

DOWNLOAD & USE **THE GOTO GUIDE APP**

CREATE YOUR SCHEDULE
NETWORK WITH OTHERS
ASK QUESTIONS
RATE SESSIONS



Kafka Meets Iceberg

Streaming Data from Kafka to Data Lakes and Data Warehouses

Kasun Indrasiri
Sr. Product Manager @Confluent

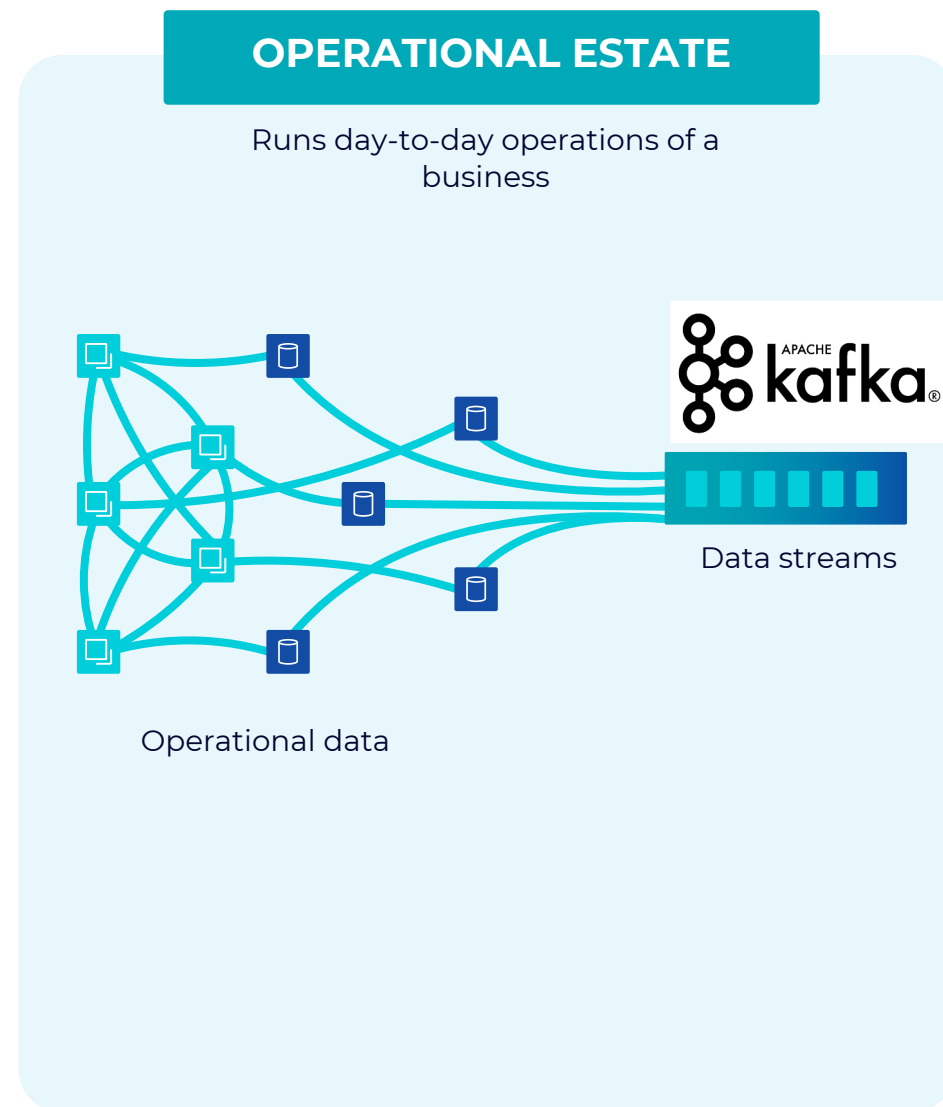
Overview

- **Apache Kafka** for operational data
- **Apache Iceberg** for analytical data
- **Kafka → Iceberg** : Feeding operational data to Data Lakes/ Data Warehouses



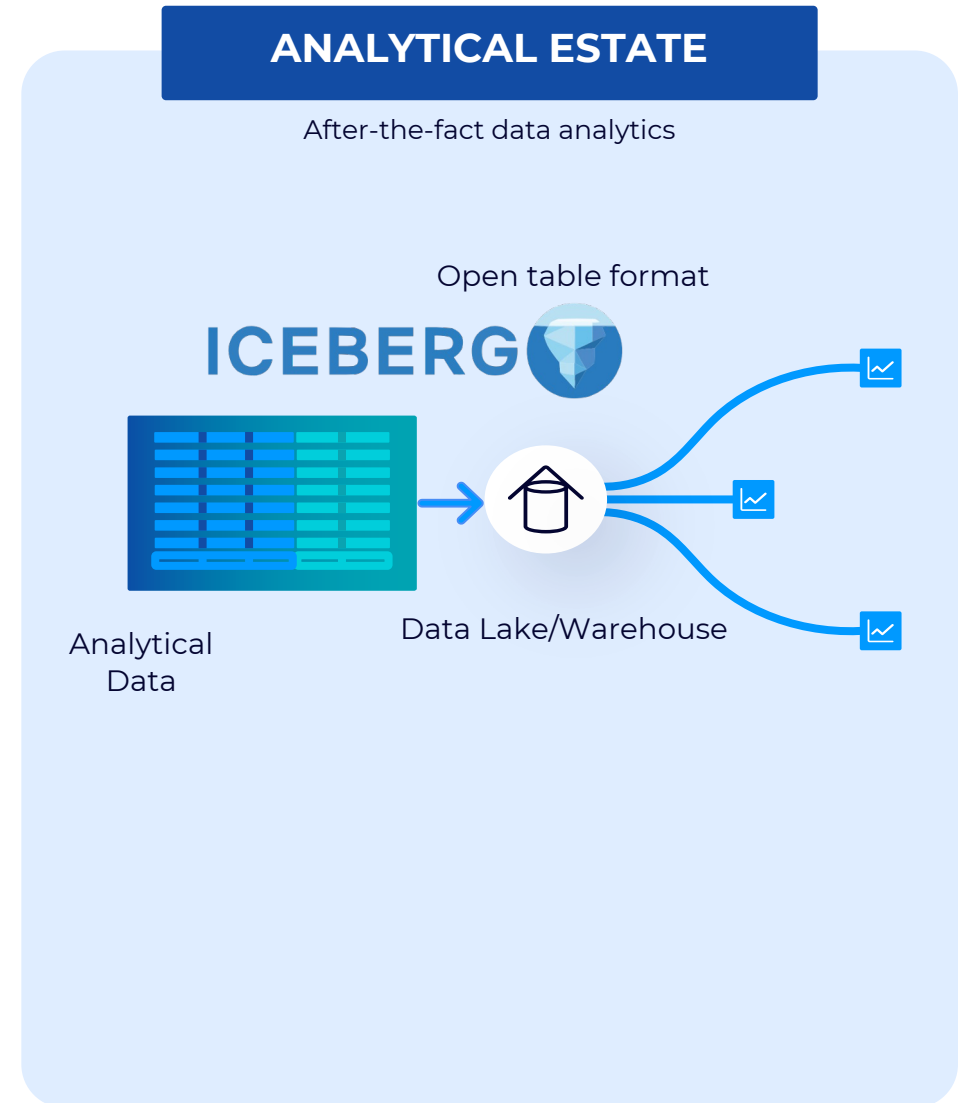
Kafka is the standard for Operational Data

- Operational Estate
 - We run our applications/business here.
- Operational data
 - Data generated by the day-to-day operations of a business.
 - *Transaction records, Clickstream data, Inventory levels, System logs*
- Apache Kafka
 - Capture, organize and store operational data as data streams.



Iceberg for Analytical data in Data Lakes

- **Analytical Estate**
 - Runs after-the-fact data analytics to derive business insights
- **Analytical data**
 - Data collected for analysis and decision-making
 - Stored in data warehouses or data lakes.
 - **Apache Iceberg** is the de-facto standard for tables.



Apache Iceberg

- An open table format for analytic datasets
- Brings simplicity of SQL tables to data lakes and warehouses with reliability and consistency.



Why Iceberg?

- **Expressive SQL support**
 - Provides advanced SQL capabilities for querying large datasets with ease and efficiency.
 - Handles Complex data types, aggregations, filtering, and grouping effectively.

```
SELECT
    customer_id,
    COUNT(order_id) AS total_orders,
    SUM(order_amount) AS total_spent,
    MAX(order_date) AS last_order_date
FROM
    orders
WHERE
    order_date >= '2024-01-01'
    AND order_status = 'completed'
GROUP BY
    customer_id
HAVING
    total_spent > 1000
ORDER BY
    total_spent DESC;
```

Why Iceberg?

- **ACID Transactions**
 - Iceberg provides robust ACID (Atomicity, Consistency, Isolation, Durability) transaction support to ensure data integrity and reliability in data lakes

Why Iceberg?

- **Schema Evolution**
 - Modify schema (add, drop, rename columns) without table rewrites or losing historical data.

```
CREATE TABLE orders (  
    order_id INT,  
    customer_id STRING,  
    order_amount DOUBLE,  
    order_date DATE  
) USING iceberg;  
  
-- Evolve the Schema by Adding a New Column  
ALTER TABLE orders  
    ADD COLUMN order_status STRING;  
  
-- Type change from DOUBLE to DECIMAL  
ALTER TABLE orders  
    ALTER COLUMN order_amount SET TYPE DECIMAL(10, 2);  
  
-- Update Existing Records with the New Column  
UPDATE orders  
SET order_status = 'completed'  
WHERE order_id IN (1, 2);  
  
SELECT * FROM orders;
```

Why Iceberg?

- Time Travel

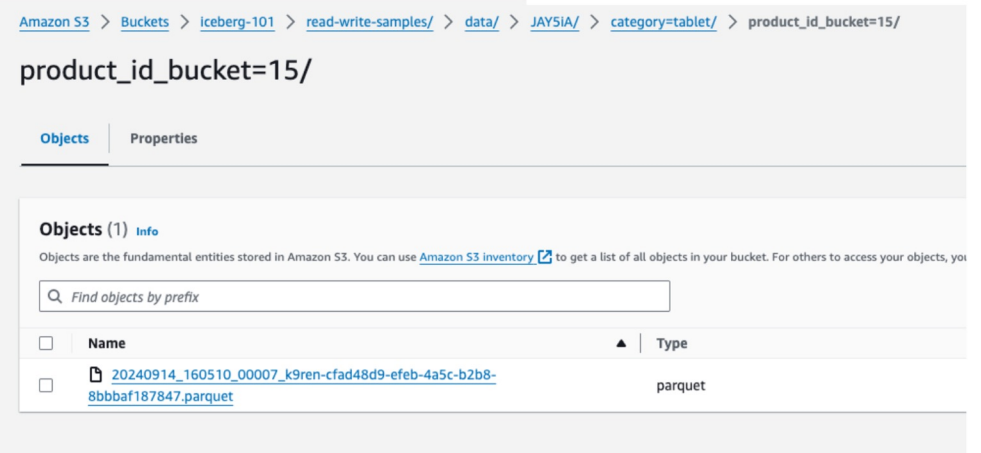
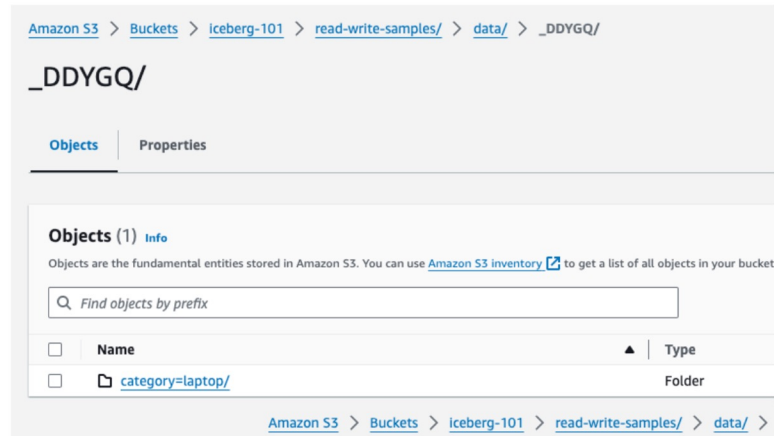
- Allows users to access historical data by querying previous versions of a table, based on snapshots or timestamps.

```
SELECT * FROM products  
    TIMESTAMP AS OF '2024-10-20T10:00:00'
```

Why Iceberg?

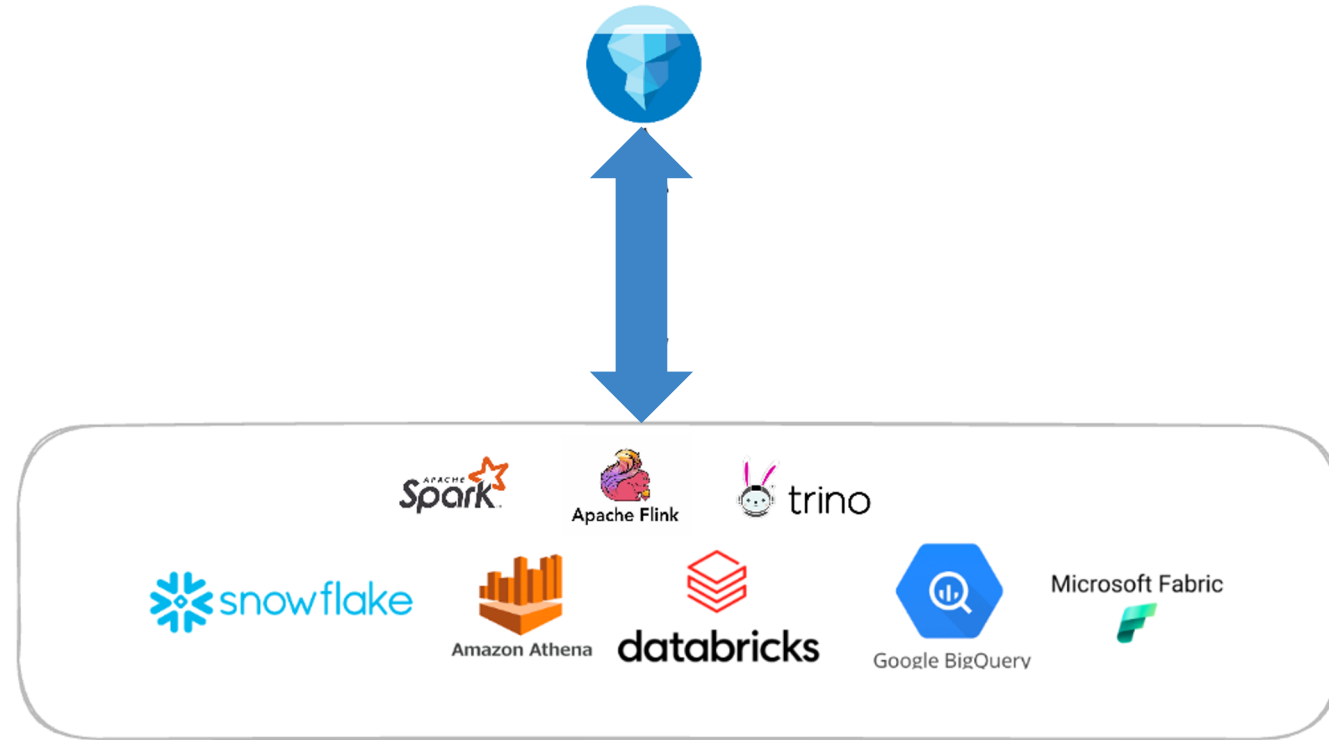
- **Hidden Partitioning**
 - Seamlessly partitioning data without having to create additional columns.
 - No need to manage additional partitioning columns.
 - Better query performance.
 - Lower storage costs by minimizing redundant data in files.

```
CREATE TABLE products (  
  product_id BIGINT,  
  name STRING,  
  category STRING,  
  price DECIMAL(10, 2))  
PARTITIONED BY (category, bucket(16, product_id))
```



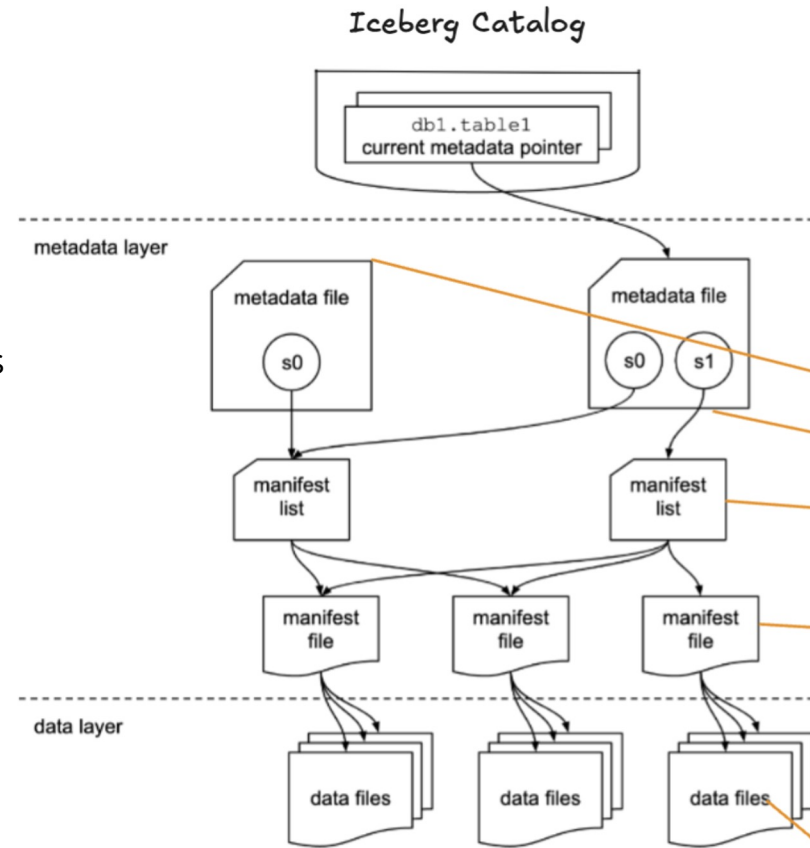
Why Iceberg?

- **Interoperability and Open Standard**
 - Seamless interoperability across various compute engines,
 - Vendor-Neutral
 - Broad Ecosystem Support



Structure of an Iceberg Table

- Data layer
 - Data files.
- Metadata Layer
 - Store the table-wide information: schema, partitioning, and pointers to the snapshots.
- Iceberg Catalog
 - Central Repository for tables
 - Table discovery, creation and modification
 - Ensures Transactional consistency
 - Glue, Polaris, Nessi, Unity, Hive Metastore



Amazon S3 > Buckets > iceberg-101 > architecture/ > metadata/

metadata/

Objects Properties

Objects (7) Info

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a

Find objects by prefix

☐	Name
☐	00000-e6472dac-cabe-4baf-9f7f-f8f2dc013886.metadata.json
☐	00001-7143f7ee-42d5-4600-b802-66d6b765773b.metadata.json
☐	00002-8e868625-3d70-4bb8-a2f9-acb243400445.metadata.json
☐	34158220-b56d-492f-a573-bd84162261cb-m0.avro
☐	dd0e1ad3-a7d2-423f-b0ae-8a015f96d687-m0.avro
☐	snap-1019593025534849376-1-dd0e1ad3-a7d2-423f-b0ae-8a015f96d687.avro
☐	snap-847503569033545781-1-34158220-b56d-492f-a573-bd84162261cb.avro

Amazon S3 > Buckets > iceberg-101 > architecture/ > data/ > B4r4kw/

B4r4kw/

Objects Properties

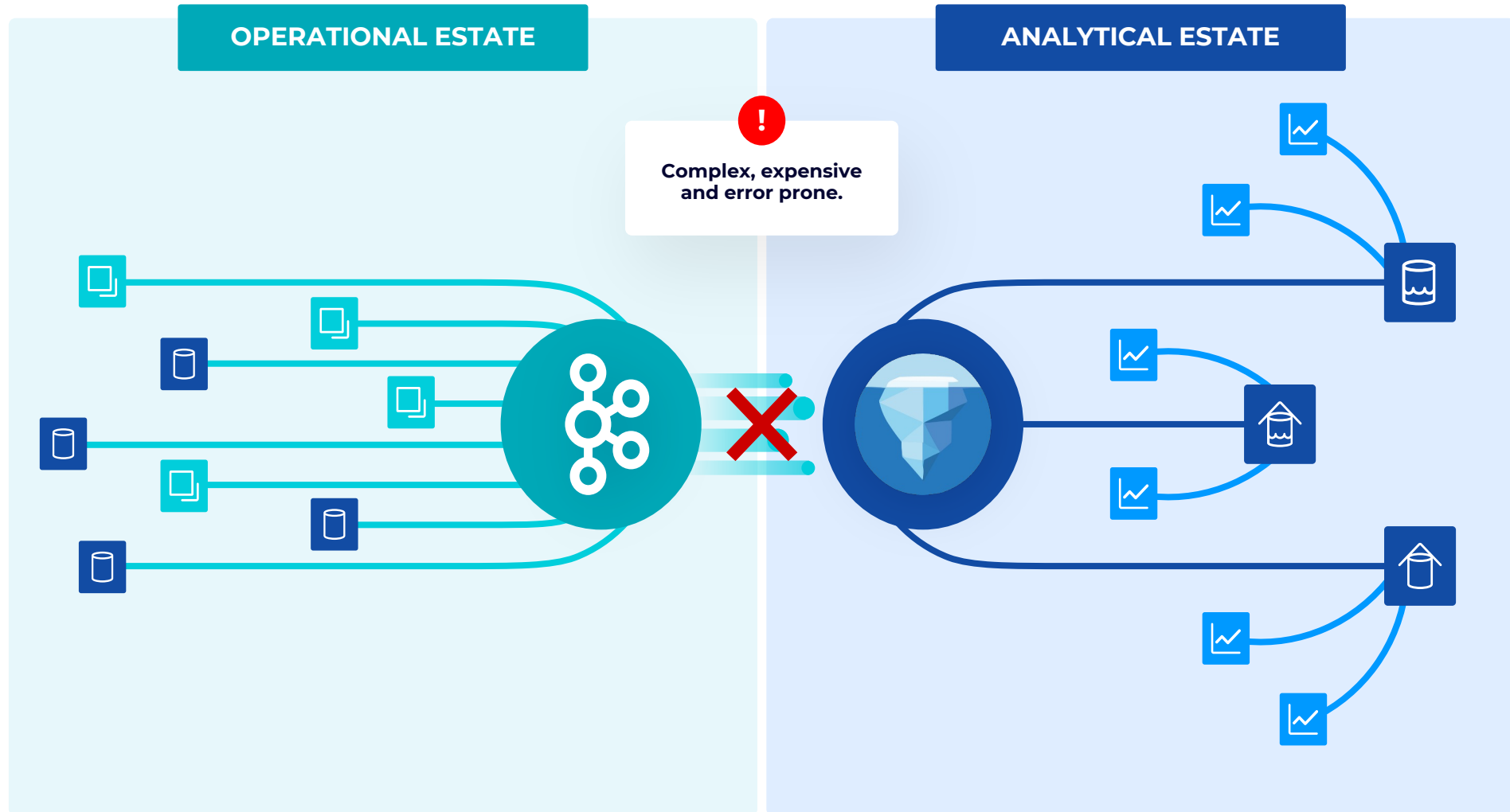
Objects (1) Info

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all o

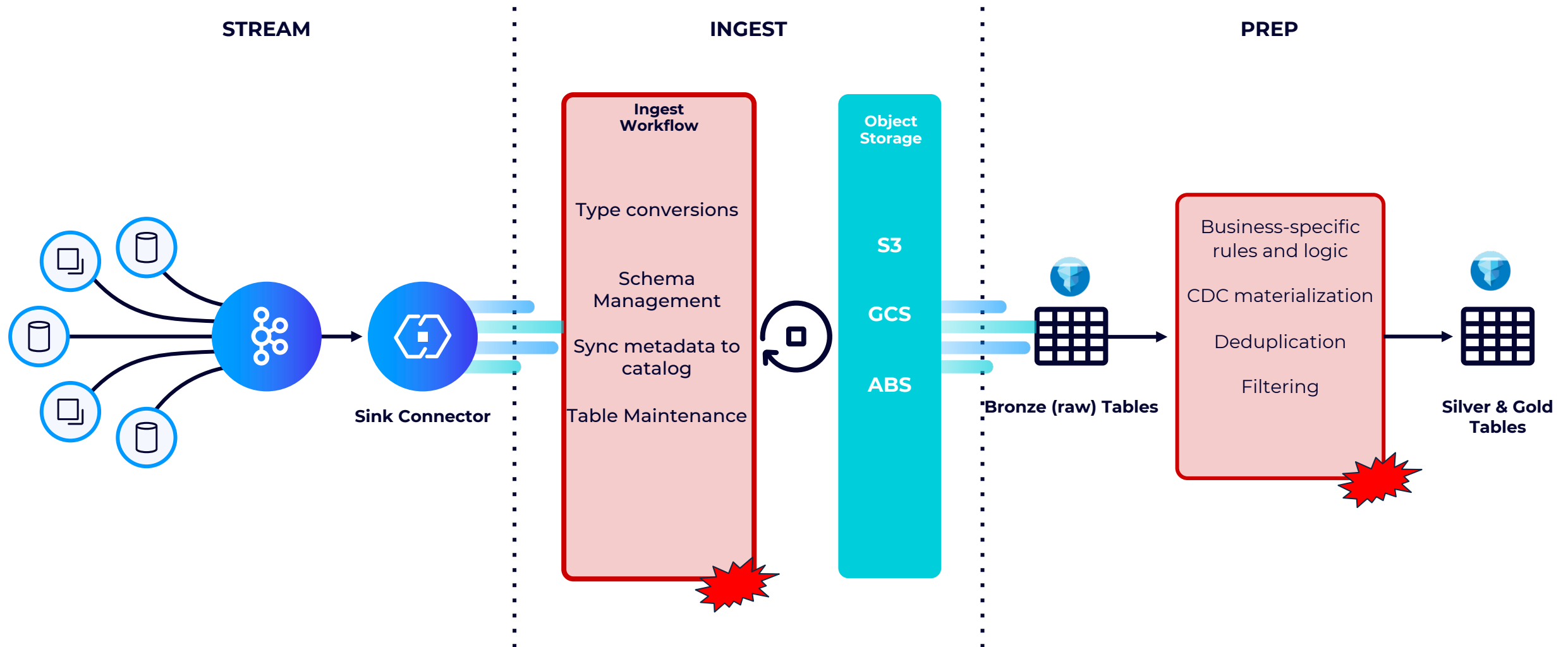
Find objects by prefix

☐	Name	Type
☐	20240708_003516_00022_6fnir-1a41205c-de8b-4adf-adf7-3bde279c5b3d.parquet	parquet

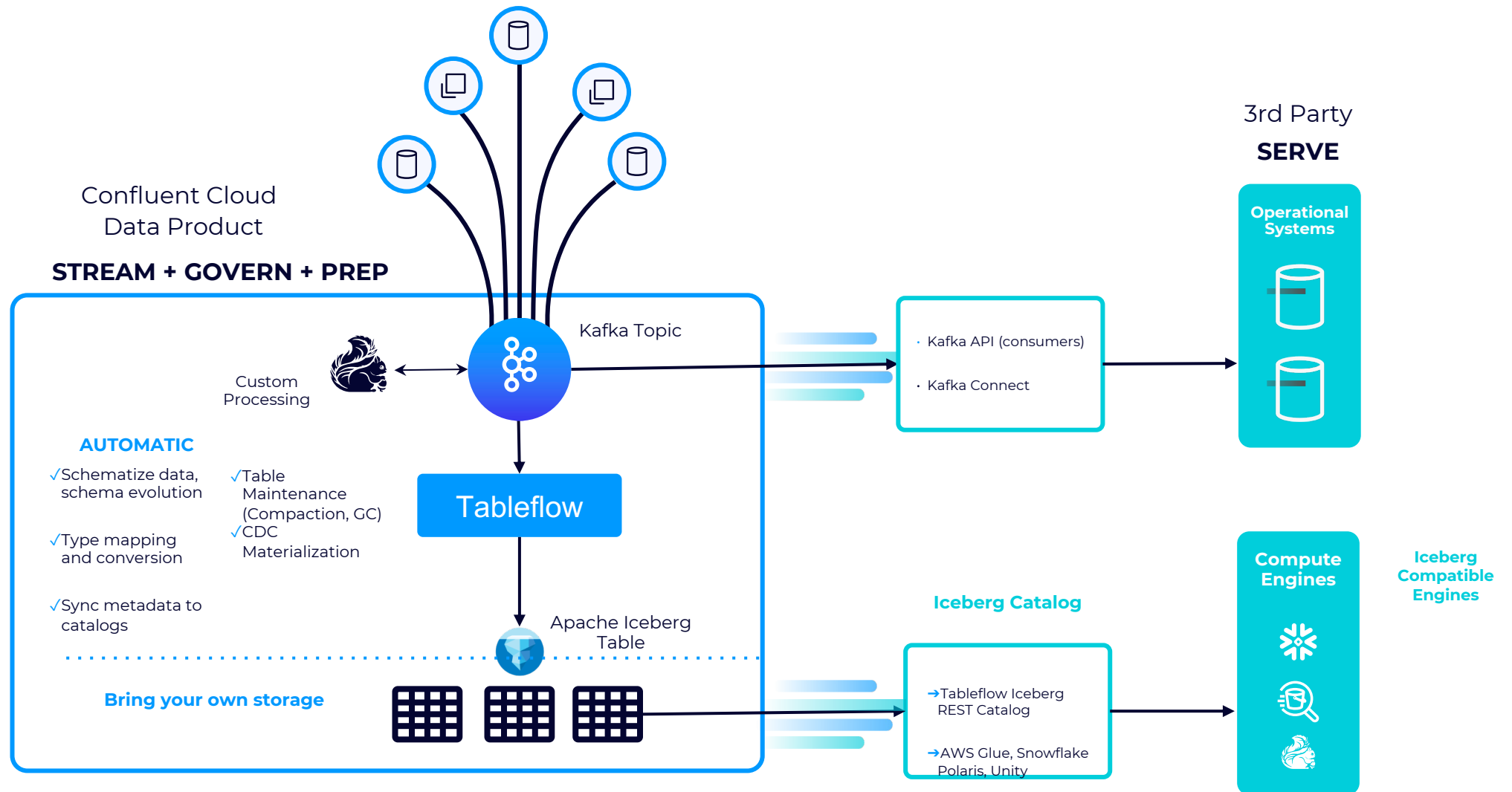
Streaming to data lakes is complicated



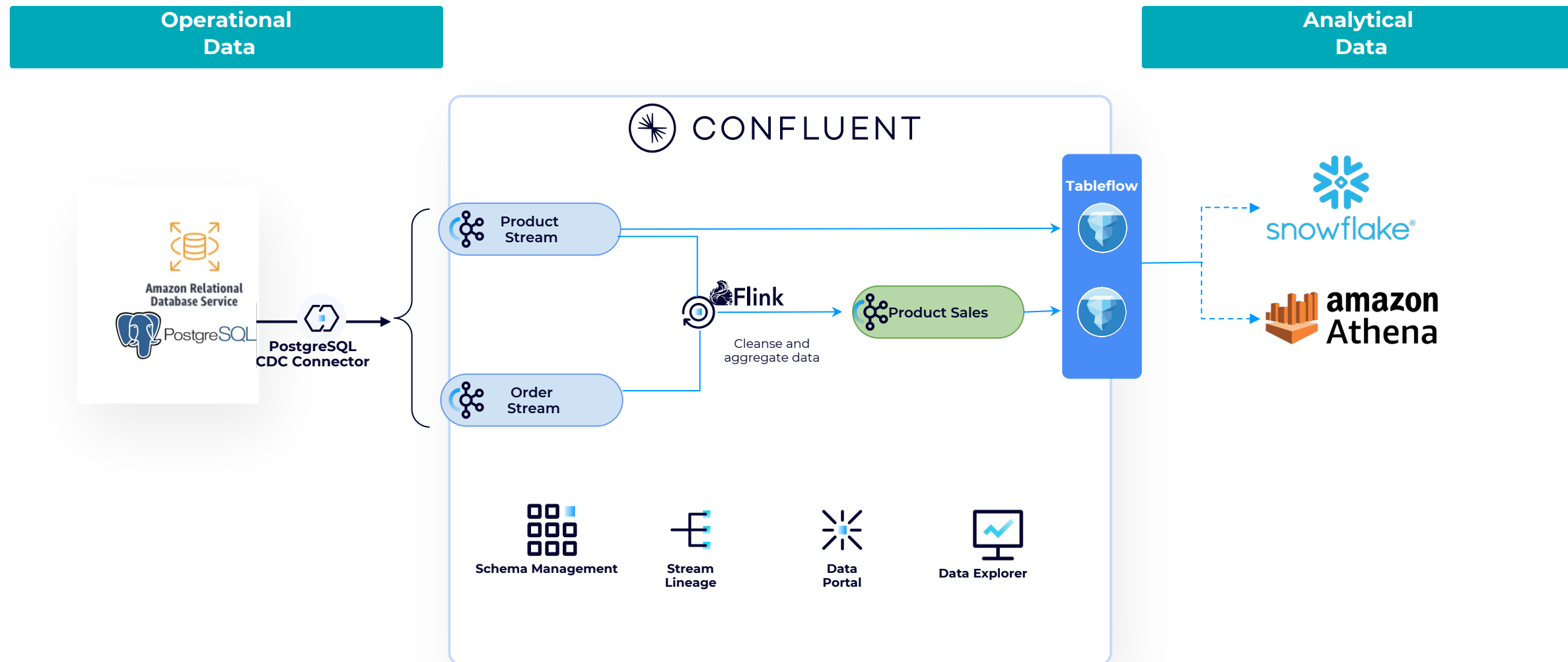
Streaming to data lakes is complicated



Tableflow materialize Kafka topics as Iceberg Tables



Demo: Tableflow to feed Kafka data to Data Lakes



Thank You!

**Rate this session
in the GOTO Guide app**

Help us make GOTO even better