**Case Study: Launching a CDC-based Delta Live Table Pipeline with Data Quality Enforcement and Slack Alerts**

**Use Case Overview**

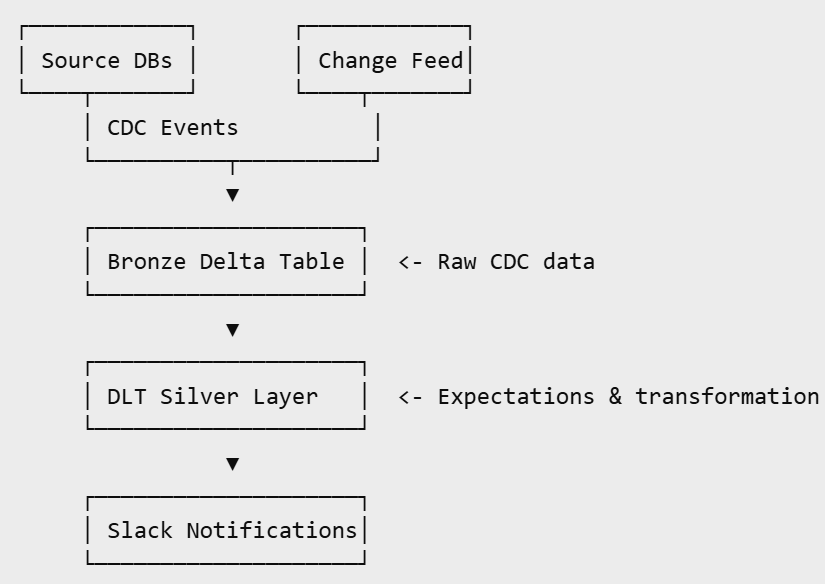
A retail enterprise collects customer order transactions from various regional databases. These databases emit **Change Data Capture (CDC)** events into a unified bronze table in Databricks. The organization wants to:

* **Ingest CDC events using Delta Live Tables (DLT)**
* **Apply at least three data quality expectations**
* **Trigger a Slack alert when the DLT pipeline fails due to expectation violations**

**Objectives**

1. Launch a DLT pipeline that ingests data from a CDC stream using APPLY CHANGES INTO.
2. Define ≥ 3 expectations on the source data for quality enforcement.
3. Automatically send Slack notifications when any expectation fails the pipeline execution.

**Architecture Overview**



**Step-by-Step Implementation**

**1. Create the Bronze CDC Table**

**Assume raw Debezium-like CDC data is arriving into a Delta table:**

sql

CREATE TABLE IF NOT EXISTS bronze.cdc\_orders (

order\_id STRING,

customer\_id STRING,

product\_id STRING,

quantity INT,

order\_status STRING,

last\_updated TIMESTAMP,

operation STRING

);

**2. DLT Configuration (YAML)**

yaml

name: cdc\_dlt\_orders

clusters:

- label: default

autoscale:

min\_workers: 1

max\_workers: 4

libraries:

- notebook: /Repos/user@org.com/dlt\_cdc\_orders\_pipeline

configuration:

env: prod

source\_table: bronze.cdc\_orders

development: false

**3. DLT Pipeline Code (Notebook)**

python

import dlt

from pyspark.sql.functions import col

# CDC Table

@dlt.table(

comment="Raw CDC stream from bronze layer."

)

def raw\_cdc\_orders():

return spark.readStream.table("bronze.cdc\_orders")

# Silver Table with Expectations

@dlt.table(

comment="Validated and deduplicated orders"

)

@dlt.expect("valid\_order\_id", "order\_id IS NOT NULL")

@dlt.expect("valid\_quantity", "quantity > 0")

@dlt.expect("valid\_status", "order\_status IN ('CONFIRMED', 'SHIPPED', 'CANCELLED')")

def silver\_orders():

return dlt.read\_stream("raw\_cdc\_orders") \

.filter("operation != 'delete'")

**4. CDC Merge Using APPLY CHANGES INTO**

sql

APPLY CHANGES INTO live.orders\_clean

FROM stream(LIVE.silver\_orders)

KEYS (order\_id)

SEQUENCE BY last\_updated

COLUMNS \* EXCEPT(operation) STORED AS SCD TYPE 1;

**5. Slack Alert Setup on Failure**

**A. Generate Slack Webhook URL**

* Go to Slack API Webhooks
* Create a new webhook for your channel (e.g., #pipeline-alerts)
* Save the generated webhook URL

**B. Create a Job to Send Slack Alerts**

python

import requests

def notify\_slack(pipeline\_name, error\_message):

webhook\_url = dbutils.secrets.get(scope="slack", key="webhook")

message = {

"text": f"🚨 \*DLT Pipeline Failed\*: `{pipeline\_name}`\n\n\*Error:\* {error\_message}"

}

requests.post(webhook\_url, json=message)

**C. Monitor event\_log for Failures**

You can periodically run a script or schedule a job to check:

sql

SELECT timestamp, details:error\_message

FROM delta.`/pipelines/<pipeline-id>/system/events`

WHERE event\_type = 'flow\_definition\_error' OR event\_type = 'flow\_progress' AND level = 'ERROR'

**Outcome**

* Data flows from CDC source to Silver table with schema validation.
* Three expectations (non-null ID, valid quantity, status whitelist) ensure trust in downstream tables.
* Failures trigger Slack alerts, enabling proactive response by the engineering team.