Real-Time Feature Aggregation at Scale: iFood's Path to Sub-Second Latency

Willian Moreira, ML Engineer Lead, iFood

FEATURE STORE SUMMIT 2025

Organized by HOPSWORKS

BIGGEST FOODTECH IN LATIN AMERICA

~14B
Real-time prediction per month

~120M
Orders per month

~116
Real-time predictions per order

~1800
Features in production

Discovery & Checkout

- Restaurants recommendations
- Dishes recommendations
- Fraud detection

Logistics

- Optimize the drivers allocation
- Estimate the delivery time
- Find the most efficient route

Marketing

- Optimize the use of marketing ads
- Optimize the use of coupons

Feature Platform

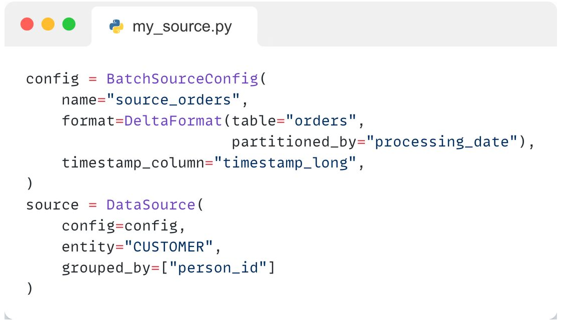
From day one: what we wanted

- ➔ Simple, declarative feature definitions.
- ➔ Stream and batch processing.
- ➔ Support for multiple windows computed simultaneously.
- ➔ Support for online, offline, and time-travel consumption.

So we build the a custom engine that supports all that using spark structured streaming and flatMapGroupsWithState

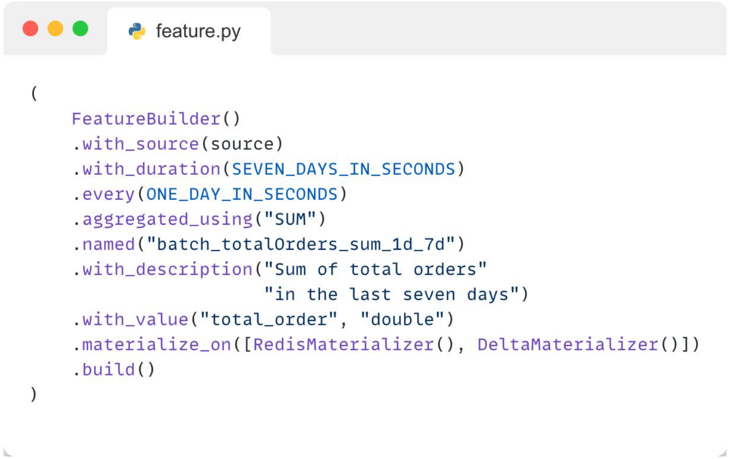
Feature Platform

How we declare a feature



my_source.py

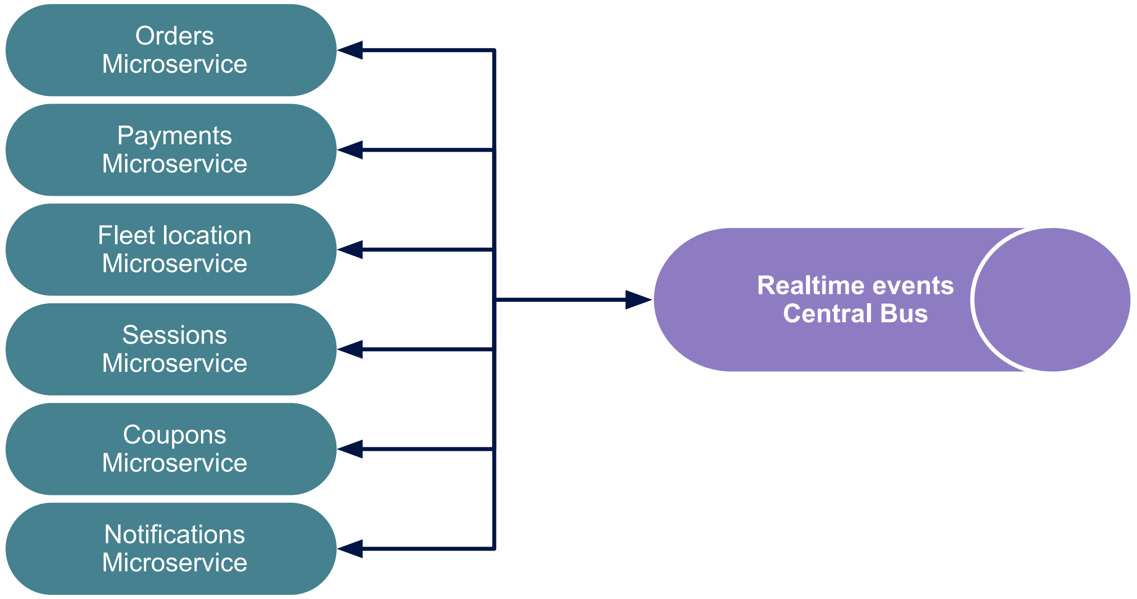
```python
config = BatchSourceConfig(
    name="source_orders",
    format=DeltaFormat(table="orders",
                       partitioned_by="processing_date"),
    timestamp_column="timestamp_long",
)
source = DataSource(
    config=config,
    entity="CUSTOMER",
    grouped_by=["person_id"]
)
```



feature.py

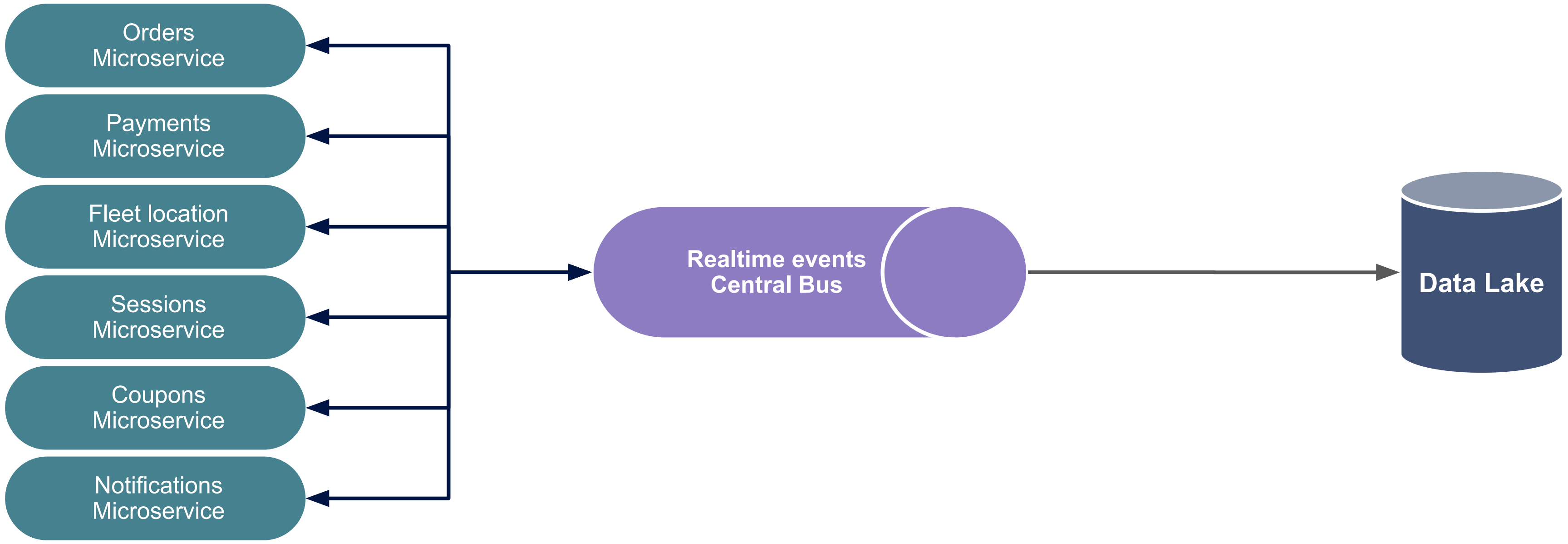
```python
(
    FeatureBuilder()
    .with_source(source)
    .with_duration(SEVEN_DAYS_IN_SECONDS)
    .every(ONE_DAY_IN_SECONDS)
    .aggregated_using("SUM")
    .named("batch_totalOrders_sum_1d_7d")
    .with_description("Sum of total orders"
                      "in the last seven days")
    .with_value("total_order", "double")
    .materialize_on([RedisMaterializer(), DeltaMaterializer()])
    .build()
)
```

iFood Software Architecture
Streaming as a first-class citizen

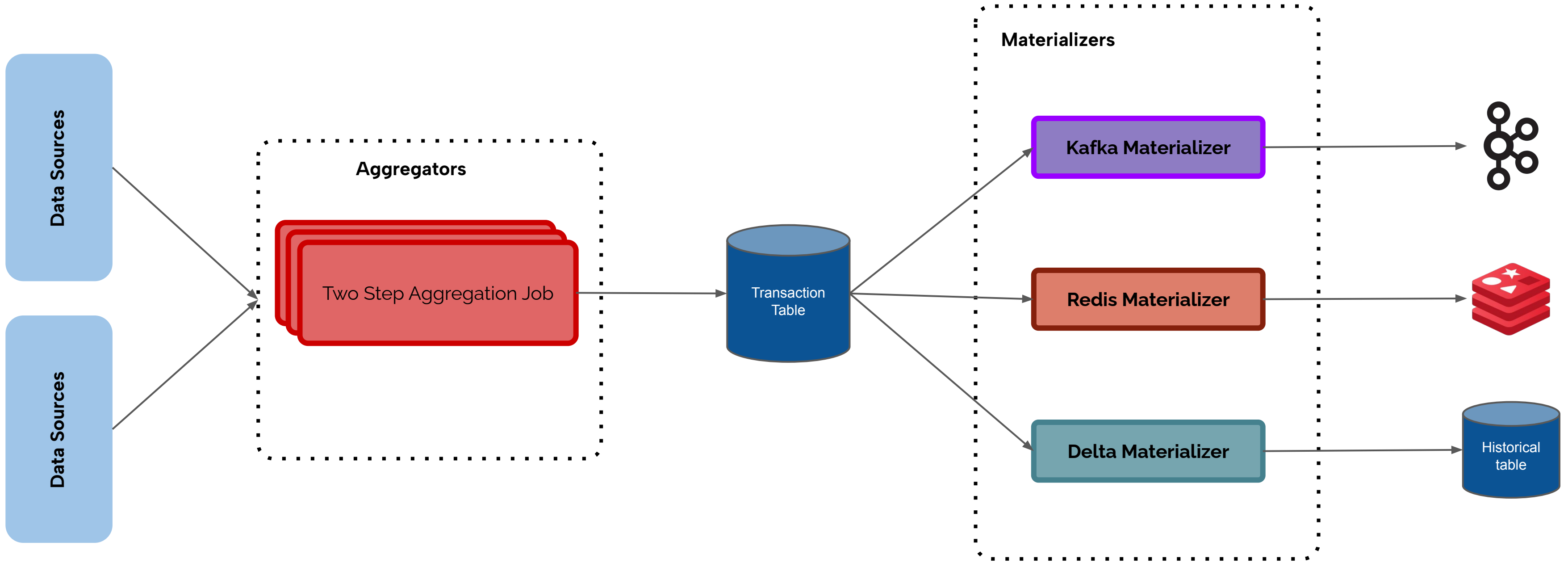


iFood Software Architecture

Streaming as a first-class citizen

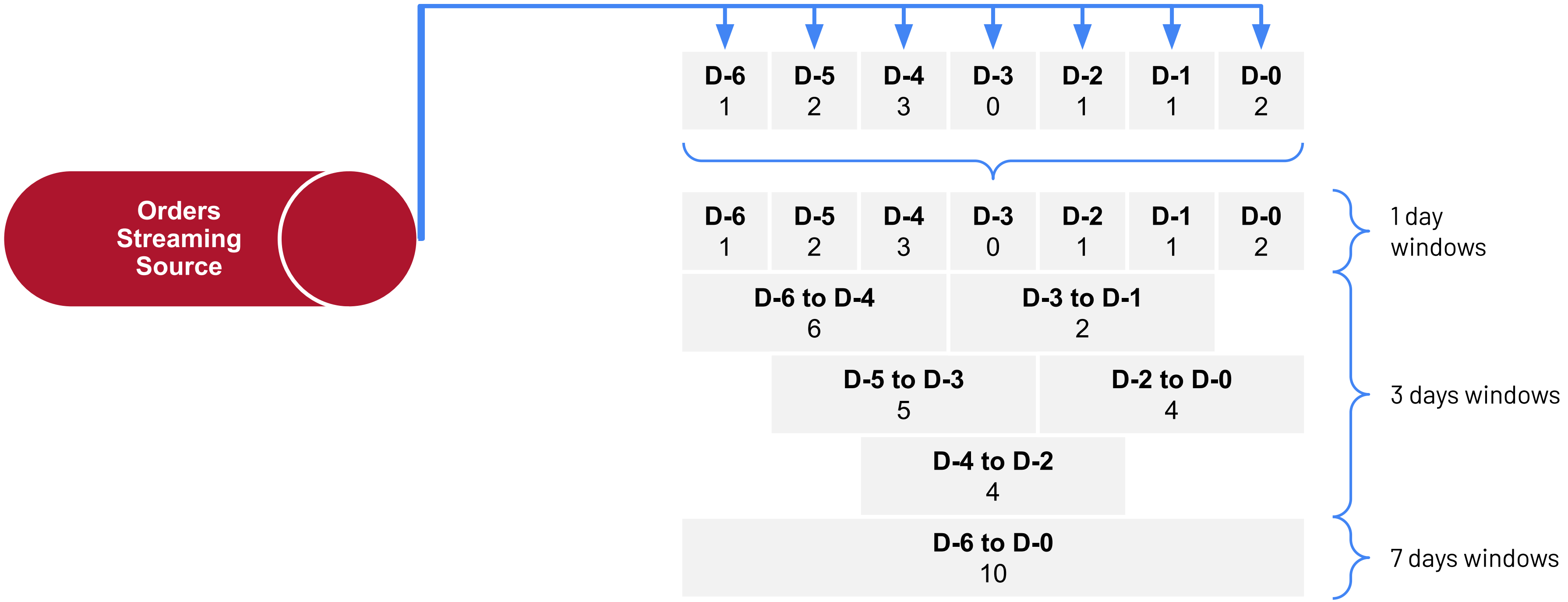


Near Real Time Architecture

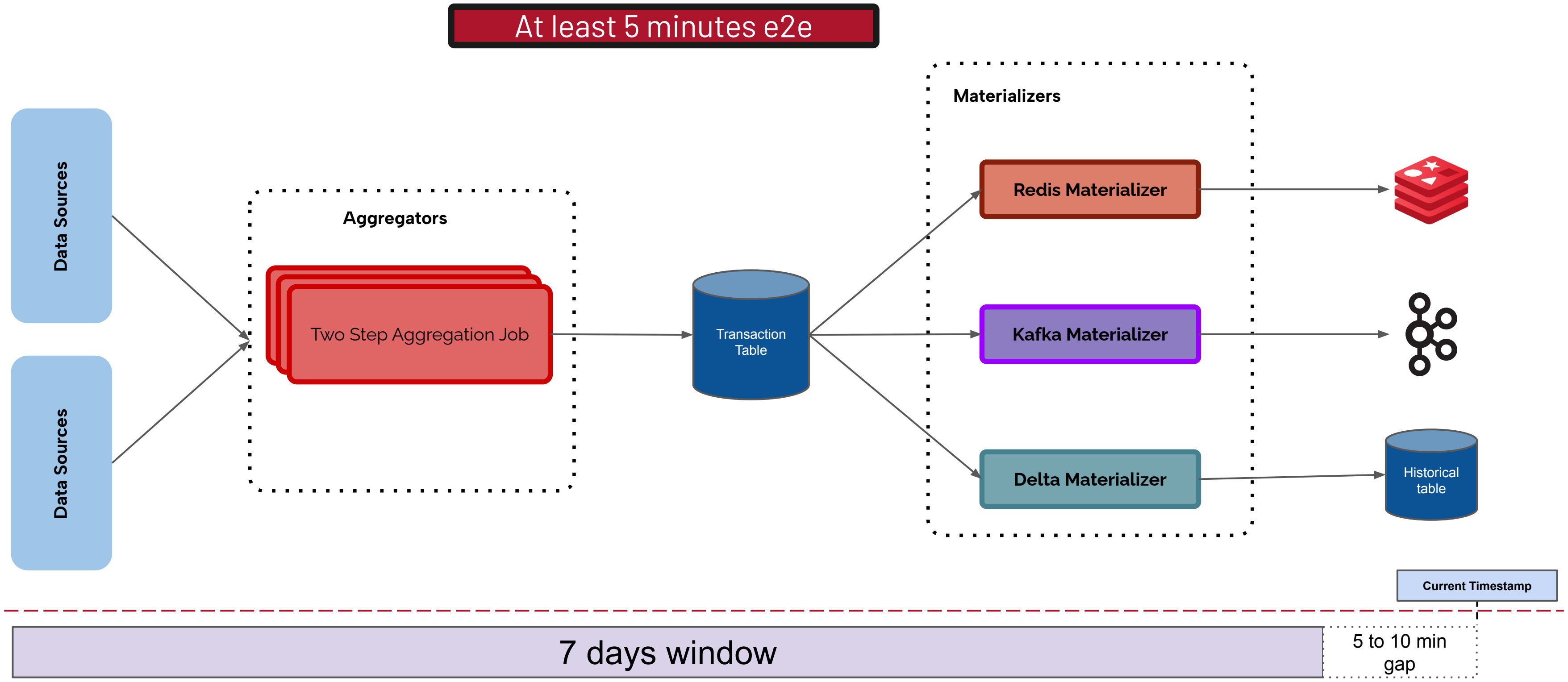


iFood Feature Engine Architecture

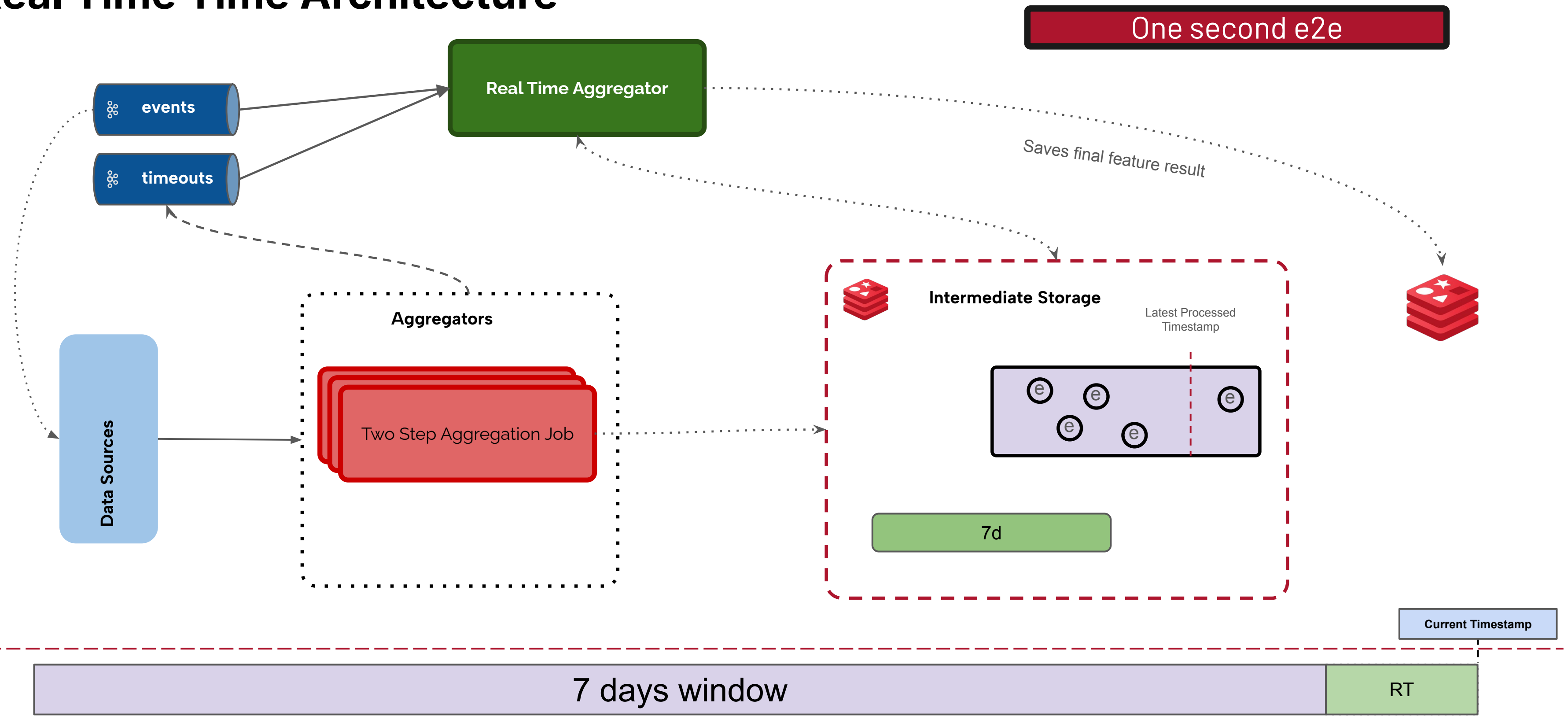
The aggregation jobs - Two-step aggregation logic

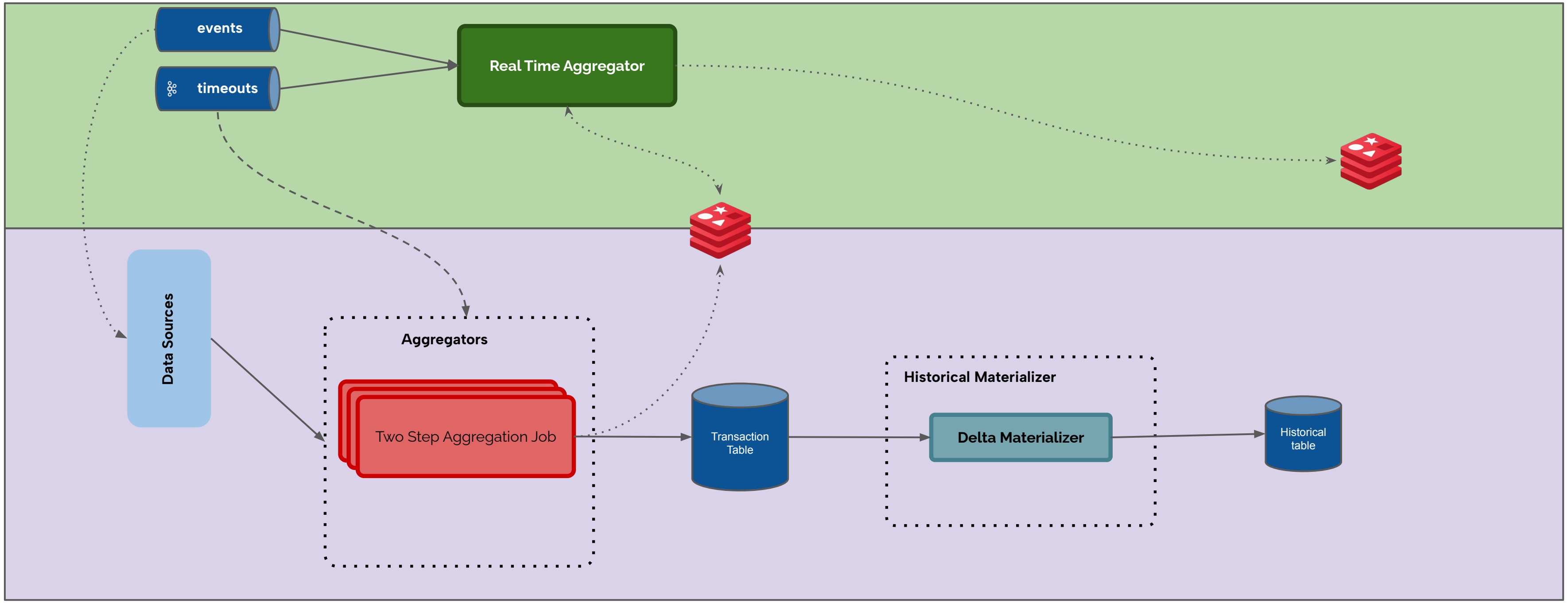


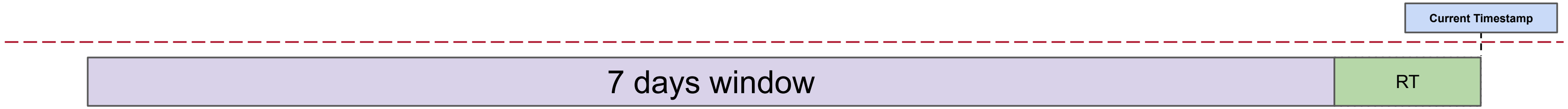
Near Real Time Architecture



Real Time Time Architecture




events
timeouts
Real Time Aggregator
Data Sources
Aggregators
Two Step Aggregation Job
Transaction Table
Historical Materializer
Delta Materializer
Historical table


Current Timestamp
7 days window
RT

Real time in Production

- ➔ 200+ real-time features
- ➔ ~4M online feature retrievals/sec (peak)
- ➔ ~1 s p95 processing latency (ingest → feature ready)
- ➔ 1,800 features available for consumption
- ➔ 400+ Spark jobs in production

Closing thoughts

➔ **One codebase, two modes**. Spark Structured Streaming runs batch and streaming from the same codebase, keeping them aligned and reducing operational work.

➔ **Dual-path architecture**. Heavy/slow for the bigger load + light/continuous for the leading edge of the window. Together they keep latency low.

➔ **Mind the cost**. Two jobs per feature aren’t free; keep real time for signals that change fast and move online metrics.

➔ **Redis as the speed layer**. Ultra fast read/write intermediate store for flexible state sharing.

Thanks!