Modernizing Banking ETL and Analytics with Delta Live Tables

Jian Zhou, Navy Federal Credit Union
Ricardo Portilla, Databricks

Introduction

Ricardo Portilla

- Industry Principal Architect, 6 years Databricks
- Previously led Market Surveillance at FINRA
- Open source creator and industry SME

2005

2007

2010

Databricks Data Intelligence Platform

Data Science & AI	ETL & Real-time Analytics	Orchestration	Data Warehousing
Mosaic AI	**Delta Live Tables**	**Workflows**	**Databricks SQL**

An AI powered data intelligence engine to understand the semantics of your data

DatabricksIQ

Unified security, governance, and cataloging

Unity Catalog

Unified data storage for reliability and sharing

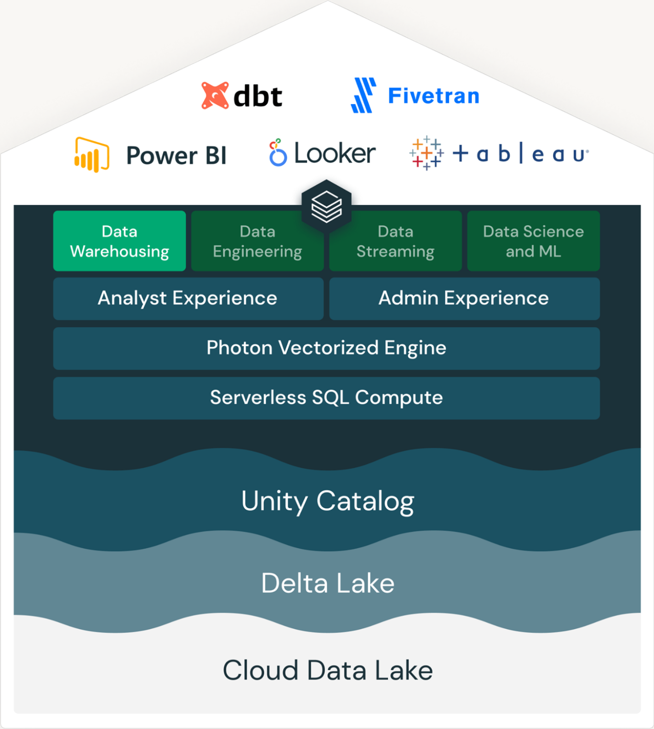
Delta Lake UniForm

Open Data Lake

All Raw Data
(Logs, Texts, Audio, Video, Images)

The best data warehouse is a lakehouse

Powered by Databricks SQL



Seamless Integration with the Ecosystem

Ease of Use

Real-world Performance

Centralized Governance

Open and Reliable Data Lake as the Foundation

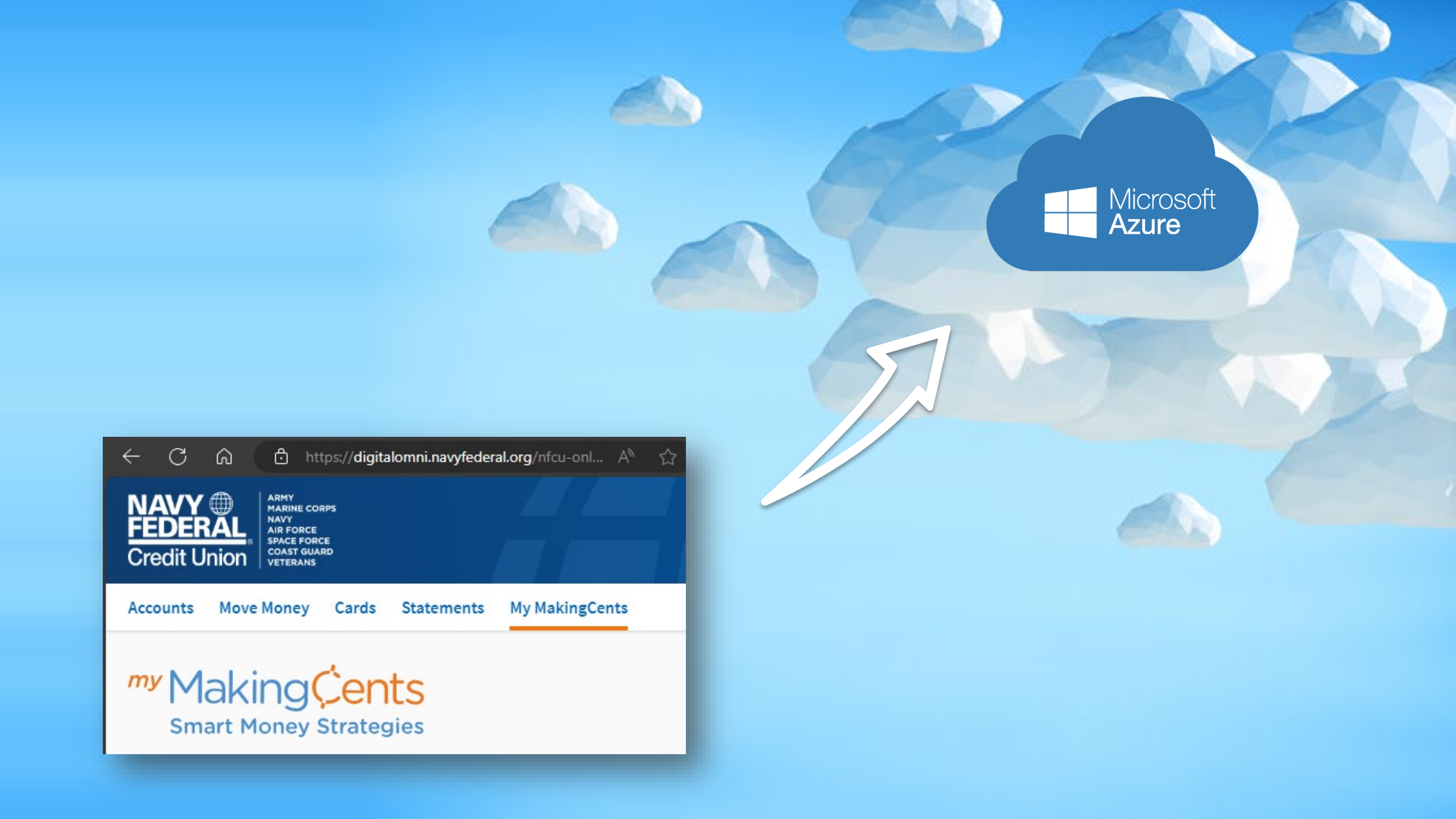
Introduction

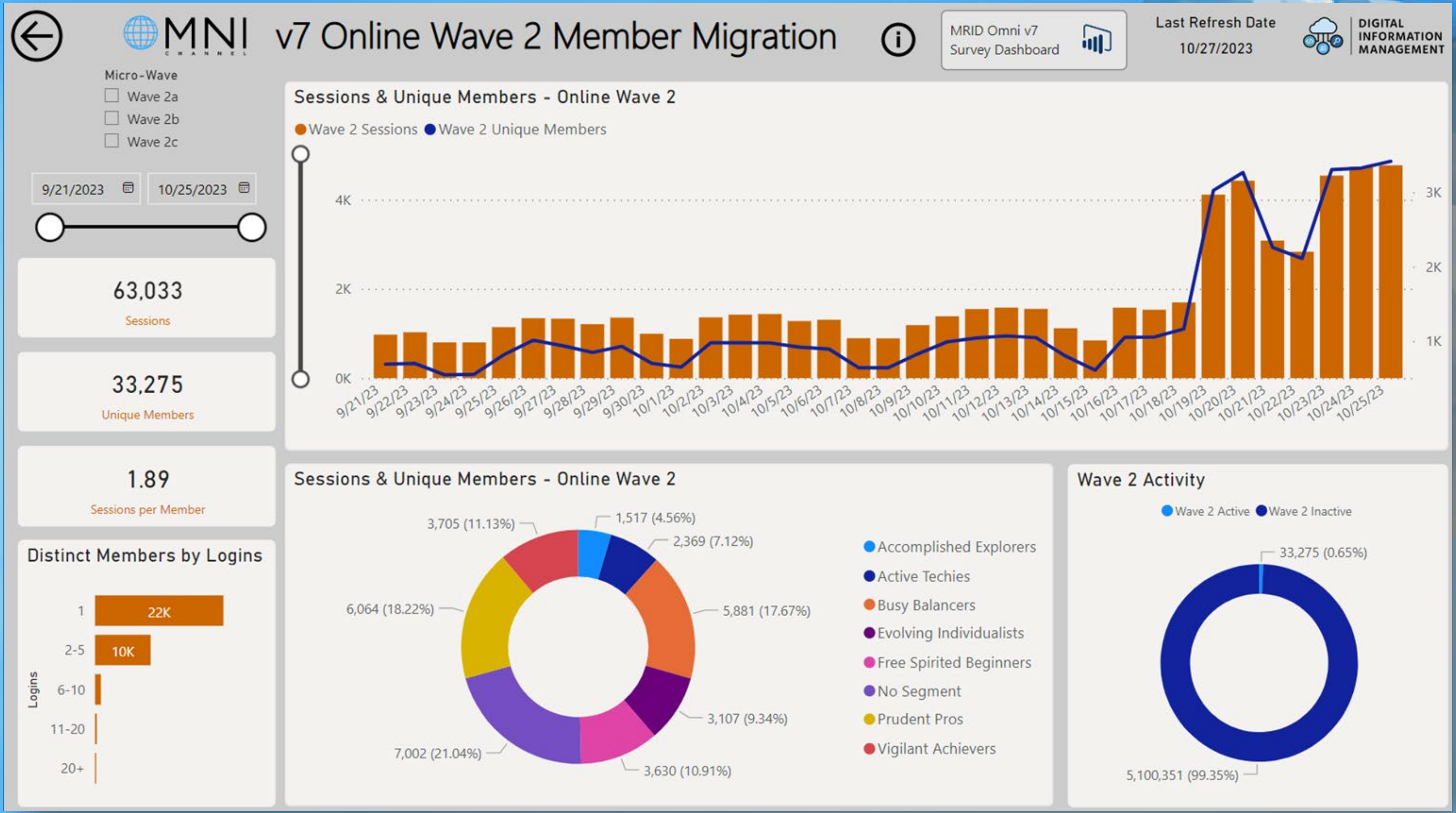
Jian (Miracle) Zhou

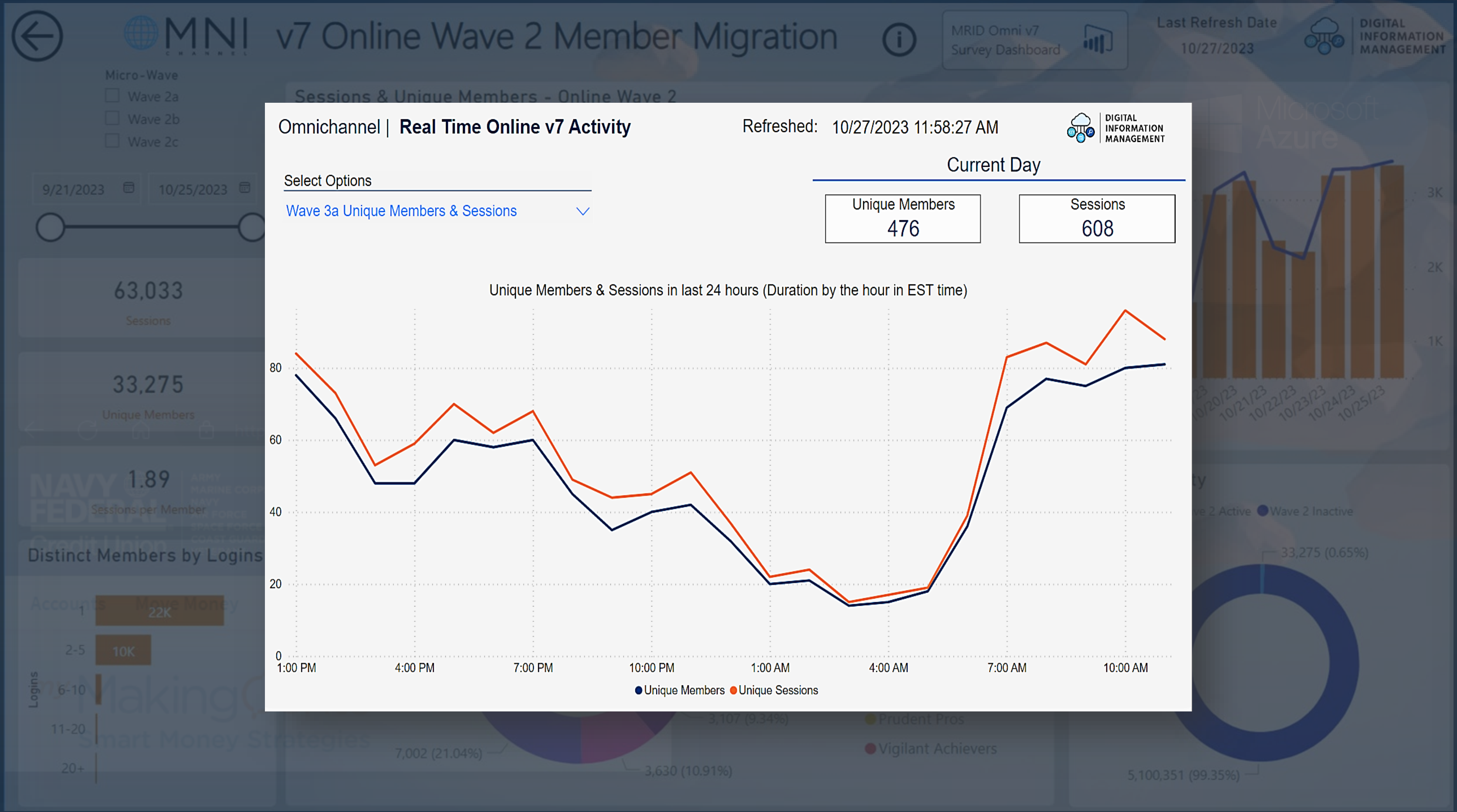
- Sr. Manager, Digital Data Engineering
- Navy Federal Credit Union
- https://www.linkedin.com/in/miraclezhou/

NAVY FEDERAL Credit Union

June 2023


Microsoft
Azure
https://digitalomni.navyfederal.org/nfcu-onl...
NAVY FEDERAL Credit Union
ARMY
MARINE CORPS
NAVY
AIR FORCE
SPACE FORCE
COAST GUARD
VETERANS
Accounts
Move Money
Cards
Statements
My MakingCents
my MakingCents
Smart Money Strategies


v7 Online Wave 2 Member Migration
MRID Omni v7 Survey Dashboard
Last Refresh Date
10/27/2023
DIGITAL INFORMATION MANAGEMENT
Micro-Wave
Wave 2a
Wave 2b
Wave 2c
9/21/2023
10/25/2023
63,033
Sessions
33,275
Unique Members
1.89
Sessions per Member
Distinct Members by Logins
Logins
1
22K
2-5
10K
6-10
11-20
20+
Sessions & Unique Members - Online Wave 2
Wave 2 Sessions
Wave 2 Unique Members
0K
2K
4K
1K
2K
3K
9/21/23 9/22/23 9/23/23 9/24/23 9/25/23 9/26/23 9/27/23 9/28/23 9/29/23 9/30/23 10/1/23 10/2/23 10/3/23 10/4/23 10/5/23 10/6/23 10/7/23 10/8/23 10/9/23 10/10/23 10/11/23 10/12/23 10/13/23 10/14/23 10/15/23 10/16/23 10/17/23 10/18/23 10/19/23 10/20/23 10/21/23 10/22/23 10/23/23 10/24/23 10/25/23
Sessions & Unique Members - Online Wave 2
1,517 (4.56%)
2,369 (7.12%)
5,881 (17.67%)
3,107 (9.34%)
3,630 (10.91%)
7,002 (21.04%)
6,064 (18.22%)
3,705 (11.13%)
Accomplished Explorers
Active Techies
Busy Balancers
Evolving Individualists
Free Spirited Beginners
No Segment
Prudent Pros
Vigilant Achievers
Wave 2 Activity
Wave 2 Active
Wave 2 Inactive
33,275 (0.65%)
5,100,351 (99.35%)


v7 Online Wave 2 Member Migration
Last Refresh Date
10/27/2023
DIGITAL INFORMATION MANAGEMENT
Omnichannel | Real Time Online v7 Activity
Refreshed: 10/27/2023 11:58:27 AM
DIGITAL INFORMATION MANAGEMENT
Select Options
Wave 3a Unique Members & Sessions
Current Day
Unique Members
476
Sessions
608
Unique Members & Sessions in last 24 hours (Duration by the hour in EST time)
80
60
40
20
0
1:00 PM
4:00 PM
7:00 PM
10:00 PM
1:00 AM
4:00 AM
7:00 AM
10:00 AM
Unique Members
Unique Sessions

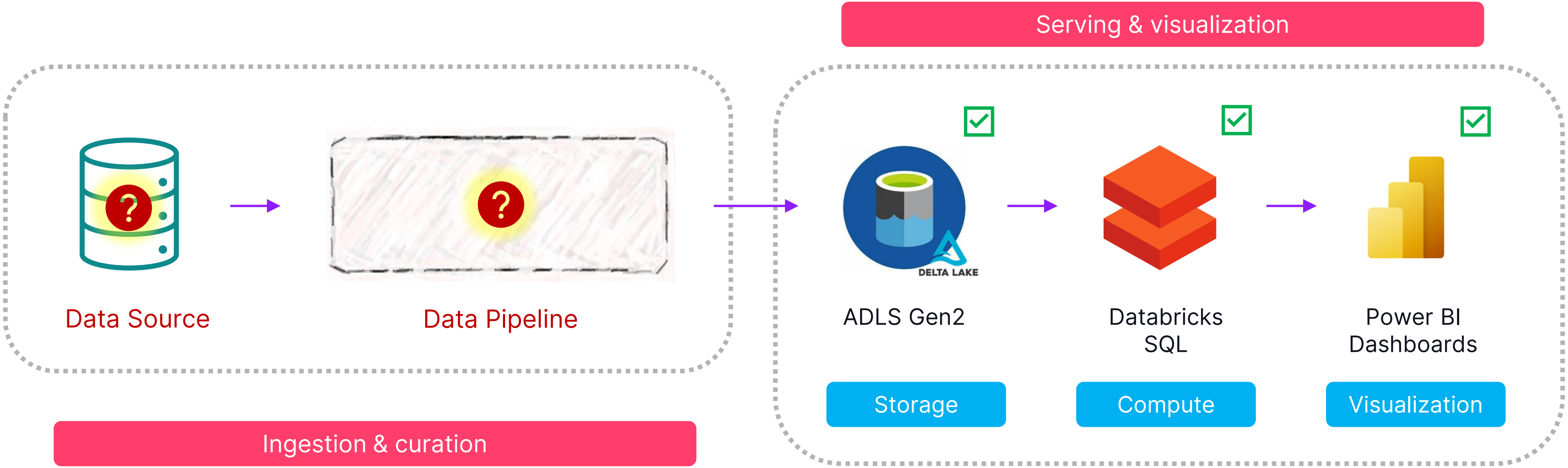
MISSION:

NEAR REAL-TIME INSIGHTS IN 6 WEEKS

MNI CHANNEL

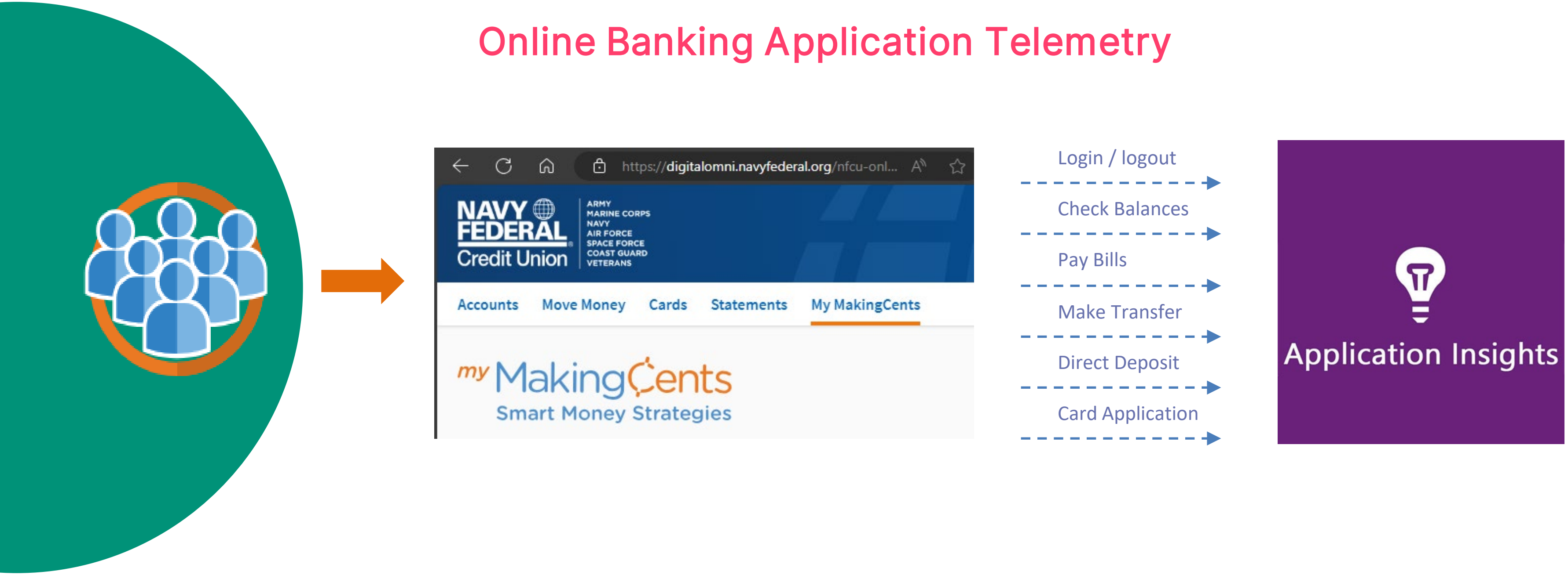
NAVY FEDERAL Credit Union

Our dataflow pattern



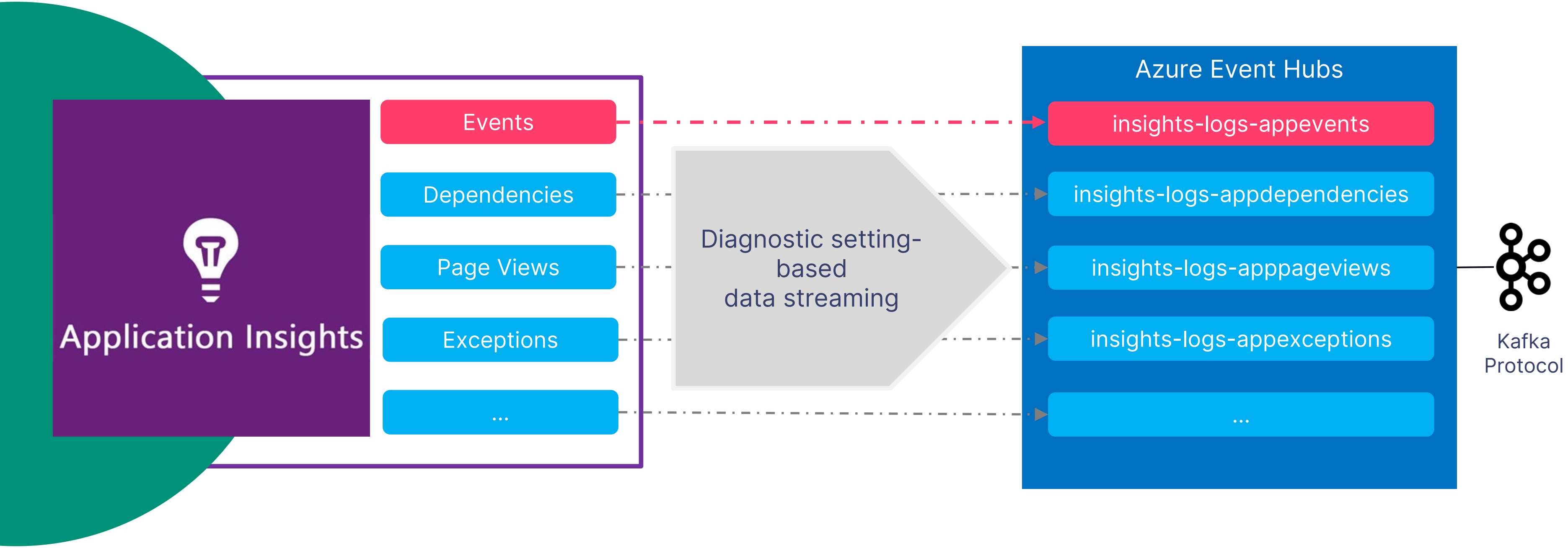
The data source

Online Banking Application Telemetry

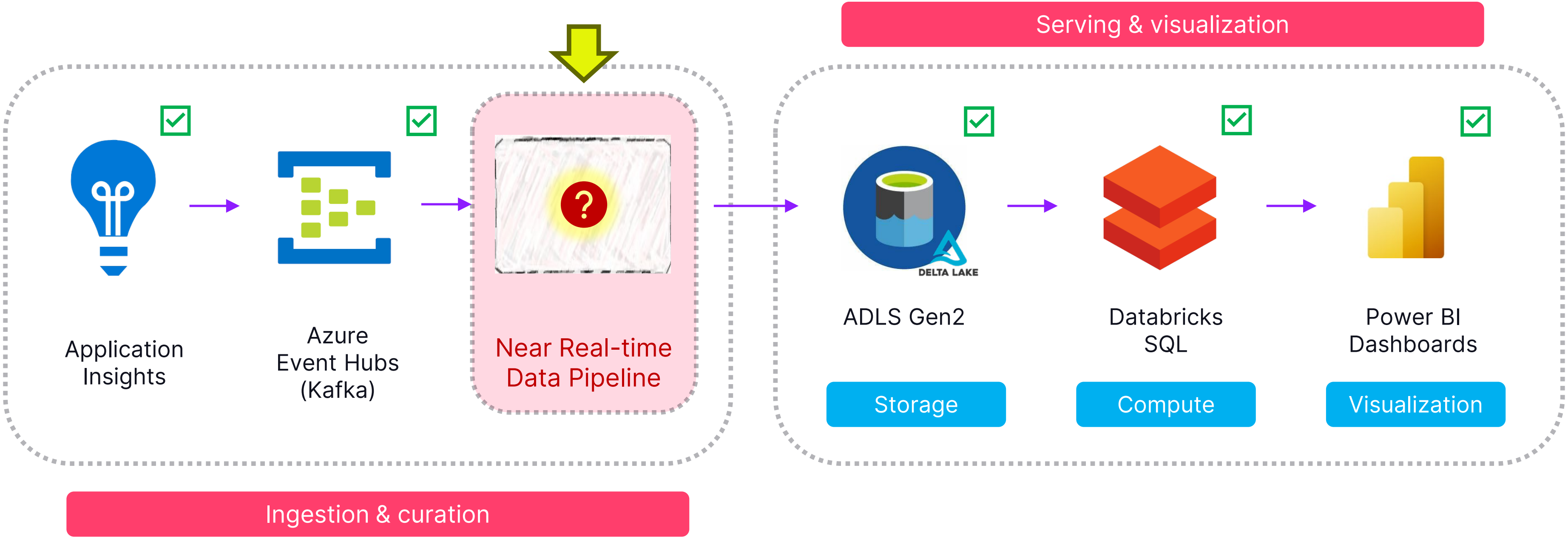


Since the initial rollout, Online v7 has logged over 8.5 billion events in Application Insights (as of April 2024)

Turn on the telemetry firehose



NRT Pipeline



The input

- Semi-structured JSON documents
- App events embedded in an array
  - 140+ event types
  - 100+ event attributes
- There are duplicates!


```
{
  "records": [
    {
      "time": "2024-04-08T14:00:03.3500000Z",
      "OperationId": "1f85fe13fa9cc460463b008bf94ca156",
      "SessionId": "40cf0031-a1db-426e-850b-1940a290b6a4",
      "Properties": {
        "activity": "login",
        "channel": "web",
        "ipAddress": "20.84.33.69"
      }
    },
    {
      "time": "2024-04-08T14:02:05.3110000Z",
      "OperationId": "88a4321463a92645bb5224238d5b7967",
      "SessionId": "40cf0031-a1db-426e-850b-1940a290b6a4",
      "Properties": {
        "activity": "ingestion",
        "channel": "web",
        "ipAddress": "20.84.33.69",
        "paymentOrderStatus":"SUCCESS"
      }
    }
  ]
}
```


App event 1

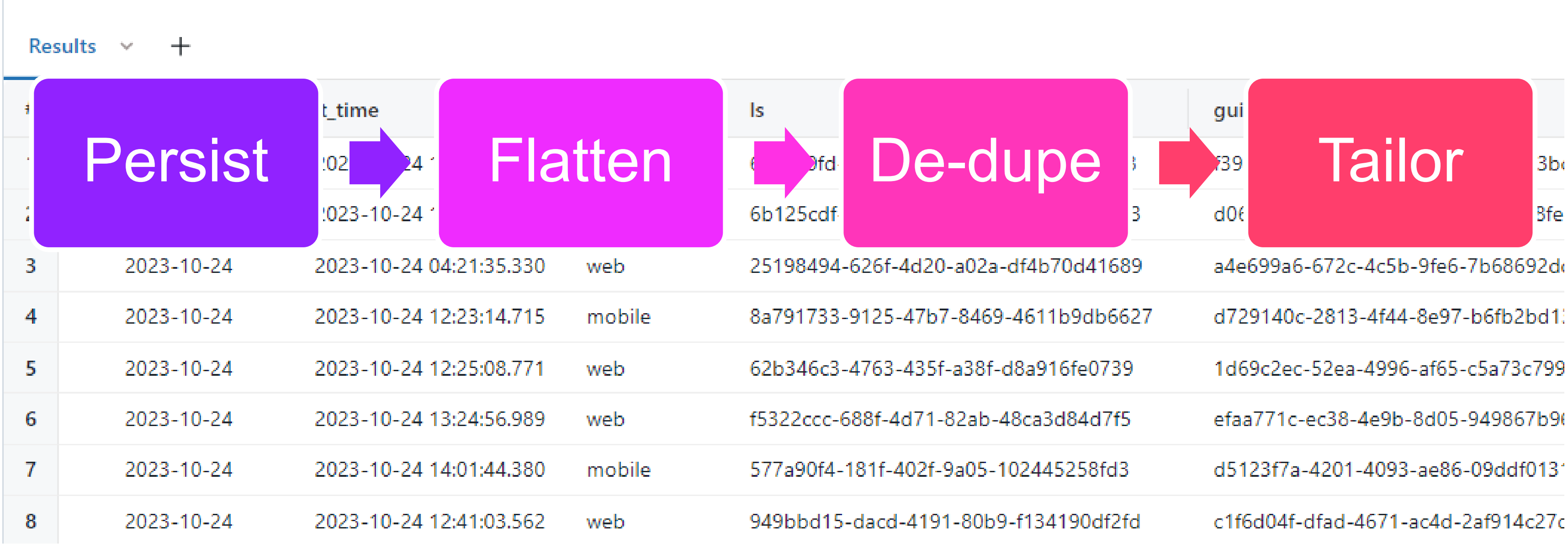
App event 2

A sample message from Event Hubs

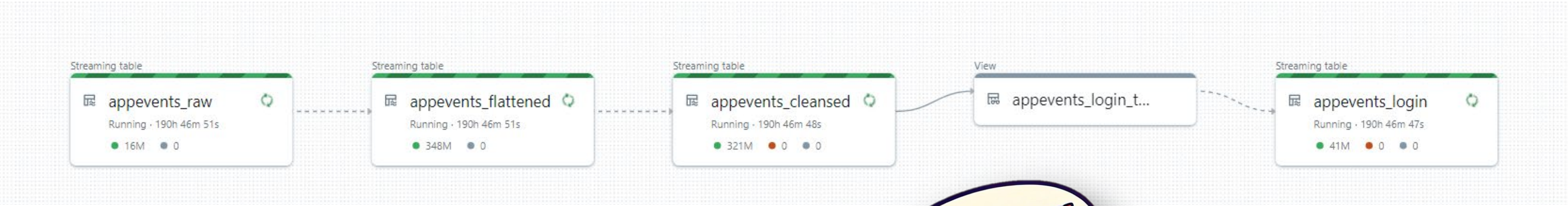
The output

```
select *
from appevents_login
where channel is not null
order by event_date desc
limit 10;
```

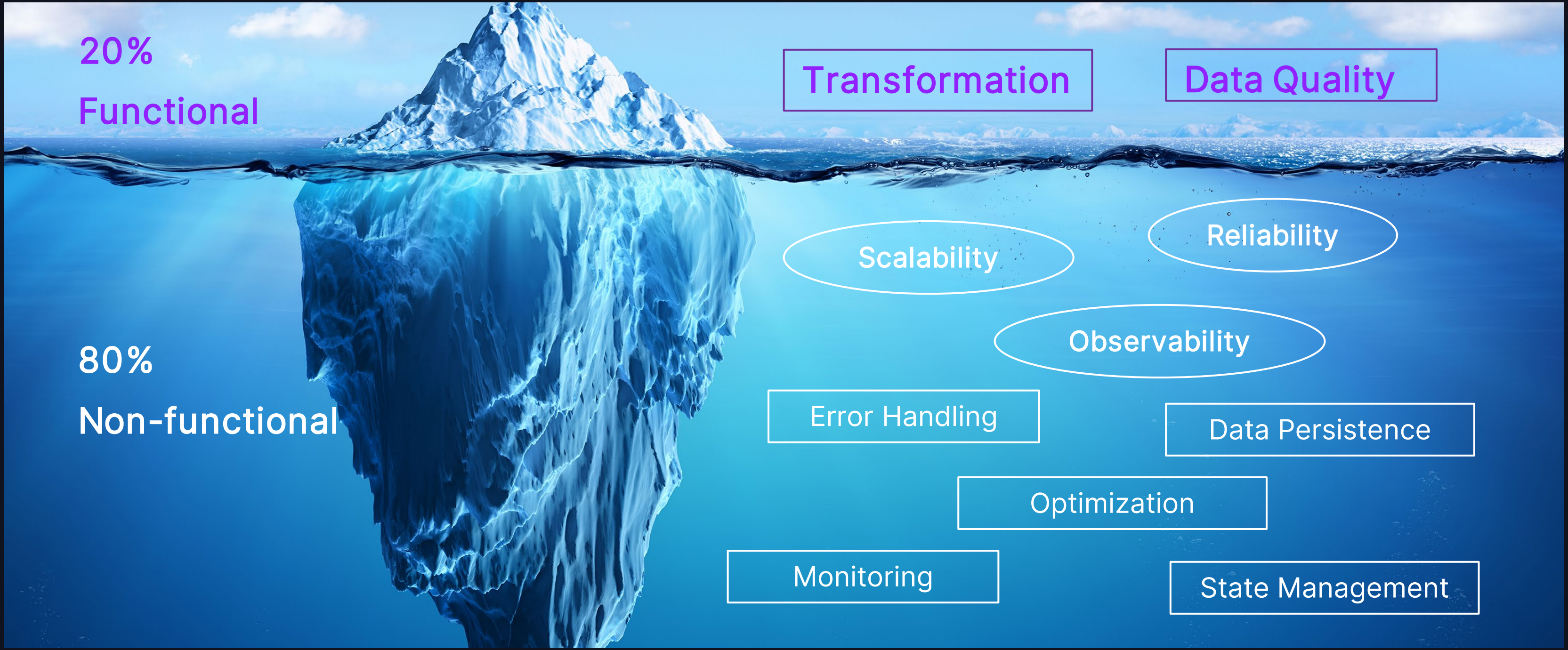
dev data sample



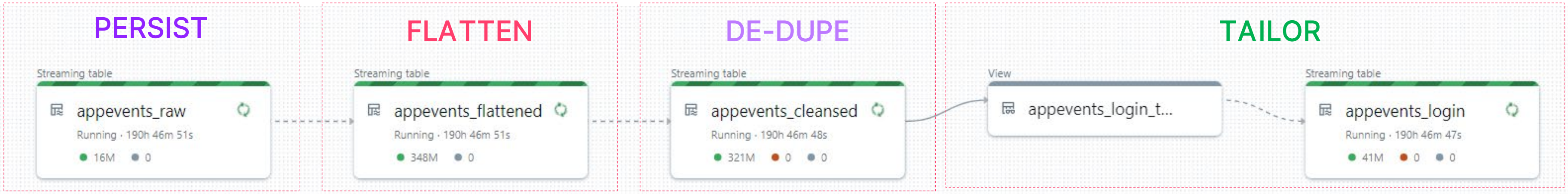
The pipeline



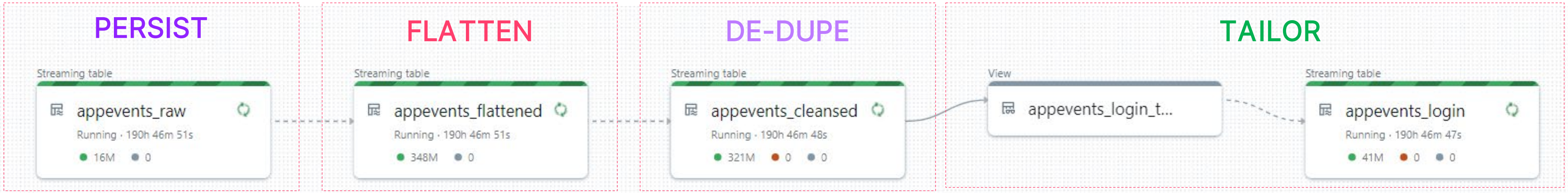
The iceberg of data engineering



Simplicity

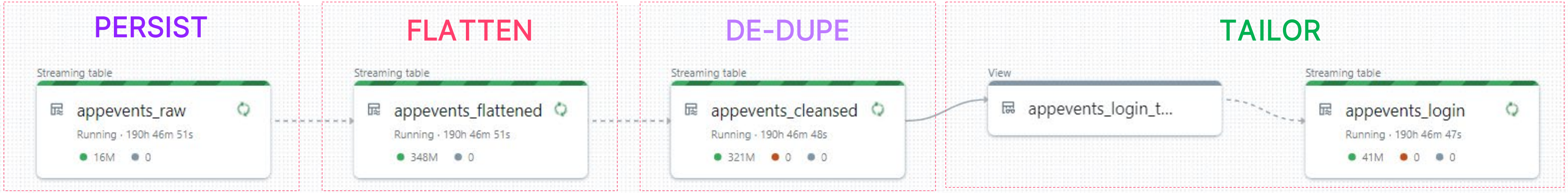

PERSIST
FLATTEN
DE-DUPE
TAILOR
Streaming table
appevents_raw
Running · 190h 46m 51s
16M 0
Streaming table
appevents_flattened
Running · 190h 46m 51s
348M 0
Streaming table
appevents_cleansed
Running · 190h 46m 48s
321M 0 0
View
appevents_login_t...
Streaming table
appevents_login
Running · 190h 46m 47s
41M 0 0

Simplicity




```python
@dlt.table()
@dlt.expect_or_drop(
  "valid records",
  "...")
def appevents_raw():
    options = get_kafka_config_options(
      spark,
      dbutils.secrets)
    return (spark
        .readStream
        .format("kafka")
        .options(**options)
        .load()
        .transform(parse_raw_appevents)
  )
```

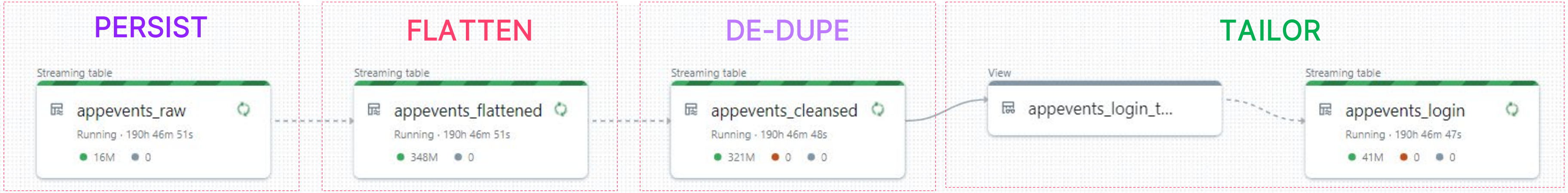
Simplicity




```python
@dlt.table()
@dlt.expect_or_drop(
  "valid records",
  "...")
def appevents_raw():
    options = get_kafka_config_options(
      spark,
      dbutils.secrets)
    return (spark
            .readStream
            .format("kafka")
            .options(**options)
            .load()
            .transform(parse_raw_appevents)
  )
```

```python
@dlt.table()
@dlt.expect_or_drop(
  "valid records",
  "...")
def appevents_flattened():
  return flatten_appevents(
    dlt.readStream("appevents_raw"))
```

Simplicity

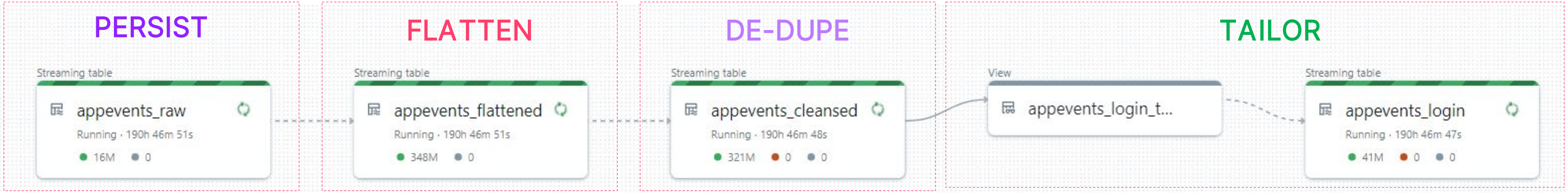



```python
@dlt.table()
@dlt.expect_or_drop(
  "valid records",
  "...")
def appevents_raw():
    options = get_kafka_config_options(
      spark,
      dbutils.secrets)
    return (spark
            .readStream
            .format("kafka")
            .options(**options)
            .load()
            .transform(parse_raw_appevents)
    )
```

```python
@dlt.table()
@dlt.expect_or_drop(
  "valid records",
  "...")
def appevents_flattened():
  return flatten_appevents(
    dlt.readStream("appevents_raw"))
```

```python
dlt.create_streaming_table(
    "appevents_cleansed"
  )

dlt.apply_changes(
  target = "appevents_cleansed",
  source = "appevents_flattened",
  keys = ["event_date", ...],
  sequence_by = col("time"),
  except_column_list = [],
  stored_as_scd_type = 1
)
```

Simplicity



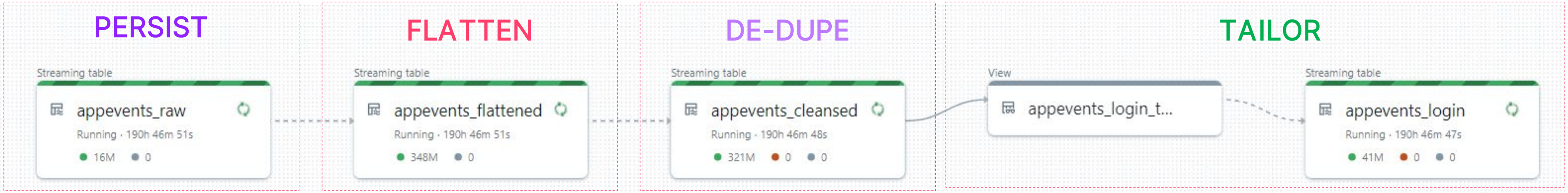

```python
@dlt.table()
@dlt.expect_or_drop(
  "valid records",
  "...")
def appevents_raw():
    options = get_kafka_config_options(
      spark,
      dbutils.secrets)
    return (spark
            .readStream
            .format("kafka")
            .options(**options)
            .load()
            .transform(parse_raw_appevents)
  )
```

```python
@dlt.table()
@dlt.expect_or_drop(
  "valid records",
  "...")
def appevents_flattened():
  return flatten_appevents(
    dlt.readStream("appevents_raw"))
```

```python
dlt.create_streaming_table(
    "appevents_cleansed"
  )

dlt.apply_changes(
  target = "appevents_cleansed",
  source = "appevents_flattened",
  keys = ["event_date", ...],
  sequence_by = col("time"),
  except_column_list = [],
  stored_as_scd_type = 1
)
```

```python
@dlt.view
def appevents_login_temp():
    df = dlt.readStream(
      "appevents_cleansed")
    return filter_login_events(df)
```

Simplicity




```python
@dlt.table()
@dlt.expect_or_drop(
  "valid records",
  "...")
def appevents_raw():
  options = get_kafka_config_options(
    spark,
    dbutils.secrets)
  return (spark
        .readStream
        .format("kafka")
        .options(**options)
        .load()
        .transform(parse_raw_appevents)
  )
```

→

```python
@dlt.table()
@dlt.expect_or_drop(
  "valid records",
  "...")
def appevents_flattened():
  return flatten_appevents(
    dlt.readStream("appevents_raw"))
```

→

```python
dlt.create_streaming_table(
    "appevents_cleansed"
  )

dlt.apply_changes(
  target = "appevents_cleansed",
  source = "appevents_flattened",
  keys = ["event_date", ...],
  sequence_by = col("time"),
  except_column_list = [],
  stored_as_scd_type = 1
)
```

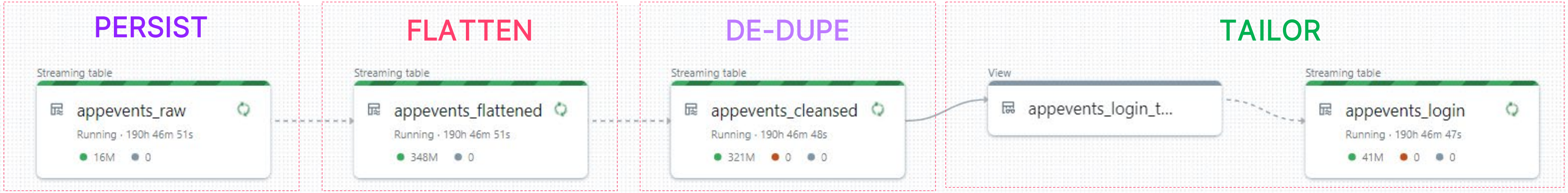
→

```python
@dlt.view
def appevents_login_temp():
  df = dlt.readStream(
    "appevents_cleansed")
  return filter_login_events(df)
```

→

```python
dlt.create_streaming_table(
    "appevents_login"
)

dlt.apply_changes(
  target = "appevents_login",
  source = "appevents_login_temp",
  keys = ["event_date", "event_id"],
  sequence_by = col("event_time"),
  except_column_list = [],
  stored_as_scd_type = 1
)
```

Simplicity




```python
@dlt.table()
@dlt.expect_or_drop(
  "valid records",
  "...")
def appevents_raw():
    options = get_kafka_config_options(
      spark,
      dbutils.secrets)
    return (spark
            .readStream
            .format("kafka")
            .options(**options)
            .load()
            .transform(parse_raw_appevents)
    )
```

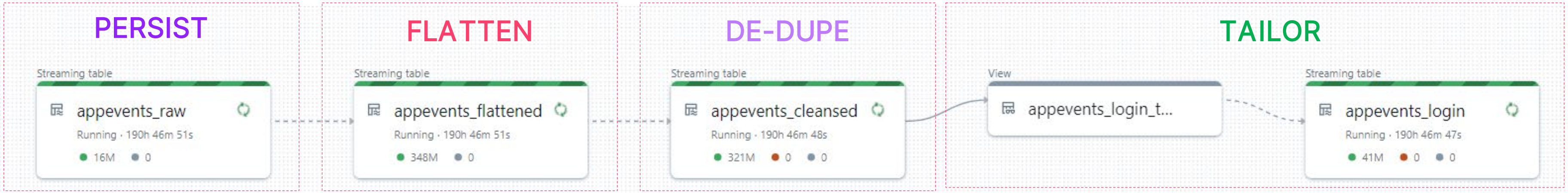
```python
@dlt.table()
@dlt.expect_or_drop(
  "valid records",
  "...")
def appevents_flattened():
  return flatten_appevents(
    dlt.readStream("appevents_raw"))
```

```python
dlt.create_streaming_table(
    "appevents_cleansed"
  )

dlt.apply_changes(
  target = "appevents_cleansed",
  source = "appevents_flattened",
  keys = ["event_date", ...],
  sequence_by = col("time"),
  except_column_list = [],
  stored_as_scd_type = 1
)
```

```python
@dlt.view
def appevents_login_temp():
    df = dlt.readStream(
      "appevents_cleansed")
    return filter_login_events(df)
```

```python
dlt.create_streaming_table(
    "appevents_login"
)

dlt.apply_changes(
  target = "appevents_login",
  source = "appevents_login_temp",
  keys = ["event_date", "event_id"],
  sequence_by = col("event_time"),
  except_column_list = [],
  stored_as_scd_type = 1
)
```

Simplicity




```python
@dlt.table()
@dlt.expect_or_drop(
  "valid records",
  "...")
def appevents_raw():
    options = get_kafka_config_options(
      spark,
      dbutils.secrets)
    return (spark
        .readStream
        .format("kafka")
        .options(**options)
        .load()
        .transform(parse_raw_appevents)
    )
```

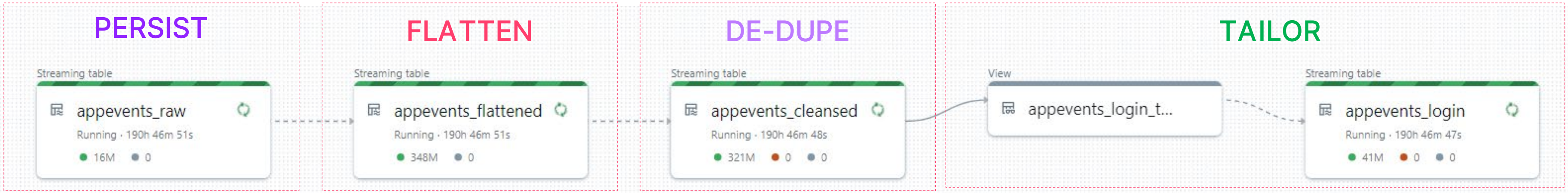
```python
@dlt.table()
@dlt.expect_or_drop(
  "valid records",
  "...")
def appevents_flattened():
  return flatten_appevents(
    dlt.readStream("appevents_raw"))
```

```python
dlt.create_streaming_table(
  "appevents_cleansed"
)

dlt.apply_changes(
  target = "appevents_cleansed",
  source = "appevents_flattened",
  keys = ["event_date", ...],
  sequence_by = col("time"),
  except_column_list = [],
  stored_as_scd_type = 1
)
```

```python
@dlt.view
def appevents_login_temp():
  df = dlt.readStream(
    "appevents_cleansed")
  return filter_login_events(df)
```

```python
dlt.create_streaming_table(
  "appevents_login"
)

dlt.apply_changes(
  target = "appevents_login",
  source = "appevents_login_temp",
  keys = ["event_date", "event_id"],
  sequence_by = col("event_time"),
  except_column_list = [],
  stored_as_scd_type = 1
)
```

Simplicity




```python
@dlt.table()
@dlt.expect_or_drop(
  "valid records",
  "...")
def appevents_raw():
    options = get_kafka_config_options(
      spark,
      dbutils.secrets)
    return (spark
            .readStream
            .format("kafka")
            .options(**options)
            .load()
            .transform(parse_raw_appevents)
    )
```

```python
@dlt.table()
@dlt.expect_or_drop(
  "valid records",
  "...")
def appevents_flattened():
  return flatten_appevents(
    dlt.readStream("appevents_raw"))
```

```python
dlt.create_streaming_table(
    "appevents_cleansed"
  )

dlt.apply_changes(
  target = "appevents_cleansed",
  source = "appevents_flattened",
  keys = ["event_date", ...],
  sequence_by = col("time"),
  except_column_list = [],
  stored_as_scd_type = 1
)
```

```python
@dlt.view
def appevents_login_temp():
    df = dlt.readStream(
      "appevents_cleansed")
    return filter_login_events(df)
```

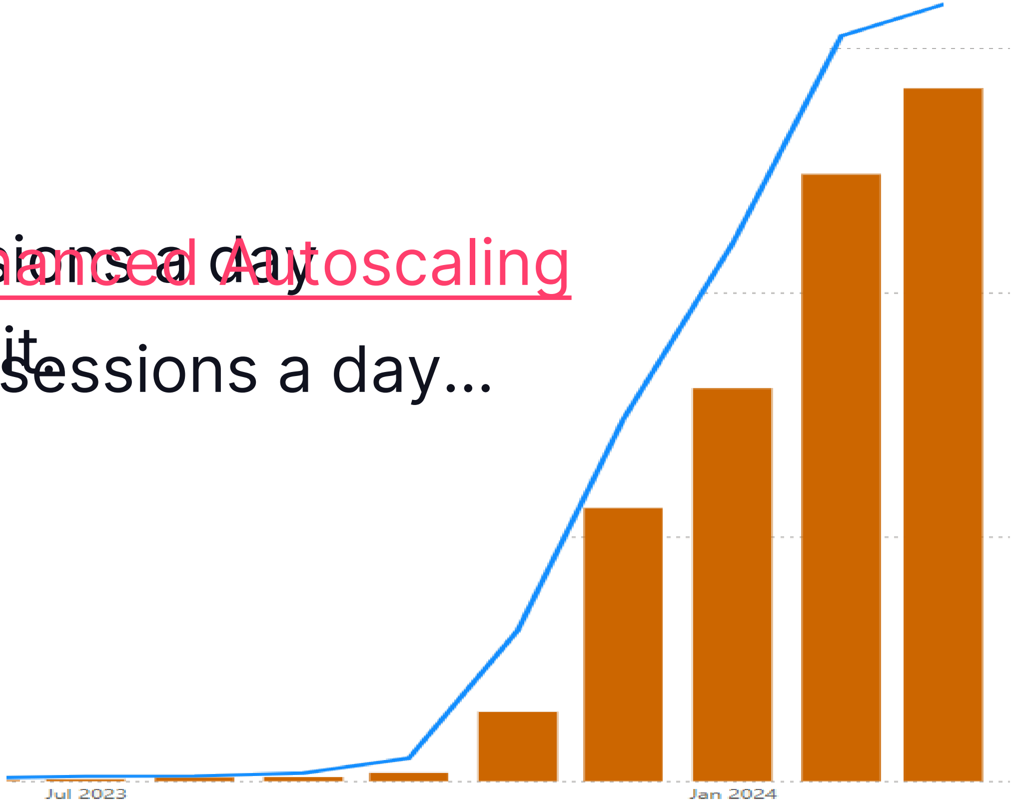
```python
dlt.create_streaming_table(
    "appevents_login"
)

dlt.apply_changes(
  target = "appevents_login",
  source = "appevents_login_temp",
  keys = ["event_date", "event_id"],
  sequence_by = col("event_time"),
  except_column_list = [],
  stored_as_scd_type = 1
)
```

3 queries. 2 CDC calls. ~100% for functional requirement.

Scalability

From <1,000 sessions a day
to >1,000,000 of sessions a day…

We turned on Enhanced Autoscaling
and forgot about it.



Reliability

24*7

continuous streaming
for 9 month

9 billion

events processed

100%

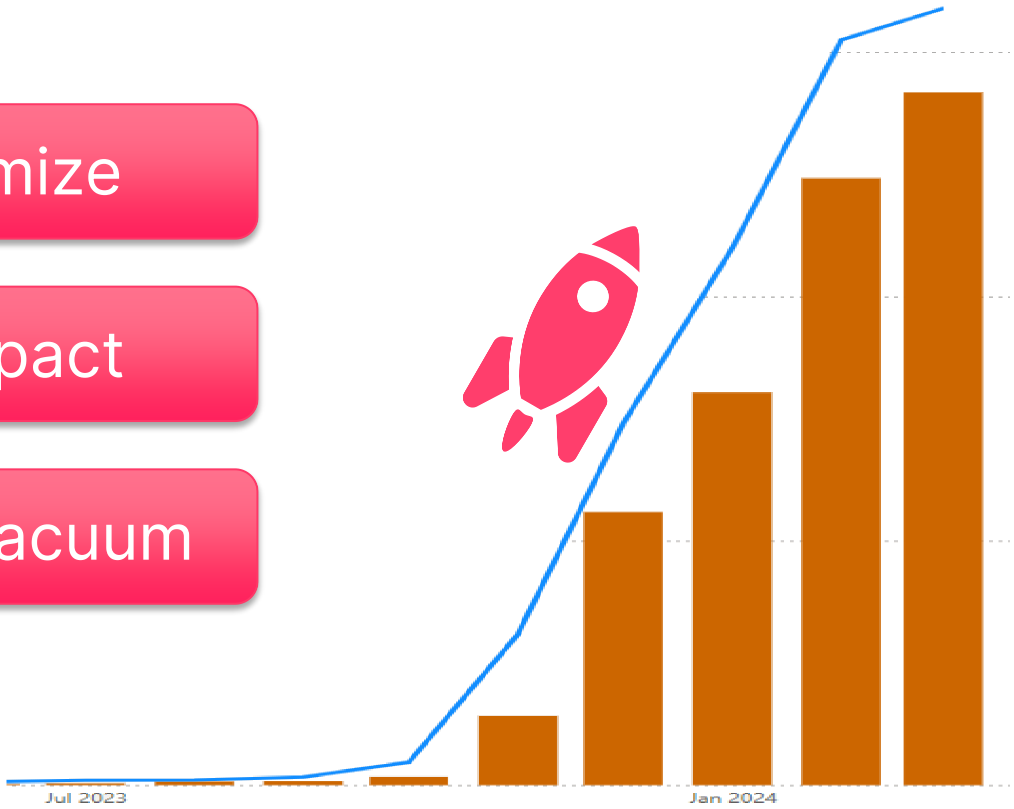
self-recovery from
cloud failures

Performance

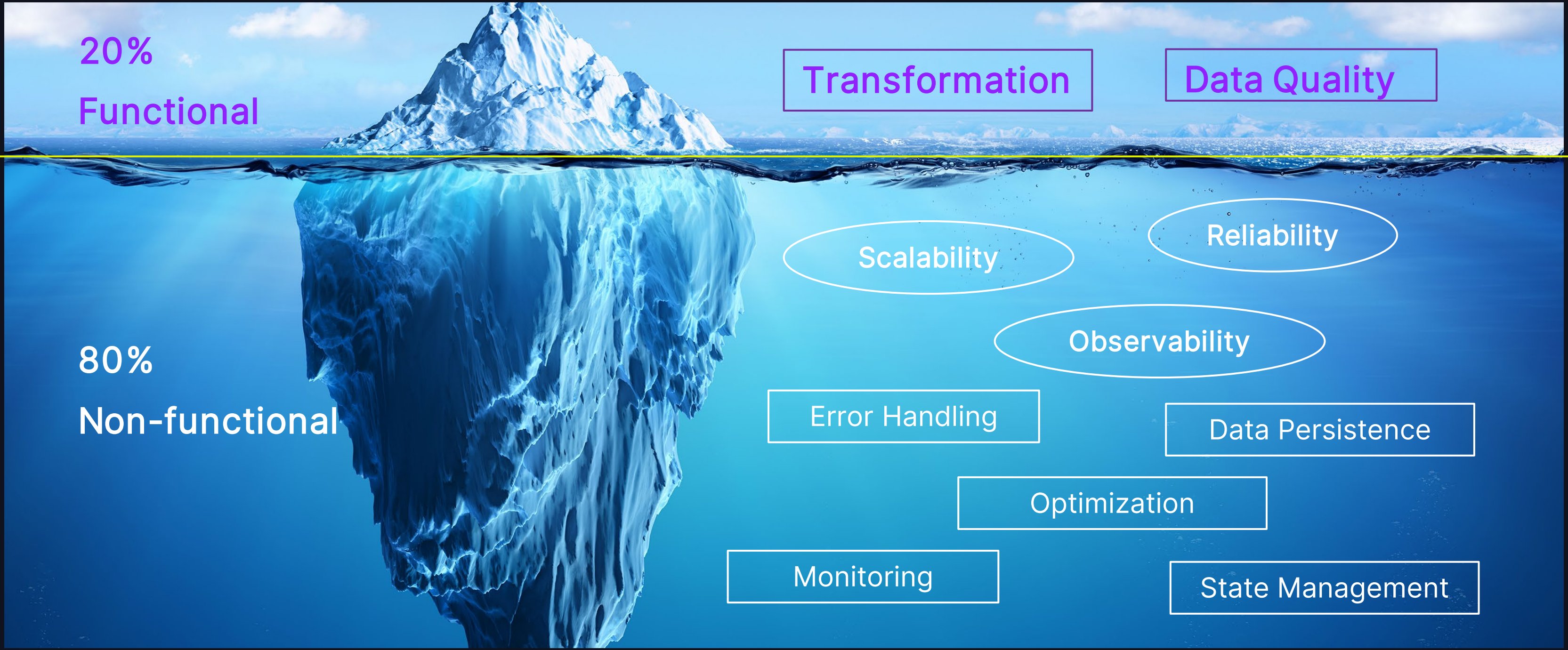
Auto-Optimize

Auto-Compact

Scheduled Vacuum



Where DLT shines



- Time

Engineers

DLT

+ Scale

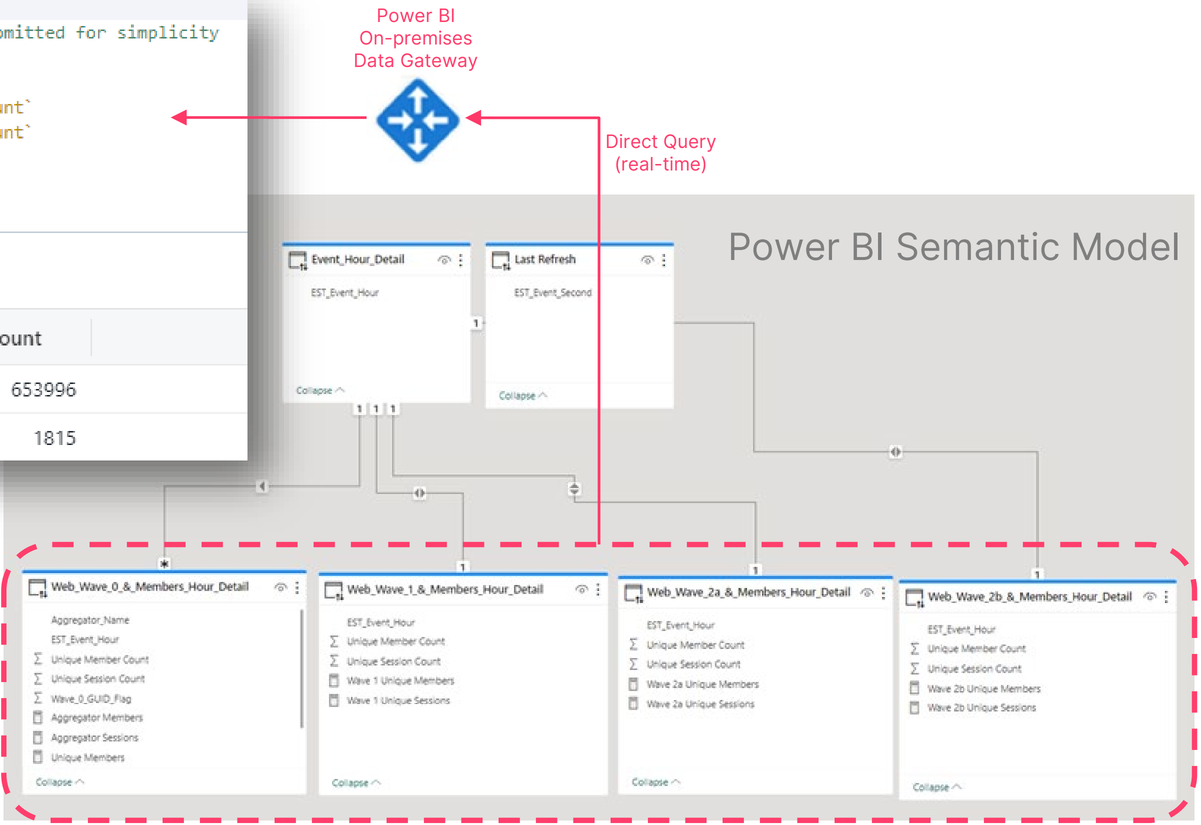
Power BI + Databricks SQL

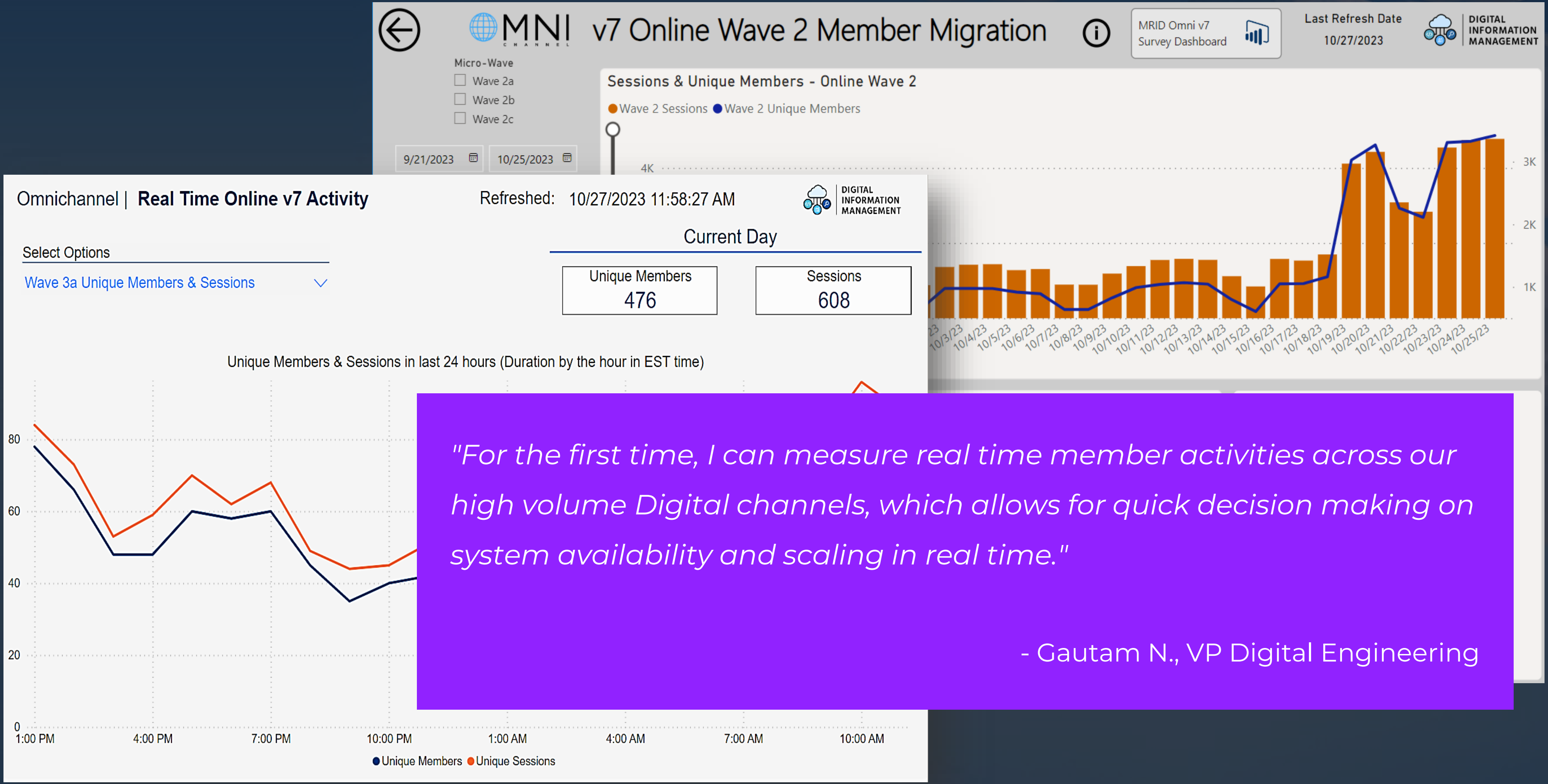
```sql
WITH CTE AS
(SELECT ... FROM appevents_login /* JOIN...WHERE...*/) -- omitted for simplicity
SELECT
  Flag
, COUNT( DISTINCT GUID_V7_Member_ID ) AS `Unique Member Count`
, COUNT( DISTINCT Login_Session_ID ) AS `Unique Session Count`
FROM CTE
GROUP BY Flag
```

Results

#	Flag	Unique Member Count	Unique Session Count
1	0	525163	653996
2	1	1403	1815

Databricks SQL

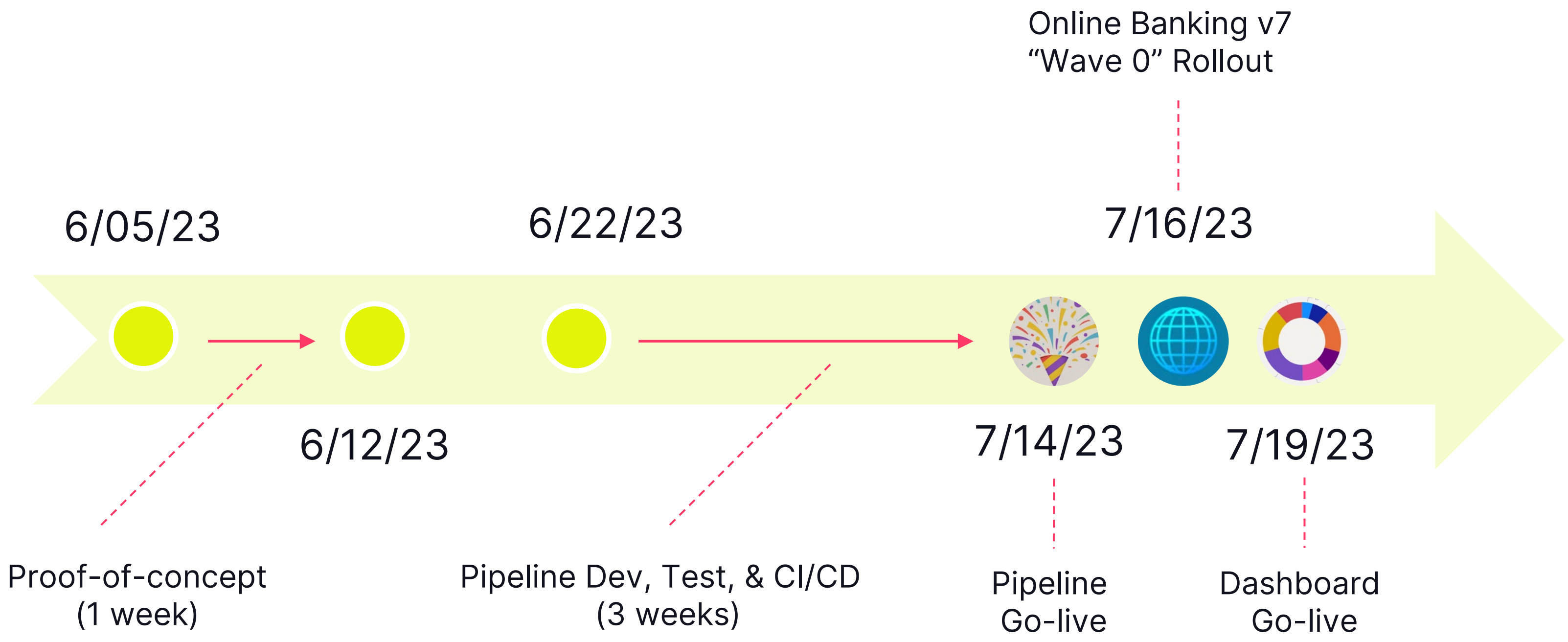




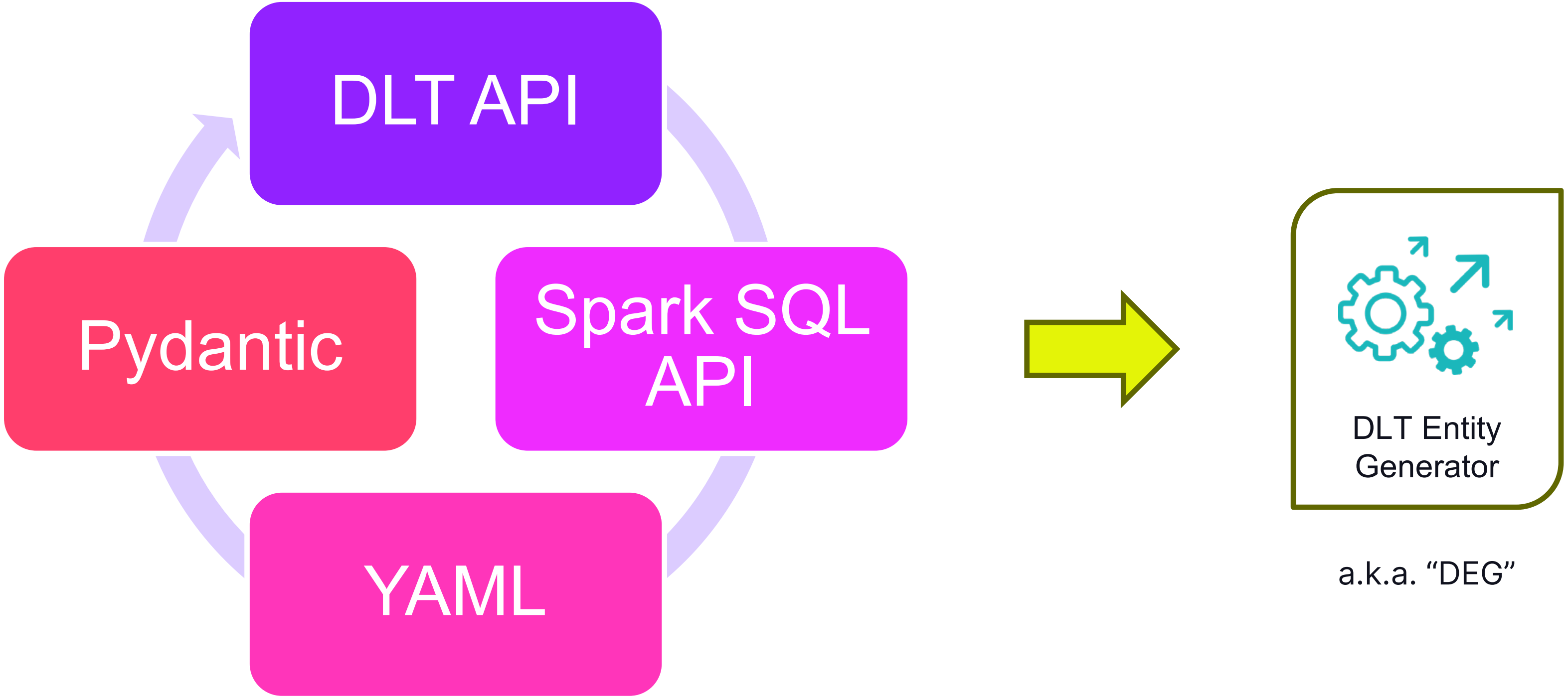
"For the first time, I can measure real time member activities across our high volume Digital channels, which allows for quick decision making on system availability and scaling in real time."

- Gautam N., VP Digital Engineering

Speed to market



METADATA-DRIVEN PIPELINE DEVELOPMENT


DLT API
Pydantic
Spark SQL API
YAML
DLT Entity Generator
a.k.a. "DEG"

Source Definition

Table Definition

View Definition

DQ Expectations

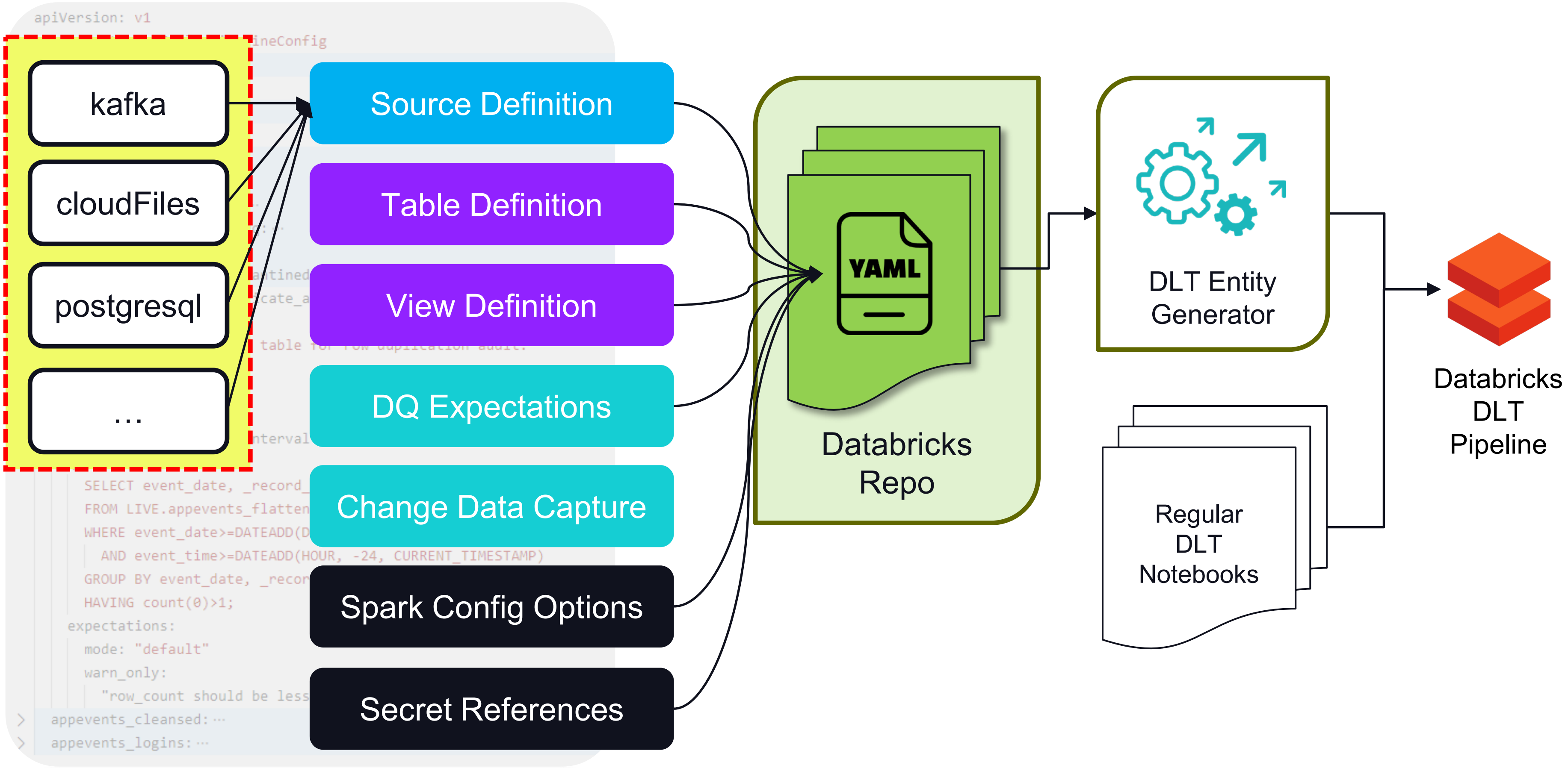
Change Data Capture

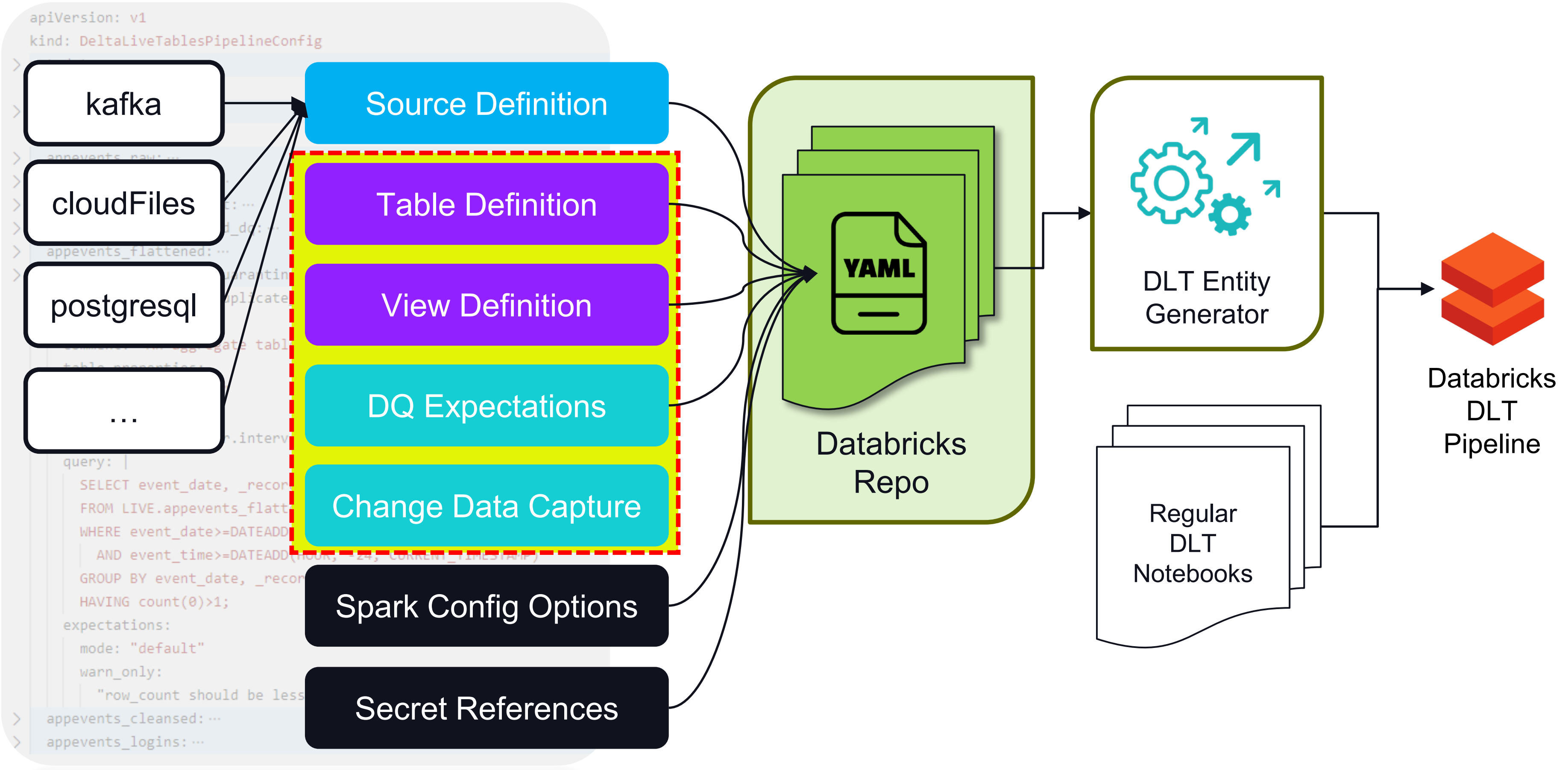
Spark Config Options

Secret References

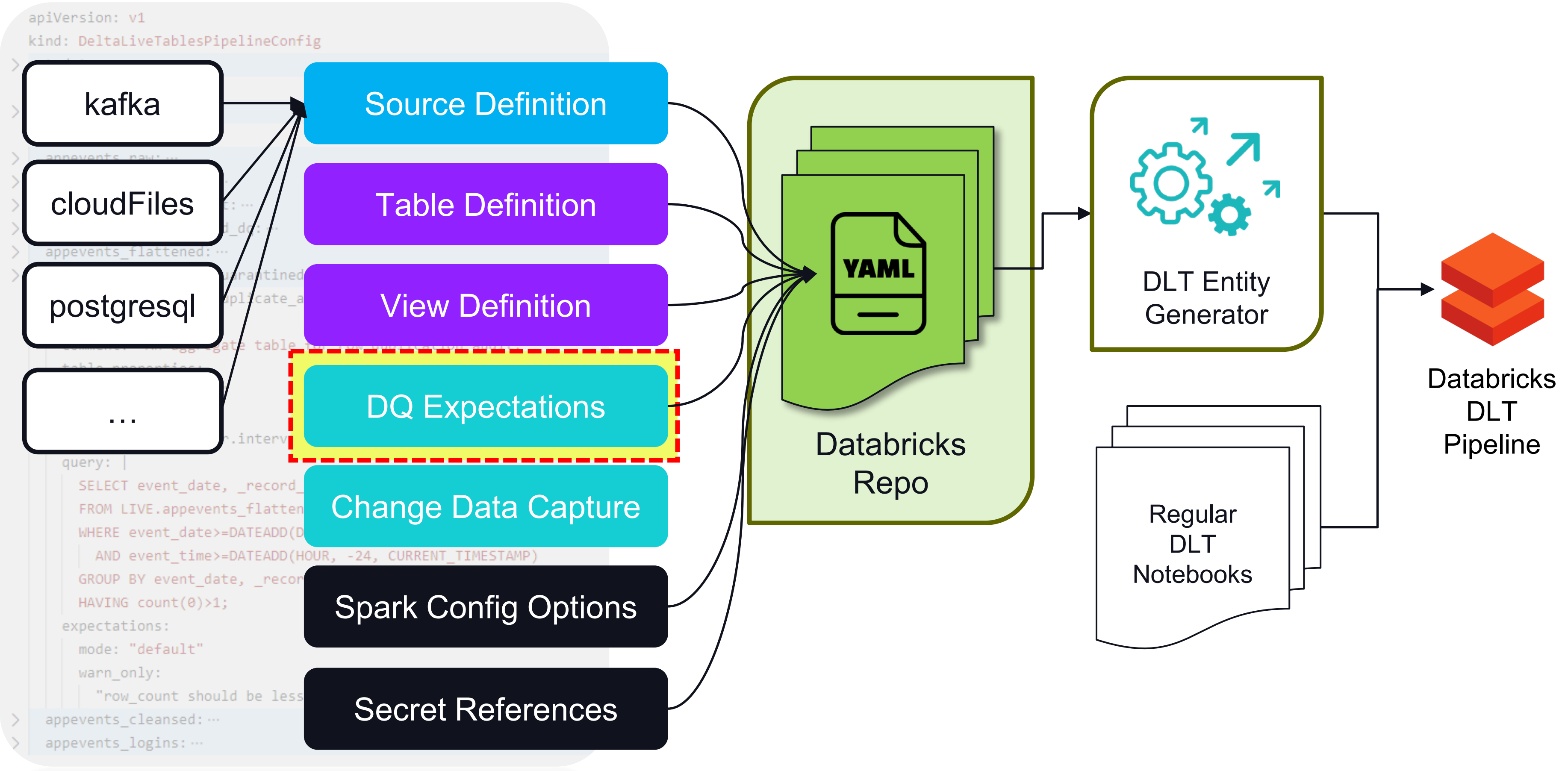
Pipeline Metadata

```yaml
apiVersion: v1
kind: DeltaLiveTablesPipelineConfig
metadata: ...
sources:
  appevents_source: ...
live_table_queries:
  appevents_raw: ...
  vw_appevents_raw_dq: ...
  appevents_raw_dq_audit: ...
  vw_appevents_flattened_dq: ...
  appevents_flattened: ...
  appevents_flattened_quarantined: ...
  appevents_flattened_duplicate_audit:
    type: "table"
    comment: "An aggregate table for row duplication audit."
    table_properties:
      "quality": "audit"
    spark_conf:
      "pipelines.trigger.interval" : "1 hour"
    query: |
      SELECT event_date, _record_hash_sha256, count(0) as row_count
      FROM LIVE.appevents_flattened
      WHERE event_date>=DATEADD(DAY, -1, CURRENT_DATE)
        AND event_time>=DATEADD(HOUR, -24, CURRENT_TIMESTAMP)
      GROUP BY event_date, _record_hash_sha256
      HAVING count(0)>1;
    expectations:
      mode: "default"
      warn_only:
        "row_count should be less than 2": "row_count<2"
  appevents_cleansed: ...
  appevents_logins: ...
```

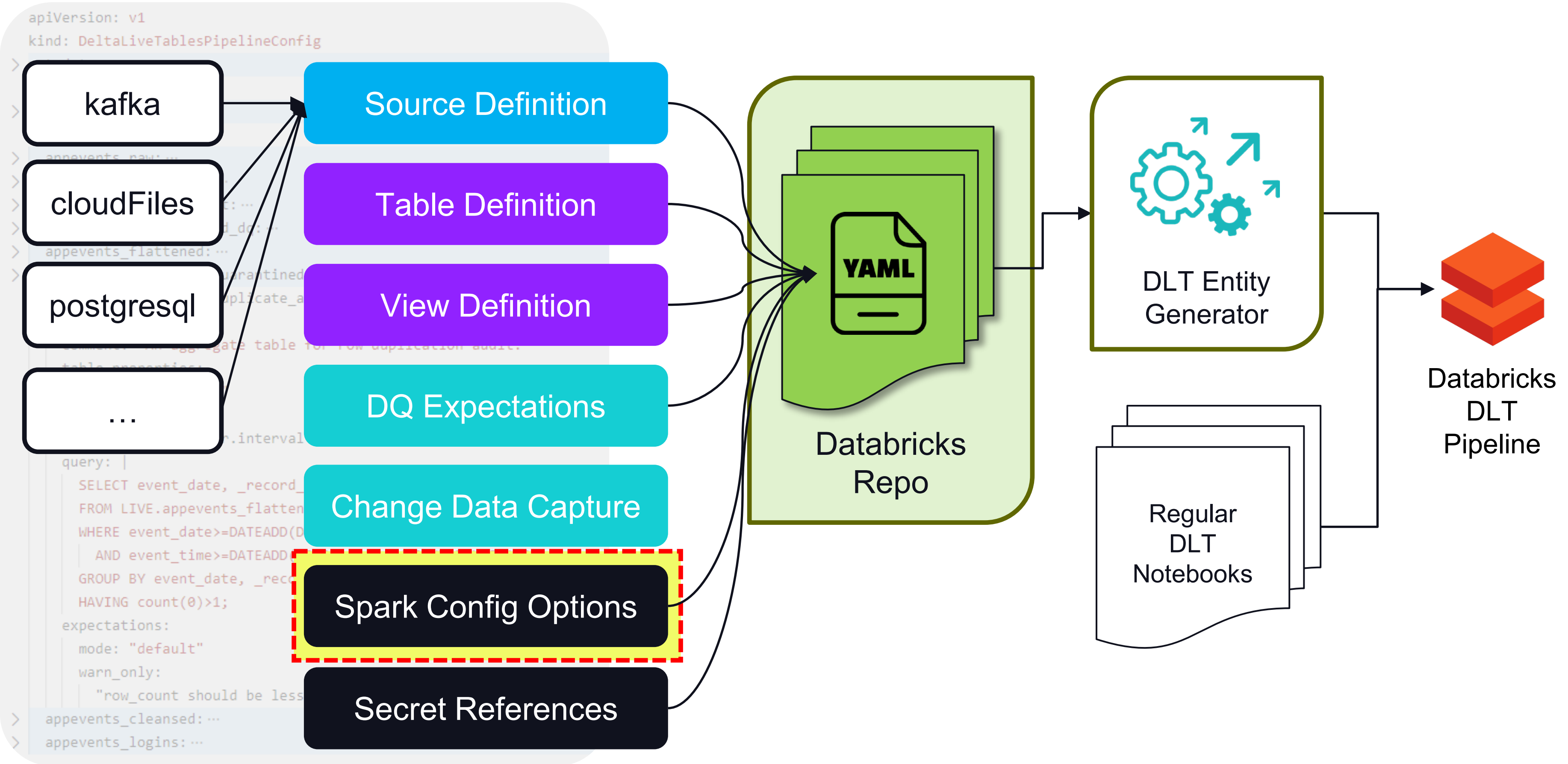

kafka
cloudFiles
postgresql
...
Source Definition
Table Definition
View Definition
DQ Expectations
Change Data Capture
Spark Config Options
Secret References
YAML
Databricks Repo
DLT Entity Generator
Regular DLT Notebooks
Databricks DLT Pipeline

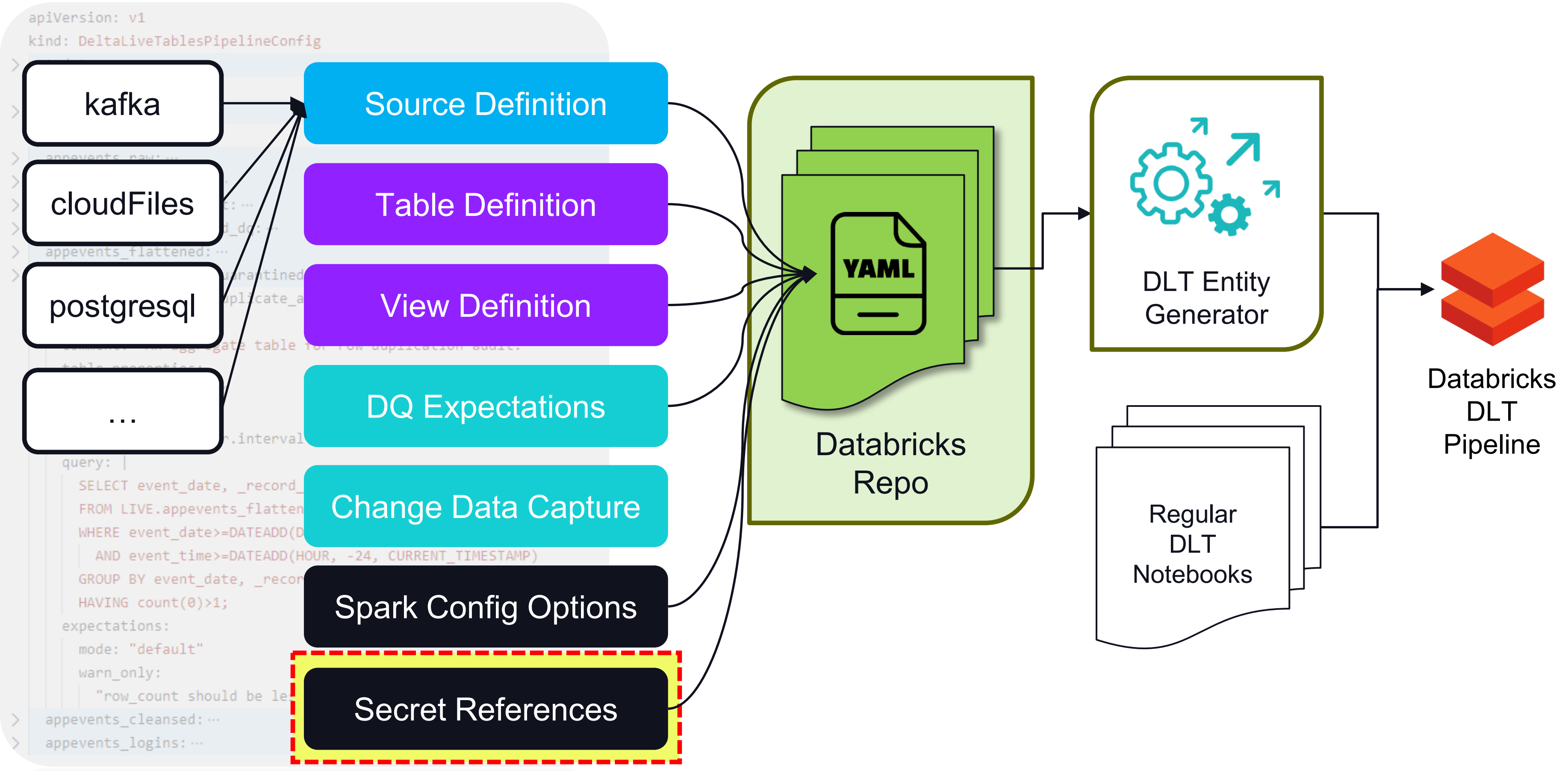


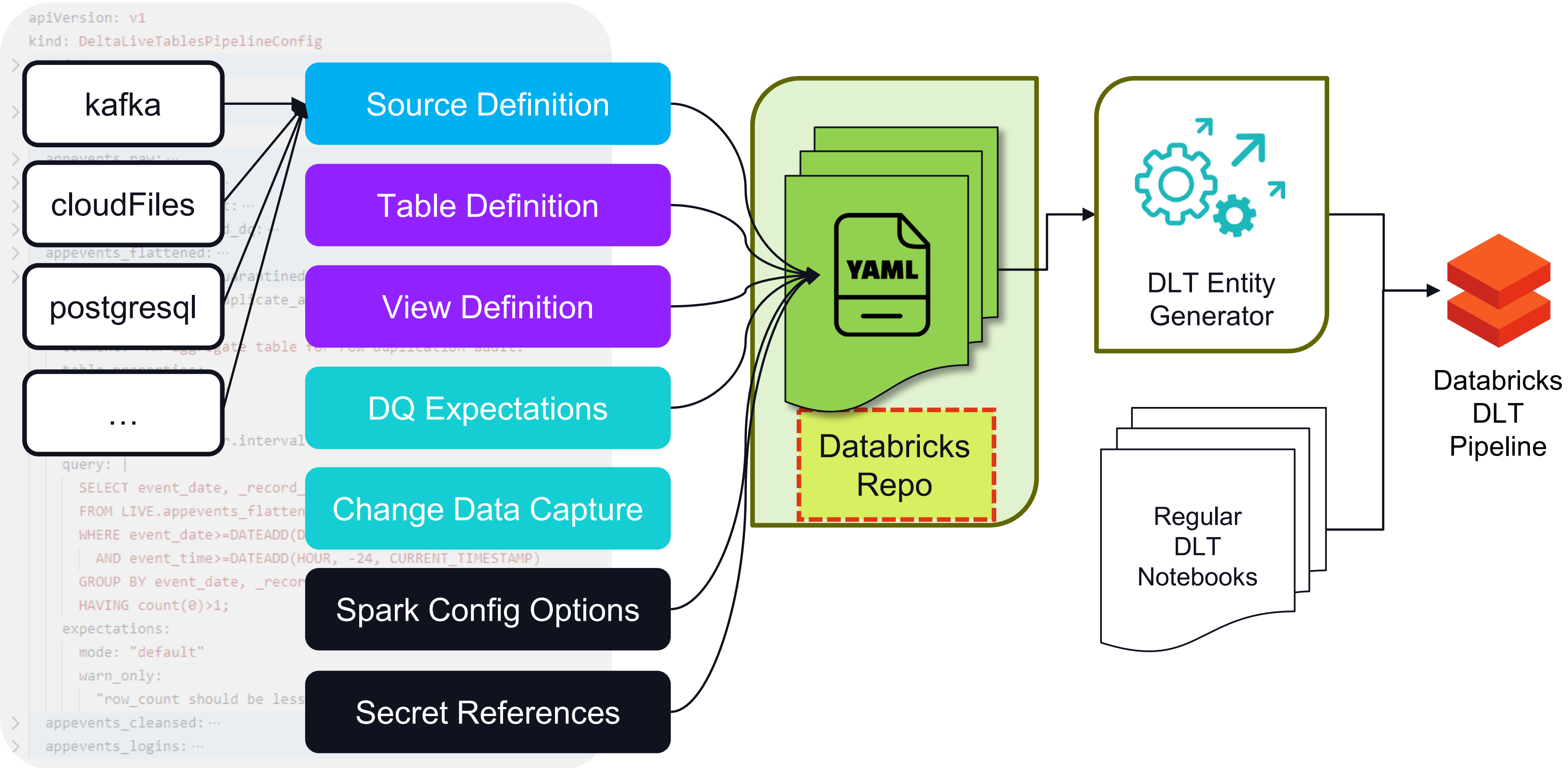
apiVersion: v1
kind: DeltaLiveTablesPipelineConfig
kafka
cloudFiles
postgresql
...
Source Definition
Table Definition
View Definition
DQ Expectations
Change Data Capture
Spark Config Options
Secret References
YAML
Databricks Repo
DLT Entity Generator
Regular DLT Notebooks
Databricks DLT Pipeline

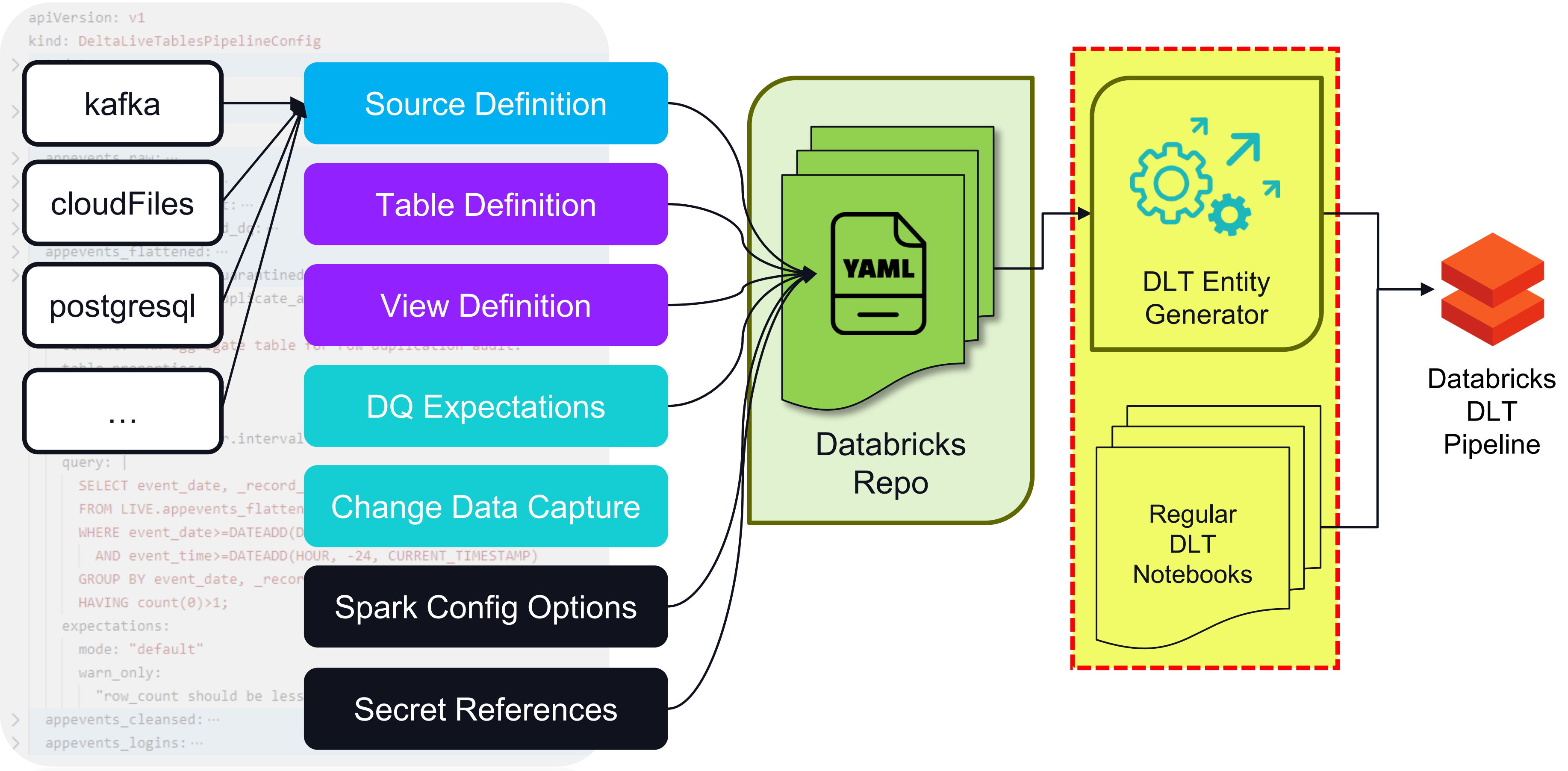


apiVersion: v1
kind: DeltaLiveTablesPipelineConfig
kafka
cloudFiles
postgresql
...
Source Definition
Table Definition
View Definition
DQ Expectations
Change Data Capture
Spark Config Options
Secret References
YAML
Databricks Repo
DLT Entity Generator
Regular DLT Notebooks
Databricks DLT Pipeline

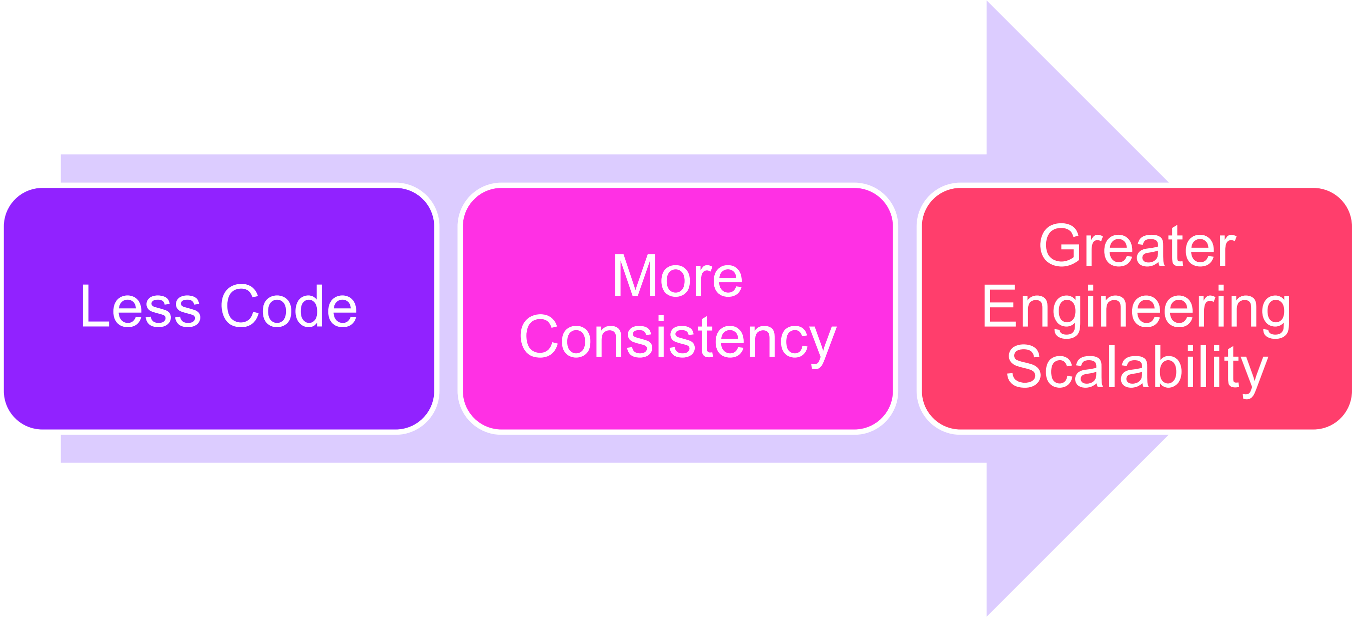

apiVersion: v1
kind: DeltaLiveTablesPipelineConfig
kafka
cloudFiles
postgresql
...
Source Definition
Table Definition
View Definition
DQ Expectations
Change Data Capture
Spark Config Options
Secret References
YAML
Databricks Repo
DLT Entity Generator
Regular DLT Notebooks
Databricks DLT Pipeline


kafka
cloudFiles
postgresql
…
Source Definition
Table Definition
View Definition
DQ Expectations
Change Data Capture
Spark Config Options
Secret References
YAML
Databricks Repo
DLT Entity Generator
Regular DLT Notebooks
Databricks DLT Pipeline


apiVersion: v1
kind: DeltaLiveTablesPipelineConfig
kafka
cloudFiles
postgresql
...
Source Definition
Table Definition
View Definition
DQ Expectations
Change Data Capture
Spark Config Options
Secret References
YAML
Databricks Repo
DLT Entity Generator
Regular DLT Notebooks
Databricks DLT Pipeline


kafka
cloudFiles
postgresql
...
Source Definition
Table Definition
View Definition
DQ Expectations
Change Data Capture
Spark Config Options
Secret References
YAML
Databricks Repo
DLT Entity Generator
Regular DLT Notebooks
Databricks DLT Pipeline

Metadata-driven solution for engineering scalability



SUMMARY

DLT isn’t a silver bullet, but it worked for our use case wonderfully.

- **Simple** declarative programming model → speed to market
- Built-in **scalability**, **reliability**, and **optimization** → simplified operations
- **Programmability** → engineering creativity & scalability

QUESTIONS?

THANK YOU!

Connect on LinkedIn

Connect on LinkedIn