DATA+AI SUMMIT BY databricks

Seamless Flow: Evolving from Batch to Streaming Data Flows using DLT

Scott Gordon, Lead Data Engineer at 84.51°
Alli Hanlon, Data Engineer at 84.51°

Introductions

Scott Gordon
Lead Data Engineer at 84.51˚

Alli Hanlon
Data Engineer at 84.51˚

Agenda

About 84.51˚

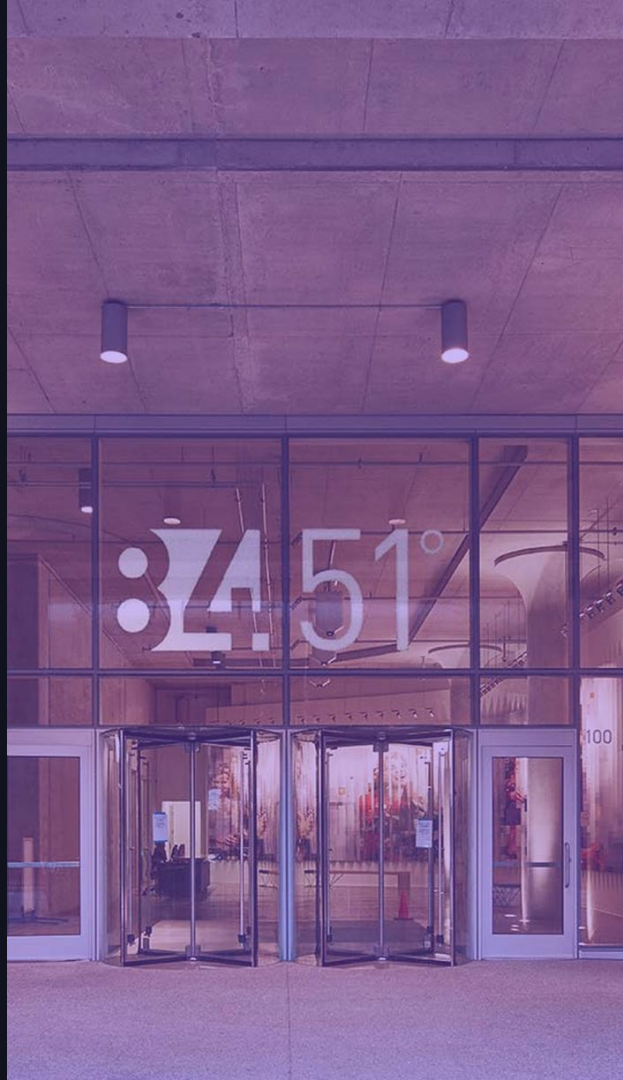
Migrating from On-prem Airflow to Azure Databricks DLT

Seamless evolution from batch to streaming with DLT

About 84.51°

“We are a **retail data science, insights and media company.** We help The Kroger Co., consumer packaged goods companies, agencies, publishers and affiliates **create more personalized and valuable experiences** for shoppers across the path to purchase.

Powered by cutting-edge science, we utilize first-party retail data from over 62 million U.S. households sourced by the Kroger Plus loyalty program to fuel a more customer-centric journey using 84.51° Insights, 84.51° Loyalty Marketing and our retail media advertising solution, Kroger Precision Marketing.”



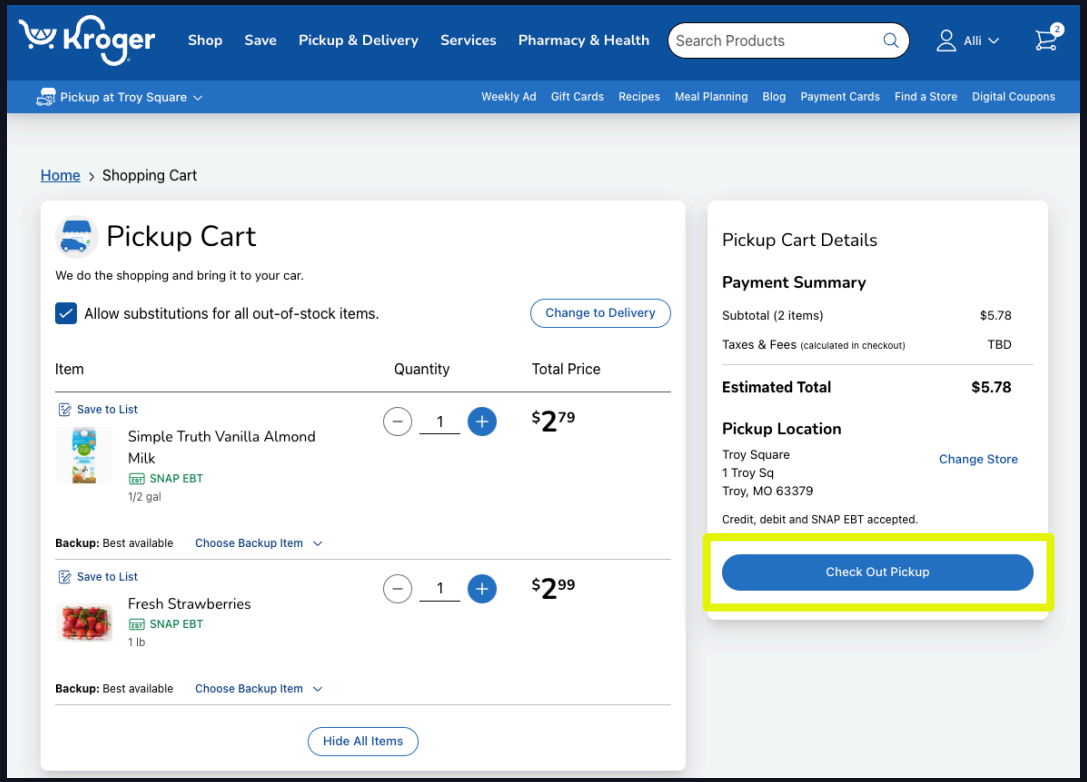
Feature: Once Flow

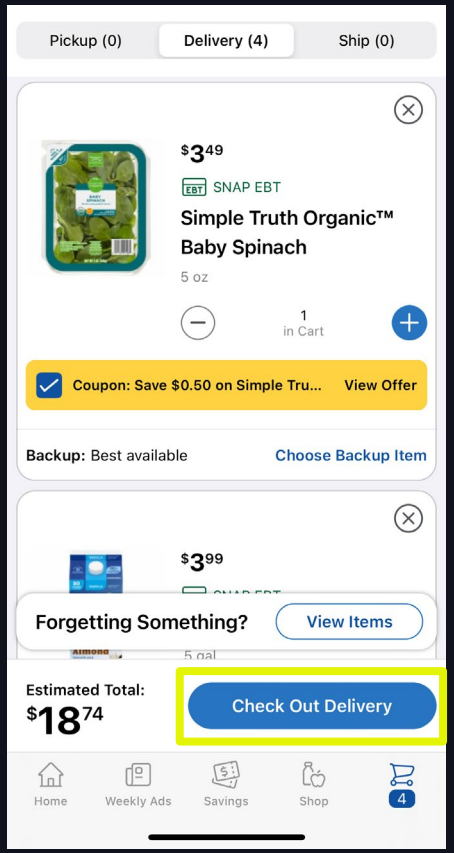
Sets apply_changes() to run only one time on a static source

Use case: Once Flow

Migrate legacy data flow from on-prem Hadoop to Azure Databricks utilizing apply_changes() and Once Flow functionality

Online Orders



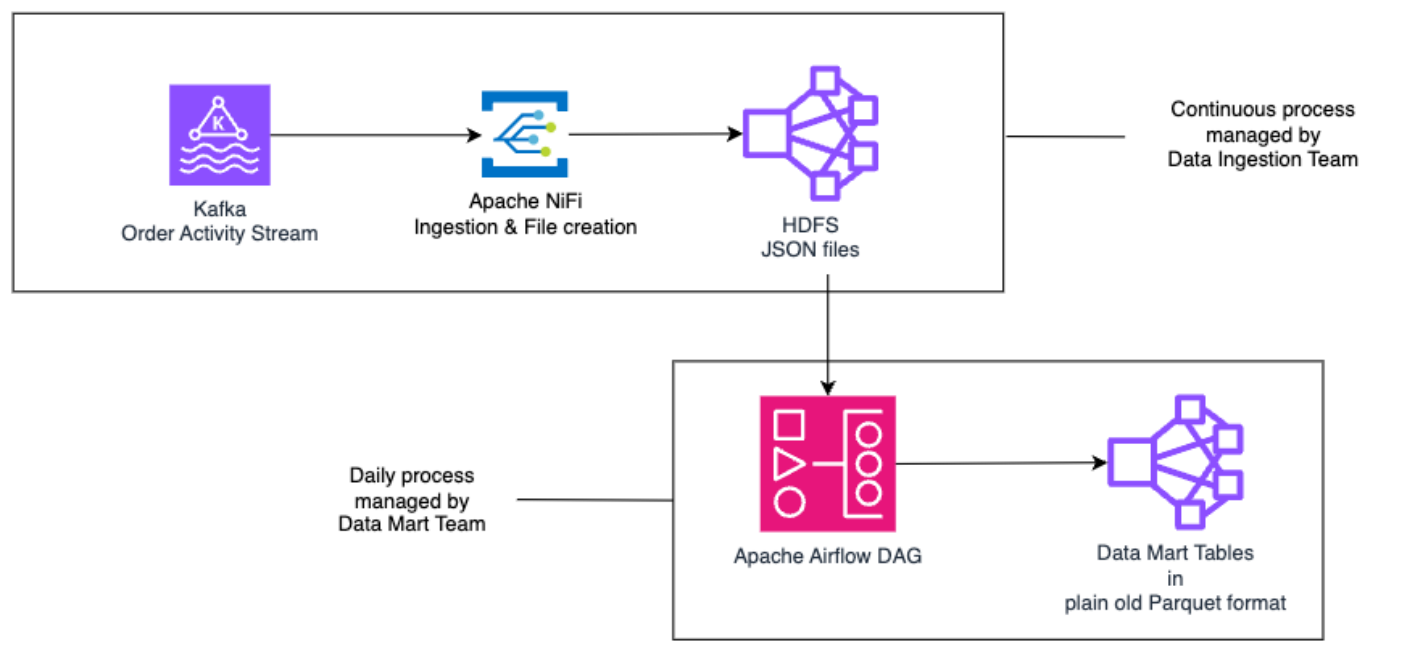


Sample Data

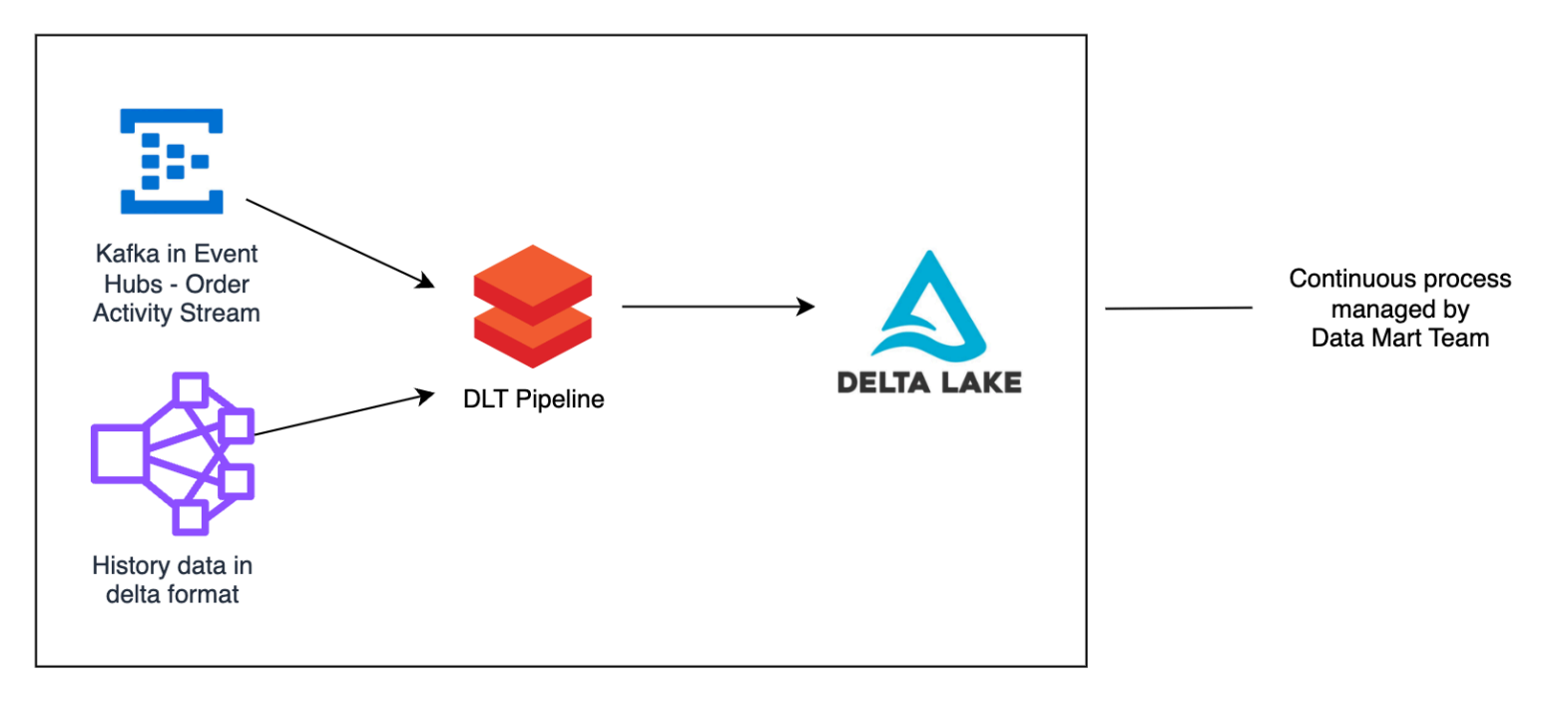
Demo data from online store pickup and delivery orders

id	status	shoppingContext	createdTime	lastUpdateTime	customerId	orderType	lineItems
123	complete	{"chain": "KROGER", "userDevice": ”WEB"}	2023-03-28 T16:44:09Z	2024-05-14T03:12:64	0242	pickup	...
223	complete	{"chain": "KROGER", "userDevice": ”IOS"}	2023-03-28 T16:44:09Z	2024-05-15T05:19:32	0244	delivery	...

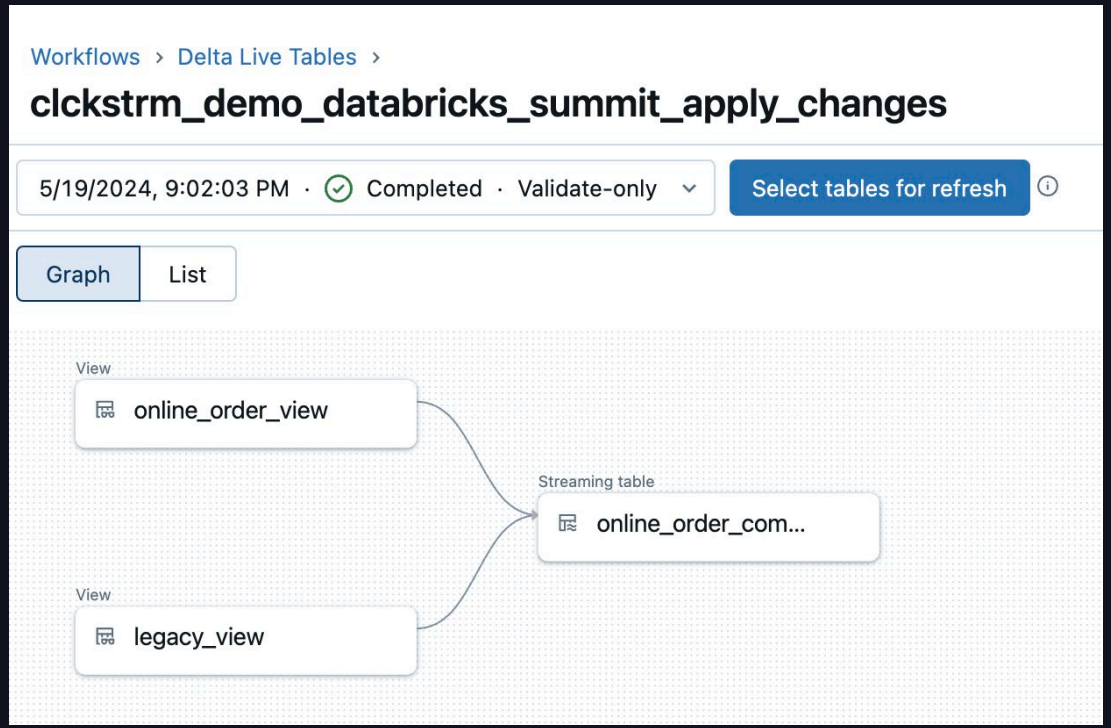
Legacy On-Prem Hadoop Solution



DLT Solution



DLT Solution - Pipeline



DLT Solution - Code

Code snippet of apply_changes

PYTHON

```
dlt.apply_changes (
    flow_name = "online_order_completed_flow",
    target = “online_order_completed",
    source = ”online_order_view",
    keys = ["id"],
    # Sequence by timestamp to get most updated order for a given id
    sequence_by = (col('lastUpdateTimestamp’)),
    # Change data capture type
    stored_as_scd_type = "1"
)
```

DLT Solution - Example

The apply_changes() block would only keep the first record in the final table

id	status	shoppingContext	createdTime	lastUpdateTime	customerId	orderType	lineItems
123	complete	{"chain": "KROGER", "userDevice": ”IOS"}	2023-03-28 T16:44:09Z	2024-05-17 T05:19:32	0244	delivery	...
123	complete	{"chain": "KROGER", "userDevice": ”IOS"}	2023-03-28 T16:44:09Z	2024-05-15 T05:19:32	0244	delivery	...

DLT Solution - Example

The apply_changes() block would only keep one copy of this record.

id	status	shoppingContext	createdTime	lastUpdateTime	customerId	orderType	lineItems
223	complete	{"chain": "KROGER", "userDevice": ”IOS"}	2023-03-28 T16:44:09Z	2024-05-15T05:19:32	0244	delivery	...
223	complete	{"chain": "KROGER", "userDevice": ”IOS"}	2023-03-28 T16:44:09Z	2024-05-15T05:19:32	0244	delivery	...

DLT Solution - Code

Code snippet of apply_changes with once option

PYTHON

```
dlt.apply_changes (
    flow_name = "online_order_legacy_completed_flow",
    # Once option is added and set to “true”
    once = True,
    target = “online_order_completed",
    source = ”legacy_view",
    keys = ["id"],
    # Sequence by timestamp to get most updated order for a given id
    sequence_by = (col('lastUpdateTimestamp’)),
    # Change data capture type
    stored_as_scd_type = "1"
)
```

Benefits

Migrating to Azure Databricks and DLT

- Simplified data flow, managed by one team
- Code is declarative, making it easy to read and maintain

Use of apply_changes() and Once Flow

- Ability to read from both an ongoing flow and a static source, with no overlap in the final table
- Ability to set the static source to only run once

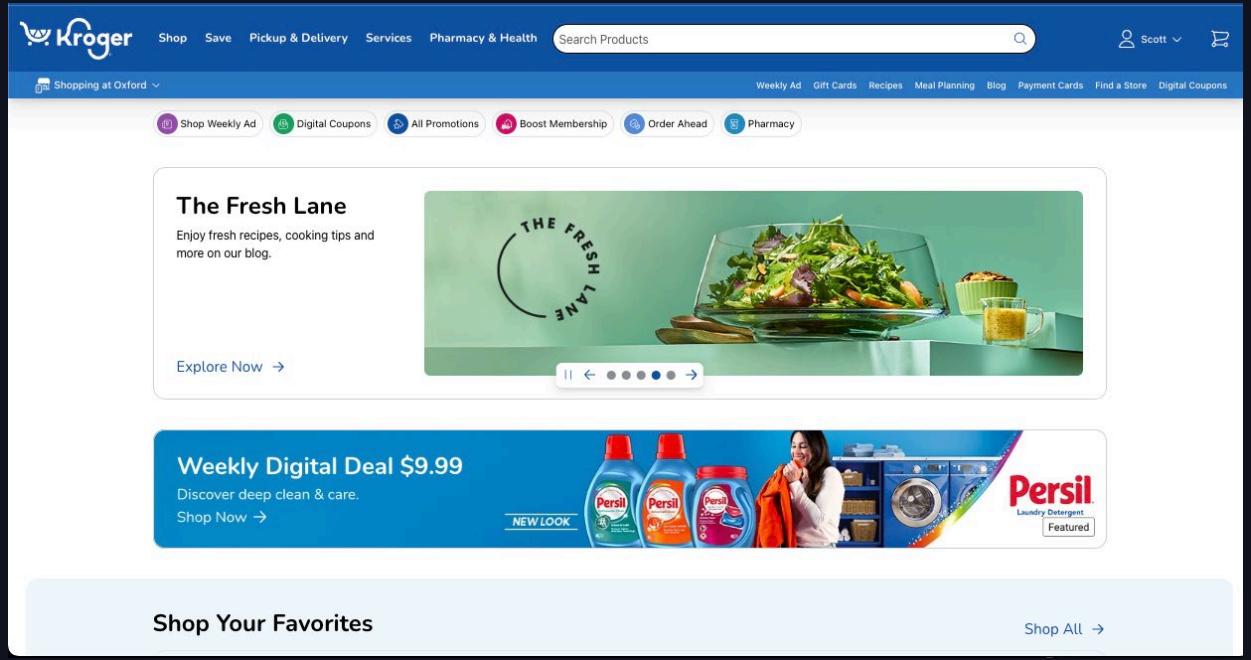
Feature: append_flow()

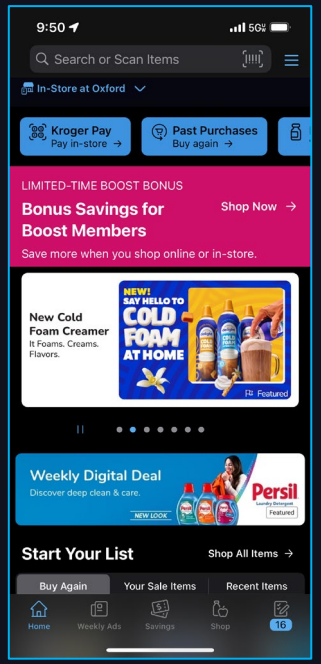
Allows you to write to a target table
from multiple sources

Use case: append_flow()

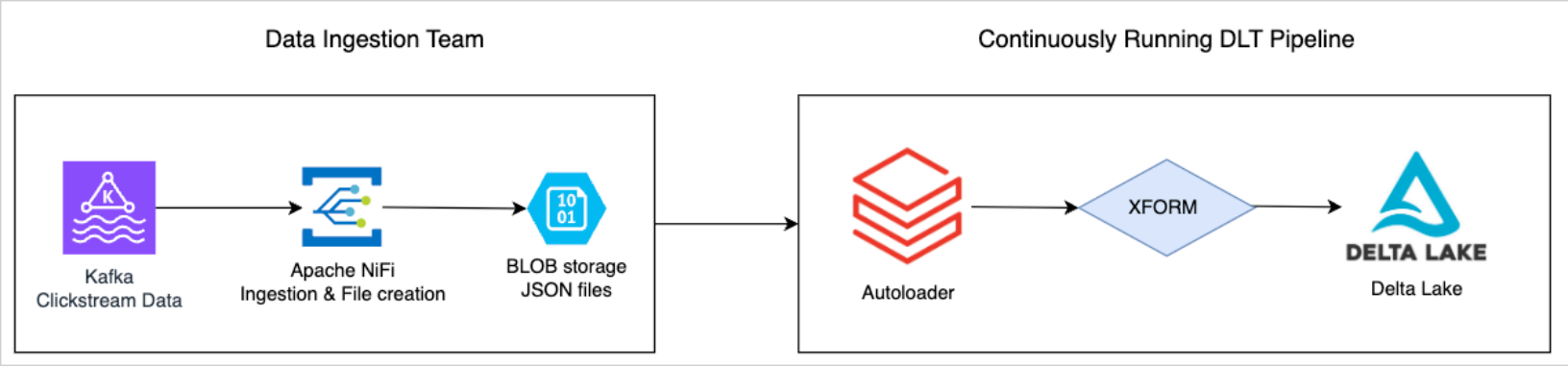
Seamlessly migrate an existing DLT data flow
from batch files to Kafka streaming
utilizing append_flow() functionality

Digital Shopping Behavior

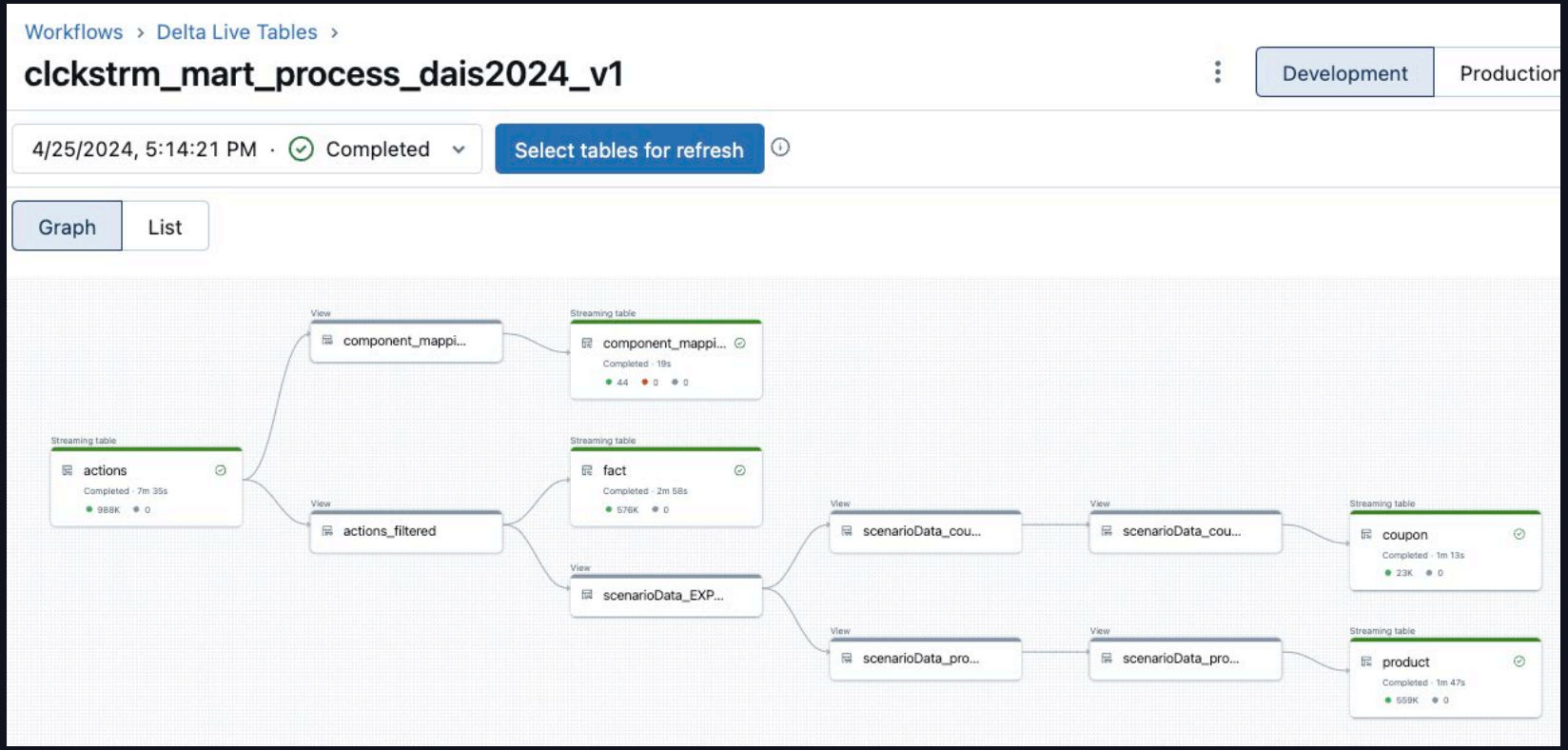




V1 - Data Flow



V1 - DLT Pipeline



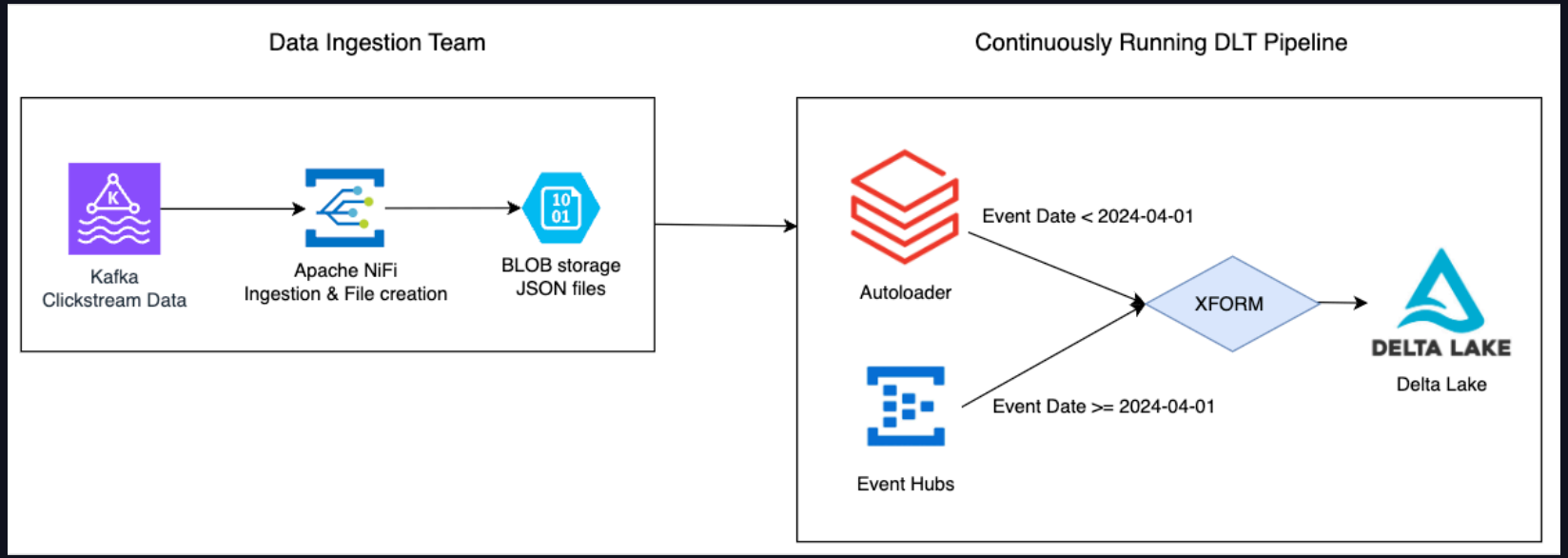
V1 - DLT Pipeline Code

Code snippet: Autoloader flow

PYTHON

```python
@dlt.table(name='actions')
def actions():
    return (
        spark.readStream
        .format('cloudFiles')
        .option('cloudFiles.format', 'json’)
        .schema(static_schema)
        .load('abfss://container@storage-account/path_to_json_files/')
    )
```

V2 - Data Flow



V2 - DLT Pipeline Code

Code snippet: Autoloader flow with Append Flow added

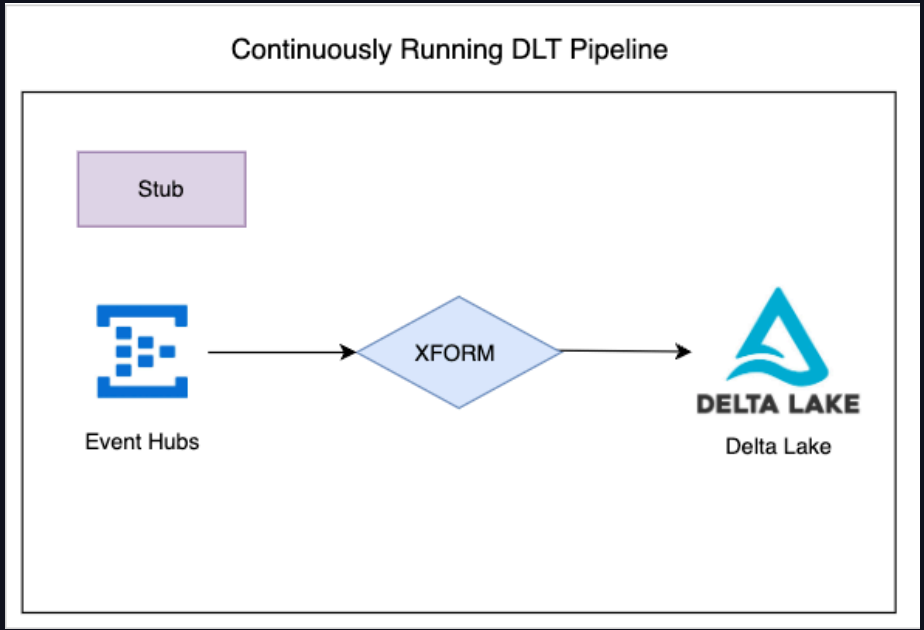
PYTHON

```
@dlt.table(name="actions")
def actions():
   return (
        spark.readStream
        .format('cloudFiles')
        .where(col("event_date") < "2024-04-01")
    )
. . . . . .
@dlt.append_flow(name = "new_append_flow", target = "actions")
def new_append_flow():
  return (
    spark.readStream.format('kafka').options(config).load()
   .where(col("event_date") >= "2024-04-01")
  )
```

V1-V2-V3 Seamless Deployment

<= Week 10	Week 11	Week 12	Week 13	Week 14
V1	deploy V2	2024-04-01	V2	deploy V3

V3 - Data Flow



V3 - DLT Pipeline Code

Code snippet: Autoloader removed leaving Append Flow only

PYTHON

```
dlt.create_streaming_table("actions")

. . . . . .

@dlt.append_flow(name = ”my_append_flow", target = "actions")
def clickstream_raw_kafka():
  return (
    spark.readStream.format('kafka').options(**config).load()
   .where(col("event_date") >= ”2024-04-01")
  )
```

Benefits

- End-to-end ownership of data flow
- Reduced latency from HOURS to MINUTES
- Migrated from batch files to streaming
  - No downtime
  - Minimal code changes

How did we choose?

Apply Changes

- Similar to “MERGE INTO”
- Can insert, update, or delete
- Change data capture (SCD 1 and 2)
- Schemas must match
- Once flow option built-in

Append Flow

- Similar to “UNION ALL”
- Data flows / appends in its entirety
- No change data capture
- Schemas can be merged
- No once flow option built-in

Our Other Talks!

Check out more from 84.51°

SciCLOps: Databricks Quick Start for Machine Learning, Powered by DABs	Unlocking Data Value: 84.51°'s Journey with Databricks Unity Catalog	Databricks Asset Bundles: A Unifying Tool for Deployment on Databricks

DATA+AI SUMMIT BY databricks

QUESTIONS?