Notification Handling, avoid pitfalls

PoC created 15.05.2023 by Ihor Shcherbyna to improve existing notification handling

Introduction:

The entire system process **customer orders** (multiple orders per customer), and each order has a current state that is changed in time by the progress of handling particular order.

Order state-changed events come for processing via Kafka messages: **one message** contains **one event**.

Some customers can register their webhook callbacks (notification handlers) to receive order state-changed notifications (**one per customer**).

Notification handling is the process of delivering notifications about each change event through a registered webhook to a particular customer.

Preconditions:

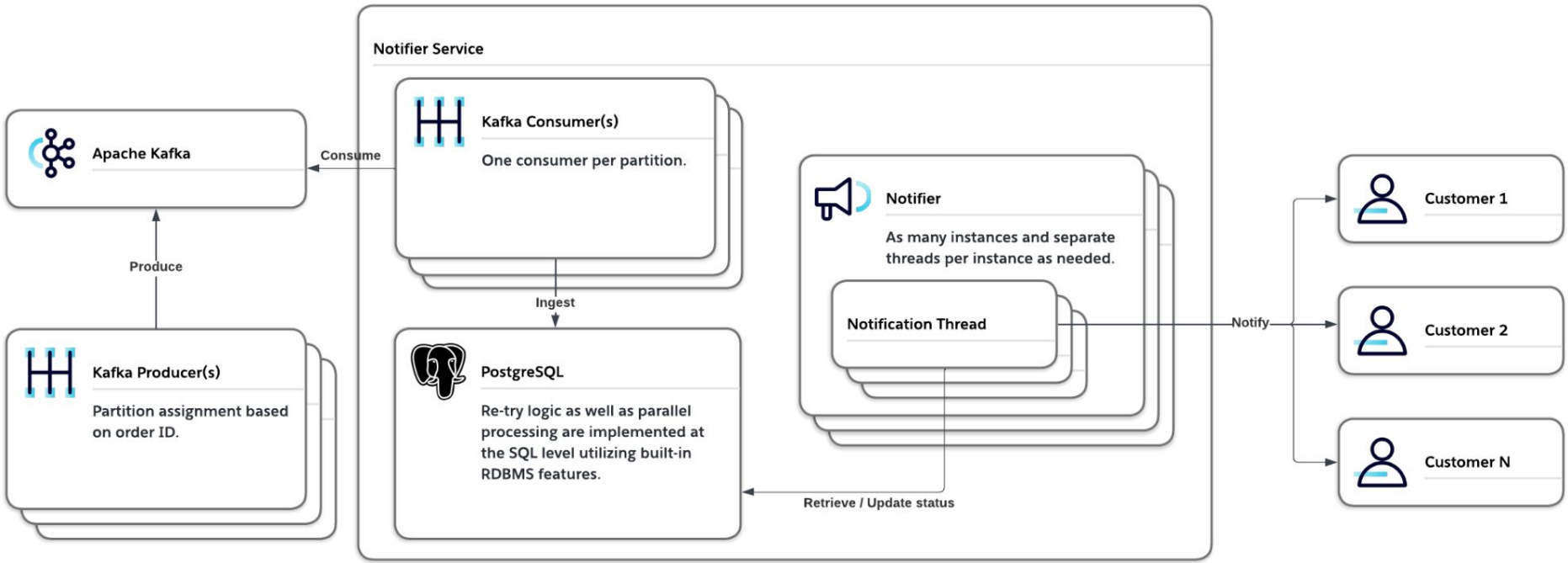
The following issues were identified:

- no order guarantee - logic is based only on that events are distributed over time;
- one slow customer webhook, slows down the entire notification delivery;
- lack of retries - there is no logic to resend event notifications if the initial send attempt fails to deliver;
- limited scalability - it is hard to scale by design.

Requirements:

- **Ordering**: order state-changed notifications within one order should be delivered in the same order as they were triggered.
- **Unaffected performance:** potential performance degradation of one or more notification handlers must not affect the performance of the rest notification handlers
- **Retried delivery**: order state-changed notifications that failed to deliver must be attempted to re-deliver with respect to the retry policy.
- **Horizontal scalability:** should mitigate (reduce) delivery latency for order state-changed notifications.

Solution Diagram

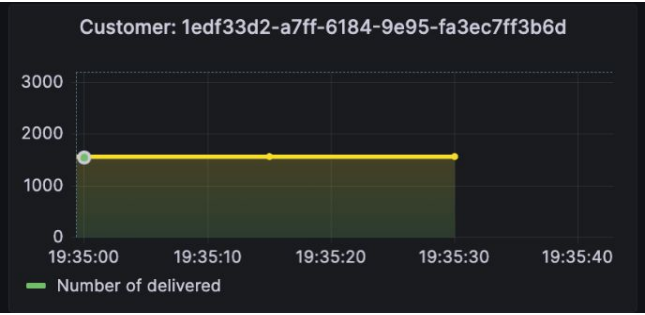


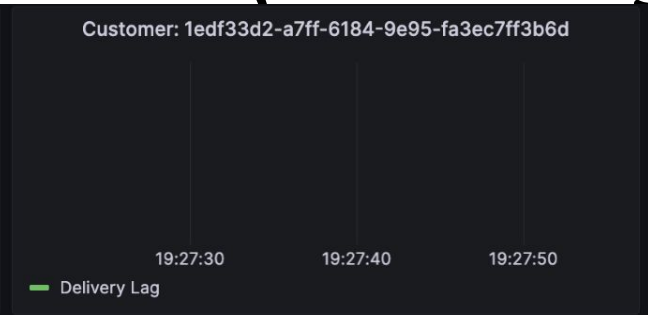
Key Points:

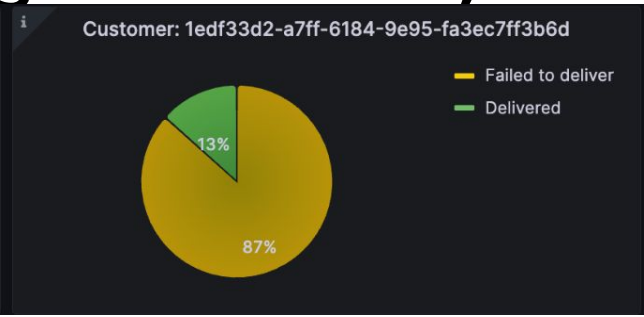
- based on existing TechStack;
- relatively low solution complexity by utilizing built-in Kafka and PostgreSQL features;
- fulfill requirements;
- flexible retry policy: easy to change, can be modified to any duration or number of attempts, even up to data eviction;
- fully scalable;
- simplified troubleshooting: full state persisted in the Database;

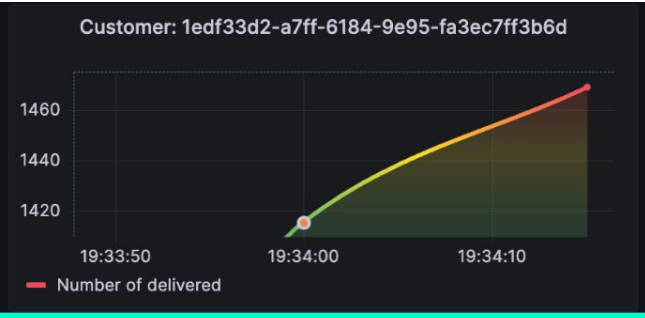
Unaffected Performance (2nd customer “freezes”):

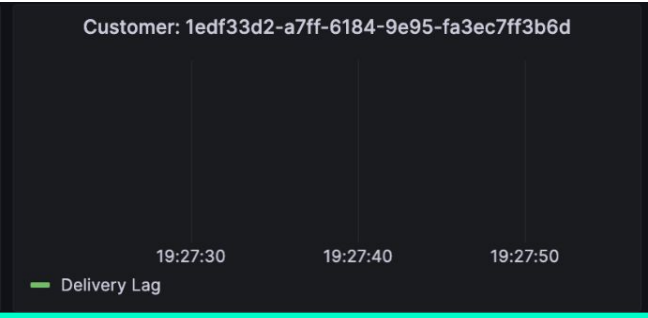
Unaffected Performance (state changes over time):

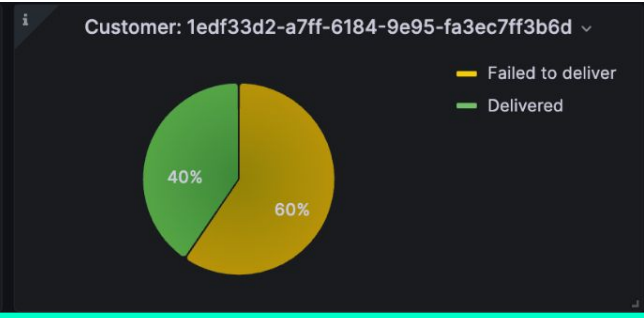


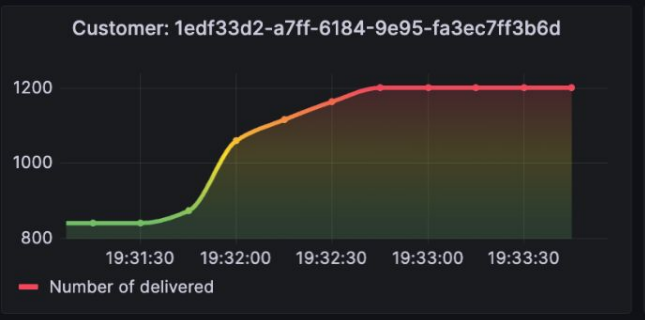


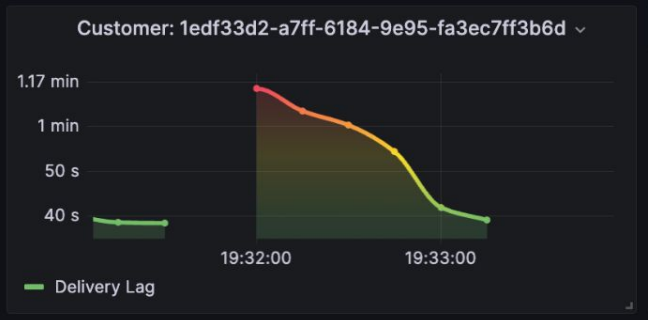


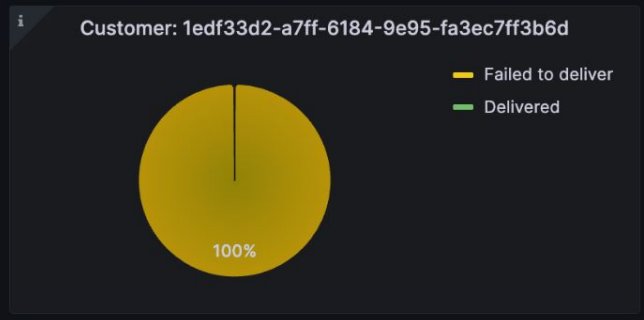




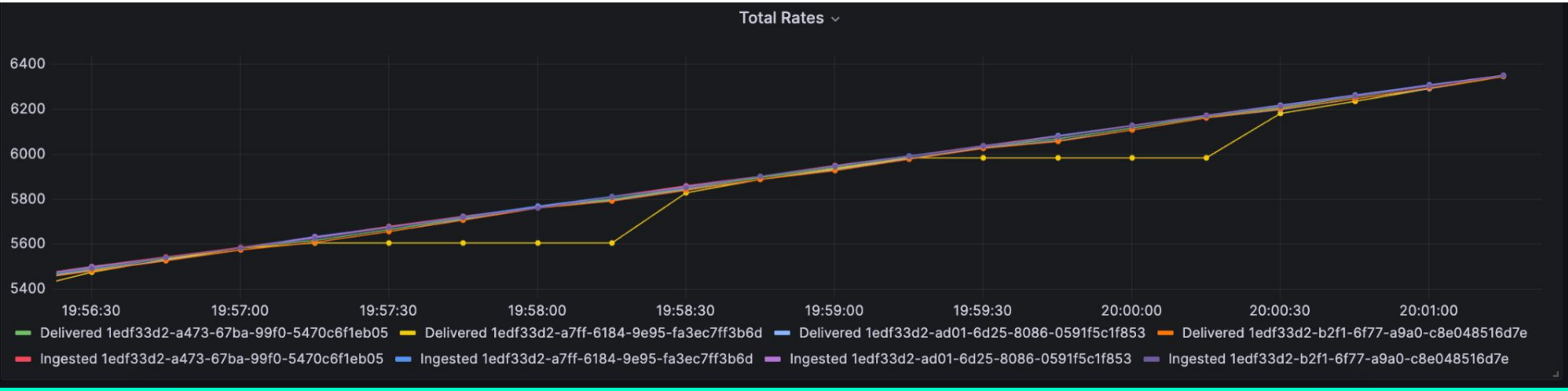








Unaffected Performance (Total Rates for customers):



Disclaimer:

- **Sensitive data** was **not used** during development and **is kept confidential.**
- All development was done **outside of working hours.**
- This project is just a **Proof of concept (POC).**