newwave

Apache Spark workshop with Hands-on Labs

NewWave Data Science Group

Agenda

- Welcome
- Follow the instruction to set up you community edition of Databricks
- We will give an overview of Spark, Databrick
- We will be walking you though an end-to-end project today

  Data and notebooks could be download at: https://github.com/Celia4444/Spark_workshop_7.7.20

Schedule

(10 mins)	Welcome and Introduction
(10 mins)	Register community addition of Databricks
(50 mins)	Lab #1. Delta lake and Data Engineering pipeline (upload data, cleaning the data, create delta tables)
(5 mins)	Break
(50 mins)	Lab #2. ML model training example using Databricks (regression)
(5 mins)	Break
(20 mins)	Data Science with MLflow
(15 mins)	Lab #3. Machine learning experiment management by MLflow

About the Speakers

Bereket Araia, Data Engineer

Jaehoon Jeong, Data Scientist

Xue Yang, Data Scientist

Coming online event

Date And Time
Thu, August 6, 2020
1:00 PM – 2:00 PM EDT

https://www.eventbrite.com/e/introduction-to-looker-for-data-analysts-tickets-107735359360

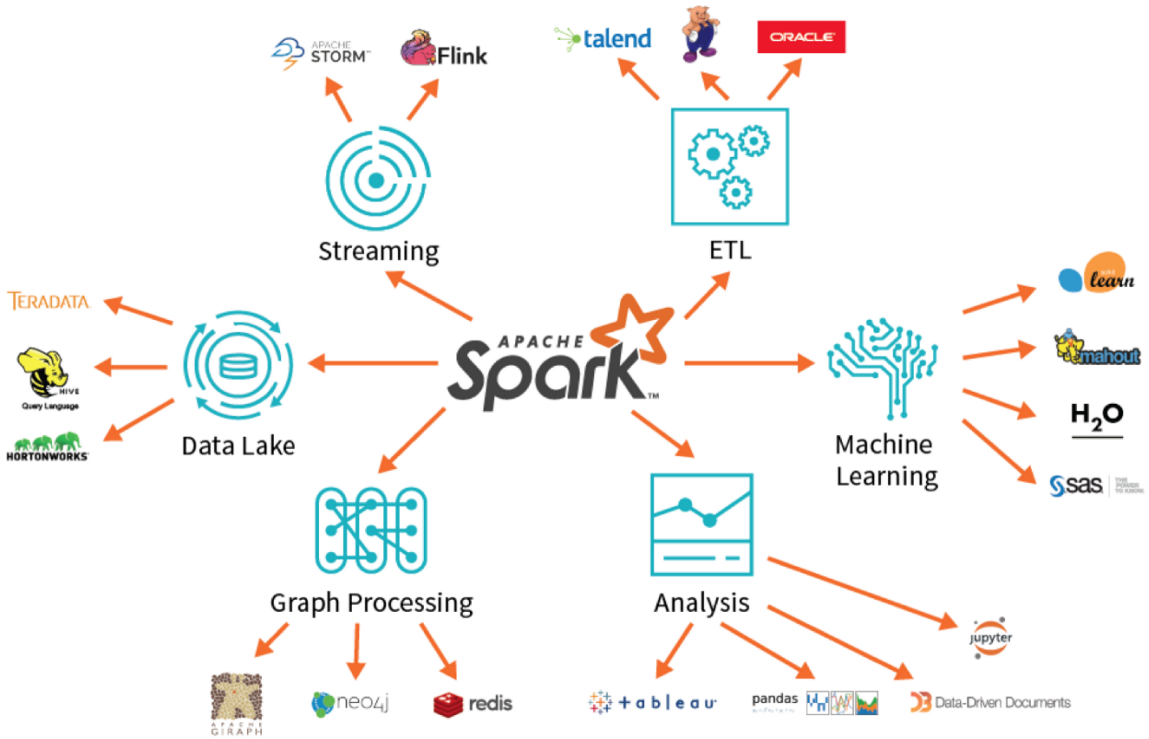
newwave

Introduction to Apache Spark and Databricks

NewWave Data Science Group
Xue Yang, Data Scientist

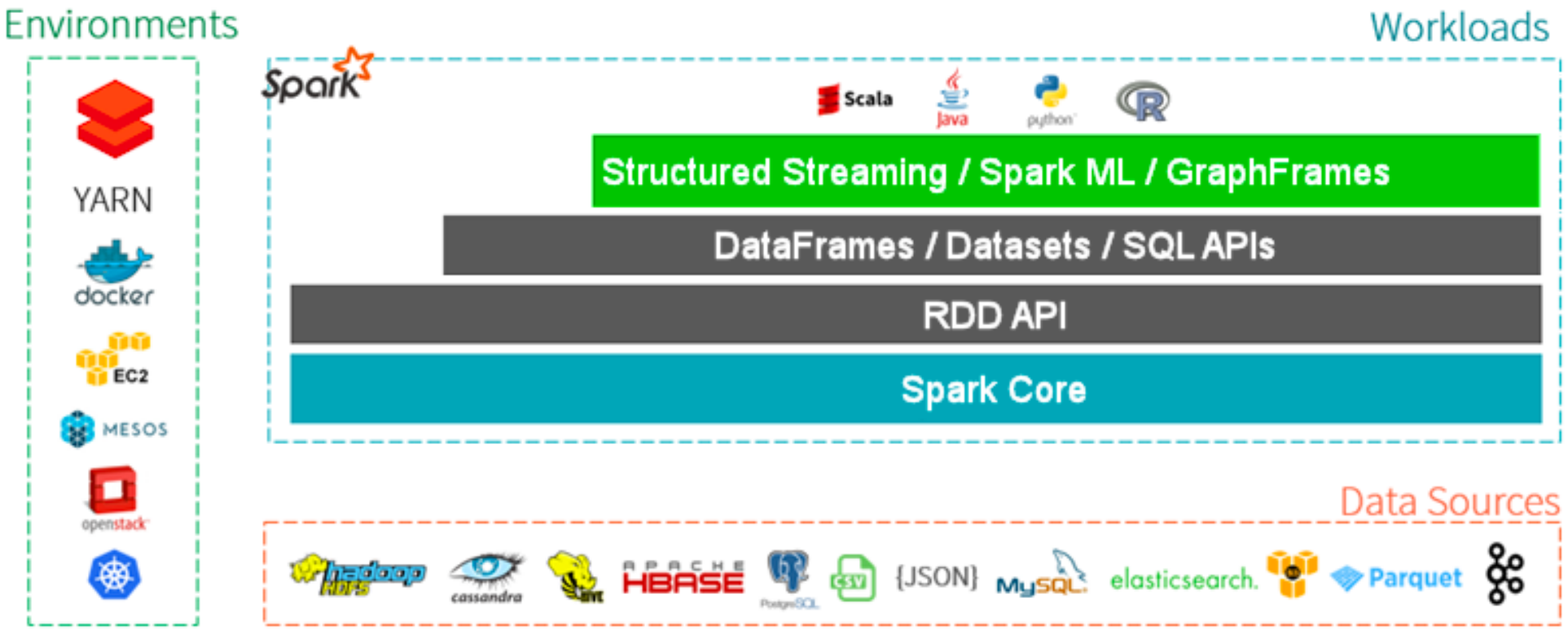
Apache Spark

"Unified analytics engine for big data processing, with built-in modules for streaming, SQL, machine learning and graph processing "



newwave

- Apache Spark started as a research project at the University of California AMPLab, in 2009 by Matei Zaharia.
- In 2013, the project was donated to the Apache Software Foundation. (open sourced)
- APIs: Scala, Java, Python, R and SQL
- In memory engine that is up to 100 times faster than Hadoop


Environments
YARN
docker
EC2
MESOS
openstack
Workloads
Spark
Scala
Java
python
Structured Streaming / Spark ML / GraphFrames
DataFrames / Datasets / SQL APIs
RDD API
Spark Core
Data Sources
hadoop HDFS
cassandra
APACHE HBASE
PostgreSQL
CSV
{JSON}
MySQL
elasticsearch.
Parquet

Who is Databricks

- Databricks was started by Spark founder Matei Zaharia.
- Today, Databricks remains the #1 contributor to Apache Spark.
- Fully committed to maintaining Apache Spark as an Open Source project.
- *"Provides a Unified Analytics Platform that accelerates innovation by unifying data science, engineering, and business."*
  - Databricks Workspace - Interactive Data Science & Collaboration.
  - Databricks Workflows - Production Jobs & Workflow Automation.
  - Databricks Runtime
  - Databricks I/O (DBIO) - Optimized Data Access Layer
  - Databricks Serverless - Fully Managed Auto-Tuning Platform
  - Databricks Enterprise Security (DBES) - End-To-End Security & Compliance

Community Edition of Databricks

- Play with free community edition provided by Databricks with 6 GB memory space.

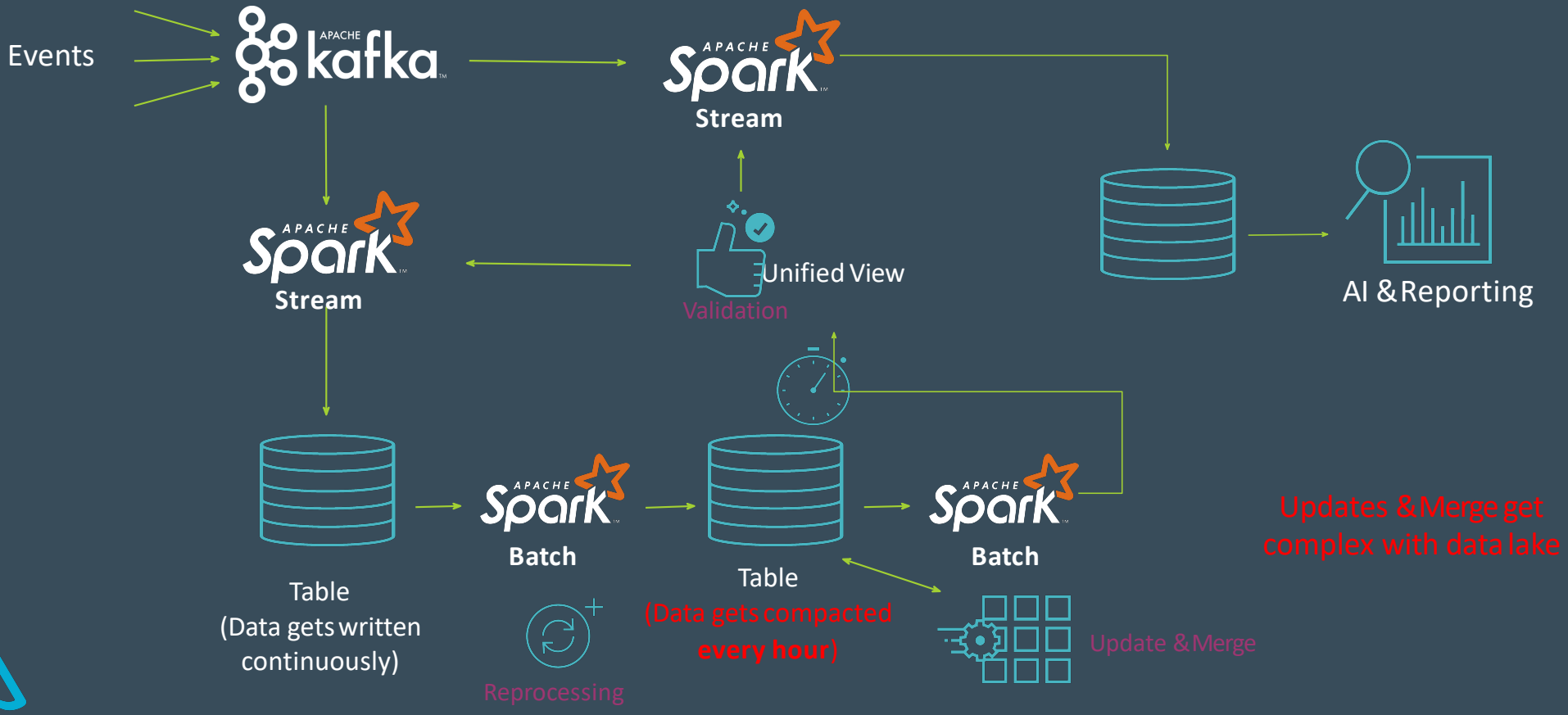
Lab 1

Delta lake + Data Engineering pipeline

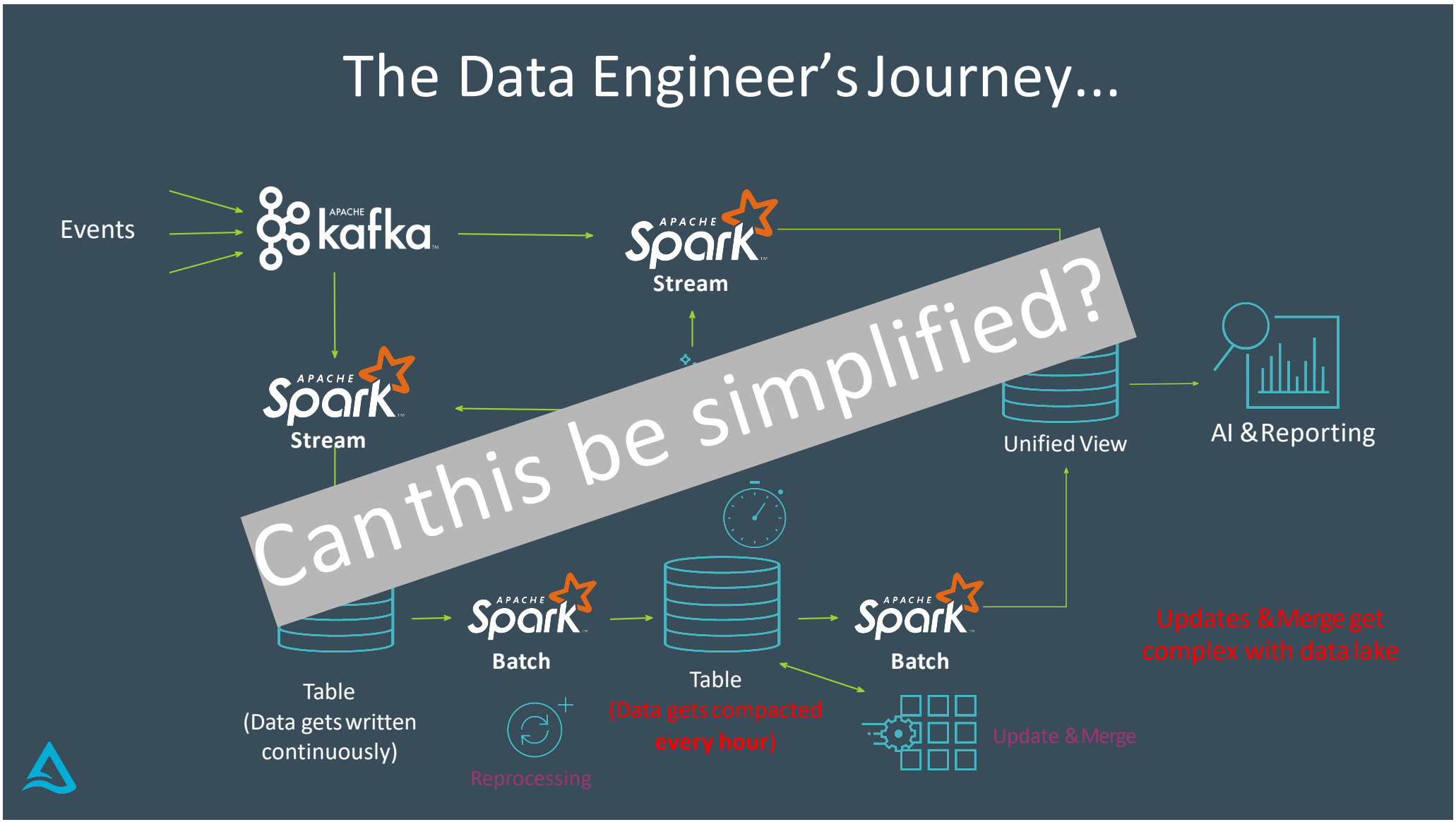
NewWave Data Science Group
Bereket Araia, Data Engineer

newwave

The Data Engineer's Journey...



The Data Engineer's Journey...



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

What's missing?

1. Ability to read consistent data while data is being written
2. Ability to read incrementally from a large table with good throughput
3. Ability to rollback in case of bad writes
4. Ability to replay historical data along new data that arrived
5. Ability to handle late arriving data without having to delay downstream processing

So… What is the answer?

STRUCTURED STREAMING

+

DELTA LAKE

=

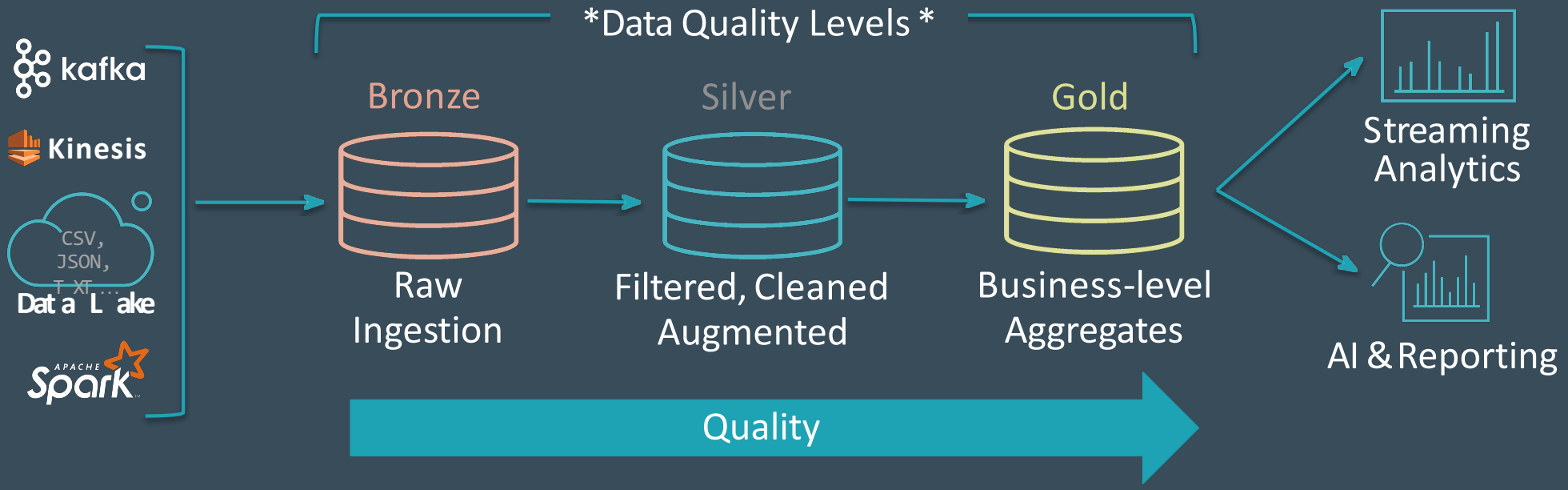
The Delta Architecture

1. Unify batch & streaming with a continuous data flow model
2. Infinite retention to replay/reprocess historical events as needed
3. Independent, elastic compute and storage to scale while balancing costs

Let’s try it instead with

DELTA LAKE

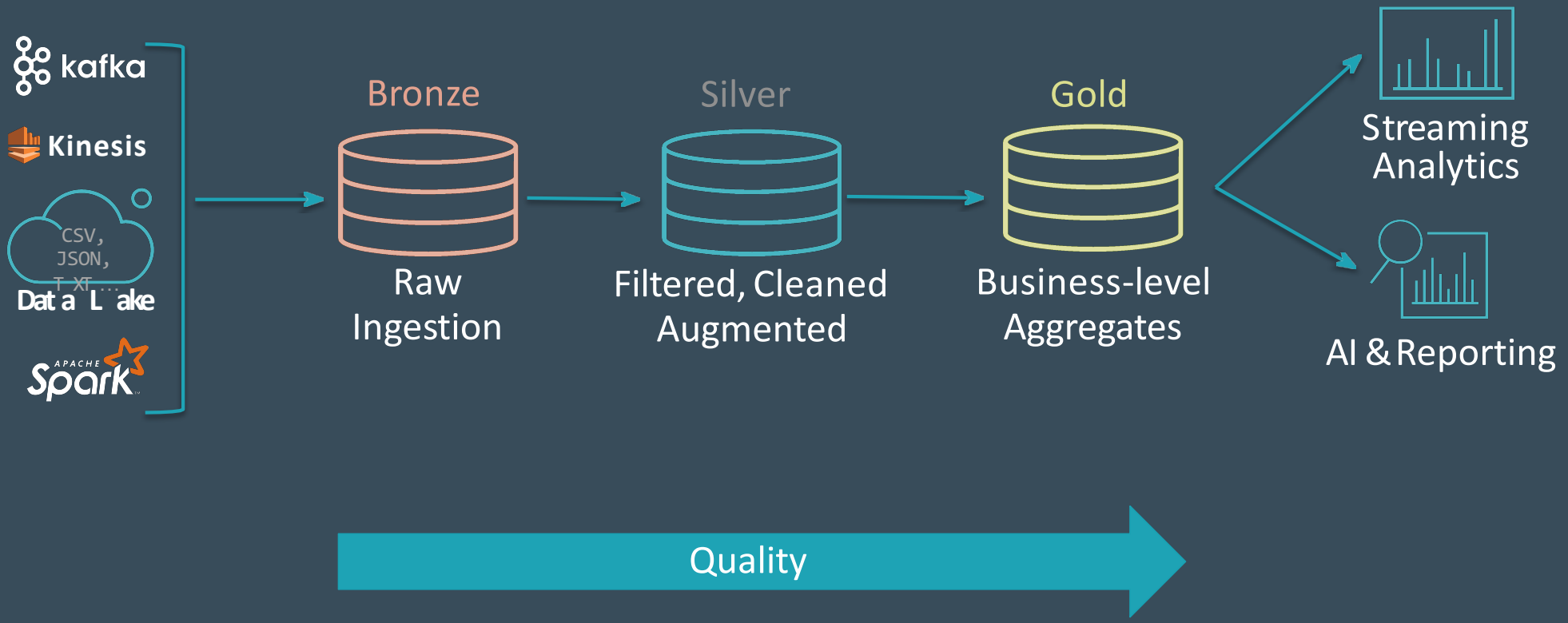
The DELTA LAKE



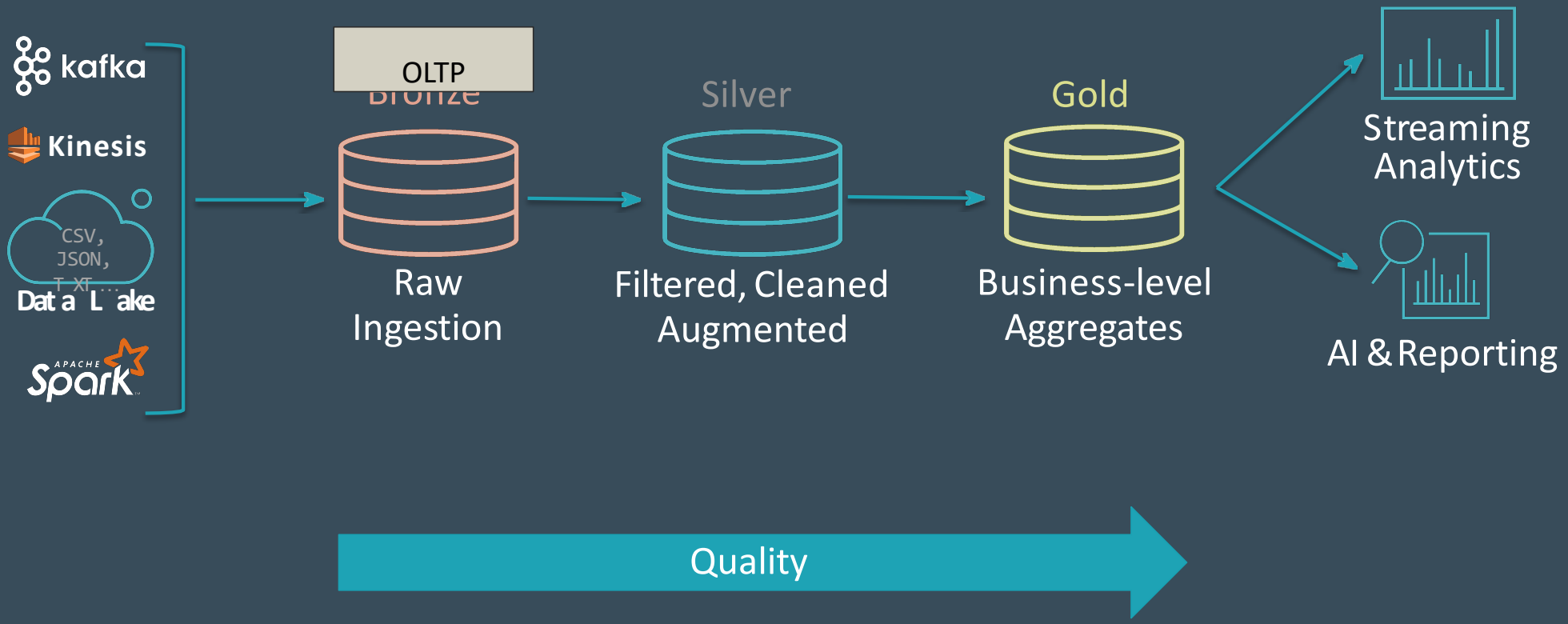
Delta Lake allows you to *incrementally* improve the quality of your data until it is ready for consumption.

What does this remind you of?

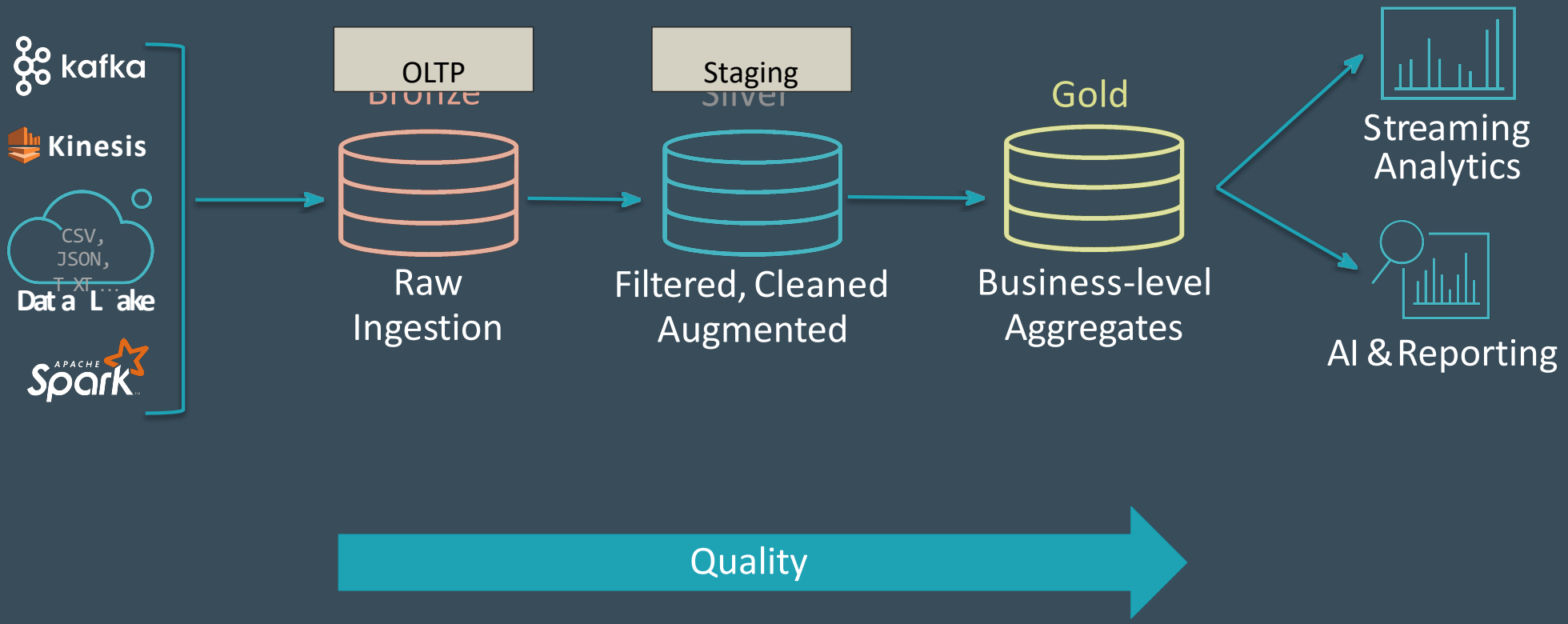
The Data Lifecycle



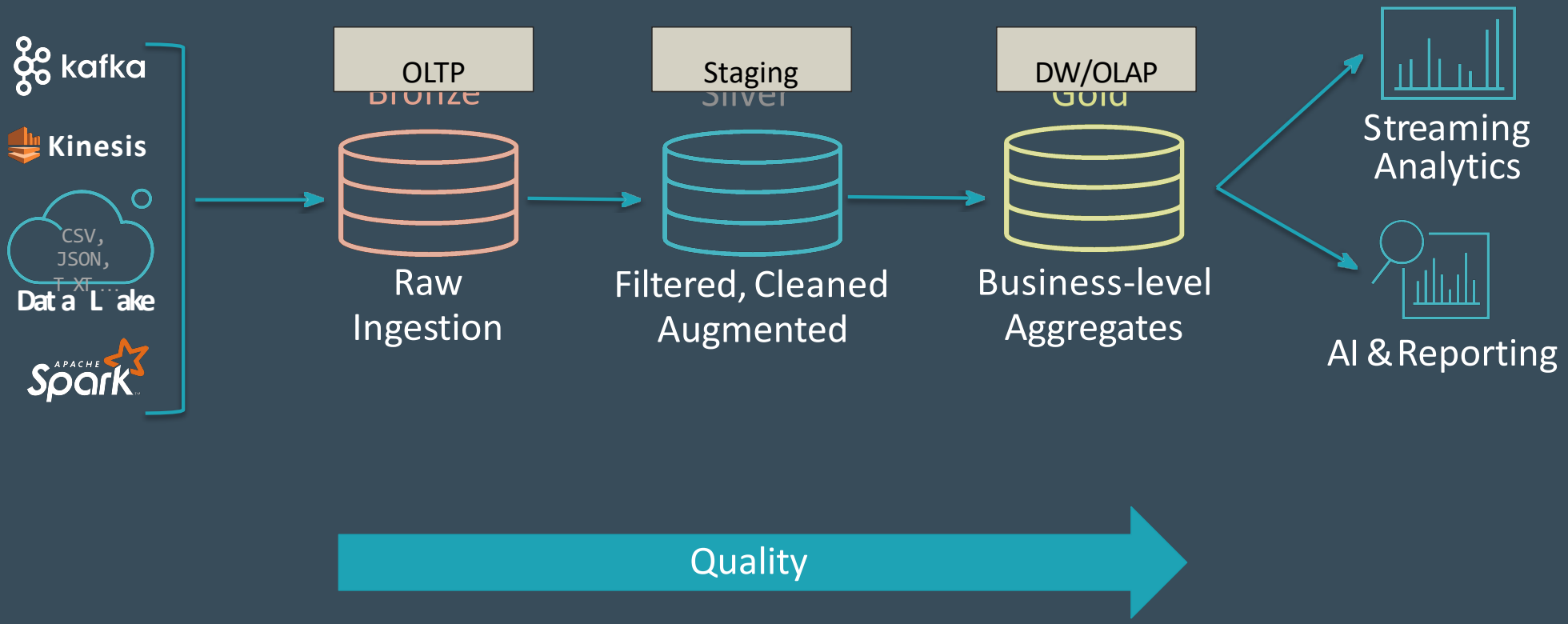
The Data Lifecycle



The Data Lifecycle

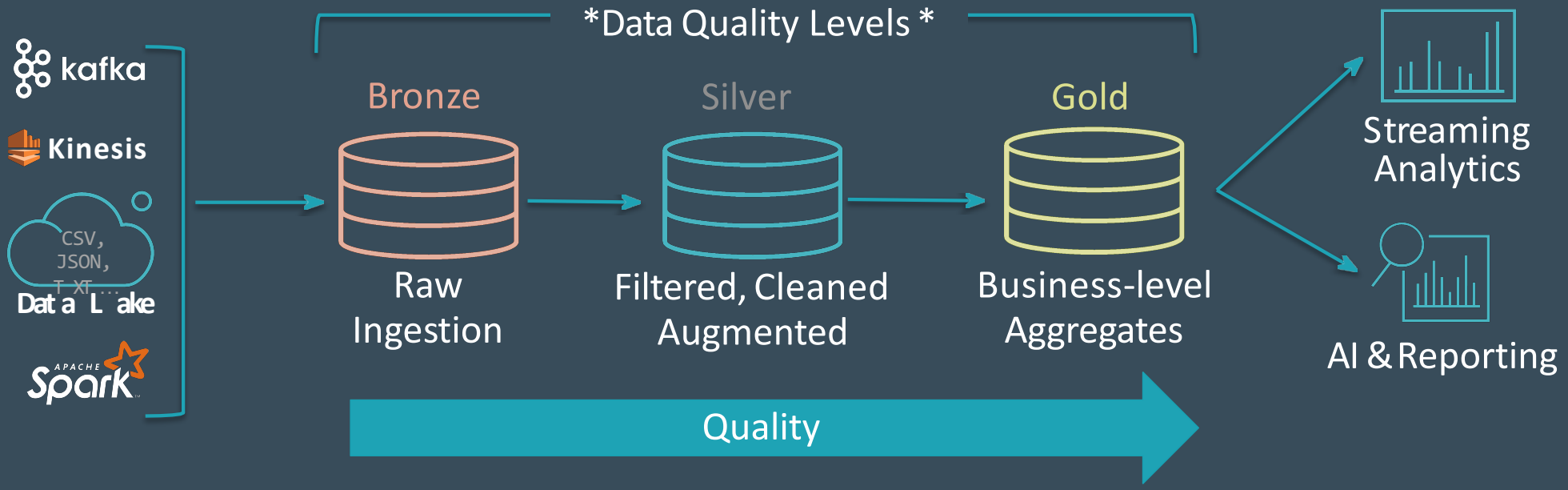


The Data Lifecycle



Transitioning from the Data Lifecycle to the Delta Lake Lifecycle

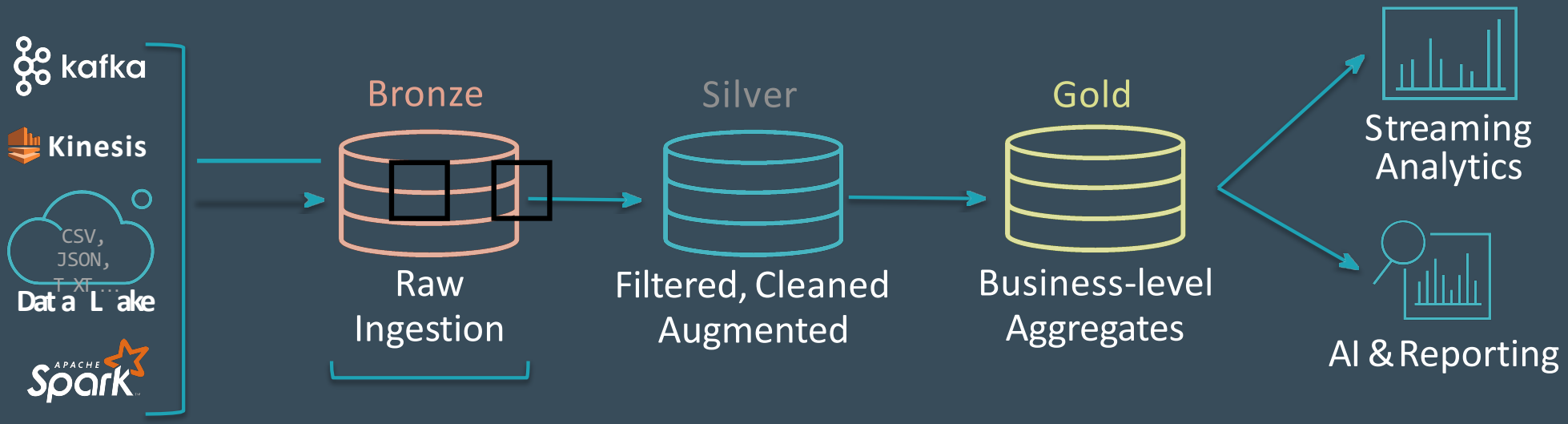
The DELTA LAKE



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

OLTP

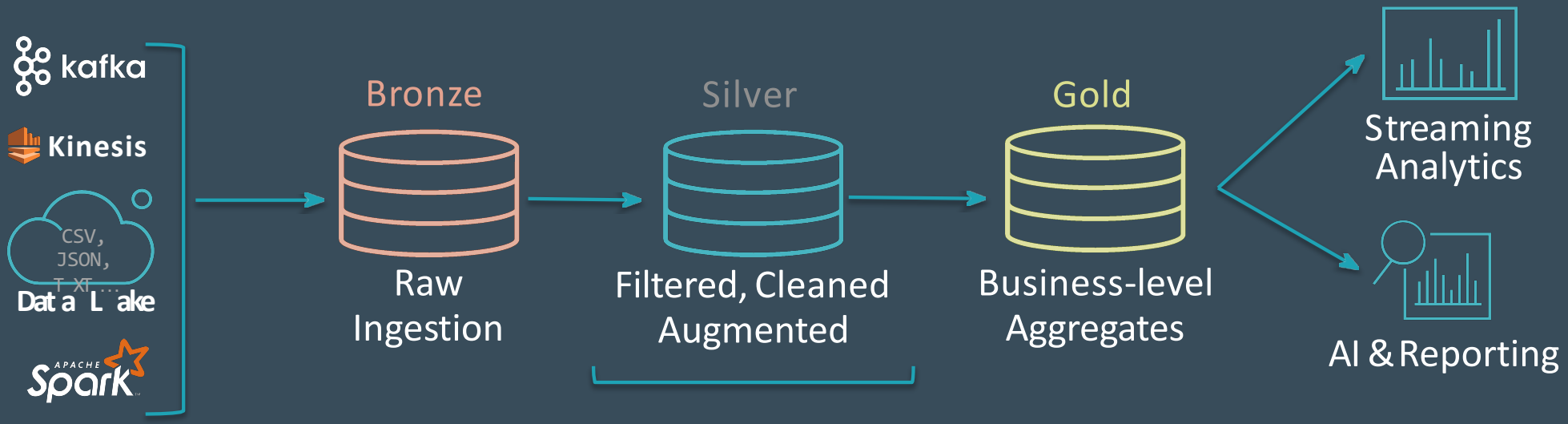
The DELTA LAKE



- Dumping ground for raw data
- Often with long retention (years)
- Avoid error-prone parsing

Staging

The DELTA LAKE

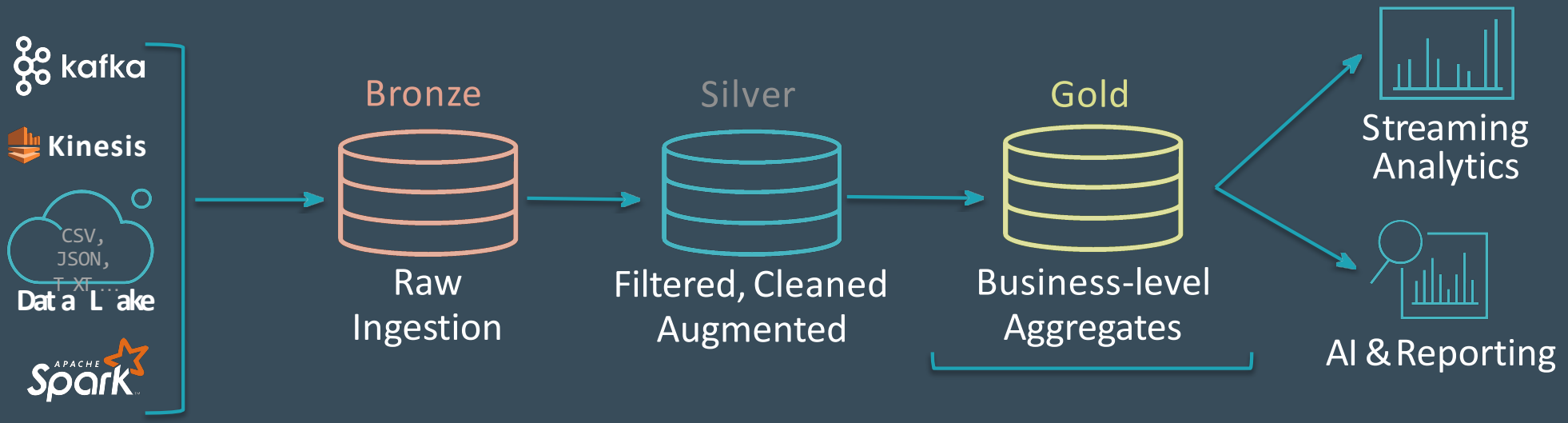


Intermediate data with some cleanup applied.

Queryable for easy debugging!

DW/OLAP

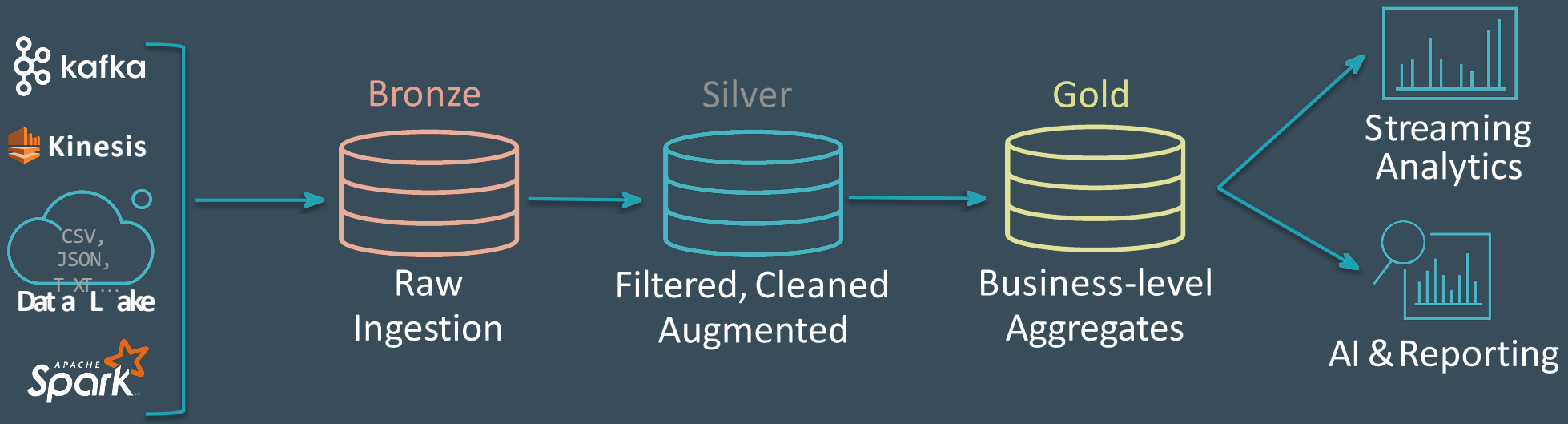
The DELTA LAKE



Clean data, ready for consumption.

Read with Spark or Presto*

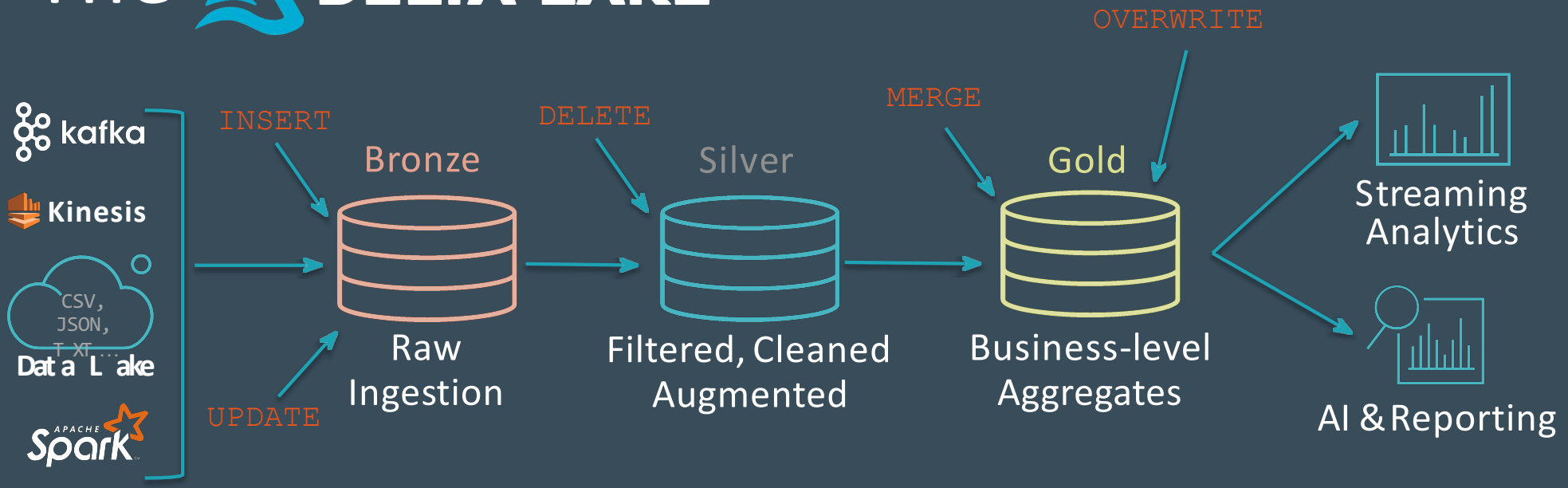
The DELTA LAKE



Streams move data through the Delta Lake

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

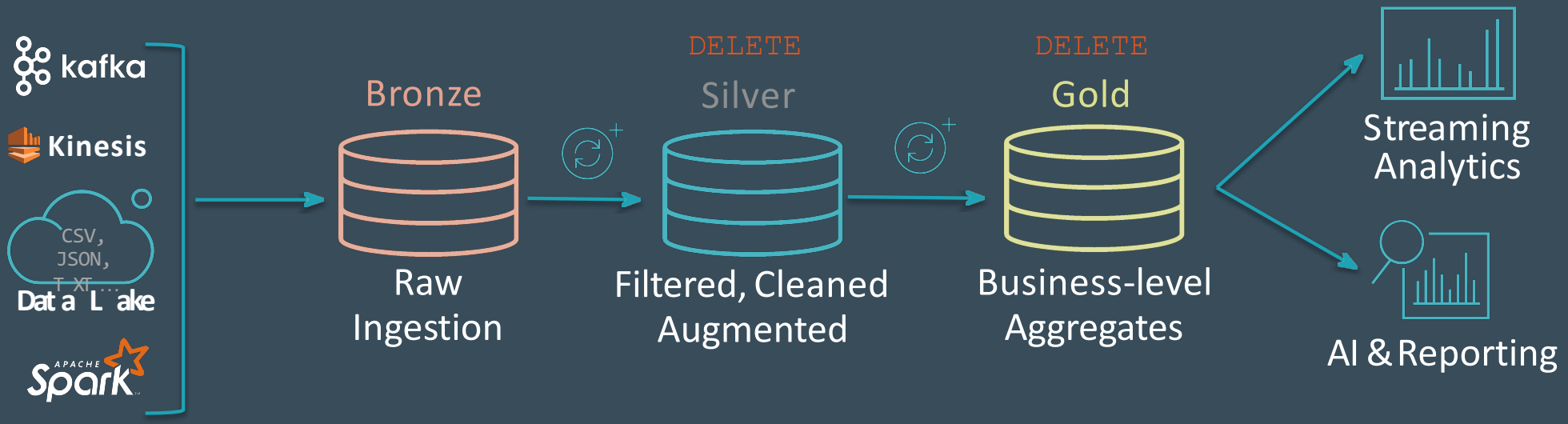
The DELTA LAKE



Delta Lake also supports batch jobs and standard DML

- Retention
- Corrections
- GDPR

The DELTA LAKE



Easy to recompute when business logic changes:

- Clear tables
- Restart streams

Let's see  DELTA LAKE in action!

The Delta Architecture

Connecting the dots...

Connecting the dots...

1. Ability to **read consistent data** while data is being written → Snapshot isolation between writers and readers

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

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

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

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

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

Who is using  DELTA LAKE?

How do I use  DELTA LAKE ?

Get Started with Delta using Spark APIs

Add Spark Package

```
pyspark --packages io.delta:delta-core_2.12:0.1.0

bin/spark-shell --packages io.delta:delta-core_2.12:0.1.0
```

Maven

```
<dependency>
  <groupId>io.delta</groupId>
  <artifactId>delta-core_2.12</artifactId>
  <version>0.1.0</version>
</dependency>
```

Instead of `parquet`...

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

... simply say `delta`

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

Build your own Delta Lake at

https://delta.io

How does  DELTA LAKE work?

Delta On Disk

Transaction Log

Table Versions

(Optional) Partition Directories

Data Files

```
my_table/
   _delta_log/
      00000.json
      00001.json
         date=2019-01-01/
            file-1.parquet
```

Table = result of a set of actions

Change Metadata – name, schema, partitioning, etc

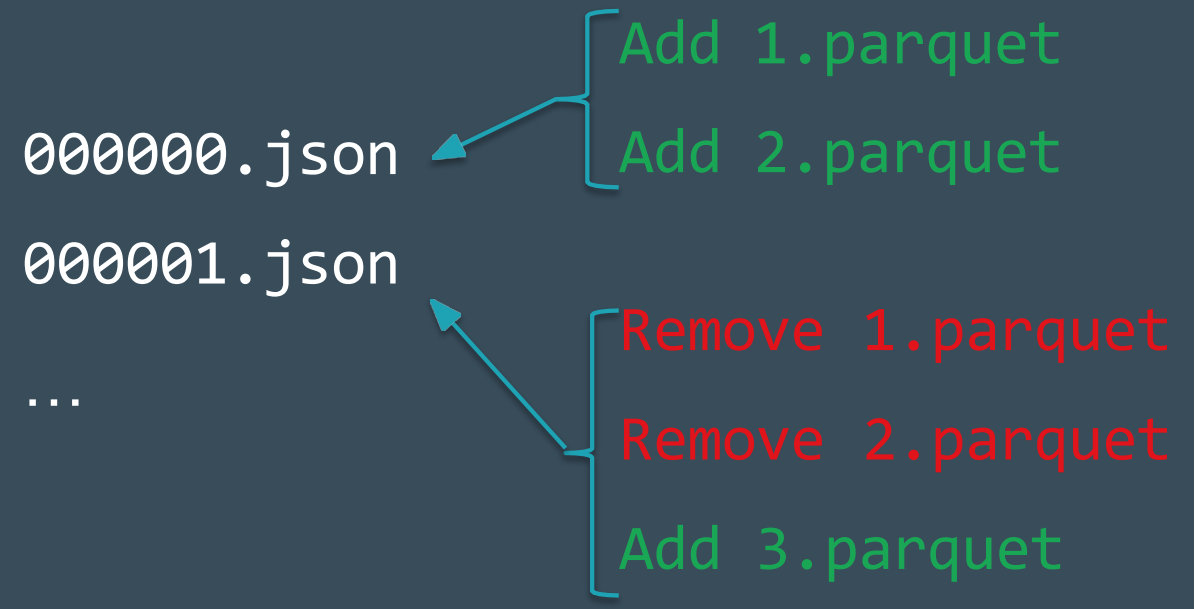
Add File – adds a file (with optional statistics)

Remove File – removes a file

Result: Current Metadata, List of Files, List of Txns, Version

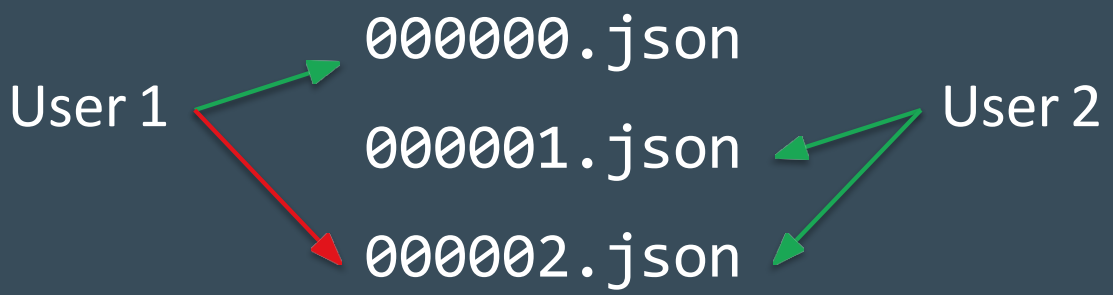
Implementing Atomicity

Changes to the table are stored as *ordered, atomic* units called commits



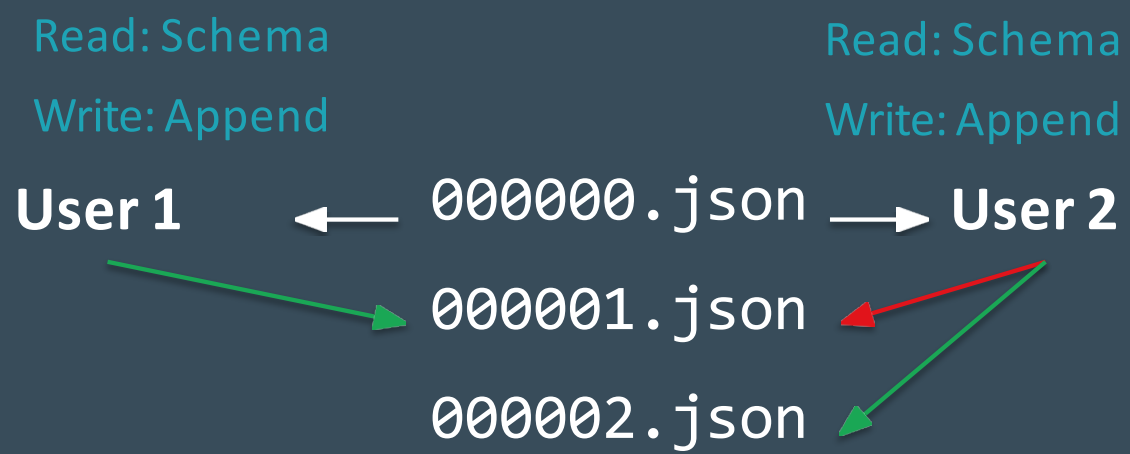
Ensuring Serializablity

Need to agree on the order of changes, even when there are multiple writers.



Solving Conflicts Optimistically

1. Record start version
2. Record reads/writes
3. Attempt commit
4. If someone else wins, check if anything you read has changed.
5. Try again.



Handling Massive Metadata

Large tables can have millions of files in them! How do we scale the metadata? Use Spark for scaling!

```
Add 1.parquet
Add 2.parquet
Remove 1.parquet
Remove 2.parquet
Add 3.parquet
```

Road Map

- 0.2.0 – Released!
  - S3 Support
  - Azure Blob Store and ADLS Support
- 0.3.0 Released!
  - UPDATE (Scala)
  - DELETE (Scala)
  - MERGE (Scala)
  - VACUUM (Scala)
- Rest of Q3
  - DDL Support / Hive Metastore

Build your own Delta Lake

https://delta.io

Try out these notebooks!

Delta Lake Primer: https://dbricks.co/dlw-01

Delta Lake + MLflow: https://dbricks.co/dlw-02

Lab 2

ML Model Training Example Using Databricks

NewWave Data Science Group
Jaehoon Jeong, Data Scientist

newwave

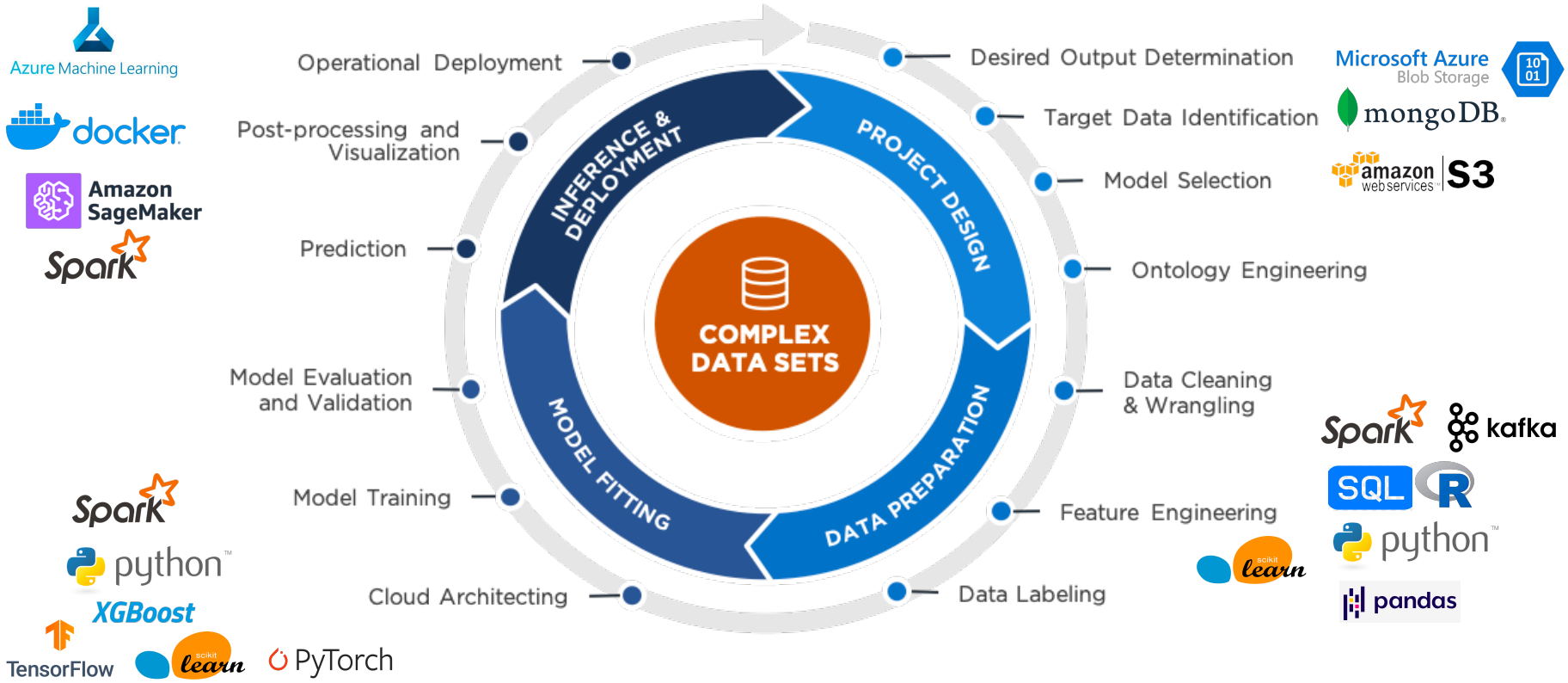
Machine Learning Model Management with MLflow

NewWave Data Science Group
Xue Yang, Data Scientist

Agenda

- Machine Learning Lifecycle
- Challenges in Managing ML Lifecycle
- How to solve with MLflow?
- Lab

Machine Learning Life Cycle



https://www.ntconcepts.com/introduction-to-the-machine-learning-lifecycle-hidden-challenges-blog-series/

More Challenges

hyper parameter tuning is a major elements of the machine learning life cycle

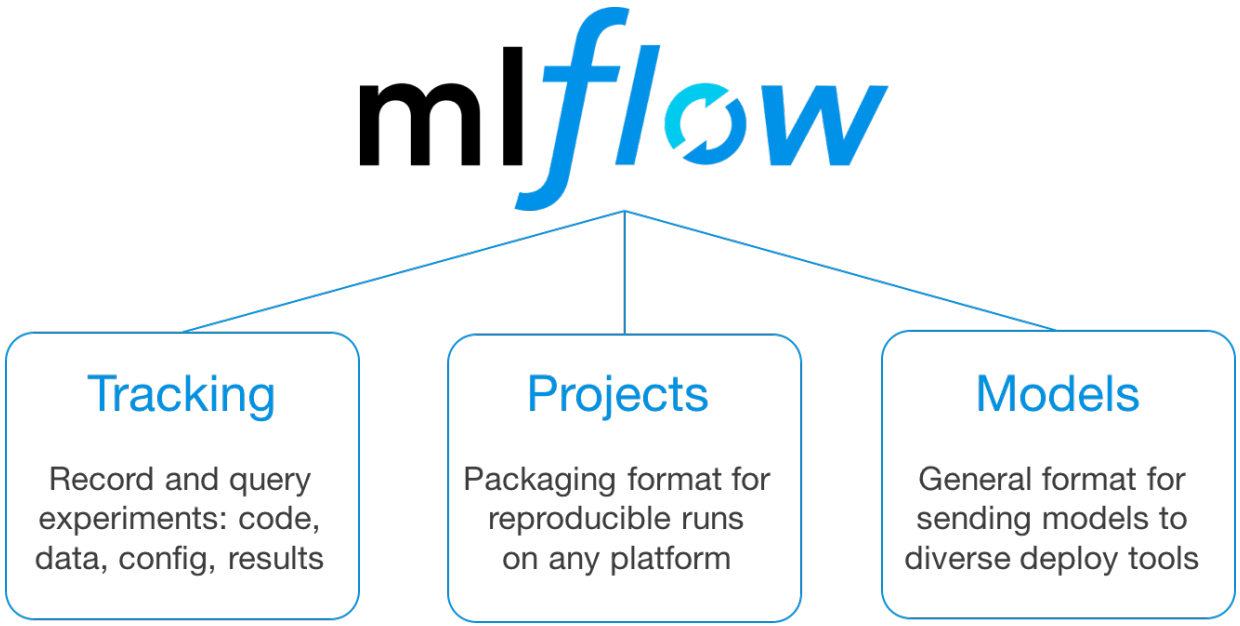
Scale

Model exchange and governance

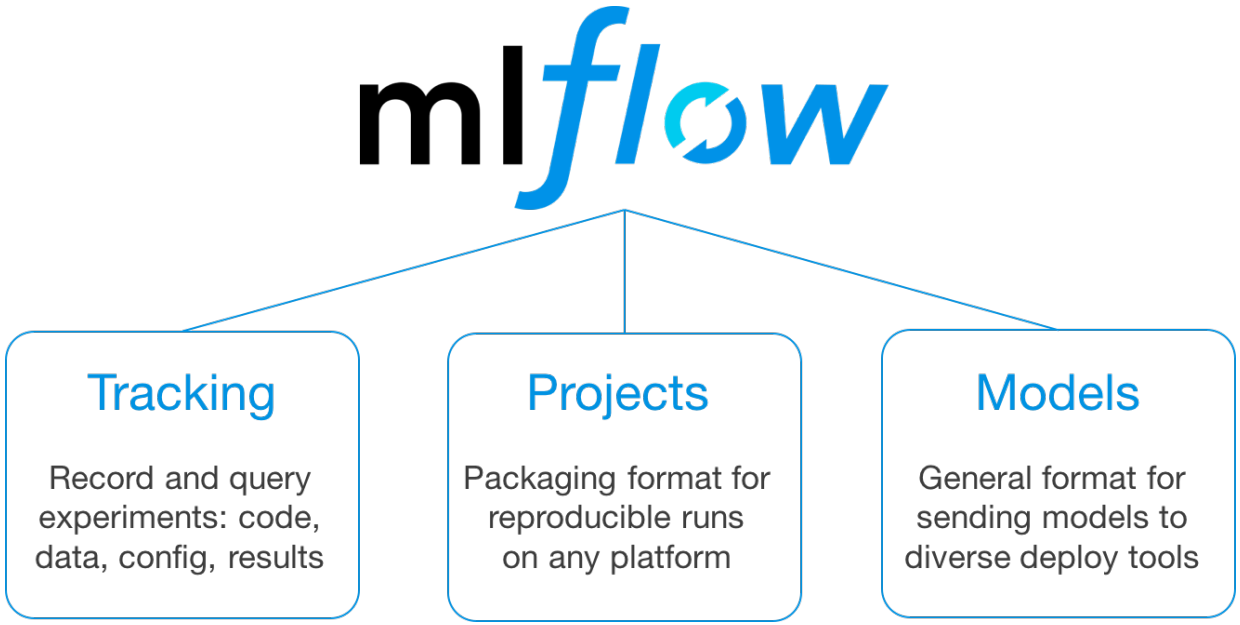
Open machine learning platform

- Works with any ML library & language
- Runs the same way anywhere(e.g. any cloud)
- Designed to be useful for 1 or 1000+ person orgs

More Challenges



More Challenges

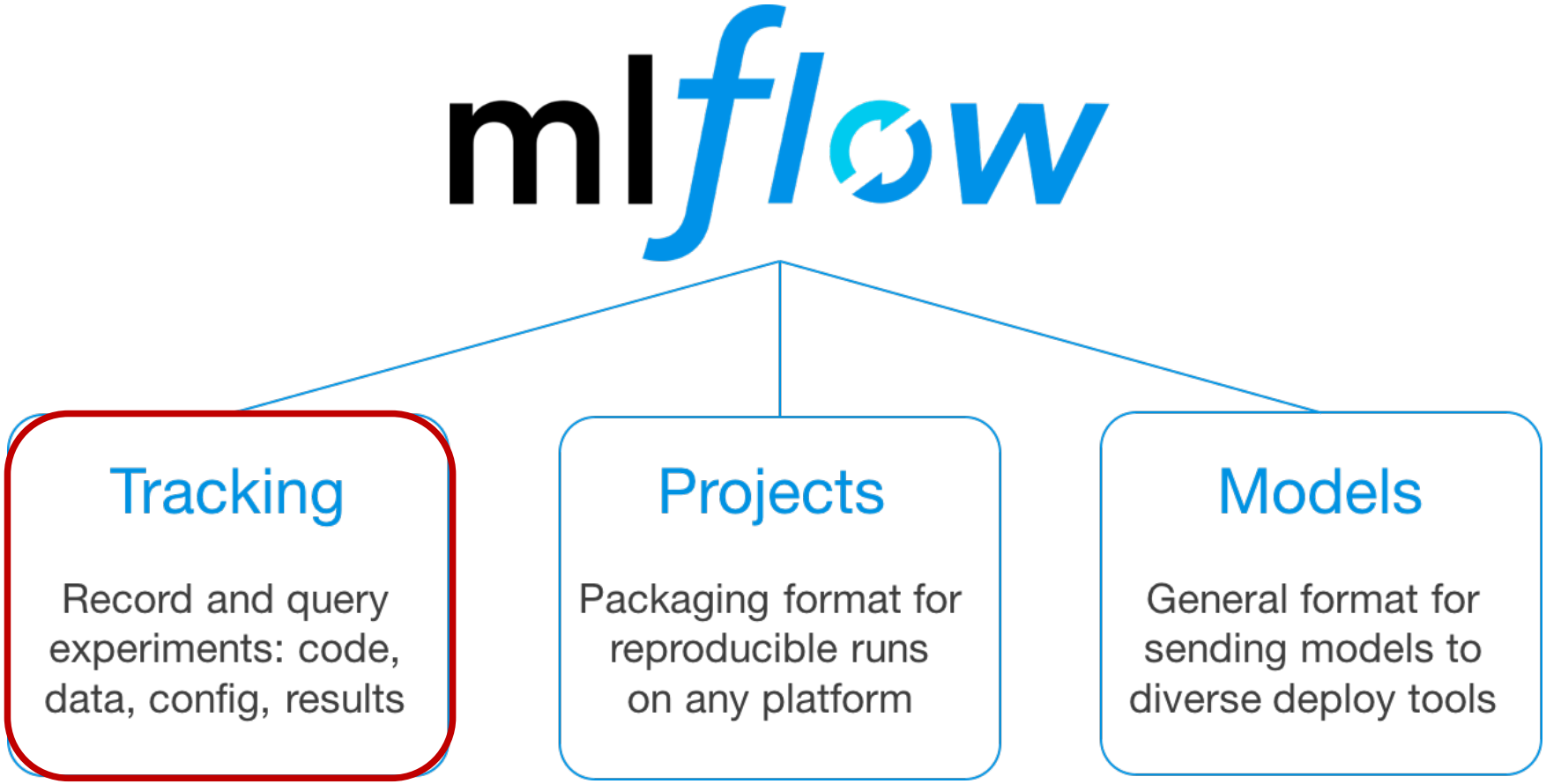


Model Registry

Store, annotate, discover, and manage models in a central repository

newwave

More Challenges



Mlflow Tracking

Parameters: key-value inputs to your code

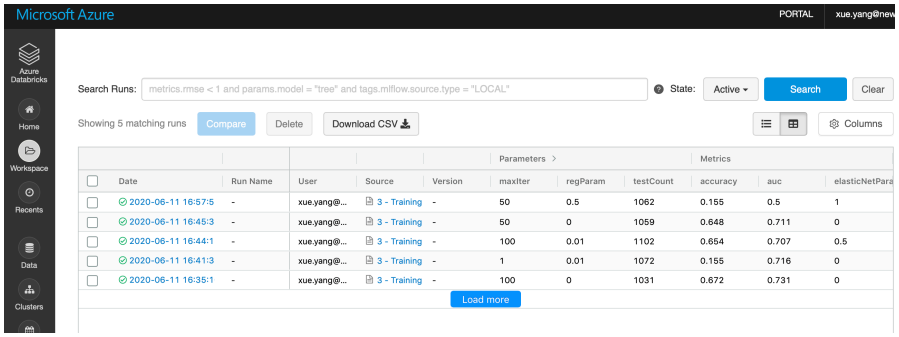
Metrics: numeric values

Source: training code that ran

Version: version of the training code

Artifacts: files, including data and models

Tags and Notes: any additional information



newwave

Mlflow Tracking

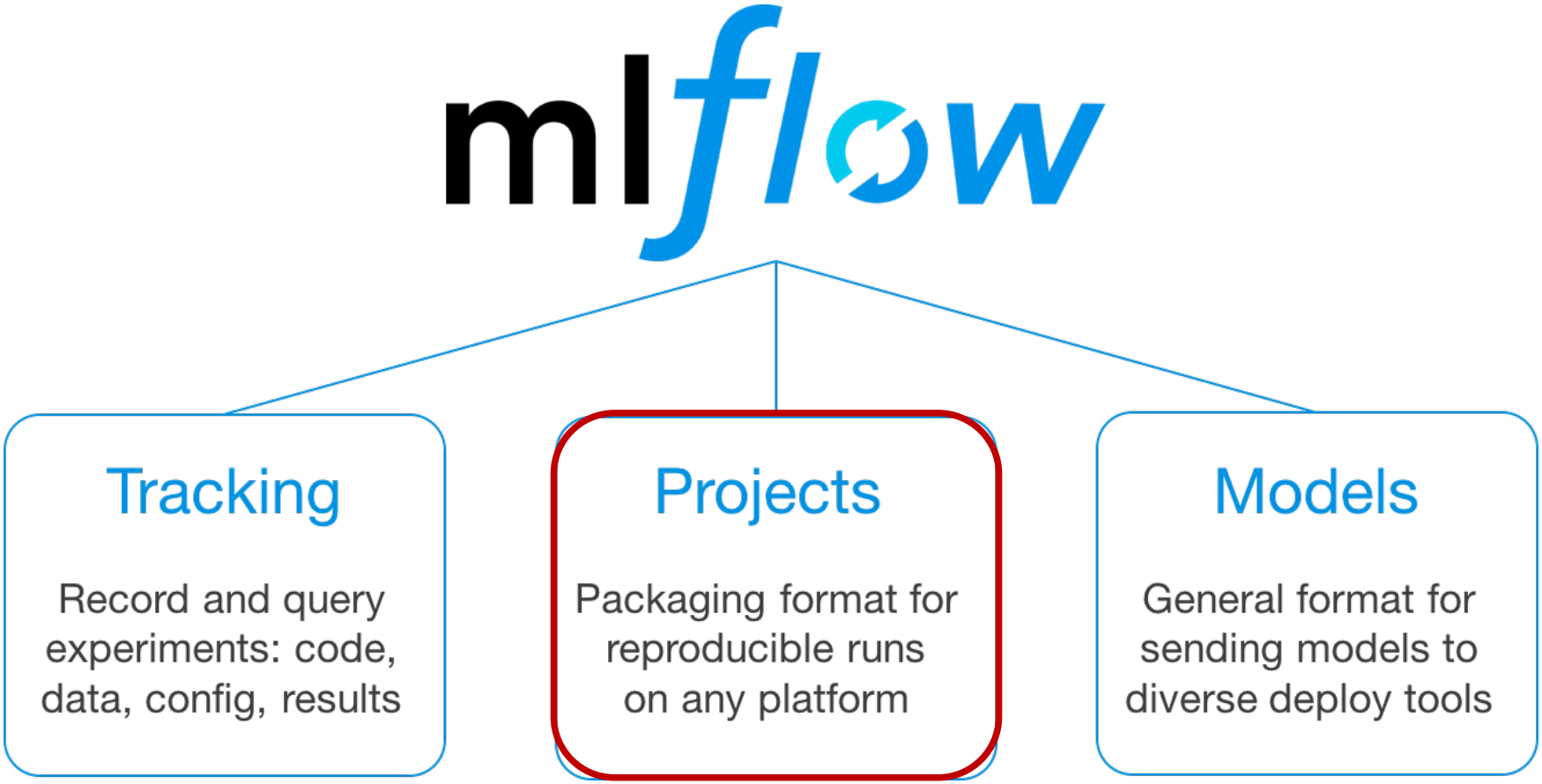
mlflow

Tracking

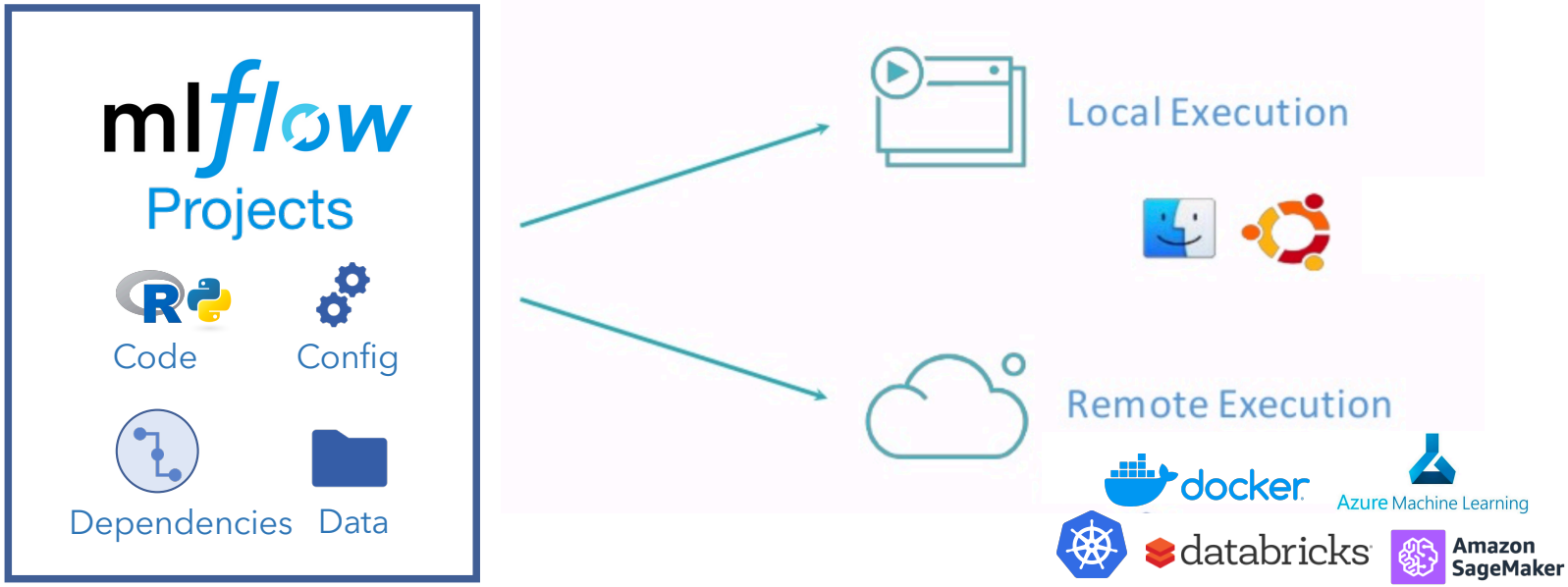
Record and query experiments: code, data, config, results

```
import mlflow
with mlflow.start_run() as run:

  # Log mlflow parameters to the model
  mlflow.log_param("A", A)
  mlflow.log_param("B", B)

  # Fit the model

  # Log mlflow metrics for the model
  mlflow.log_metric("accuracy", accuracy)
  mlflow.log_metric("auc", auc)

  # Log model generated
  mlflow.spark.log_model(model, "model")
```

More Challenges



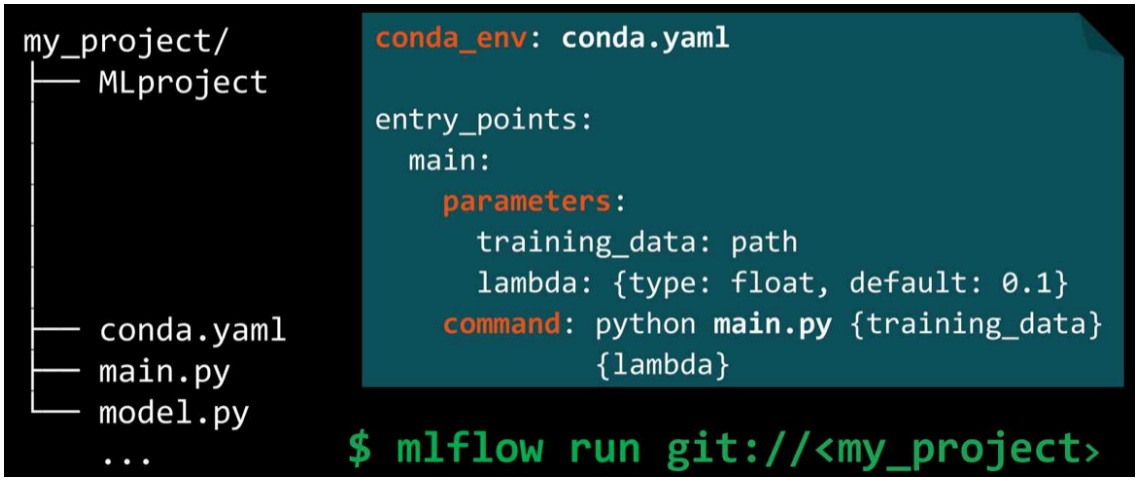
MLflow Projects



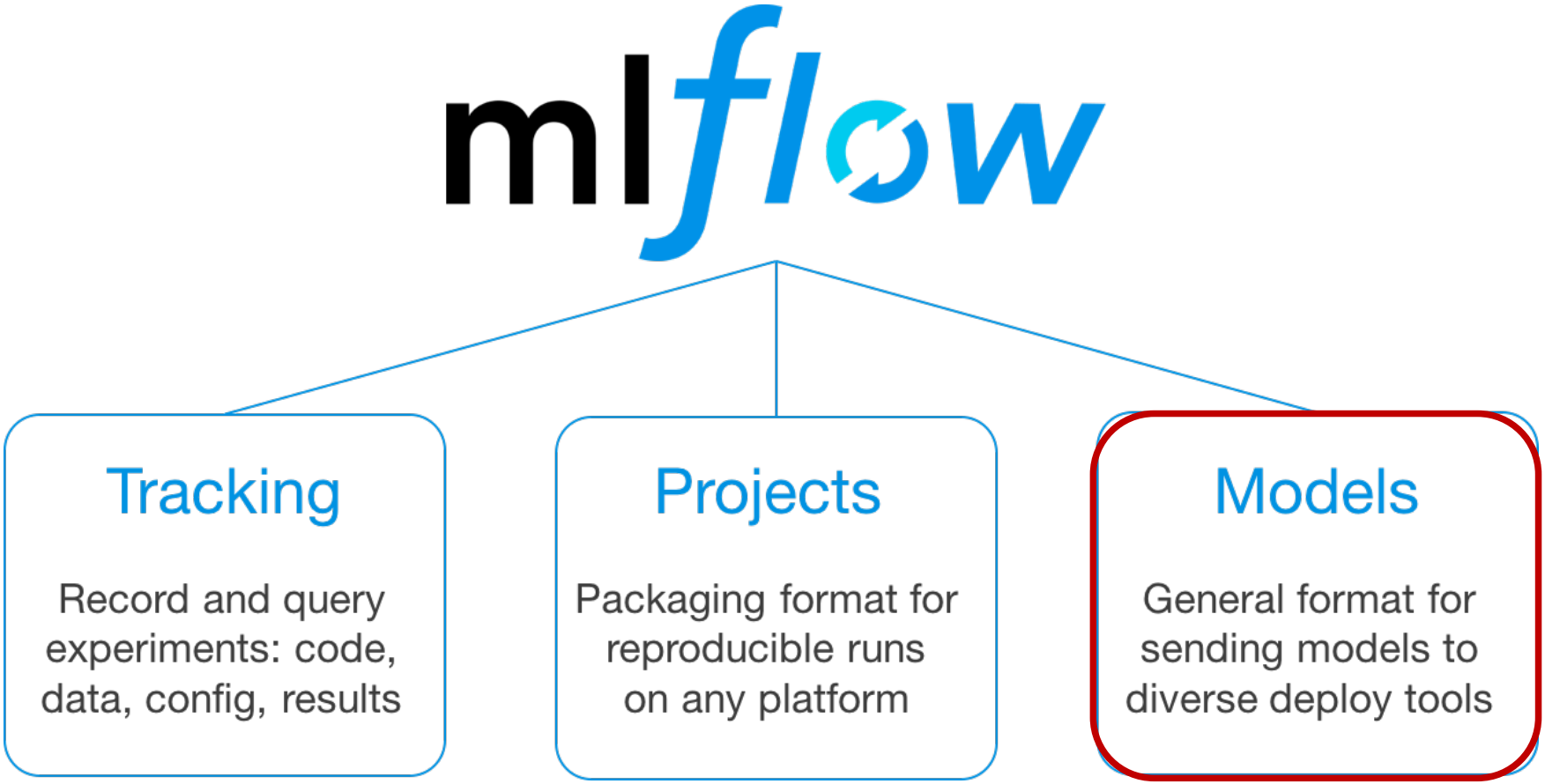
newwave

MLflow Projects

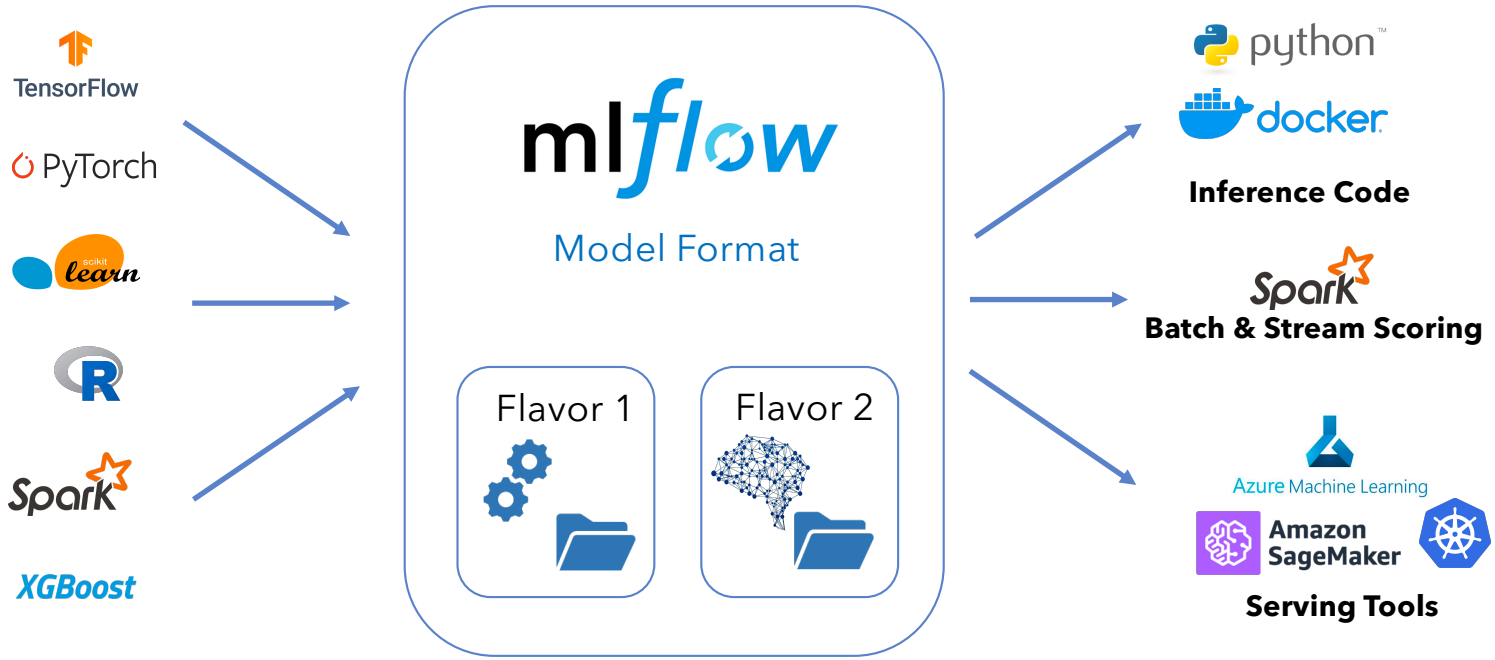
- Packaging format for reproducible ML runs.
- Define dependencies for reproducibility
- Execution API for running project locally or remote



More Challenges

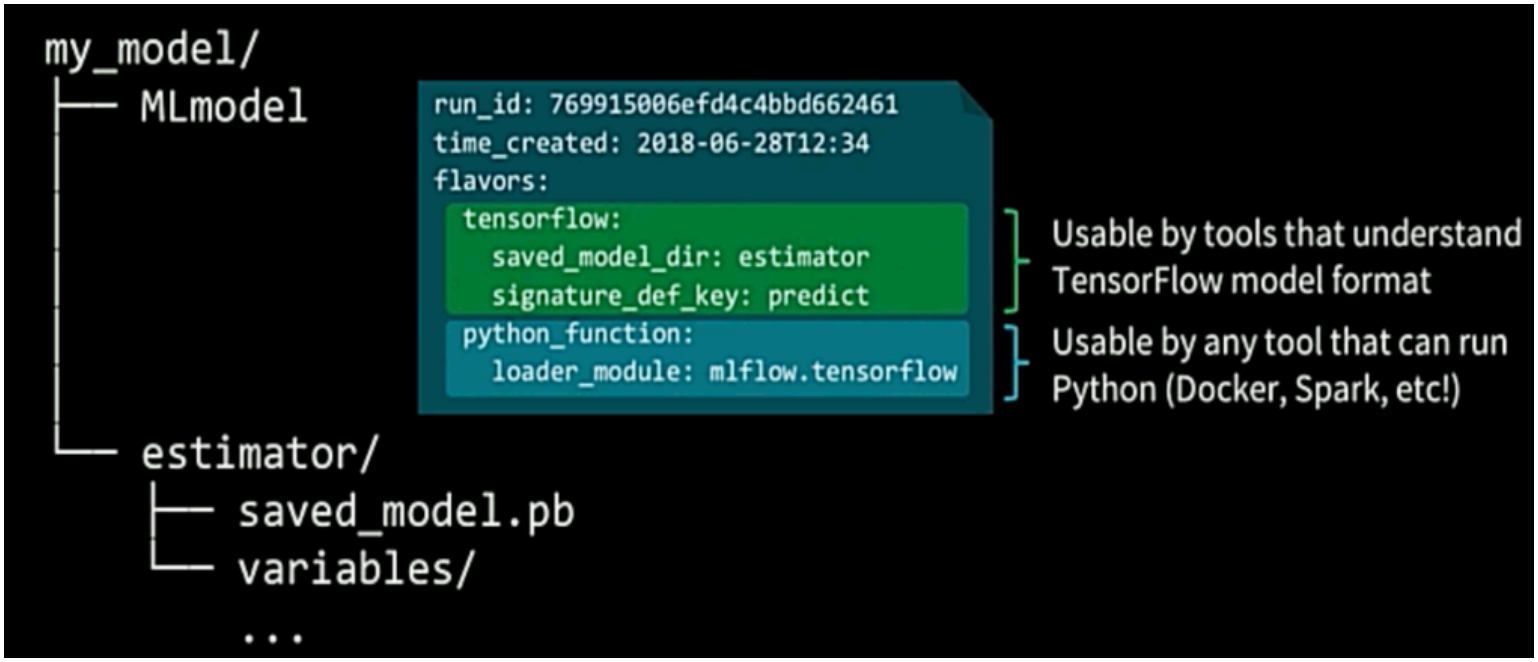


MLflow Models



Standard for ML models

Mlflow Models



newwave

How to get started with MLflow?

- Start with MLflow Tracking to monitor ML model during training and running,
- Explore dashboard to compare the metrics of results with hyperparameters at glance after training.
- Play with free community edition provided by Databricks with 6 GB memory space.

Lab 3

Machine learning experiment management by MLflow

NewWave Data Science Group
Xue Yang, Data Scientist

newwave

newwave

Thank you.