DATA+AI
SUMMIT 2022

End-to-End ETL with Databricks

ORGANIZED BY databricks

Your Instructor

Nice to meet you!

Kelly O'Malley, Sr. Solutions Architect at Databricks

Course topics/agenda

What we'll cover today

- Course Welcome: Introduction to Delta Live Tables (DLT) (40 minutes)
- Setting up your development environment (20 minutes)
- Using the DLT user interface (20 minutes)
- DLT syntax for Python/SQL (100 minutes)
- Results and monitoring (20 minutes)
- Code development and troubleshooting (20 minutes)
- Orchestration with workflows (20 minutes)

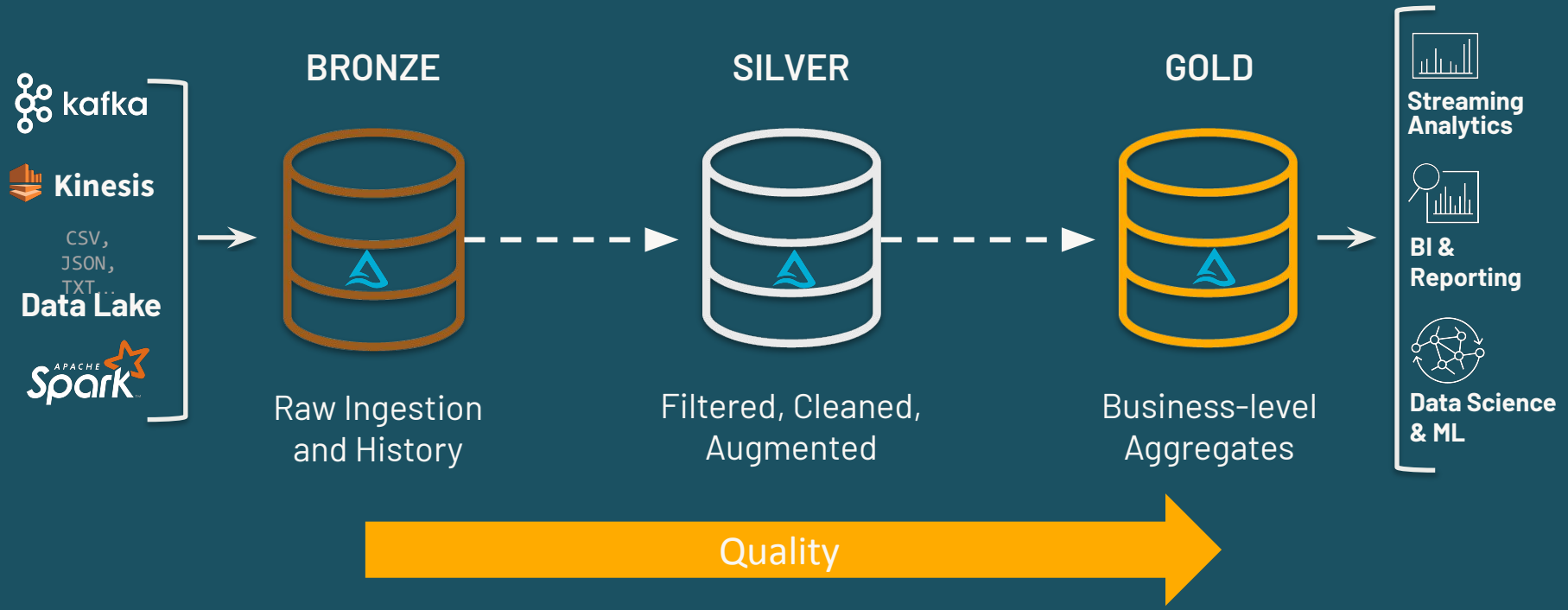
DATA+AI
SUMMIT 2022

Introduction to Delta Live Tables

ORGANIZED BY databricks

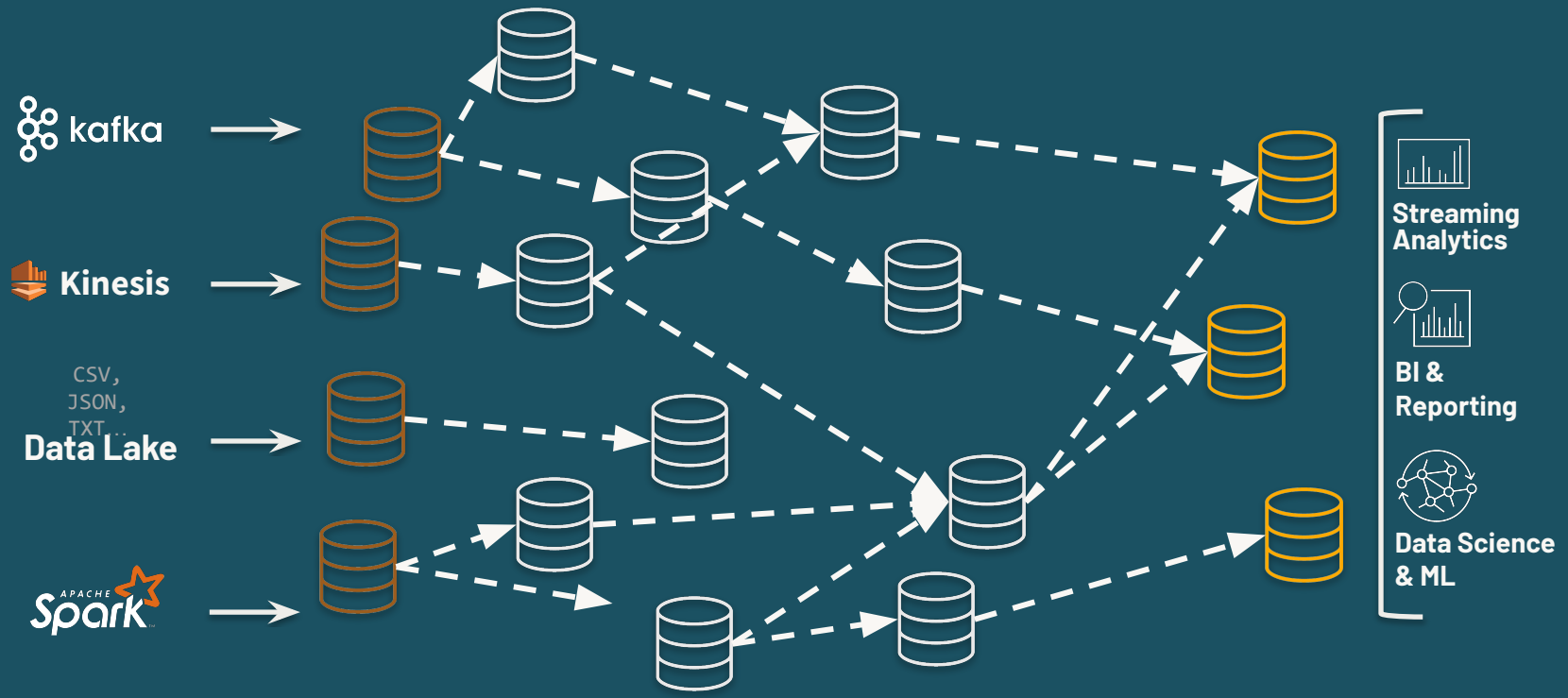
Good data is the foundation of a Lakehouse

All data professionals need clean, fresh and reliable data.



But the reality is not so simple

Maintaining data quality and reliability at scale is often complex and brittle



Life as a data professional…

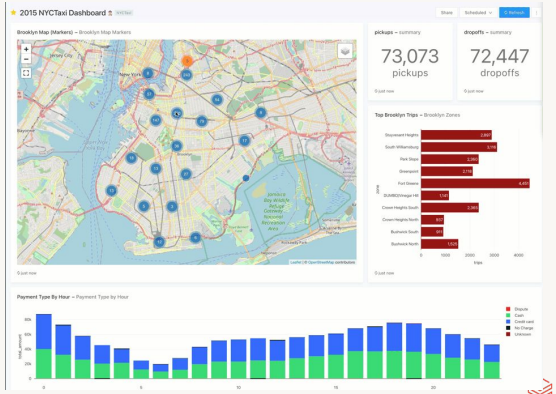
From: The CEO <ali@databricks.com>
Subject: Need an analysis ASAP!
To: Michael Armbrust
<michael@databrick.com>

Hey Michael, I need a quick analysis of our net customer retention and how it has changed over the past few quarters. Raw data can be found at s3://our-data-bucket/raw_data/...

```
1 %fs ls /data/sensors
```

	path
1	dbfs:/data/sensors/_SUCCESS
2	dbfs:/data/sensors/_committed_3908896360792309052
3	dbfs:/data/sensors/_started_3908896360792309052
4	dbfs:/data/sensors/part-00000-tid-3908896360792309052-adec30c0-9ba8-4344-a36c-7ec3
5	dbfs:/data/sensors/part-00001-tid-3908896360792309052-adec30c0-9ba8-4344-a36c-7ec3
6	dbfs:/data/sensors/part-00002-tid-3908896360792309052-adec30c0-9ba8-4344-a36c-7ec3
7	dbfs:/data/sensors/part-00003-tid-3908896360792309052-adec30c0-9ba8-4344-a36c-7ec3

???



Going from query to production

The tedious work required to turn SQL queries into reliable ETL Pipelines

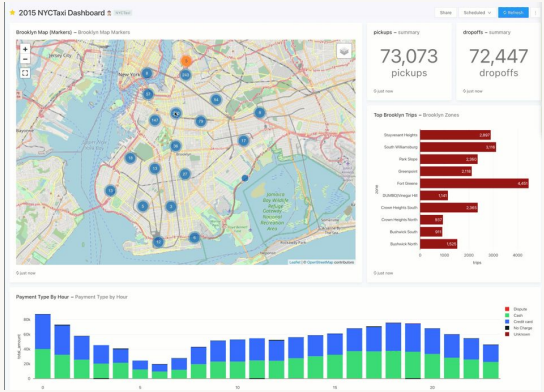
From: **The CEO** <ali@databricks.com>
Subject: Need an analysis ASAP!
To: Michael Armbrust <michael@databrick.com>

Great report! Can you update it ~~everyday~~ every minute?

```
CREATE TABLE raw_data as SELECT * FROM
json. `...`
CREATE TABLE clean_data as SELECT ... FROM raw_data
```

APACHE Spark™

DELTA LAKE



The slog from query to production

The tedious work required to turn SQL queries into reliable ETL Pipelines

Version Control

Deployment Infrastructure

Quality Checks

Governance

Data Discovery

Backfill Handling

Dependency Management

```
CREATE TABLE raw_data as SELECT * FROM json.`...`
CREATE TABLE clean_data as SELECT ... FROM raw_data
```

APACHE Spark™

DELTA LAKE

Daily Partition Computation

Checkpointing & Retries

Operational complexity dominates

Time is spent on ***tooling*** instead of on ***transforming***

Version Control

Deployment Infrastructure

Quality Checks

Governance

Data Discovery

Backfill Handling

Dependency Management

```
CREATE TABLE raw_data as SELECT * FROM json.`...`
CREATE TABLE clean_data as SELECT ... FROM raw_data
```

APACHE Spark™

DELTA LAKE

Daily Partition Computation

Checkpointing & Retries

Where you should focus your time

Getting value from data

Version Control

Deployment Infrastructure

Quality Checks

Governance

Data Discovery

Backfill Handling

Dependency Management

```
CREATE TABLE raw_data as SELECT * FROM
json. `...`
CREATE TABLE clean_data as SELECT ... FROM raw_data
```

Daily Partition Computation

Checkpointing & Retries

APACHE Spark

DELTA LAKE

Introducing Delta Live Tables

From query to production pipeline just by adding LIVE.

```
CREATE LIVE TABLE raw_data as SELECT * FROM json.`…`
CREATE LIVE TABLE clean_data as SELECT … FROM LIVE.raw_data
```

Repos

Unity Catalog*

Databricks Workflows

Delta Live Tables

Full Refresh

Dependency Management

Expectations

APACHE Spark™

DELTA LAKE

Incremental Computation *

Checkpointing & Retries

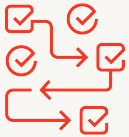
* Coming Soon

What is Delta Live Tables?

Modern software engineering for ETL processing

Delta Live Tables (DLT) is the first ETL framework that uses a simple, declarative approach to building reliable data pipelines. DLT automatically manages your infrastructure at scale so data analysts and engineers can spend less time on tooling and focus on getting value from data.

Accelerate ETL Development



Automatically manage your infrastructure

Have confidence in your data

Simplify batch and streaming

https://databricks.com/product/delta-live-tables

DLT Glossary

What is a Live Table?

Live Tables are materialized views for the lakehouse.

A live table is:

- Defined by a SQL query
- Created and kept up-to-date by a pipeline

```
LIVE
CREATE OR REPLACE TABLE report
AS SELECT sum(profit)
FROM prod.sales
```

Live tables provides tools to:

- Manage dependencies
- Control quality
- Automate operations
- Simplify collaboration
- Save costs
- Reduce latency

What is a Streaming Live Table?

Based on Spark™ Structured Streaming

A streaming live table is "stateful":

- Ensures exactly-once processing of input rows
- Inputs are only read once

```
CREATE STREAMING LIVE TABLE report
AS SELECT sum(profit)
FROM cloud_files(prod.sales)
```

- Streaming Live tables compute results over append-only streams such as Kafka, Kinesis, or autoloader (files on cloud storage)
- Streaming live tables allow you to reduce costs and latency by avoiding reprocessing of old data.

When should I use streaming?

What is Spark™ Structured Streaming?

Basis for Streaming Live Tables. Runs queries on continually arriving data.

Computation Model: Input is an ever-growing append-only table

- Files uploaded to cloud storage
- Message busses like kafka, kinesis, or eventhub
- Delta tables with delta.appendOnly=true
- Transaction logs of other databases

Rather than wait until all data has arrived, structured streaming can produce results on demand.

- Lower latency by processing less data each update
- Lower costs by avoiding redundant work

Using Spark™ Structured Streaming for ingestion

Easily ingest files from cloud storage as they are uploaded

```
CREATE STREAMING LIVE TABLE raw_data
AS SELECT *
FROM cloud_files("/data", "json")
```

This example creates a table with all the json data stored in "/data":

- `cloud_files` keeps track of which files have been read to avoid duplication and wasted work
- Supports both listing and notifications for arbitrary scale
- Configurable schema inference and schema evolution

Using Spark™ Structured Streaming for ingestion

Easily ingest records from message buses

```
import dlt

@dlt.table
def kafka_data():
  return spark.readStream \
    .option("format", "kafka") \
    .option("subscribe", "events") \
    .load()
```

This example creates a table with all the records published to the Kafka topic "event".

- The Kafka source + DLT automatically track which partitions / offsets have already been read.
- Any structured streaming source included in DBR can be used with DLT
- Message buses provide the lowest latency for ingesting data

Using the SQL STREAM() function

Stream data from any Delta table

```
CREATE STREAMING LIVE TABLE mystream
  AS SELECT *
  FROM STREAM(my_table)
```

Pitfall: `my_table` must be an append-only source.

e.g. it may not:

- be the target of APPLY CHANGES INTO
- define an aggregate function
- be a table on which you've executed DML to delete/update a row (see GDPR section)

- `STREAM(my_table)` reads a stream of new records, instead of a snapshot
- `Streaming` tables must be an append-only table
- Any append-only delta table can be read as a stream (i.e. from the live schema, from the catalog, or just from a path).

REFERENCE ARCHITECTURE

Use Delta for infinite retention

Delta provides cheap, elastic and governable storage for transient sources


```
TBLPROPERTIES (
pipelines.reset.allowed=false
)
```

CREATE STREAMING LIVE TABLE AS …

bronze

CREATE STREAMING LIVE TABLE AS …

Use a short retention period to avoid compliance risks and reduce costs

Avoid complex transformations that could have bugs or drop important data

Retain history

Easy to perform GDPR and other compliance tasks

Setting `pipelines.reset.allowed=false` ensures that downstream computation can be full-refreshed without losing data

Streaming does not always mean expensive

Delta live tables lets you choose how often to update the results.

Triggered: Manually

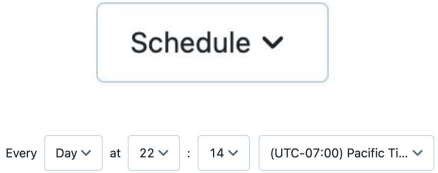
Costs: lowest

Latency: highest

Triggered: On a schedule using Databricks Jobs

Costs: depends on frequency

Latency: 10 minutes to months

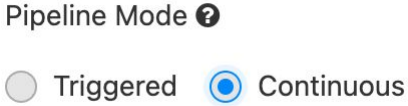


Continually

Costs: highest

Latency: minutes to seconds

(for some workloads)



DATA+AI
SUMMIT 2022

Setting up your development environment

ORGANIZED BY databricks

DATA+AI
SUMMIT 2022

Using the DLT user interface

ORGANIZED BY databricks

DATA+AI
SUMMIT 2022

DLT syntax for Python/SQL

ORGANIZED BY databricks

DATA+AI
SUMMIT 2022

Results and monitoring

ORGANIZED BY databricks

DATA+AI
SUMMIT 2022

Code development and troubleshooting

ORGANIZED BY databricks

DATA+AI
SUMMIT 2022

Orchestration with workflows

ORGANIZED BY databricks

DATA+AI
SUMMIT 2022

Thank you

Your Name
You Title