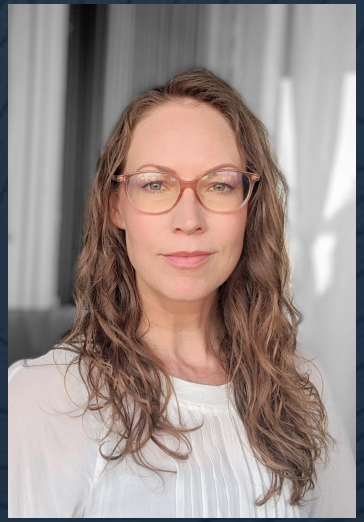
Simplify and Scale Data Engineering Pipelines with Delta Lake

Kate Sullivan
Curriculum Engineer
@ Databricks



Emma Freeman
Curriculum Engineer @
Databricks

Douglas Strodtman
Curriculum Engineer @
Databricks

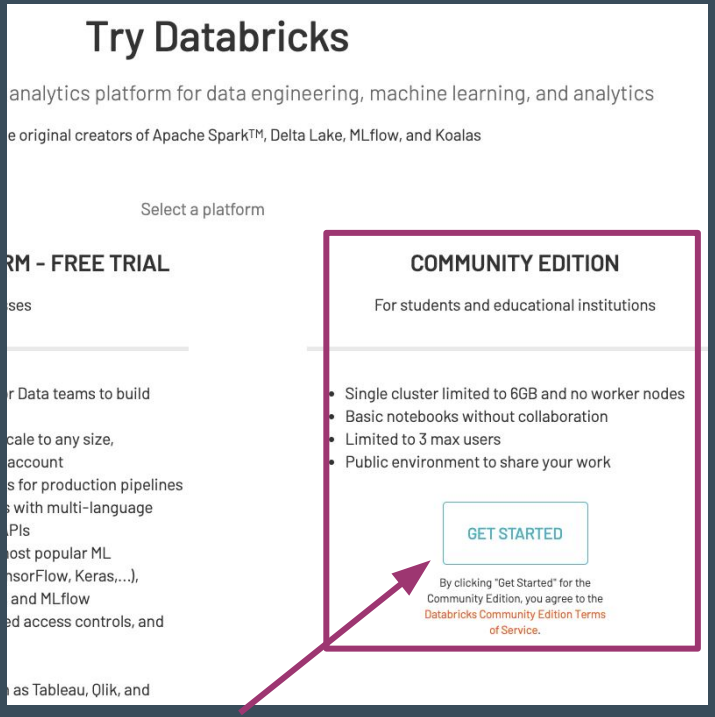
Logistics

- Slides will be available after this webinar
- Everyone is muted, put questions in the class Slack channel

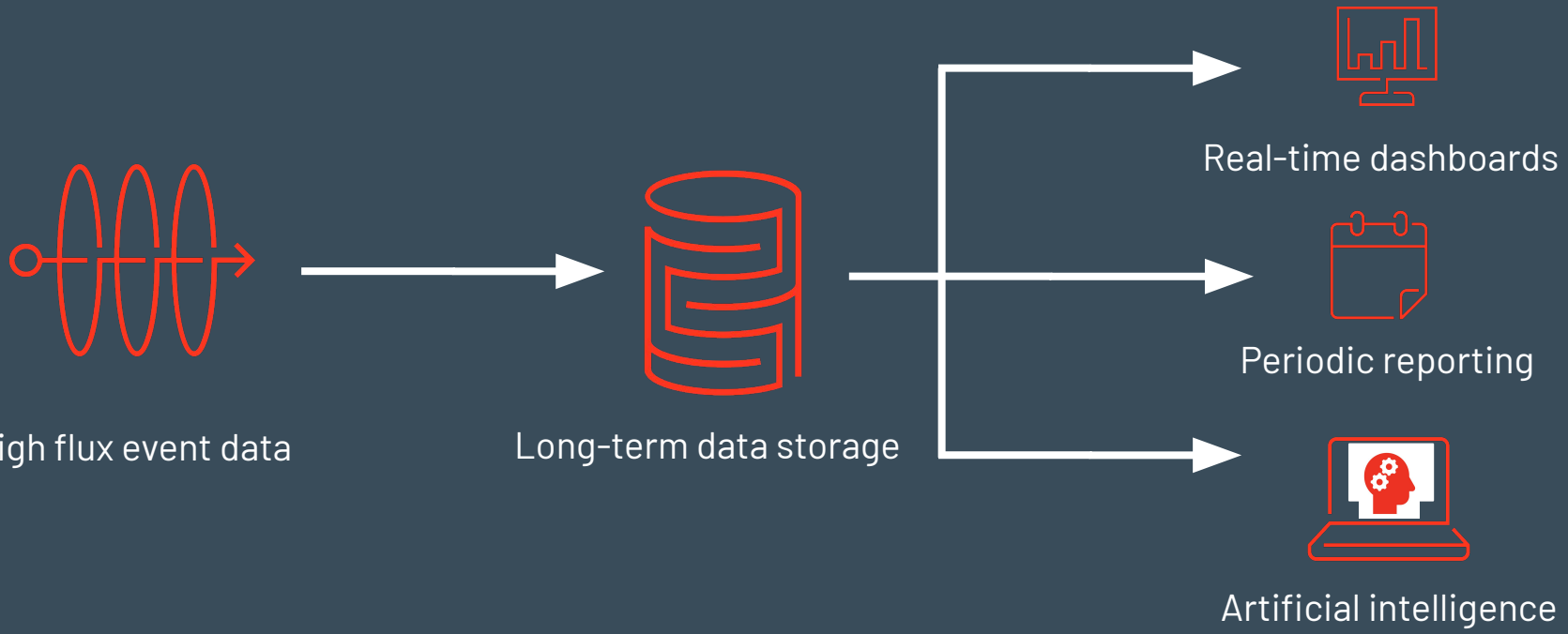
Sign up for Community Edition

1) Go to https://databricks.com/try-databricks

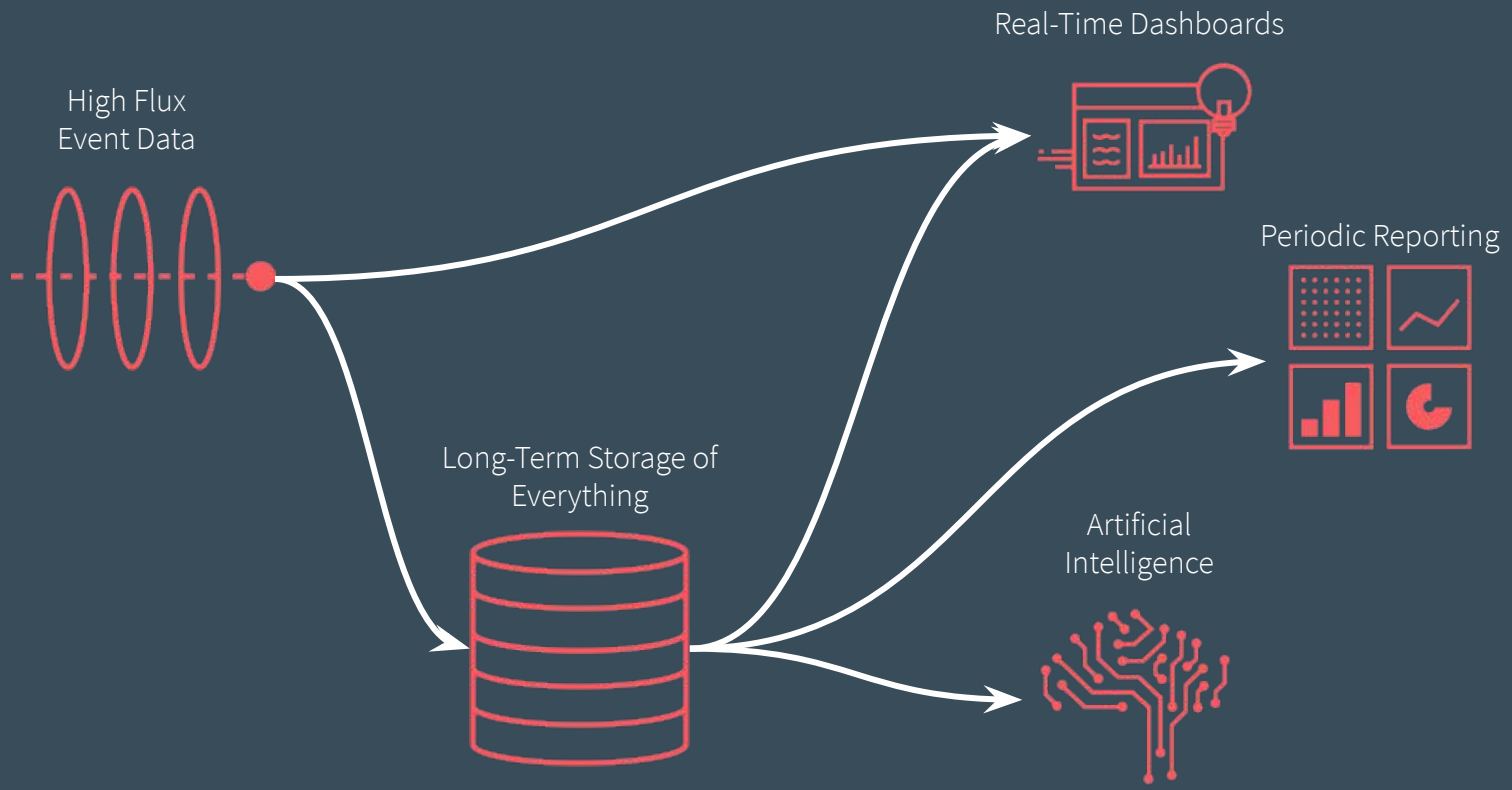
2) Enter info and click "Sign Up"
3) Select Community Edition
4) Follow on-screen instructions to login



The Big Picture



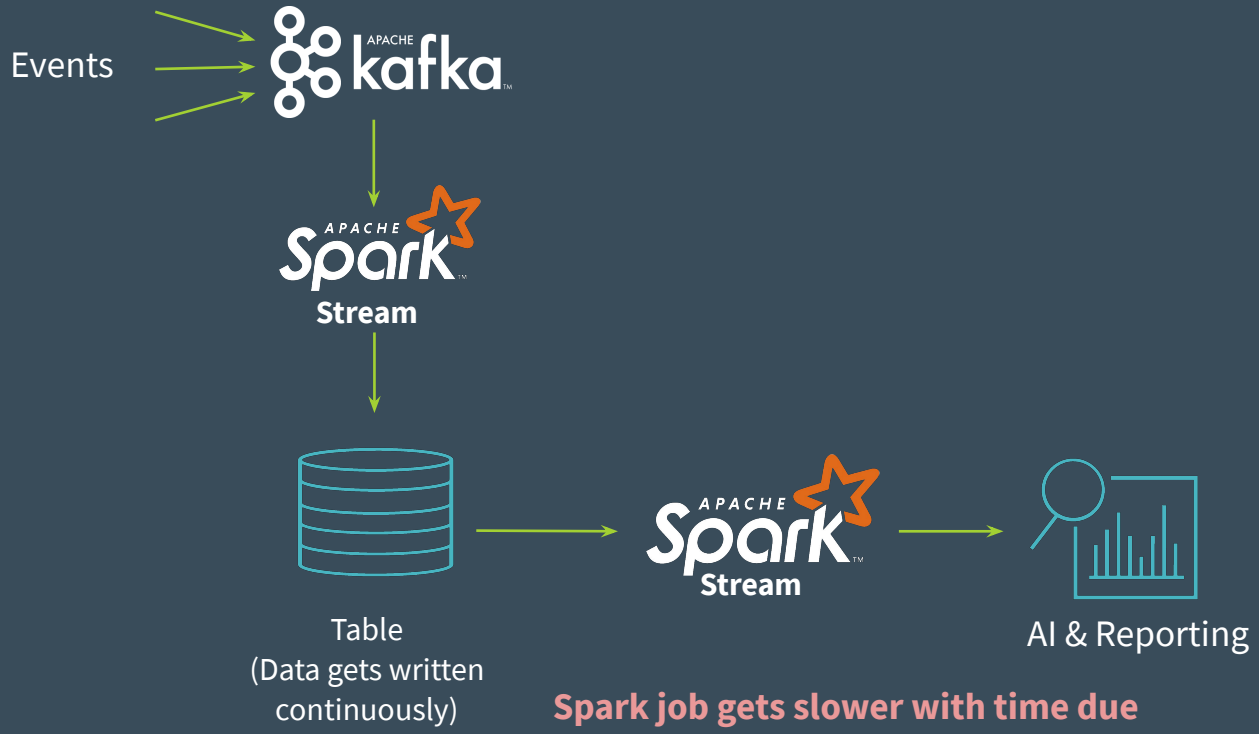
The Big Picture: the Lambda Architecture



A Data Engineer's Dream...

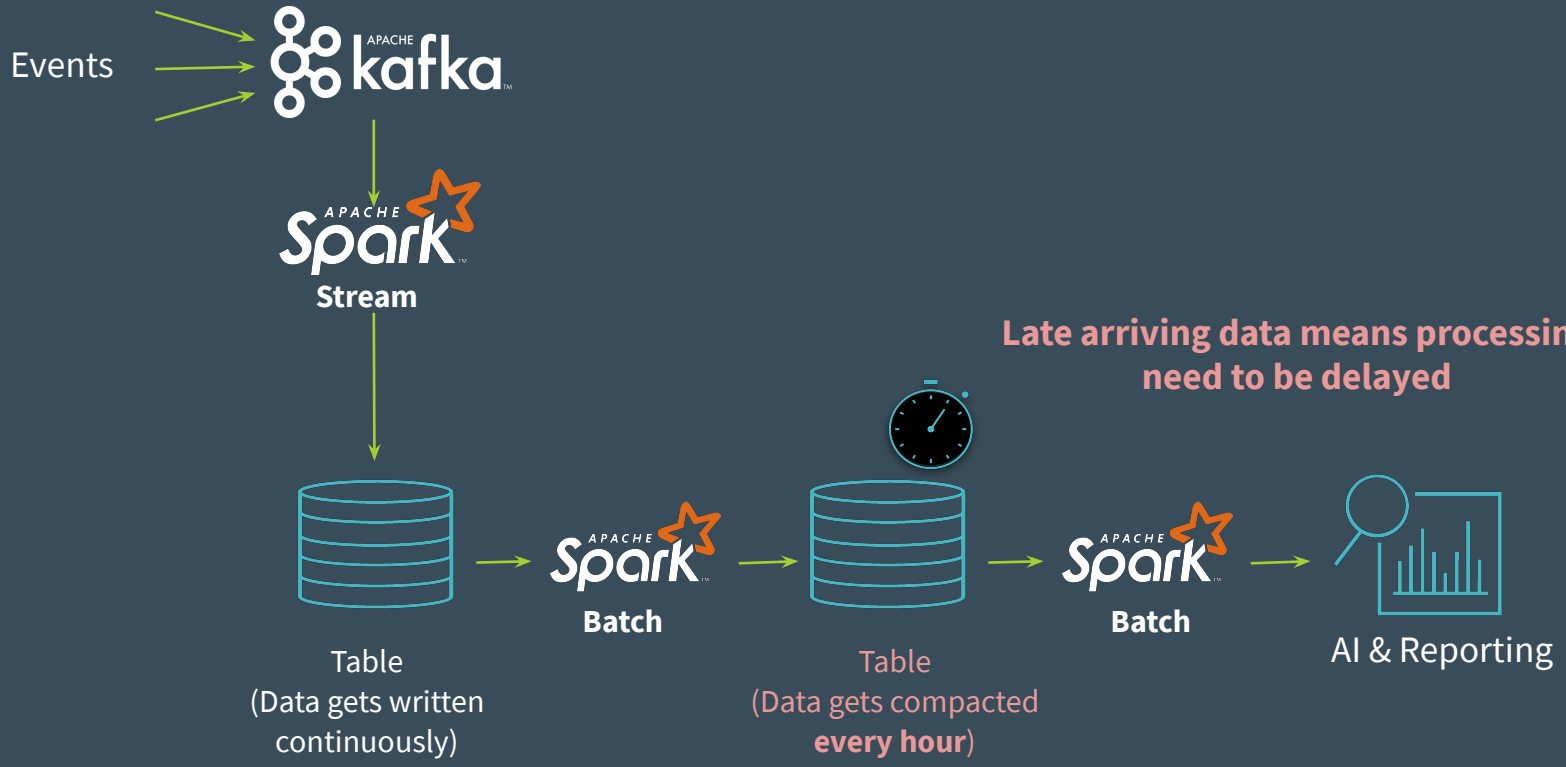
Process data **continuously** and **incrementally** as new data arrive in a **cost efficient way** without having to *choose* between batch or streaming

The Data Engineer's Journey...



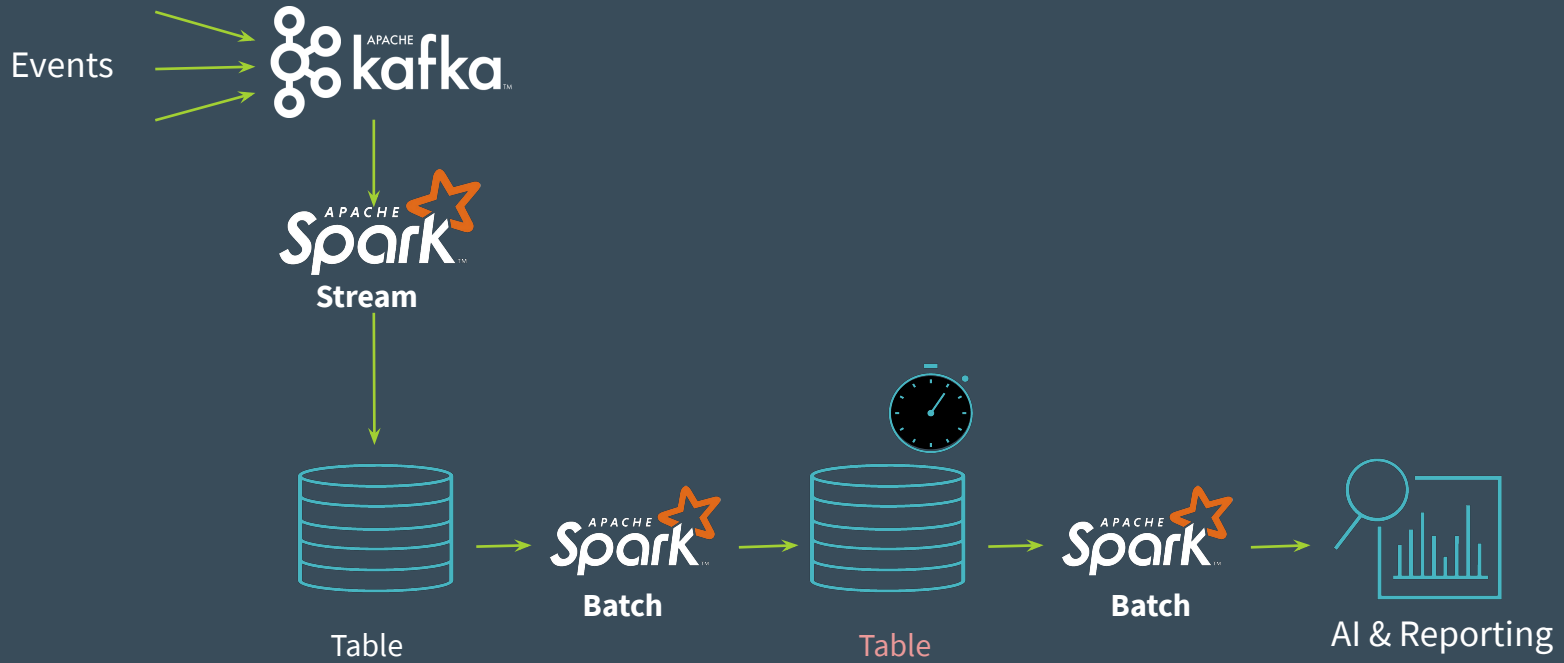
Spark job gets slower with time due to small files.

The Data Engineer's Journey...



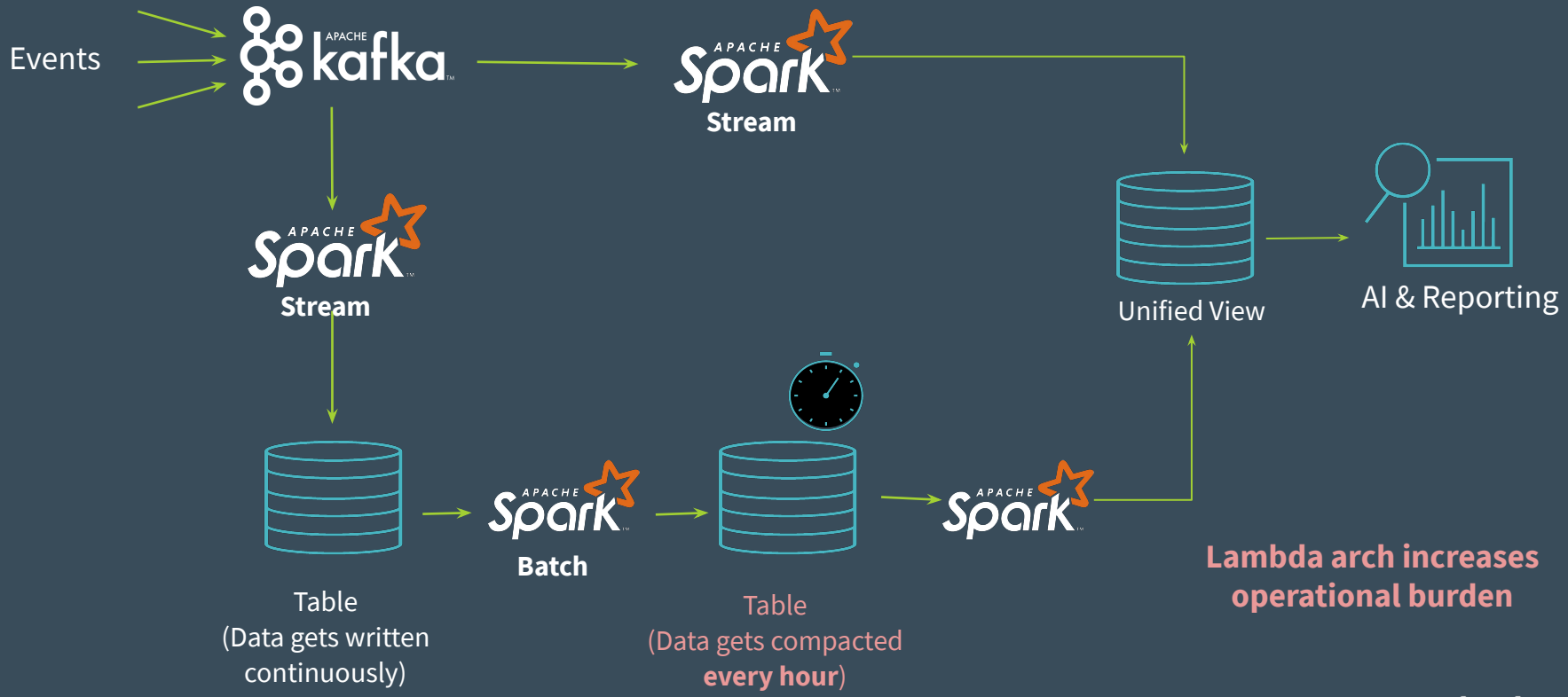
databricks

The Data Engineer's Journey...

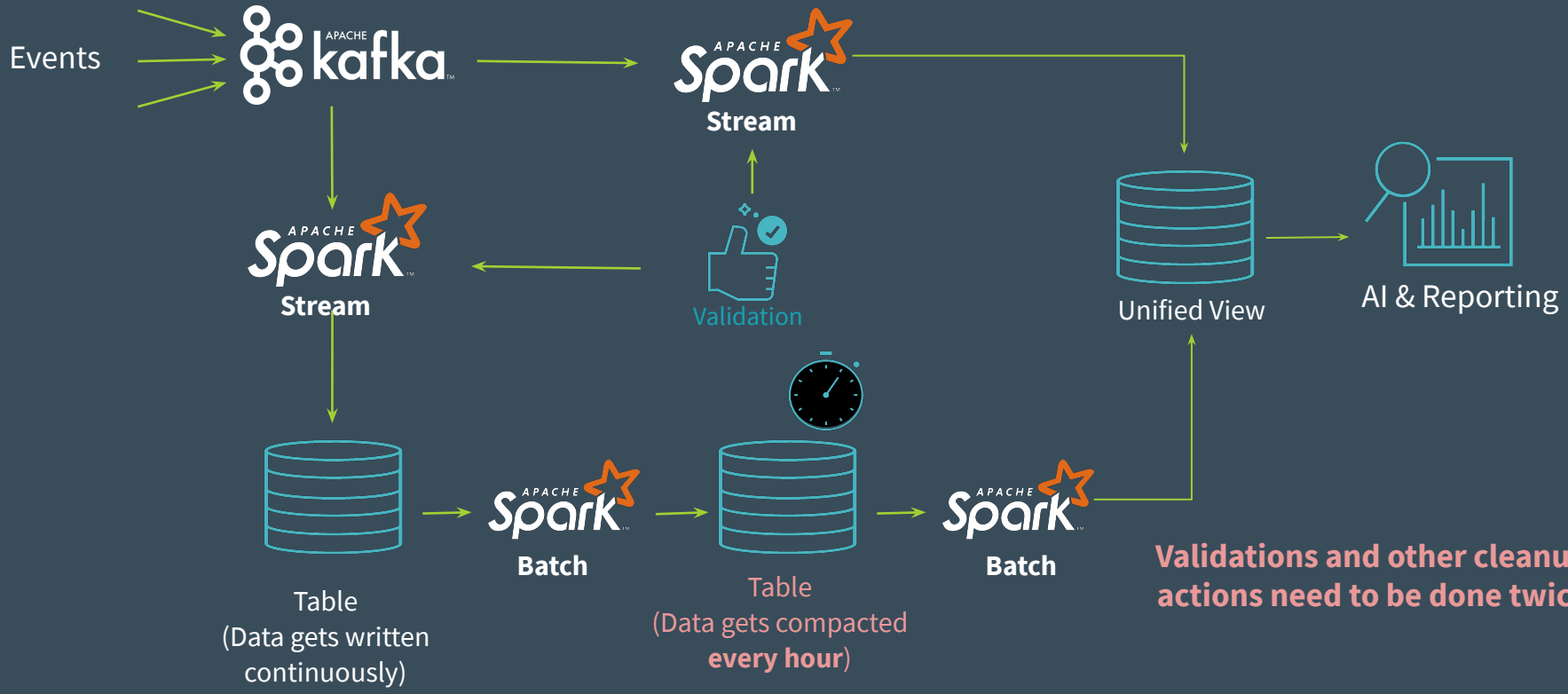


databricks

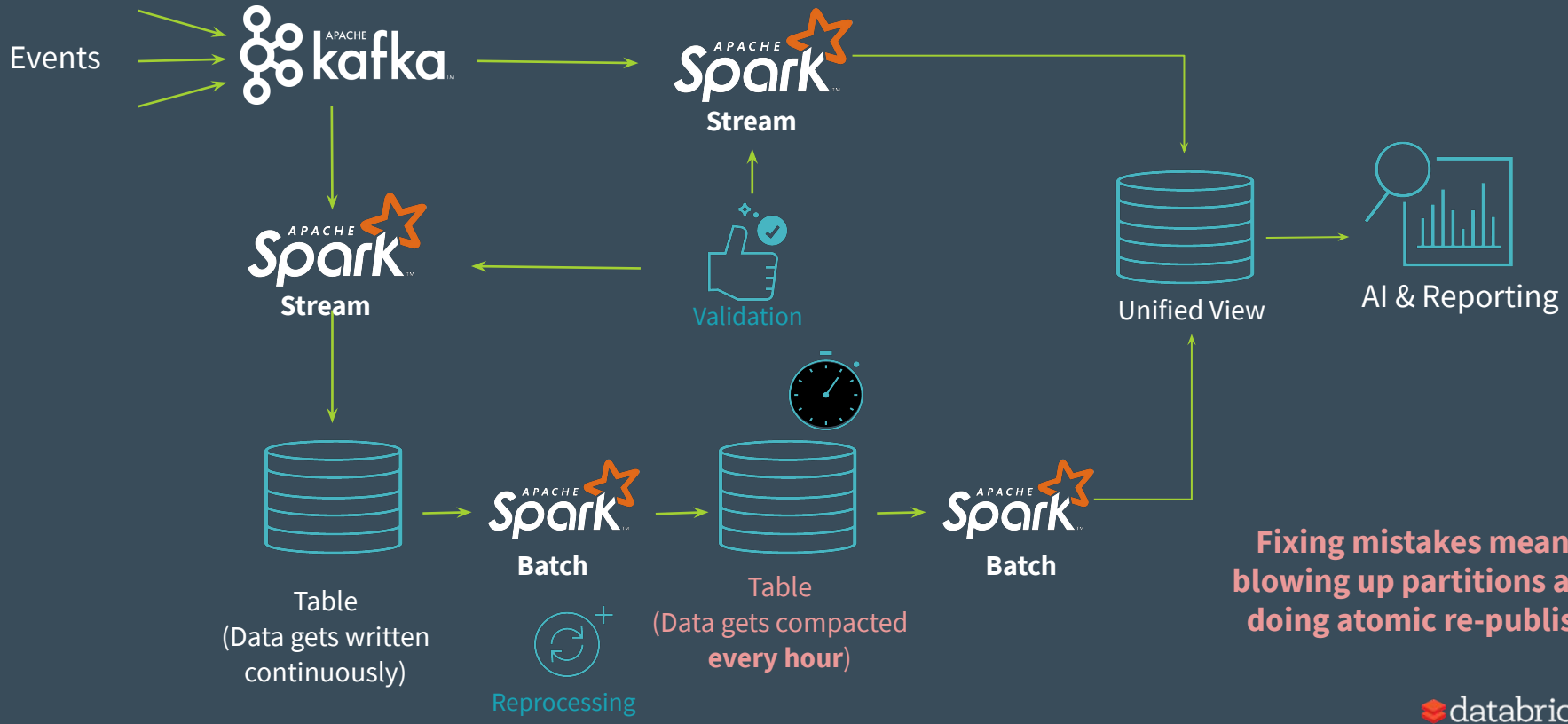
The Data Engineer's Journey...



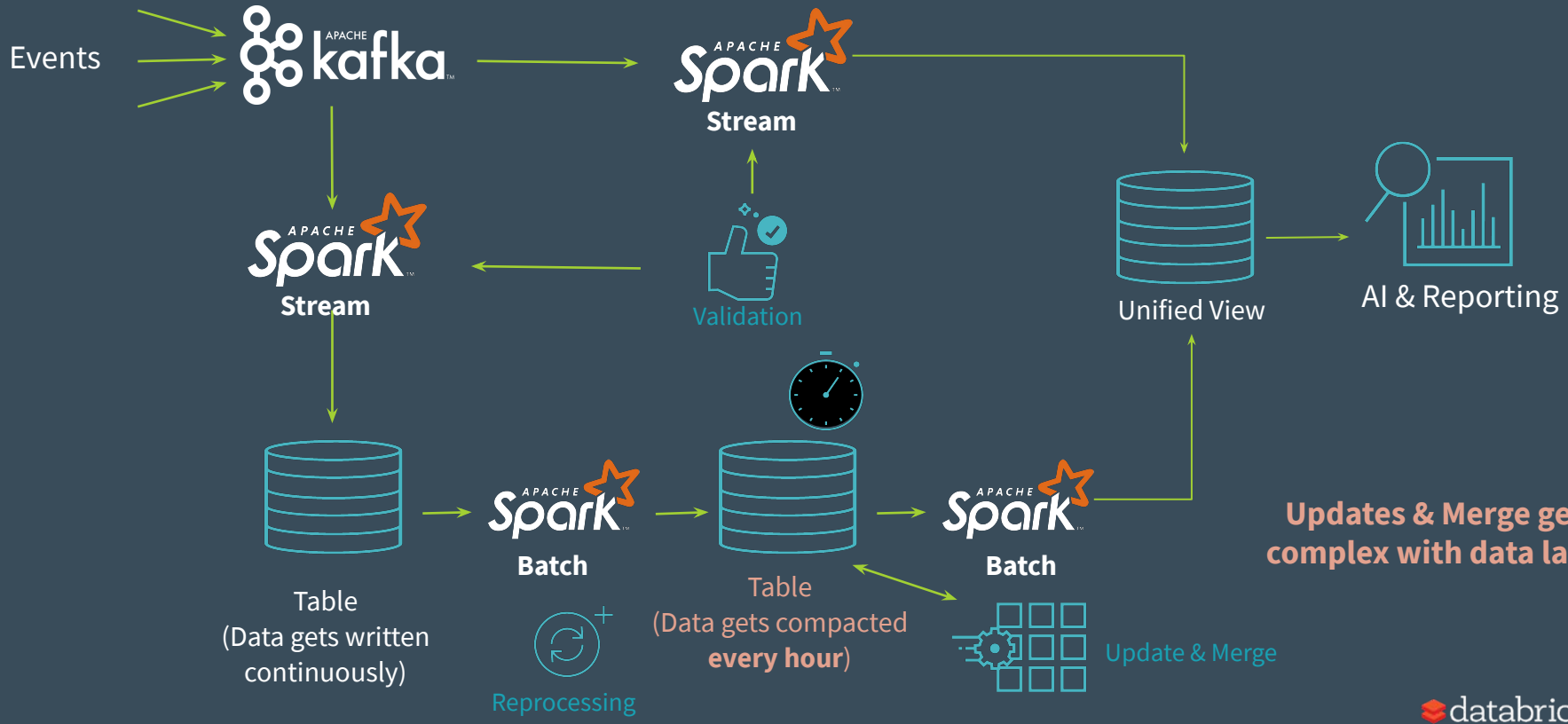
The Data Engineer's Journey...



The Data Engineer's Journey...



The Data Engineer's Journey...

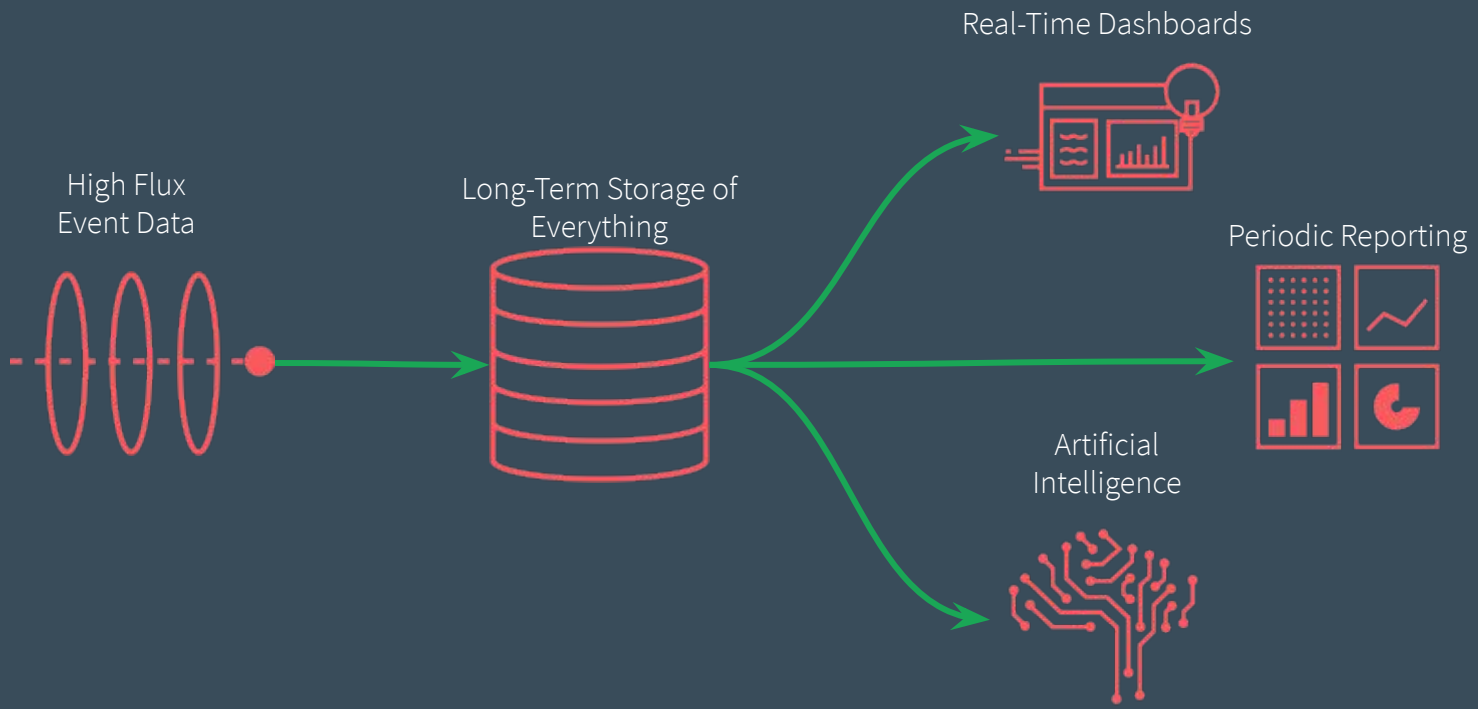


An Ideal System

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

Let’s try it instead with

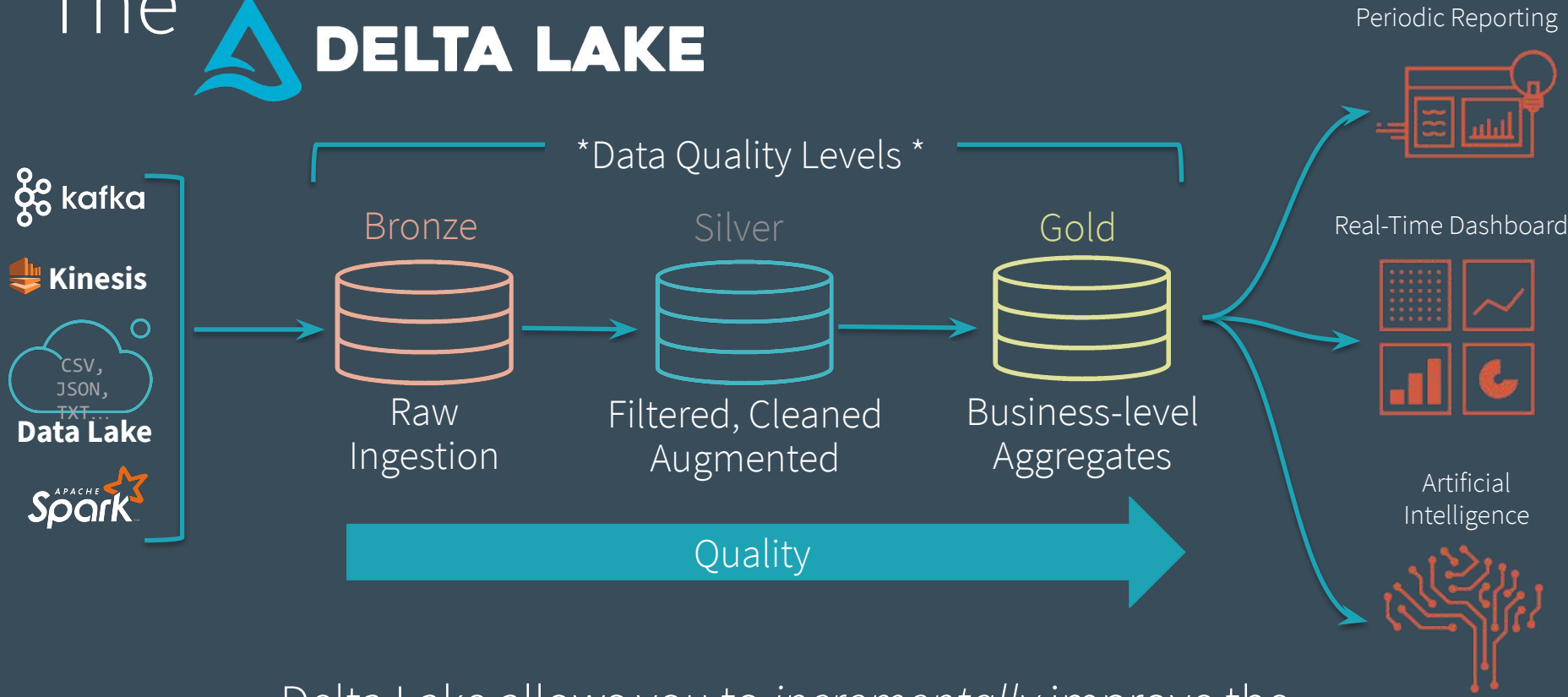
The Big Picture: the Delta Architecture



Connecting the dots...

1. Ability to read consistent data while data is being written → Snapshot isolation between writers and readers
2. Ability to read incrementally from a large table with good throughput → Optimized file source with scalable metadata handling
3. Ability to rollback in case of bad writes → Time travel
4. Ability to replay historical data along new data that arrived → Stream the backfilled historical data through the same pipeline
5. Ability to handle late arriving data without having to delay downstream processing → Stream any late arriving data added to the table as they get added

The DELTA LAKE



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

Components of Delta Lake

Delta Lake is comprised of:

- 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

- 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

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