Getting Data Ready for Data Science with Delta Lake and MLflow

Denny Lee, Developer Advocate

Logistics

- Recording and slides will be available after this webinar
- Everyone is muted, put questions in the question panel
- We will give you a link to follow up, for more information fill out the form

Today's Speakers

Denny Lee is a Developer Advocate at Databricks. He is a hands-on distributed systems and data sciences engineer with extensive experience developing internet-scale infrastructure, data platforms, and predictive analytics systems for both on-premise and cloud environments.

VISION

Accelerate innovation by unifying data science, engineering and business

---

SOLUTION

Unified Analytics Platform

WHO WE ARE

- Original creators of 

- 2000+ global companies use our platform across big data & machine learning lifecycle

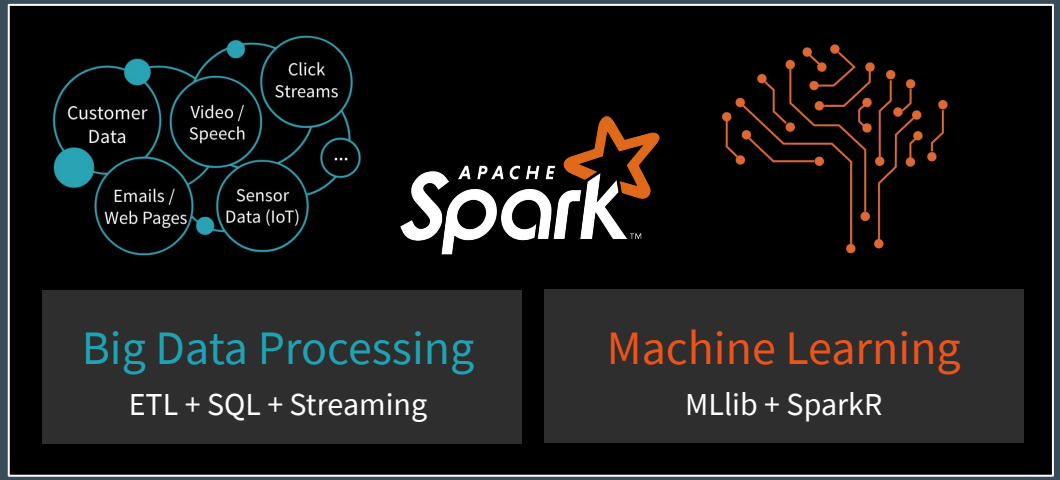
Background

Making Apache Spark™ Better with Delta Lake

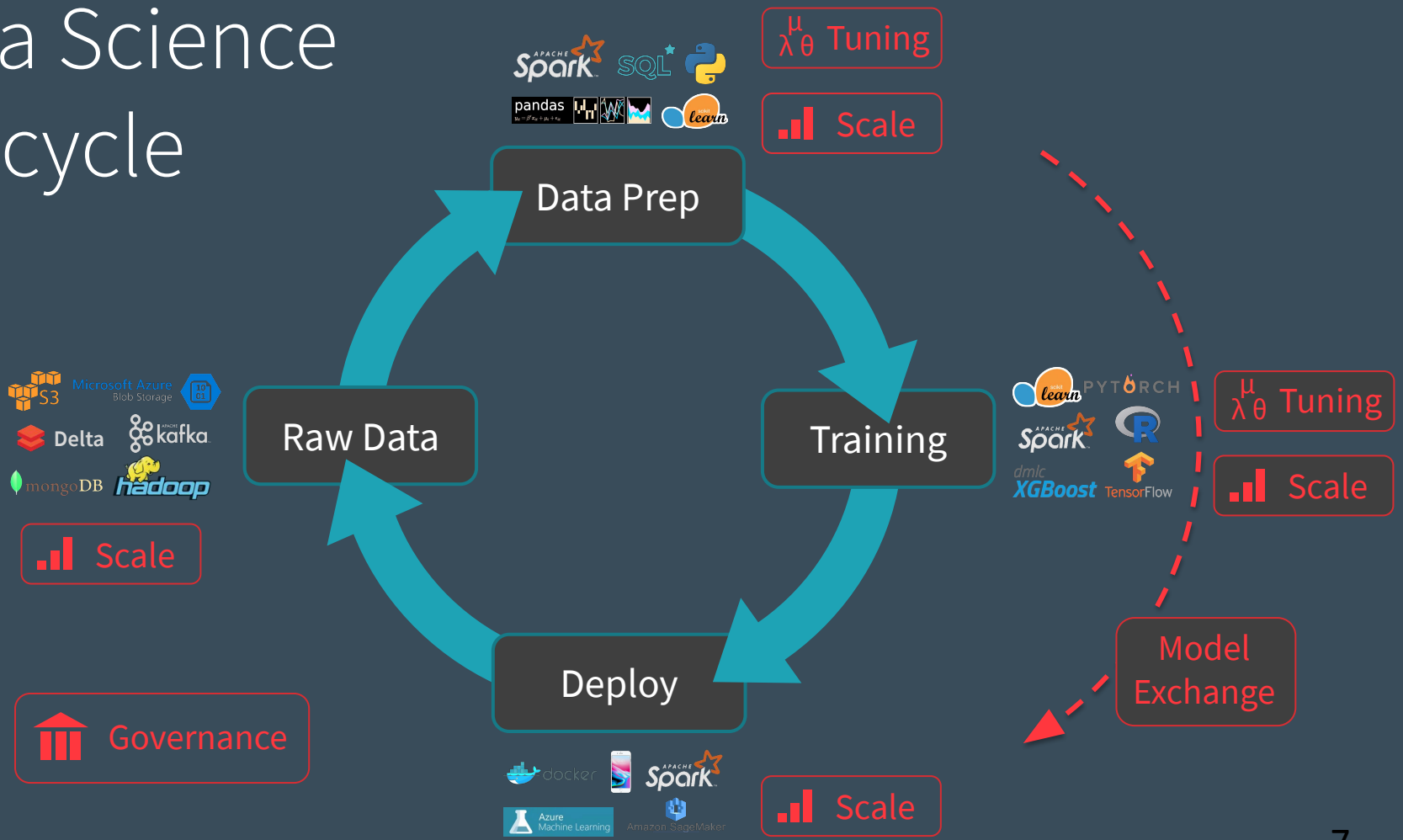
Delta Architecture, A Step Beyond Lambda Architecture

Apache Spark: De-Facto Unified Analytics Engine

Uniquely combines Data & AI technologies



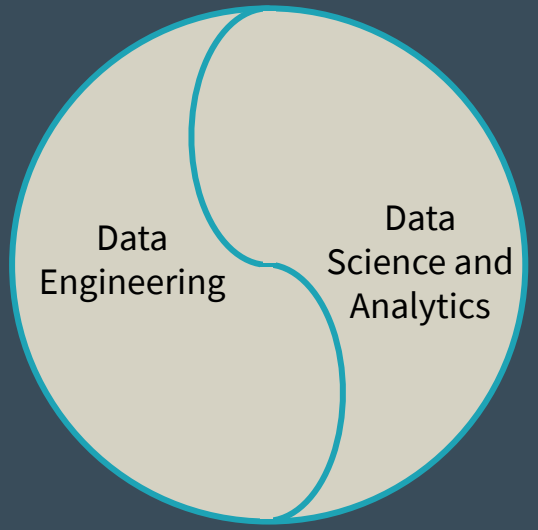
Data Science Lifecycle



The role of data engineering

Support/ enable data science and analytics

- Develop, test and maintain data pipelines that make reliable and quality data available quickly, efficiently and securely
- Help productionize data science models with robust, maintainable code



Big Data Architecture & Data Lakes

Store data in structured AND semistructured format

Pull data from various input sources

Single, central location to access data. Breaking data silos.

Open, accessible format. No vendor lock-in.

SQL and Machine Learning on a single source of truth

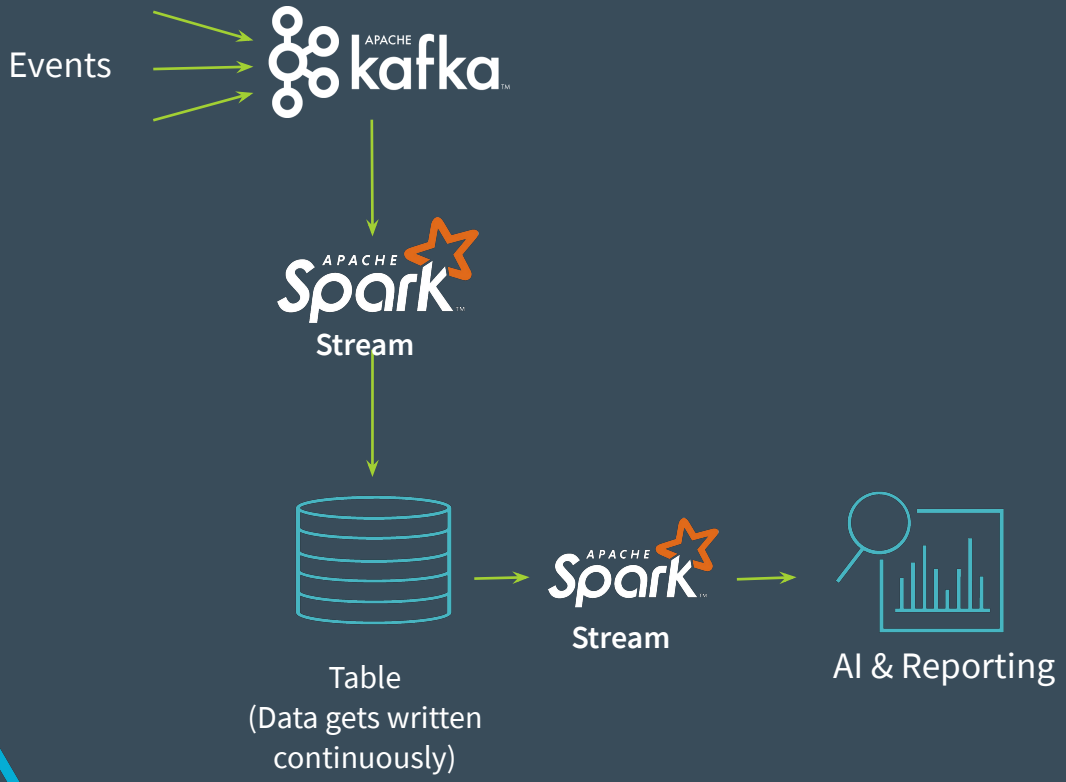
Data reliability challenges with data lakes

Failed production jobs leave data in corrupt state requiring tedious recovery

Lack of quality enforcement creates inconsistent and unusable data

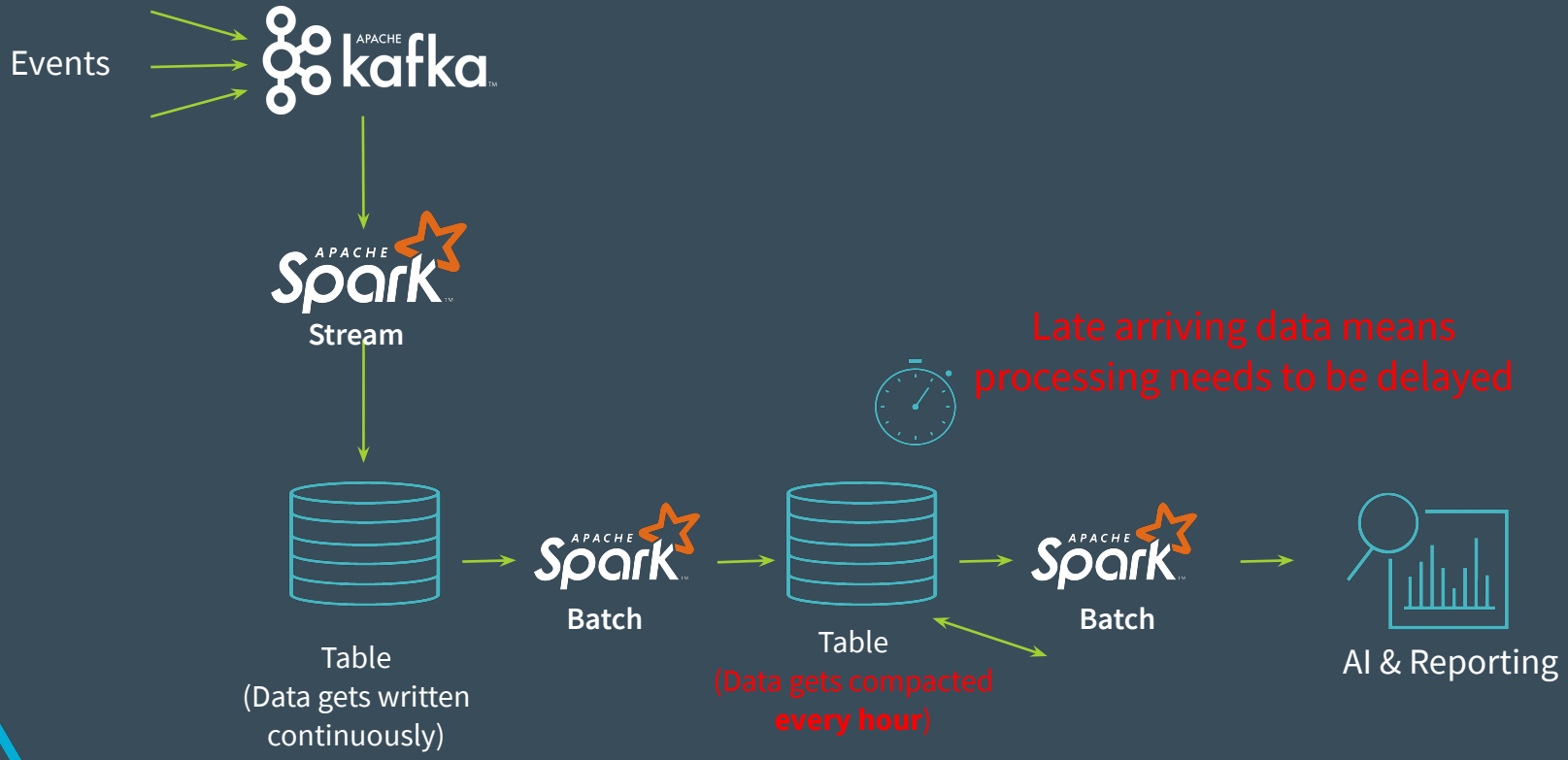
Lack of transactions makes it almost impossible to mix appends and reads, batch and streaming

Complex pipelines increase engineering overhead

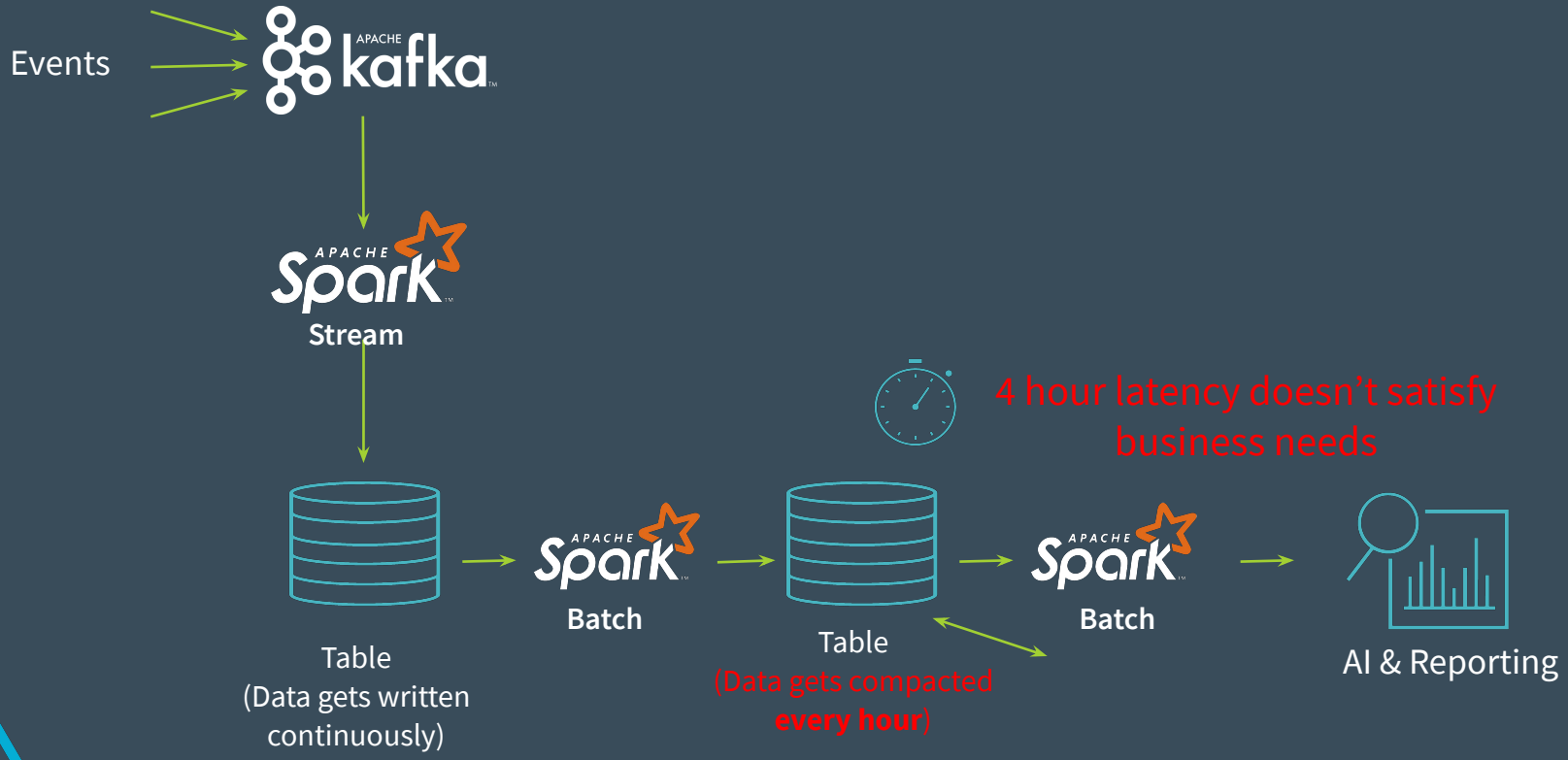


Spark job gets exponentially slower with time due to small files.

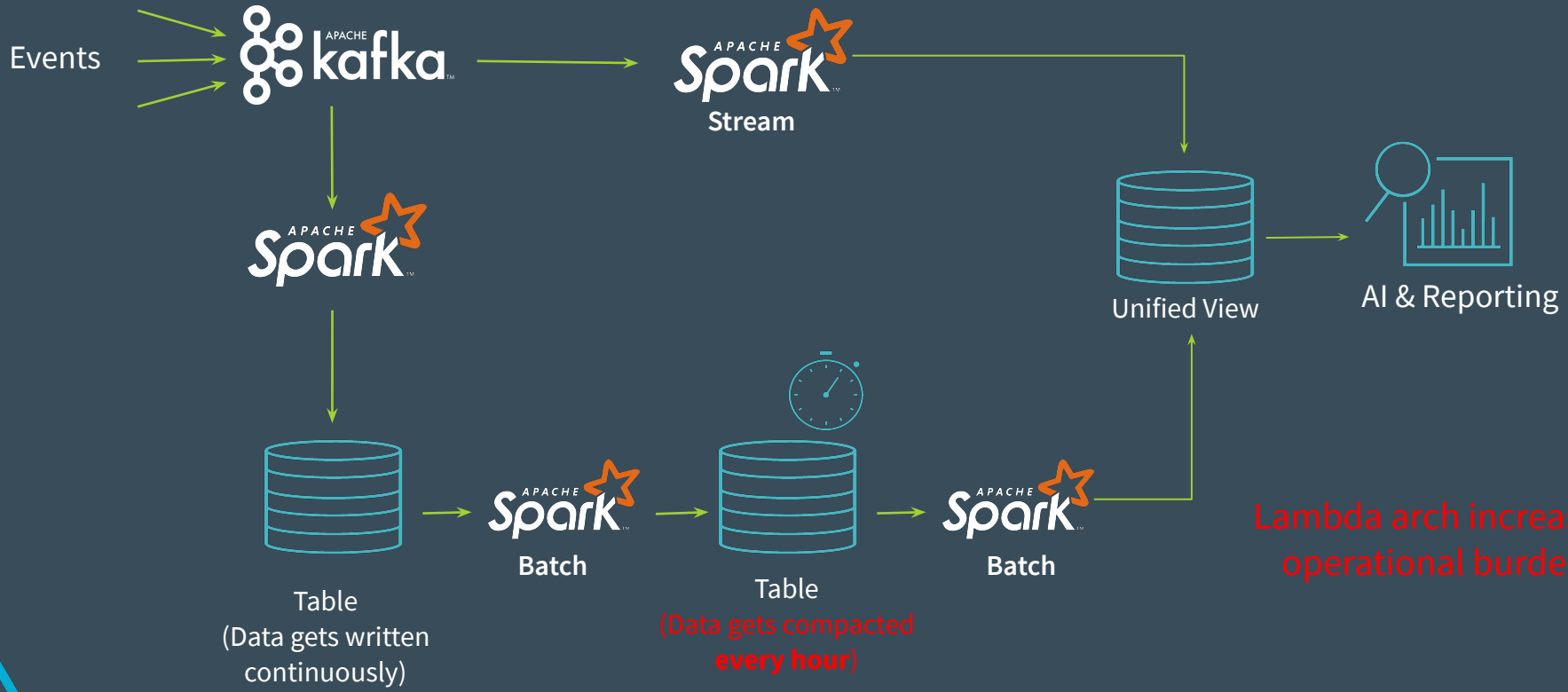
Complex pipelines increase engineering overhead



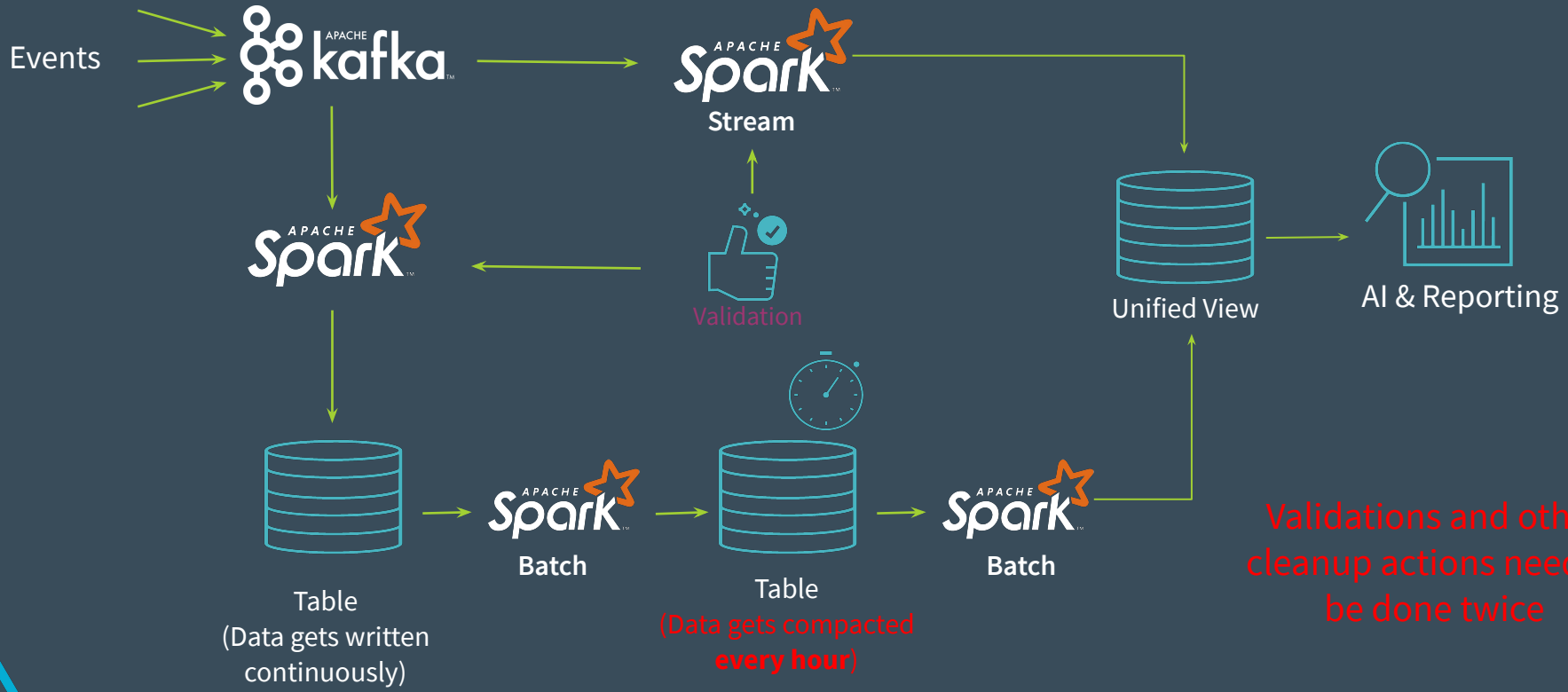
Complex pipelines increase engineering overhead



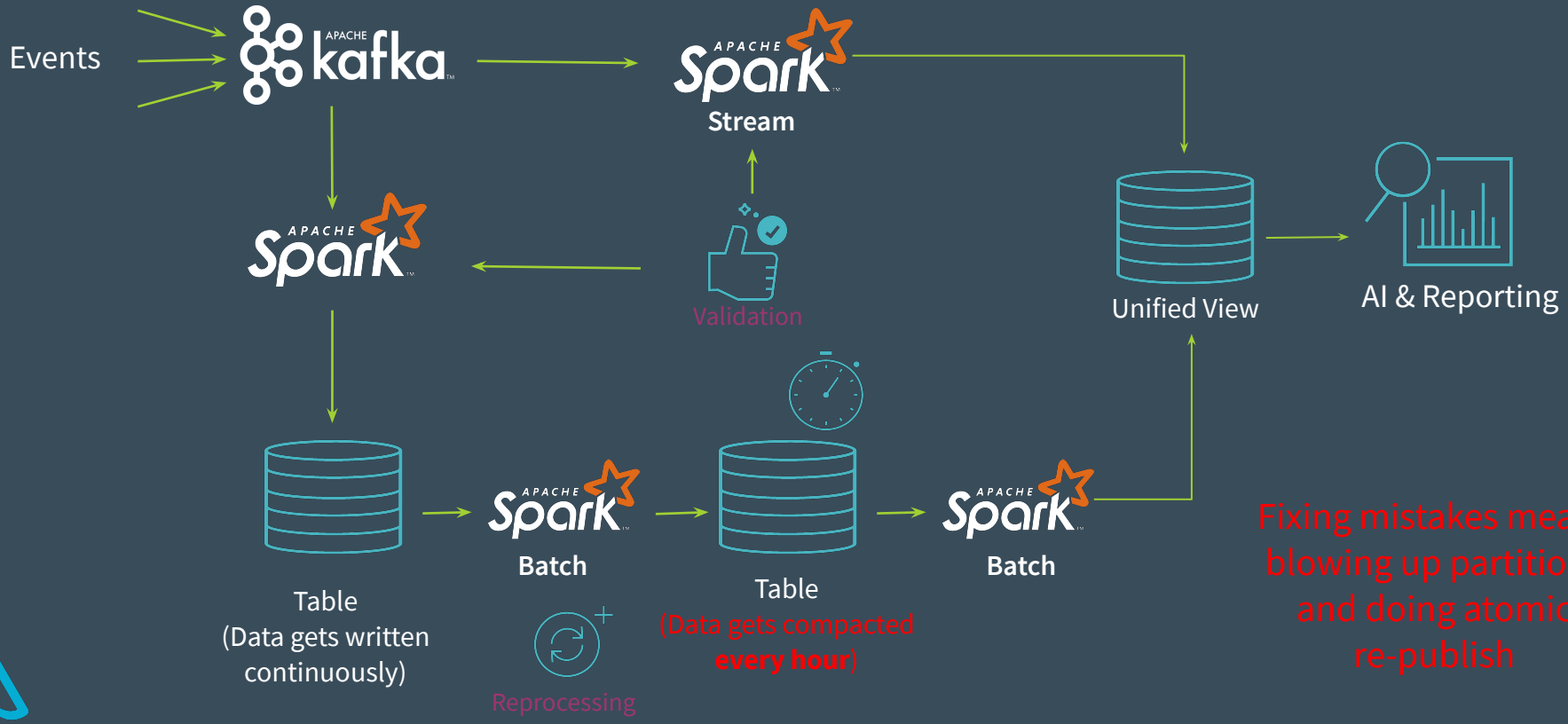
Complex pipelines increase engineering overhead



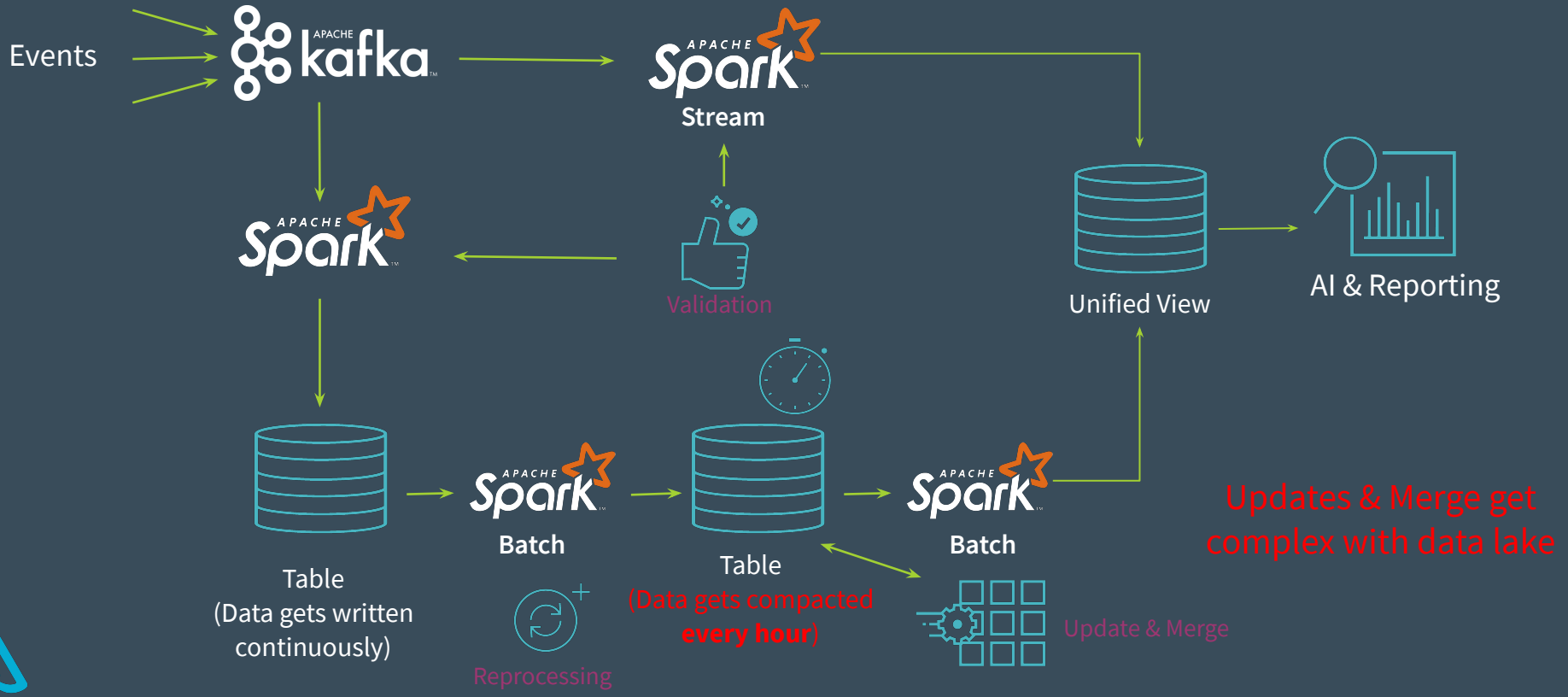
Complex pipelines increase engineering overhead



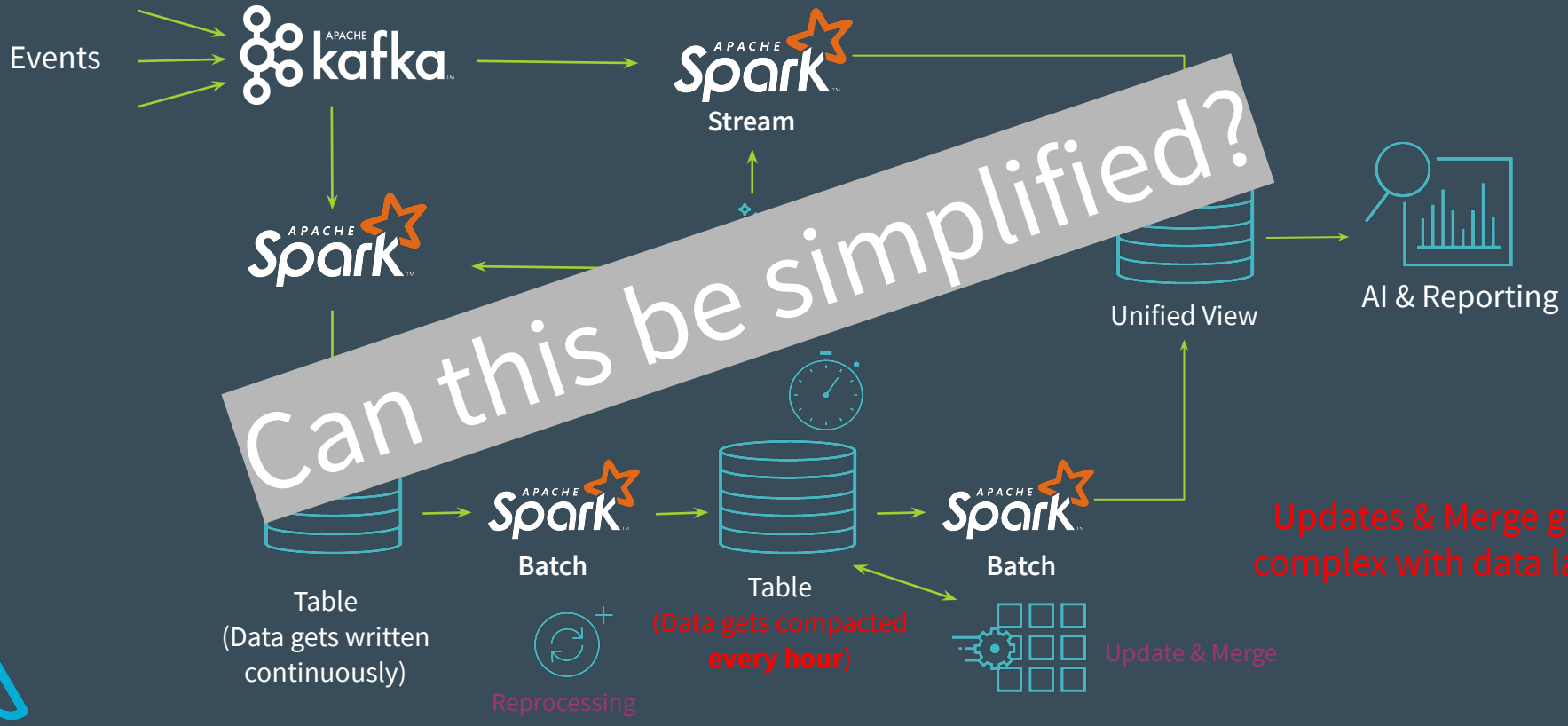
Complex pipelines increase engineering overhead



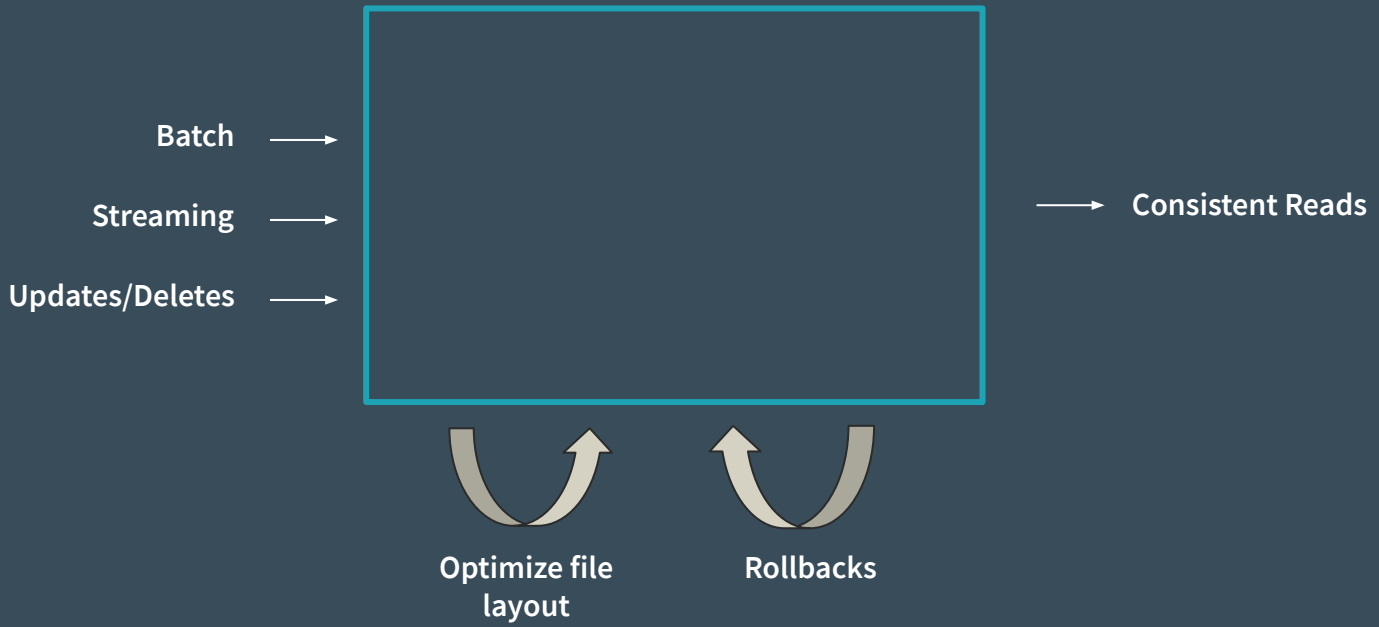
Complex pipelines increase engineering overhead



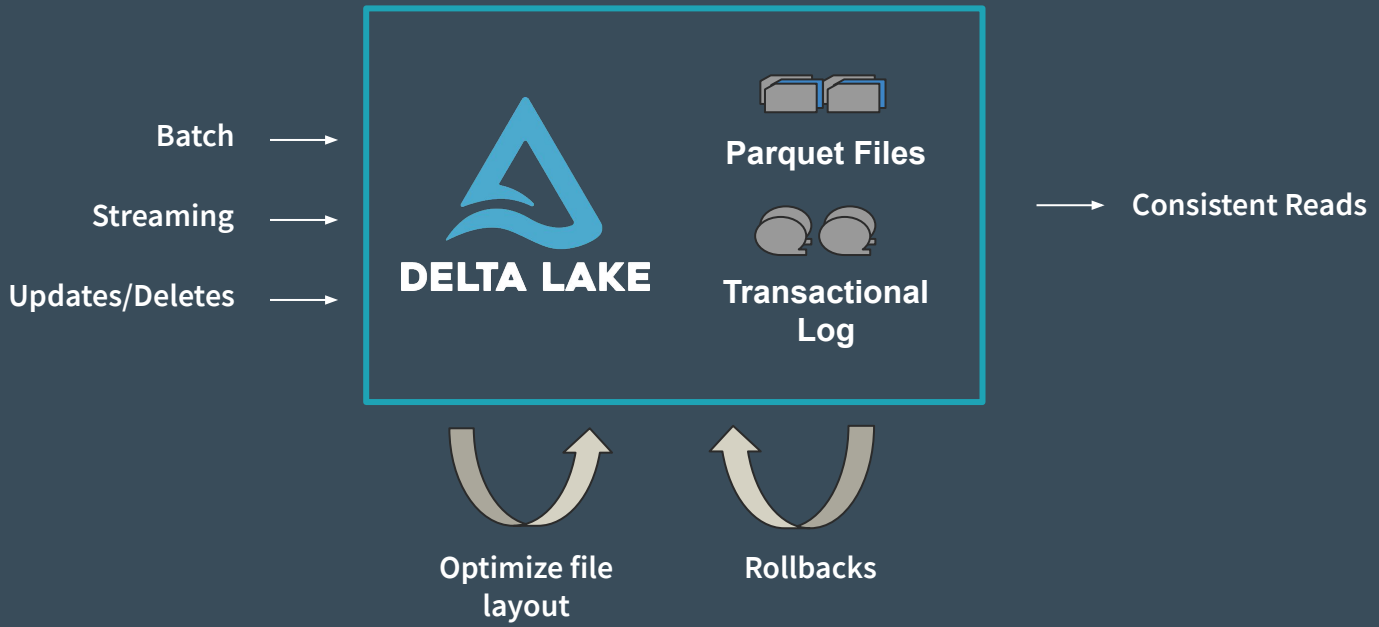
Complex pipelines increase engineering overhead



ACID Transactions



ACID Transactions



A New Open Source Standard for Building Data Lakes

Open Format Based on Parquet

With Transactions

Apache Spark™ APIs

A Data Practitioner's Dream...

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

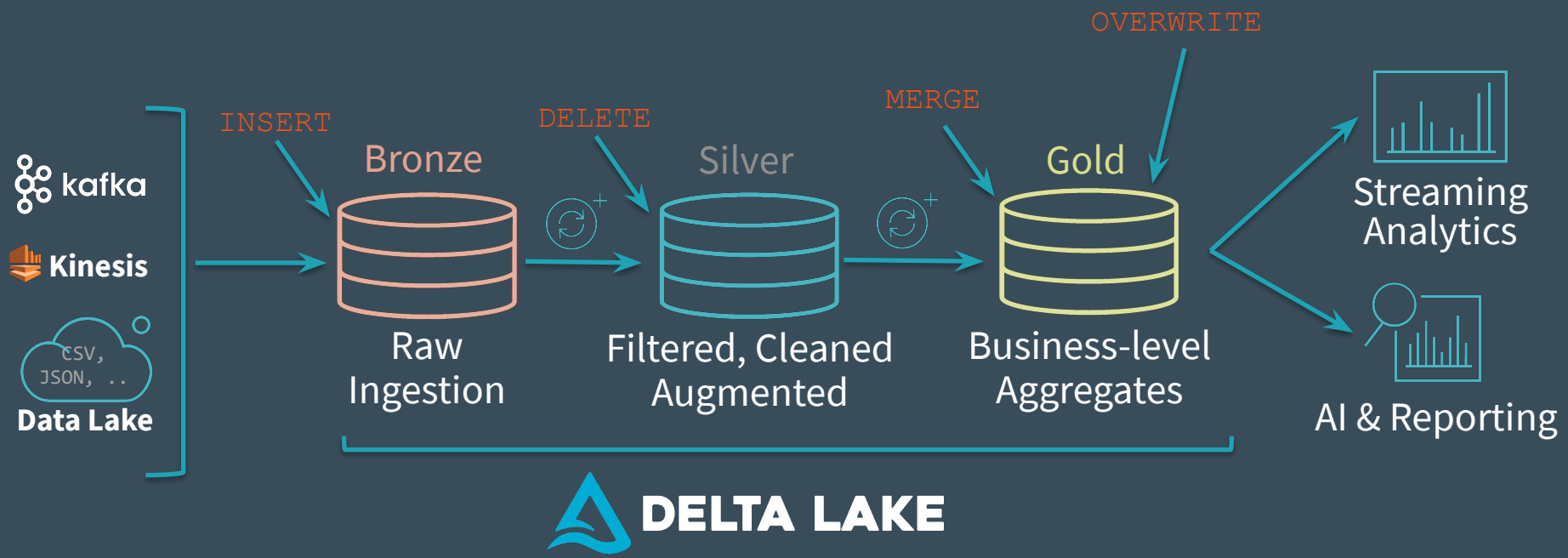
A Data Practitioner's Dream...

Key Features

- ACID Transactions
- Schema Enforcement
- Unified Batch & Streaming
- Time Travel/Data Snapshots

The Delta Architecture

A continuous data flow model to unify batch & streaming



Benefits of the Delta Architecture

- Reduce end-to-end pipeline SLA.
- Reduce pipeline maintenance burden.
- Eliminate lambda architecture for minute-latency use cases.

How we make this architecture a reality?

- **Optimized File Source** to stream efficiently from cloud stores
- **ACID Transactions** to read & write data reliably
- **Scalable metadata handling** for high throughput
- **Inserts, Updates and deletes** to handle late arriving data, change data capture and GDPR use cases
- **Data versioning** for rollbacks
- **Schema on write** option to enforce data quality

How do I use  DELTA LAKE ?

Get Started with Delta using Spark APIs

Add Spark Package

```
pyspark --packages io.delta:delta-core_2.12:0.5.0

bin/spark-shell --packages io.delta:delta-core_2.12:0.5.0
```

Maven

```
<dependency>
  <groupId>io.delta</groupId>
  <artifactId>delta-core_2.12</artifactId>
  <version>0.5.0</version>
</dependency>
```

Instead of `parquet`...

```
dataframe
    .write
    .format("parquet")
    .save("/data")
```

... simply say `delta`

```
dataframe
    .write
    .format("delta")
    .save("/data")
```

Upsert/Merge: Fine-grained Updates

```
MERGE INTO customers     -- Delta table
USING updates
ON customers.customerId = source.customerId
WHEN MATCHED THEN
    UPDATE SET address = updates.address
WHEN NOT MATCHED
    THEN INSERT (customerId, address) VALUES (updates.customerId,
updates.address)
```

Time Travel

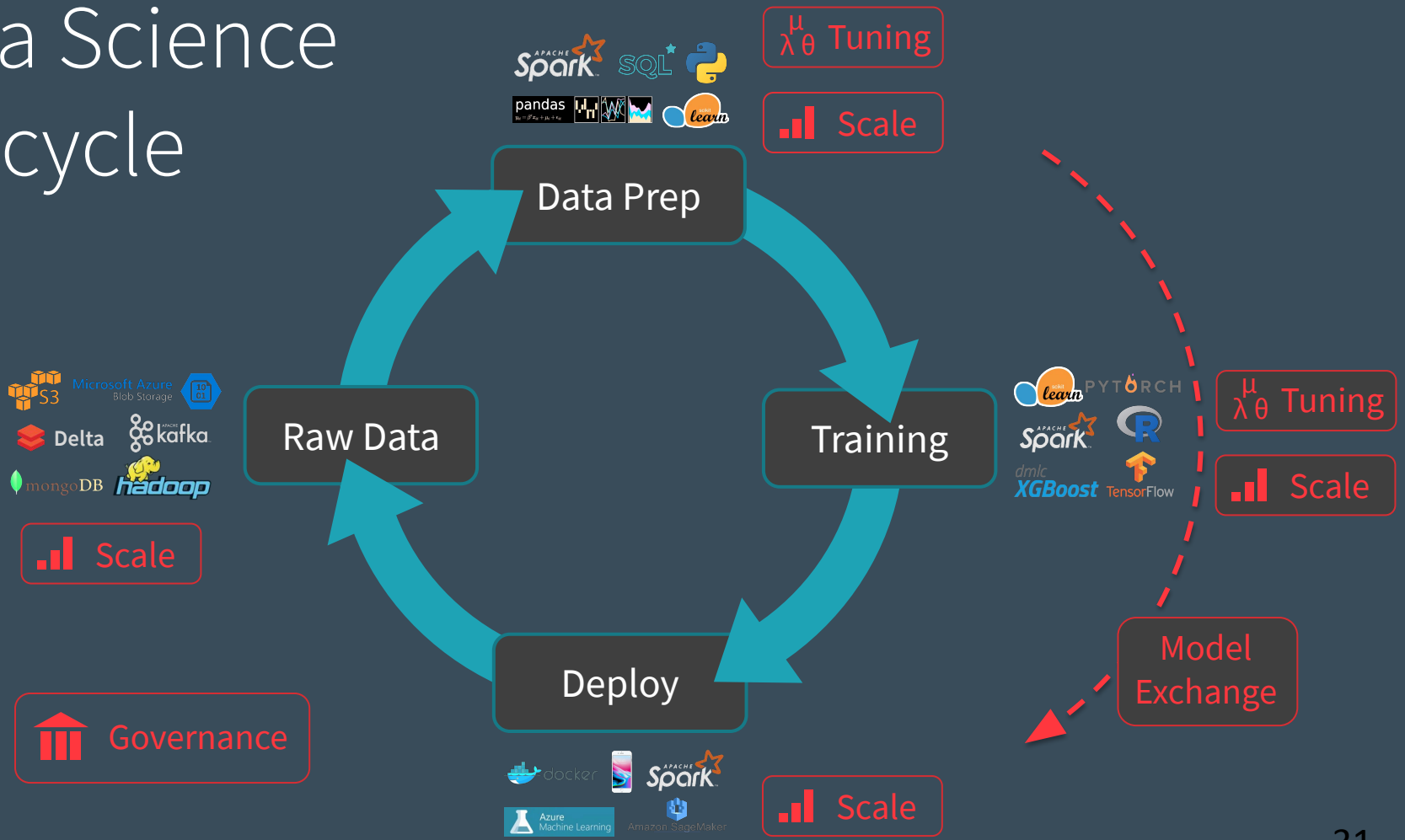
Reproduce experiments & reports

```
SELECT count(*) FROM events
TIMESTAMP AS OF timestamp

SELECT count(*) FROM events
VERSION AS OF version

spark.read.format("delta").option("timestampAsOf",
timestamp_string).load("/events/")
```

Rollback accidental bad writes

```
INSERT INTO my_table
    SELECT * FROM my_table TIMESTAMP AS OF
    date_sub(current_date(), 1)
```

Data Science Lifecycle



Build your own Delta Lake at **https://delta.io**

Delta Lake Connectors

Standardize your big data storage with an open format accessible from various tools

Delta Lake Partners and Providers

More and more partners and providers are jumping working with Delta Lake

Google Dataproc

Privacera

Informatica

Streamsets

WANDisco

Qlik

Users of Delta Lake

Tencent 腾讯 · Alibaba Group · COMCAST · viacom · ciena · JAM CITY

edmunds · databricks · healthdirect Australia · Booz | Allen | Hamilton · McAfee Together is power. · CONDÉ NAST

TILTING POINT · Mc Graw Hill Education · acorns · EVERQUOTE · Namely · usermind

ZEISS · split · upwork · DOLLAR SHAVE CLUB · wehkamp

Notebook from today's webinar

Try the notebook from today's webinar on Databricks Community Edition!

Download the notebook at https://dbricks.co/dlw-01

Questions

https://delta.io

https://mlflow.org