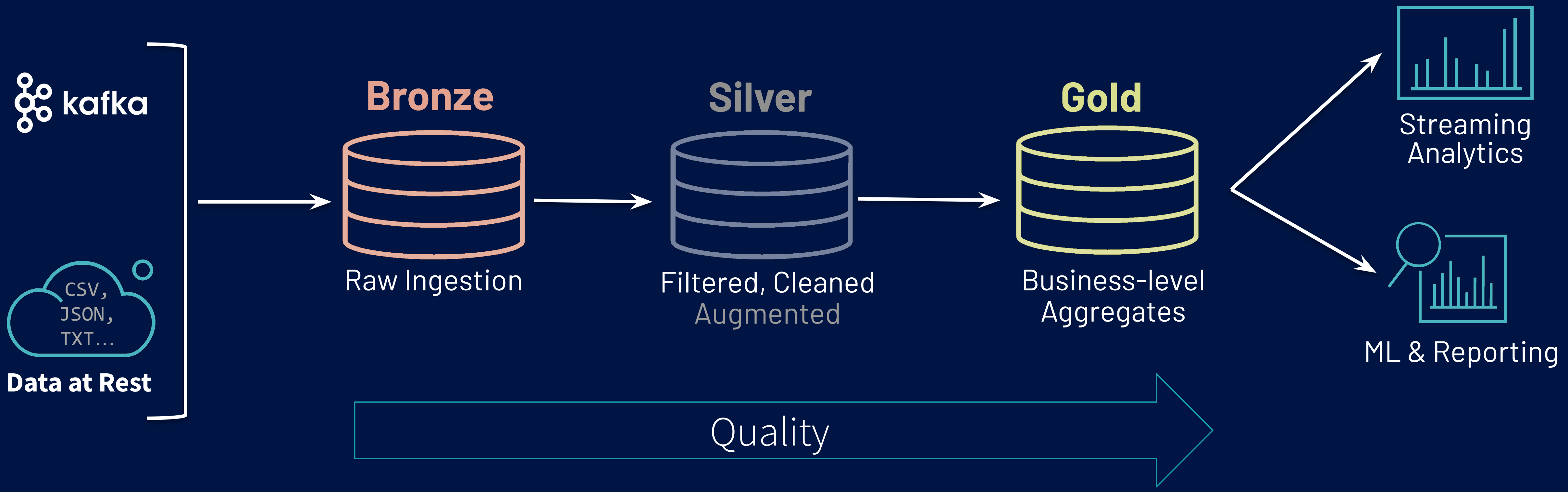
- Highly performant and persistent
- Low-cost, easily scalable object storage
- Ensures consistency
- Allows for flexibility

DELTA LAKE

The Delta Architecture Design Pattern

SPARK+AI SUMMIT

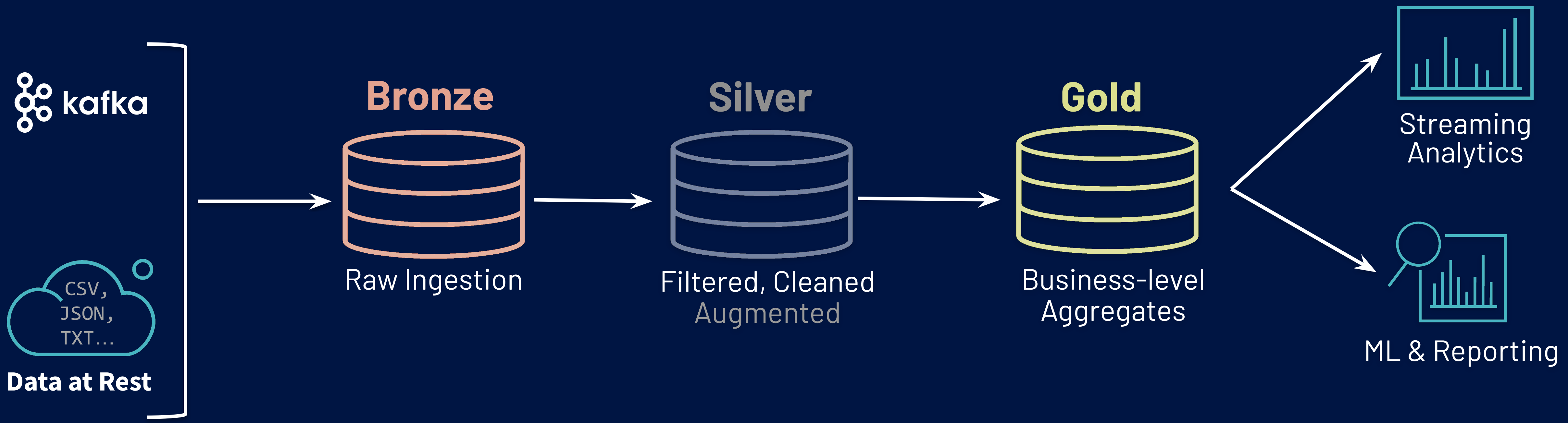
The Delta Architecture



Delta Lake allows you to *incrementally* improve the quality of your data until it is ready for consumption.

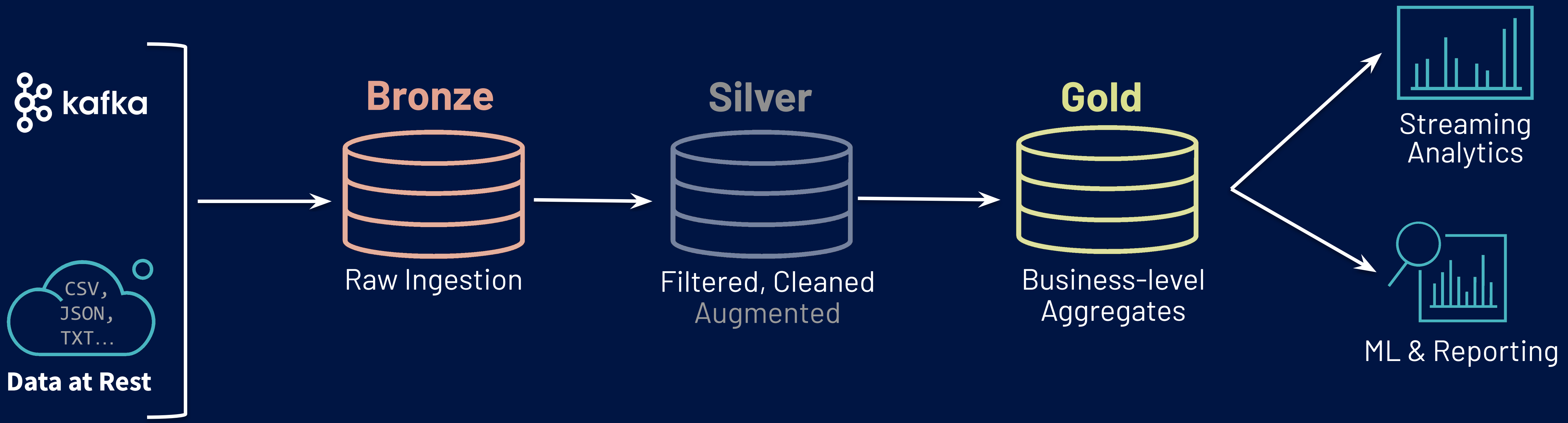
databricks

The Delta Architecture



- Dumping ground for raw data
- Often with long retention (years)
- Raw data with minimal parsing

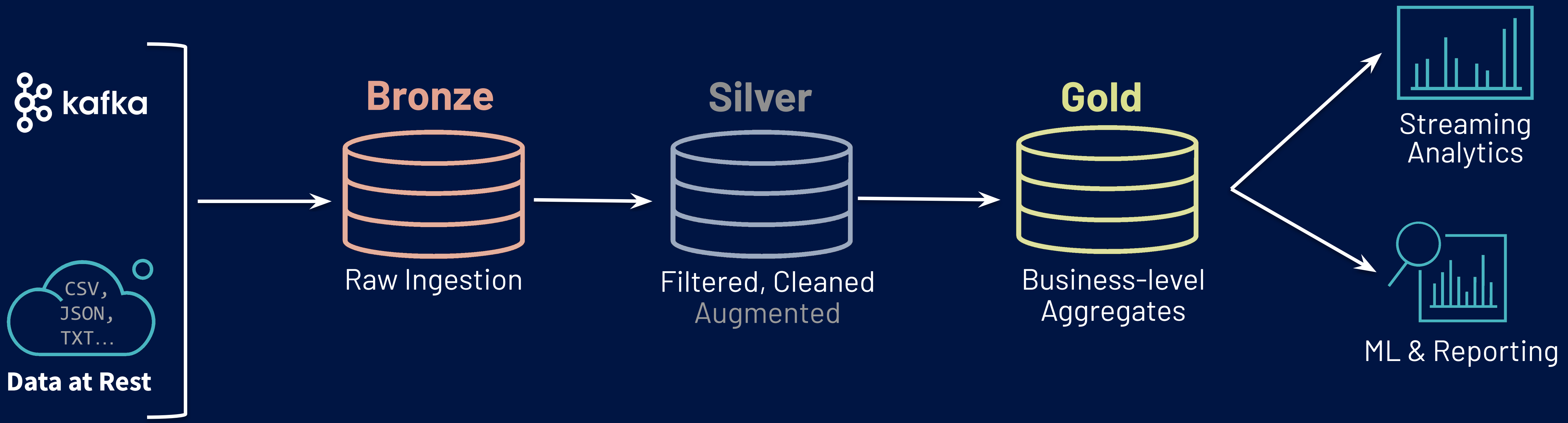
The Delta Architecture



Intermediate data with some cleanup applied.

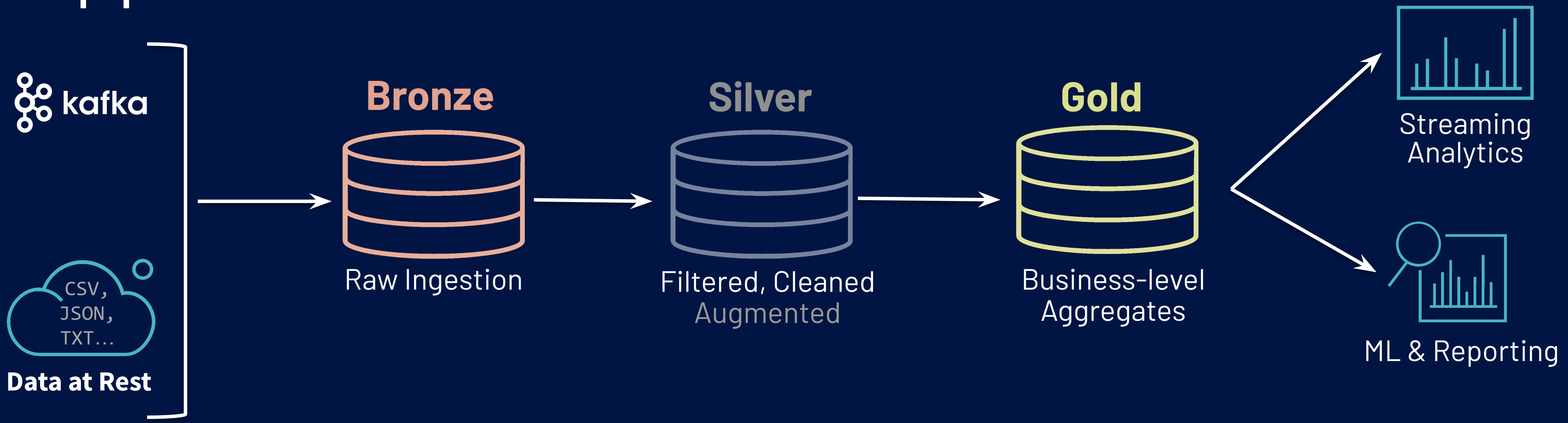
Queryable for easy debugging!

The Delta Architecture



Clean data, ready for consumption

Simplify real-time / streaming data applications



Streams move data through the Delta Lake

- Low-latency or manually triggered
- Eliminates management of schedules and jobs

databricks

10 minute break

DELTA LAKE

Lab 1: Building Delta Pipelines

SPARK+AI SUMMIT

10 minute break

SPARK+AI SUMMIT

DELTA LAKE

Features of Delta Lake

SPARK+AI SUMMIT

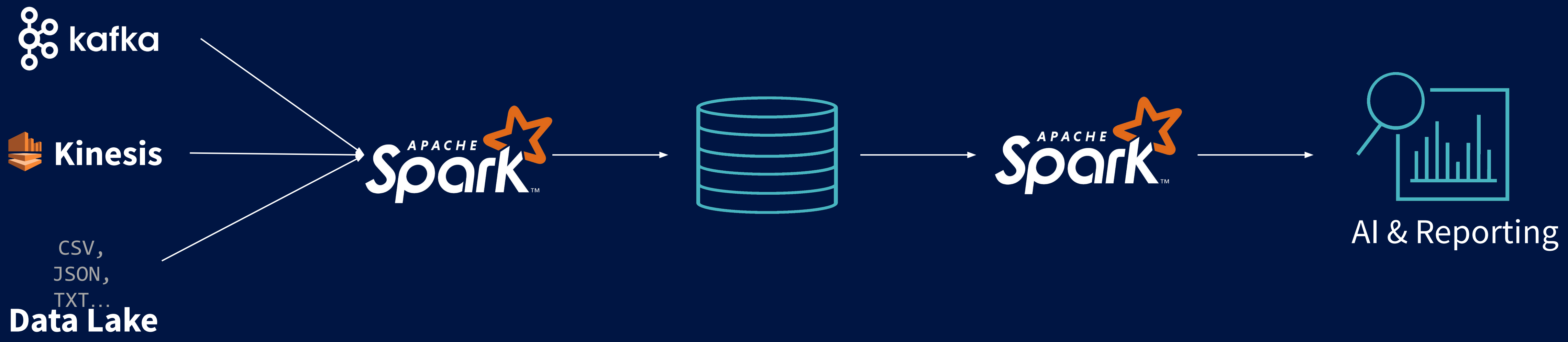
Unification of batch and streaming

DELTA LAKE

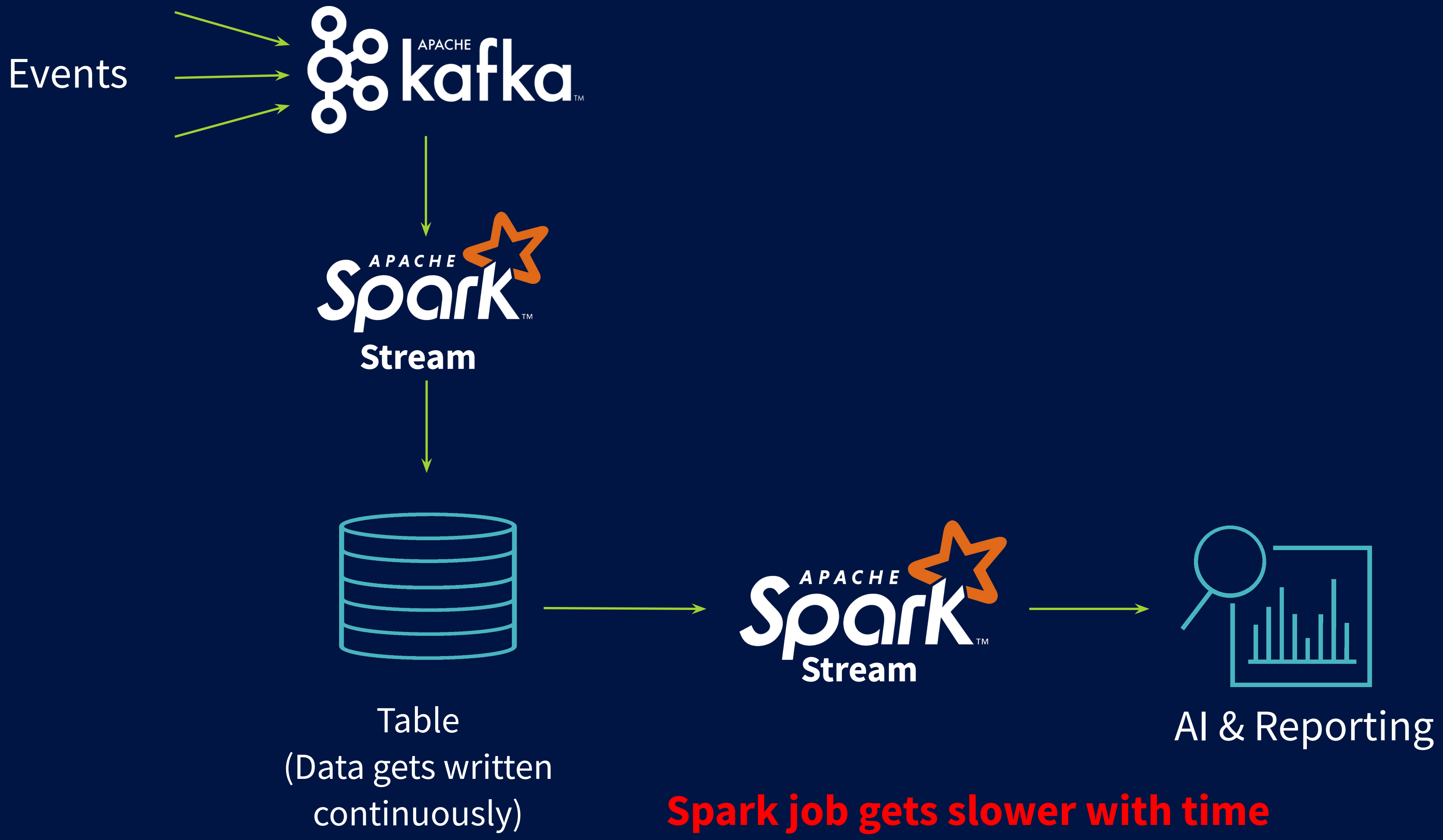
- Continuous data flow model
- Same engine. Same APIs. Same user code. No need to reason about system complexities separately.
- A table in Delta Lake is a batch table as well as a streaming source and sink.
- Can stream late arriving data and backfilled historical data through the same pipeline without having to delay downstream processing
- Incrementally load the new data efficiently. No need to do state management on what are the new files added.
- Process the data quickly as it arrives without any delays.

A Data Engineer's Dream...

Process data **continuously** and **incrementally** as new data arrive in a **cost efficient way** without having to *choose* between batch or streaming

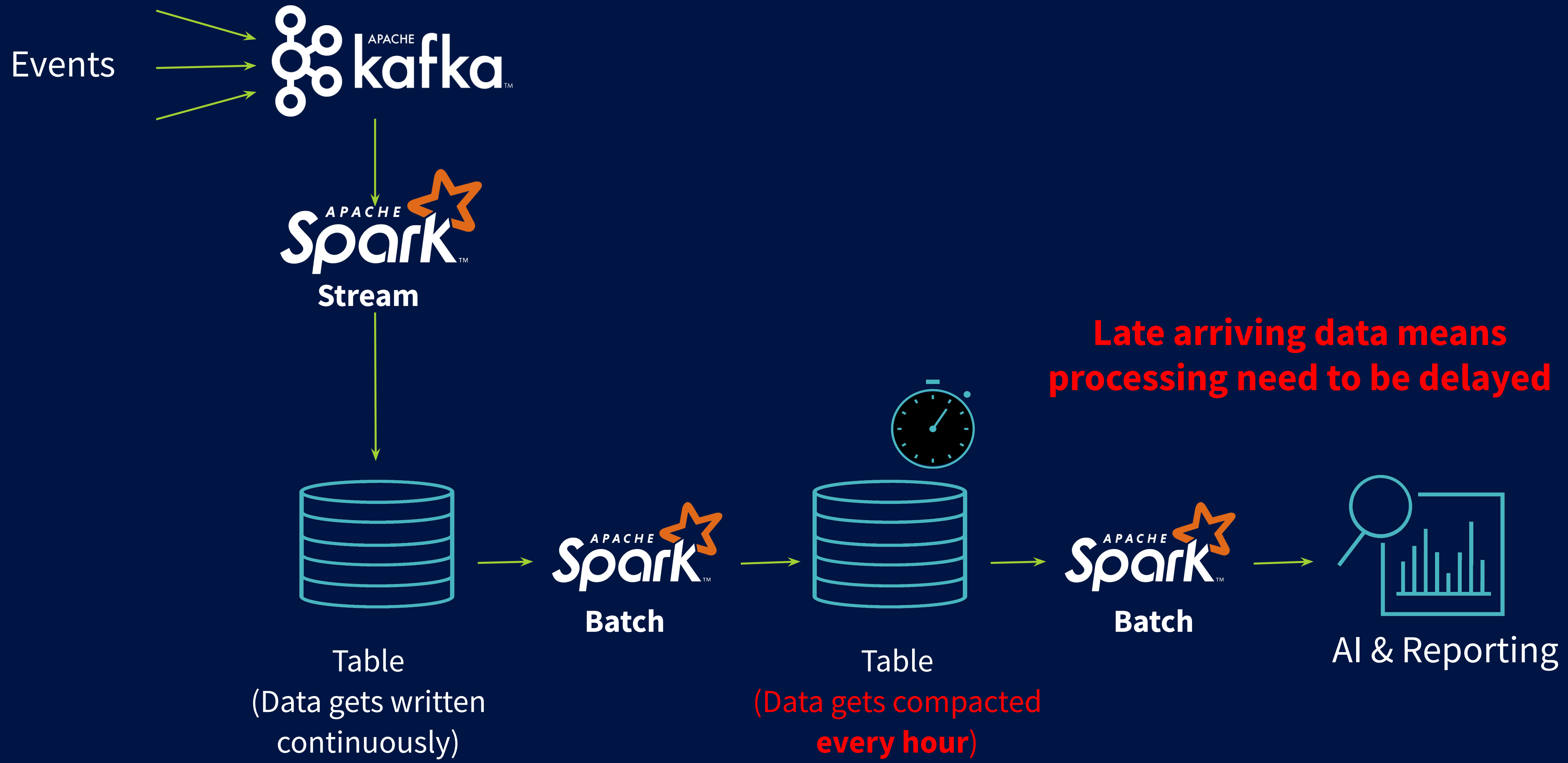


The Data Engineer's Journey...



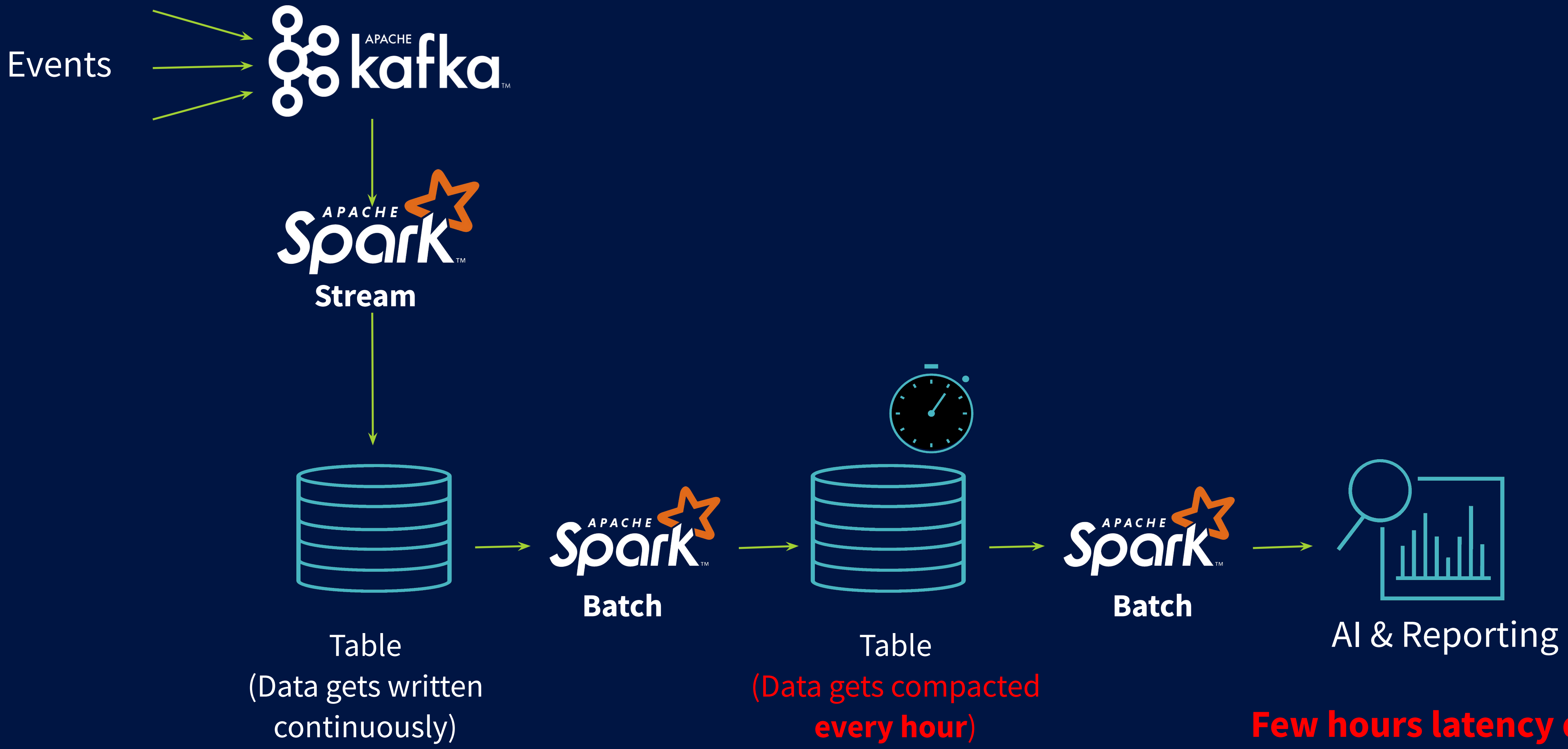
Spark job gets slower with time due to small files.

The Data Engineer's Journey...



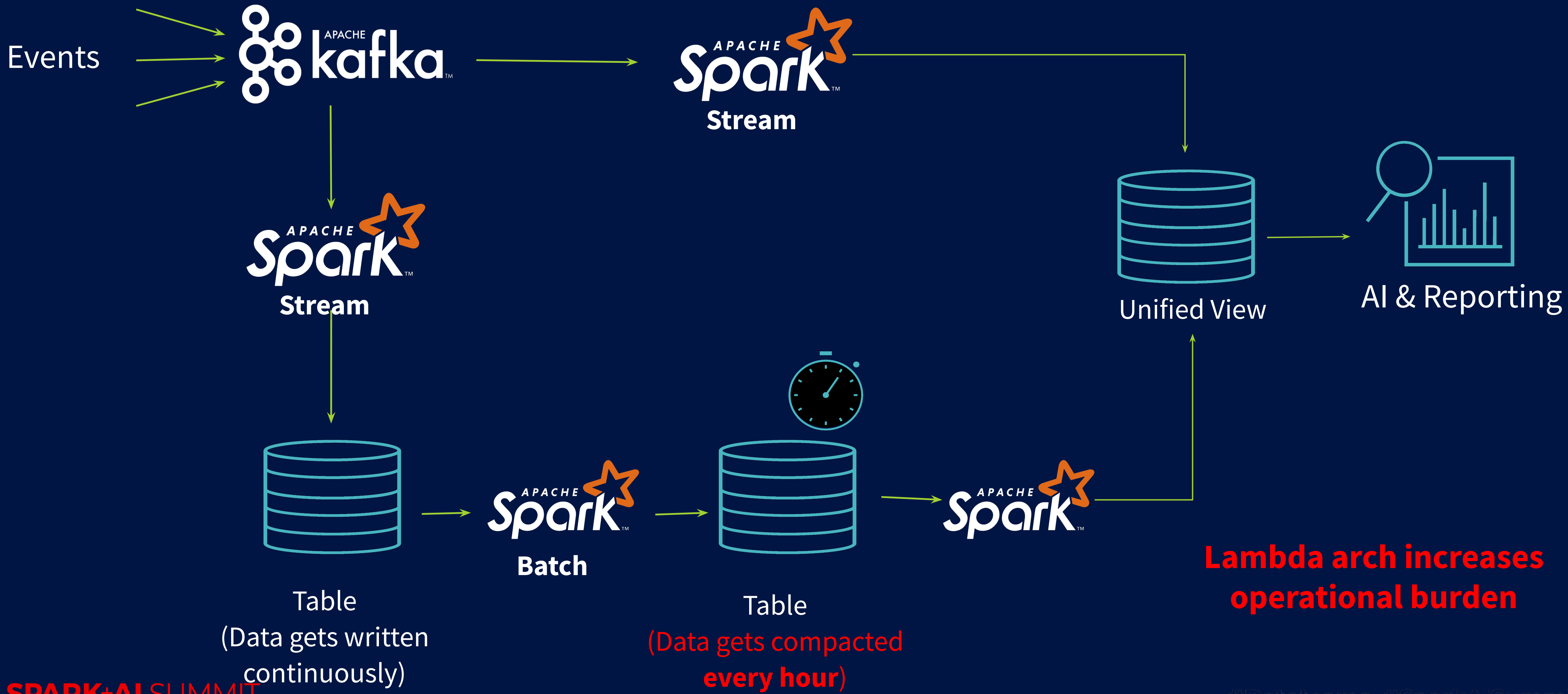
Late arriving data means processing need to be delayed

The Data Engineer's Journey...



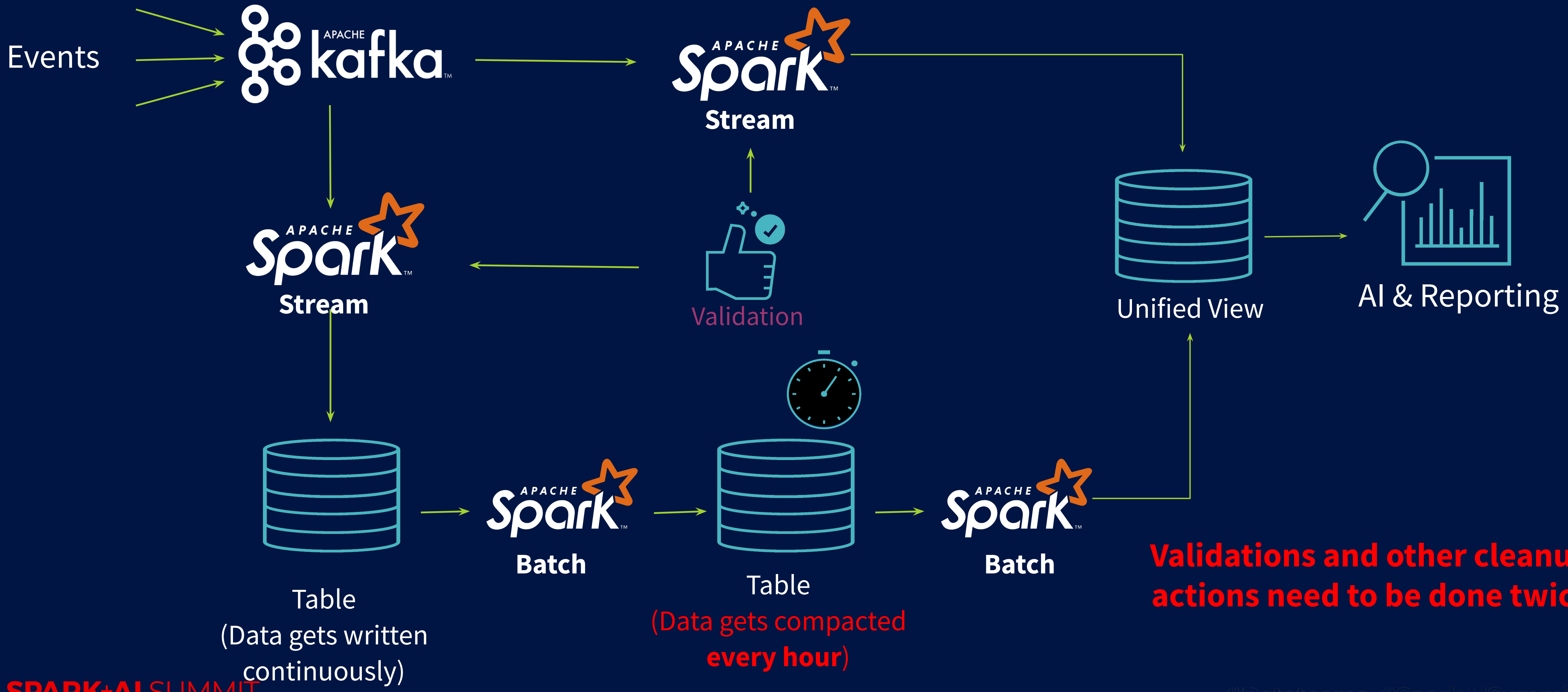
Few hours latency doesn't satisfy business needs

The Data Engineer's Journey...



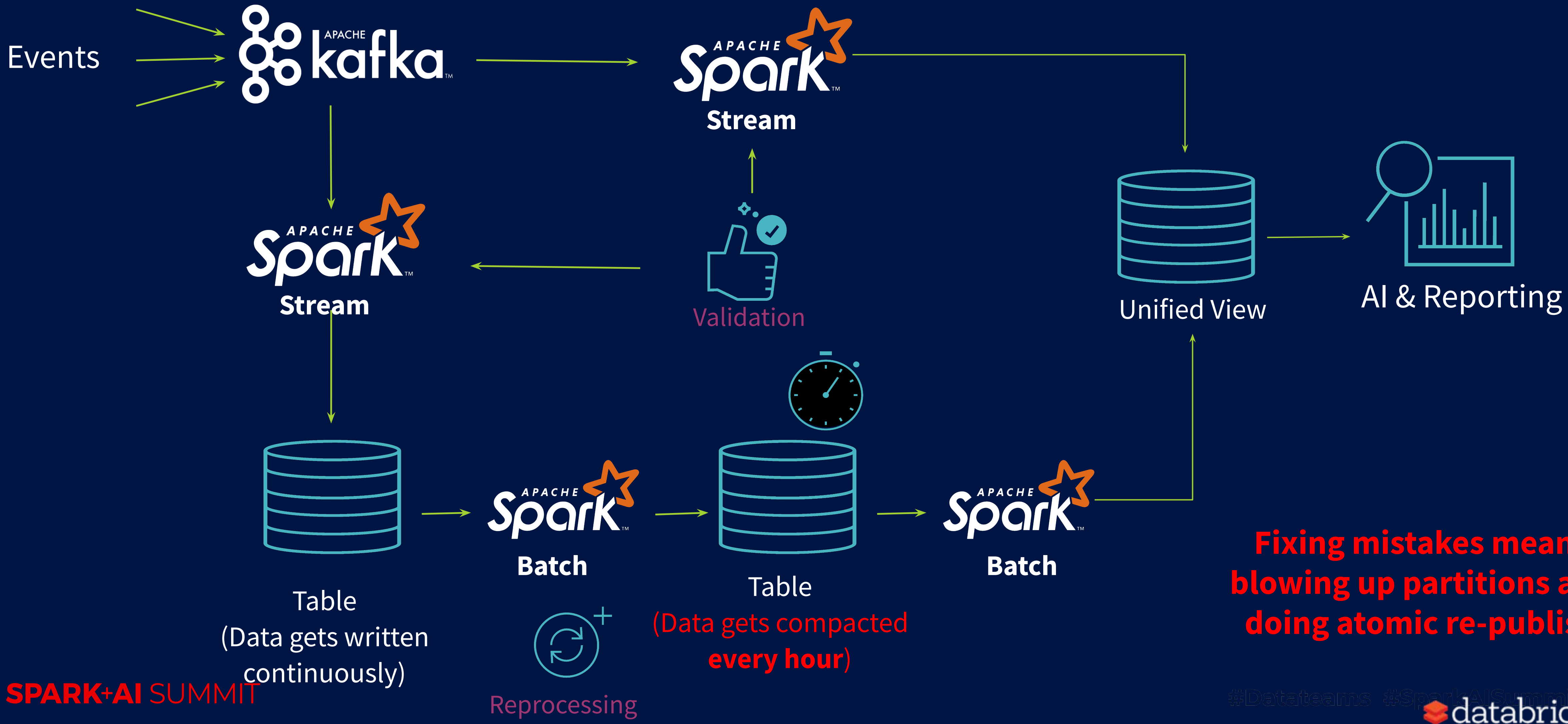
Lambda arch increases operational burden

The Data Engineer's Journey...



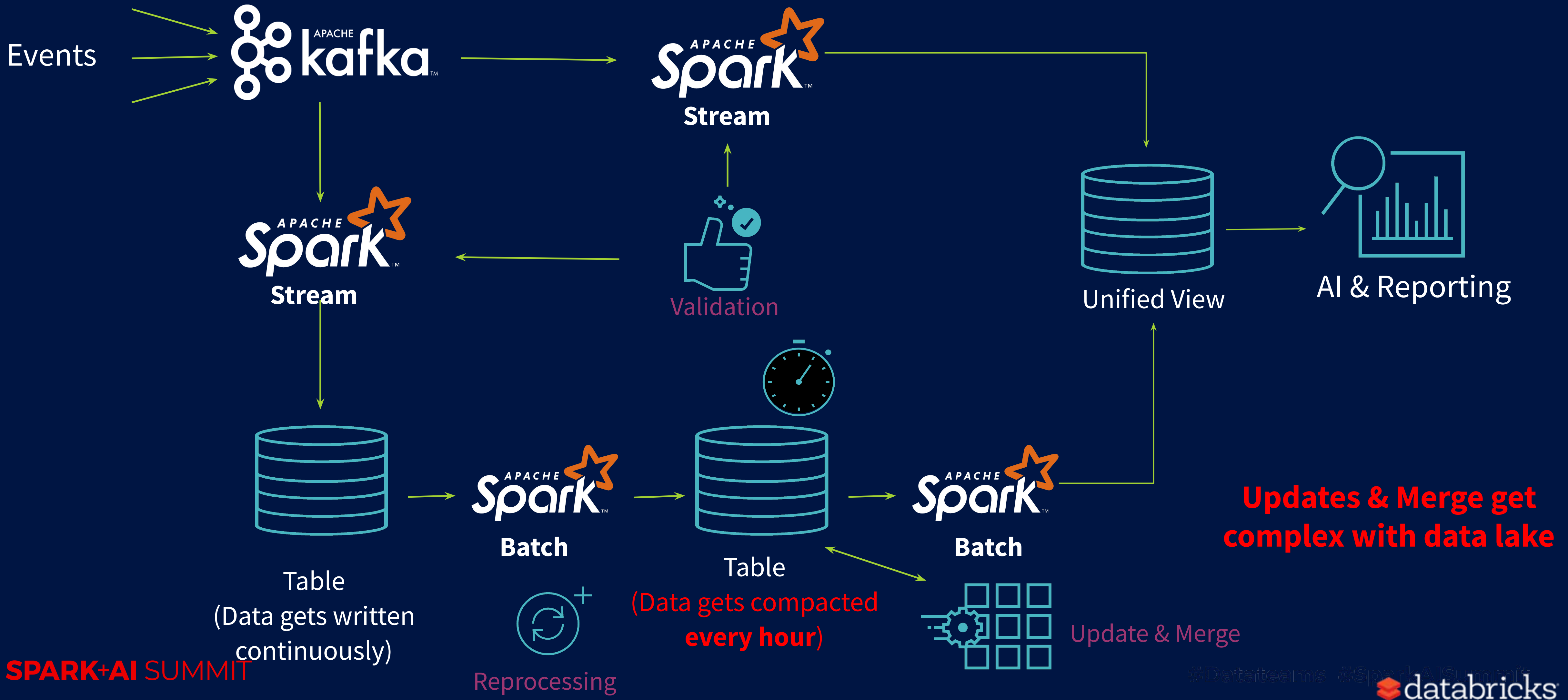
Validations and other cleanup actions need to be done twice

The Data Engineer's Journey...



Fixing mistakes means blowing up partitions and doing atomic re-publish

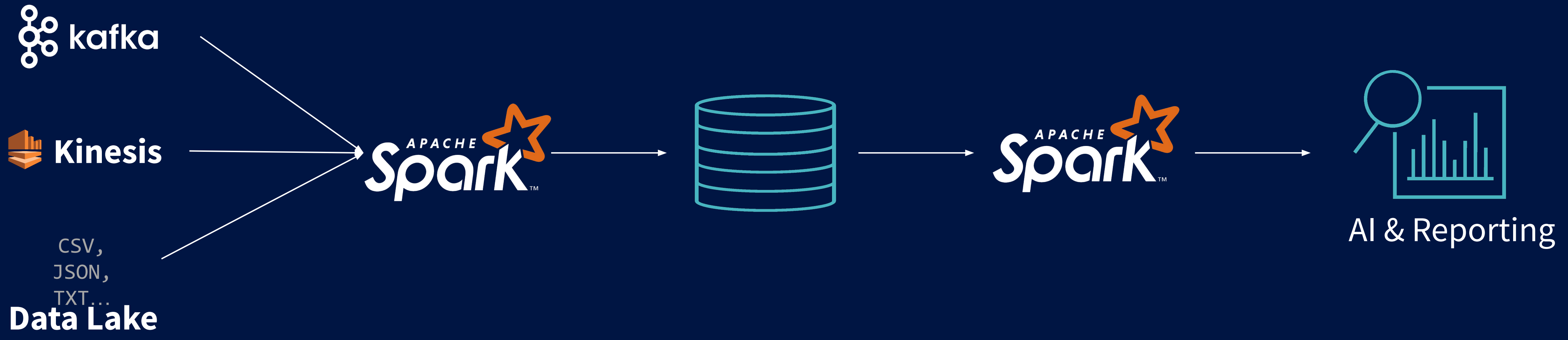
The Data Engineer's Journey...



Updates & Merge get complex with data lake

A Data Engineer’s Dream…

Process data **continuously** and **incrementally** as new data arrive in a **cost efficient way** without having to *choose* between batch or streaming



Schema Enforcement and Evolution

DELTA LAKE

- Schema enforcement (aka validation): prevents bad writes
- Schema evolution: allows for updates to data schema
- Delta saves table schema in JSON format in the transaction log
- Data to be written cannot have:
  - Additional columns
  - Data type mismatches
  - Column names that only differ by case
- Schema evolution commonly used for appends or overwrites

A brief look at time travel

- Utilizes the transaction log
- Can recreate a table's state at any point in time
- Automatic versioning
- Rollback to previous versions in case of bad writes
- Easily perform deletes

A brief look at time travel

DELTA LAKE

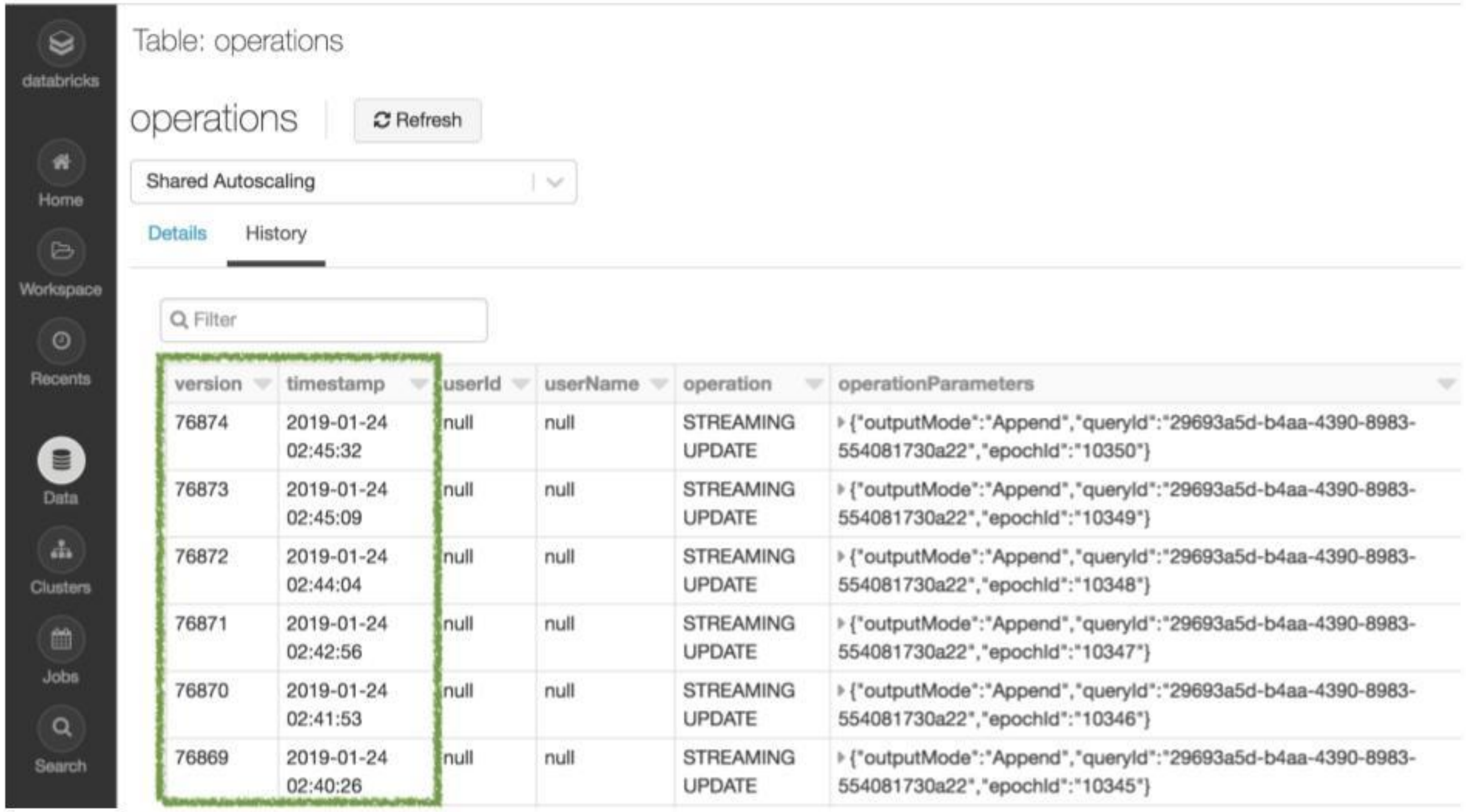


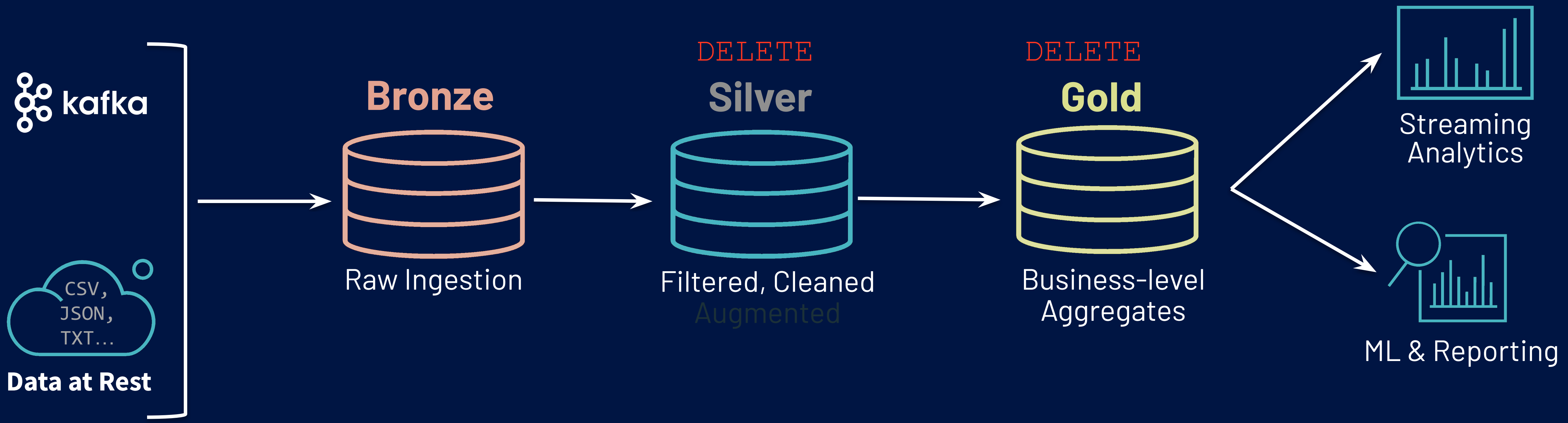
Table: operations

operations

Shared Autoscaling

version	timestamp	userId	userName	operation	operationParameters
76874	2019-01-24 02:45:32	null	null	STREAMING UPDATE	{"outputMode":"Append","queryId":"29693a5d-b4aa-4390-8983-554081730a22","epochId":"10350"}
76873	2019-01-24 02:45:09	null	null	STREAMING UPDATE	{"outputMode":"Append","queryId":"29693a5d-b4aa-4390-8983-554081730a22","epochId":"10349"}
76872	2019-01-24 02:44:04	null	null	STREAMING UPDATE	{"outputMode":"Append","queryId":"29693a5d-b4aa-4390-8983-554081730a22","epochId":"10348"}
76871	2019-01-24 02:42:56	null	null	STREAMING UPDATE	{"outputMode":"Append","queryId":"29693a5d-b4aa-4390-8983-554081730a22","epochId":"10347"}
76870	2019-01-24 02:41:53	null	null	STREAMING UPDATE	{"outputMode":"Append","queryId":"29693a5d-b4aa-4390-8983-554081730a22","epochId":"10346"}
76869	2019-01-24 02:40:26	null	null	STREAMING UPDATE	{"outputMode":"Append","queryId":"29693a5d-b4aa-4390-8983-554081730a22","epochId":"10345"}

The Delta Architecture



Easy to recompute when business logic changes:

- Clear tables
- Restart streams

databricks

DELTA LAKE

Lab 2: Exploring Delta Lake features

SPARK+AI SUMMIT

Wrap Up

Summary

In this class, we learned:

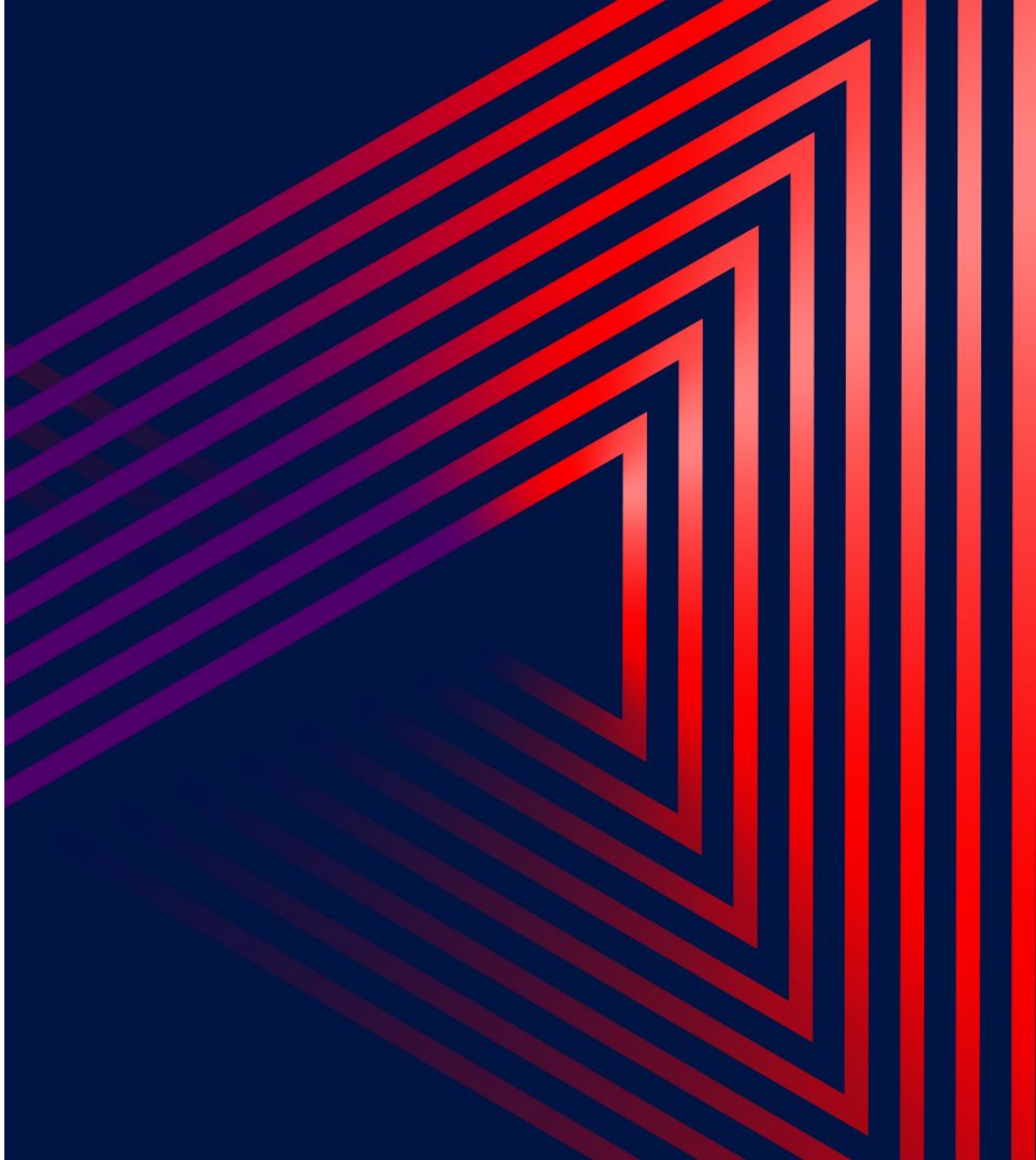
- The benefits of the Delta architecture
- How to build a data pipeline using Delta Lake
- How to incorporate batch and streaming data in a Delta Lake pipeline
- How Delta Lake schema enforcement and evolution impacts data writes and updates
- How to perform data deletions and version rollbacks

Q & A

Course evaluation

Thank you for taking this class!

Please take a moment to fill out the feedback form.



SPARK+AI SUMMIT

Feedback

Your feedback is important to us.

Don’t forget to rate and review the sessions.

SPARK+AI SUMMIT

#Datateams #SparkAISummit

SPARK+AI SUMMIT 2020

Organized by databricks