Real-Time Data Processing Pipeline

Bala Dutt, Charu Garg, Prabhat Hegde

Stage	Status	Update date	Remarks
Requirements	Done	08/Feb/2020	
Design	WIP	08/Feb/2020	
Implementation	Not yet started		
Submitted	No		

Background

According to a recent report by IBM Marketing cloud, "*90 percent of the data in the world today has been created in the last two years alone, 2.5 quintillion bytes daily — and with new devices, sensors and technologies, the data growth rate will likely accelerate even more*".

By data, we mean, high, "time-value" data. "Time Value of Data" – the notion that the value of information to the business drops quickly after it is created or the value of data decays with time. Here, data value effectively drops to zero over a very short period of time. Following are three example use-cases for 'perishable insights',

1. Fraud detection – detect frauds as they happen and block them or prevent them

   Ex. An OAuth use-case that saw a dictionary attack at a low rate to keep it under the radar. Also, 1:n attacks where n users login from one IP address or 1 user logins from n IP addresses.

   Balbix, a startup, uses more than 20 ML algorithms to assess risk across more than 200 attack vectors. Capsule8, another startup, uses BigQueryML, distributed analytics, etc. for this. LogZilla boasts of 108 billion events searches in 1s and 855,000 events per second to solve for NetOps and SecOps. While SecOps includes intrusion detection, infrastructure security.

   Our focus is application and user-level security, to prevent account takeovers (ATOs) and money fraud. In the fraud domain, it is also necessary to respond quickly to new situations. So, the system should enable new algorithms to be easily hooked in.

2. Moving large data

   Large amounts of data need to be moved from the OLTP system to end-user facing reporting system. For example, accounting products may have OLTP applications to store transactions for end-users, but when users are interested in Business Insights or reports like Profit and Loss or Balance sheet, they would use another app, which needs to have data in sync at low latency.

3. Anomaly detection systems

   Here the system's metrics are collected and monitored at runtime to detect anomalies like bugs, changing user patterns, rogue applications or misbehaving components.

Scope

Many use-cases for stream data processing exist in Intuit. Data in legacy applications are locked in monoliths, each with their data-store. As the move towards Services gains steam, real-time streaming is the only way to reduce time to decision-making using data that is spread across these products, while also unburdening the monoliths.

Quickbooks Online (a.k.a QBO) is Intuit's flagship product for Small Businesses. It is a SAS product with >3 million customers worldwide. It is largely a monolithic application with Oracle as RDS. The scope of our work is to move data across two microservices, each with its data store in near real-time, using open-source or Intuit inner-source components.

Use Cases

Our target use-cases, in order, are:

1. As a developer, I need a configurable data-movement solution, with flexibility to optimise the platform for specific business use-cases, like Throughput Vs Reliability.
2. As a developer, I need to be able to perform transformations on data, like computing aggregate Invoice amounts over a time-window, like a day.
3. As a SRE, I need to be able to shorten the Risk-Screening cycle, detecting anomalies (in Transactions) in real-time and enabling decisions for OLTP systems.
4.

Capabilities

The use-cases above translate to the following capabilities in the solution.

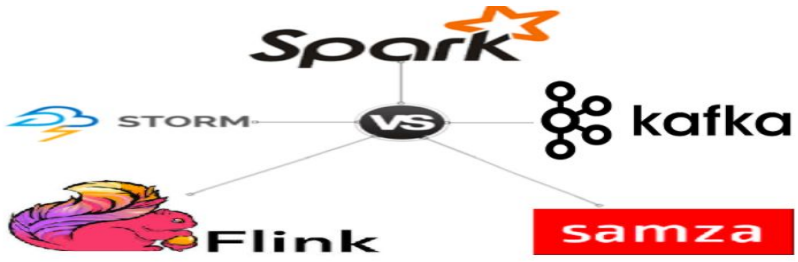
	Use-case	**Capability**
1	Stateless Transformer: Move transactional (OLTP) data from one service to another	Scale: TPS, Data Freshness: 6 sec Data size: 10 GBps
2	Simple Stateful Transformer: Support basic stateful data transformations during stream processing (For ex. Aggregates, Counting)	Transformations, ex. windowing aggregations
3	Stateful Transformer: Complex processing	Joins across Streams, Aggregation of relational data, Anomaly detection

Architectural Principles

1. Extensibility/Integratibility
    a. Developers have SSI models, they are not exposed to the internals
2. Non-Functional
    a. Latency and Throughput, Latency over throughput
3. Composable guarantees and Flexibility
    a. Resilience, at-least-once semantics, transactions, consistency all are add-ons to the superfast/super-scalable core

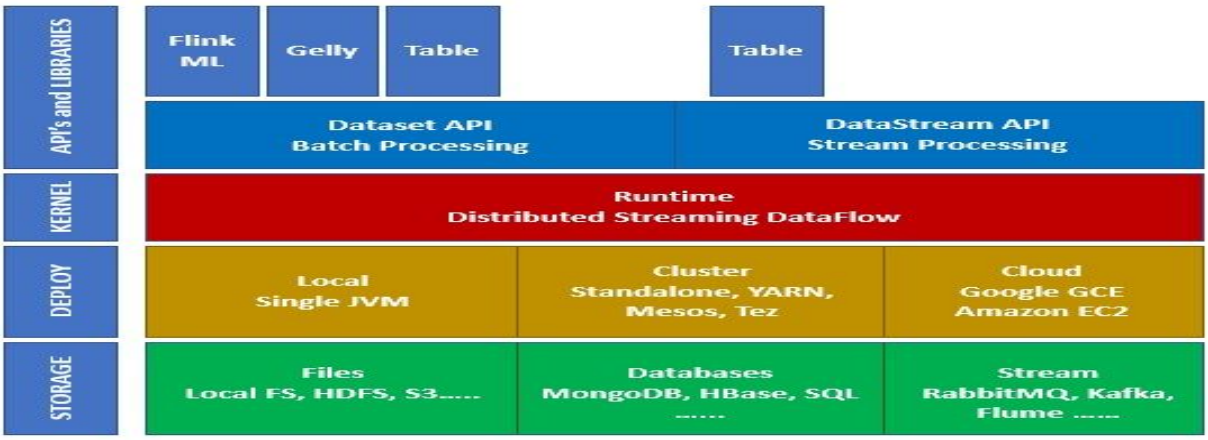
Frameworks Investigated

Stream processing definition - a data processing engine designed with infinite data sets in mind.



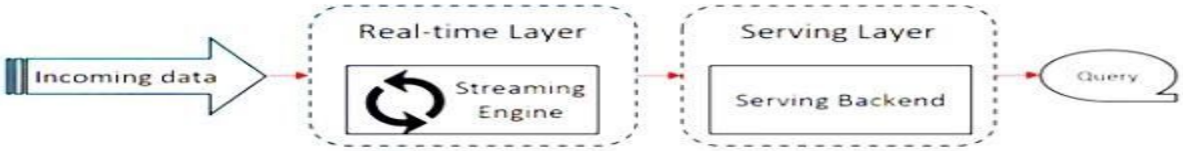
Evaluation Criteria and Findings (link)

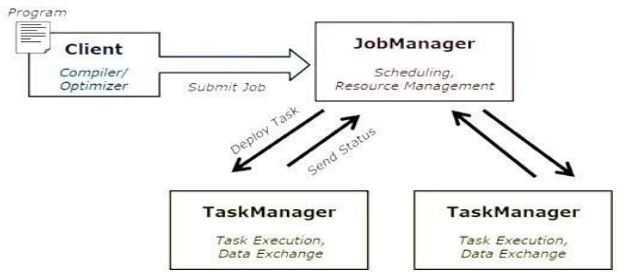
Final Tech Stack choice - Flink



Apache Flink works on Kappa architecture. Kappa architecture has a single processor stream, which treats all input as stream and the streaming engine processes the data in real-time. Batch data in this architecture is a special case of streaming.

The following diagram shows Apache Flink Architecture. The key idea in Kappa architecture is to handle both batch and real-time data through a single stream processing engine.



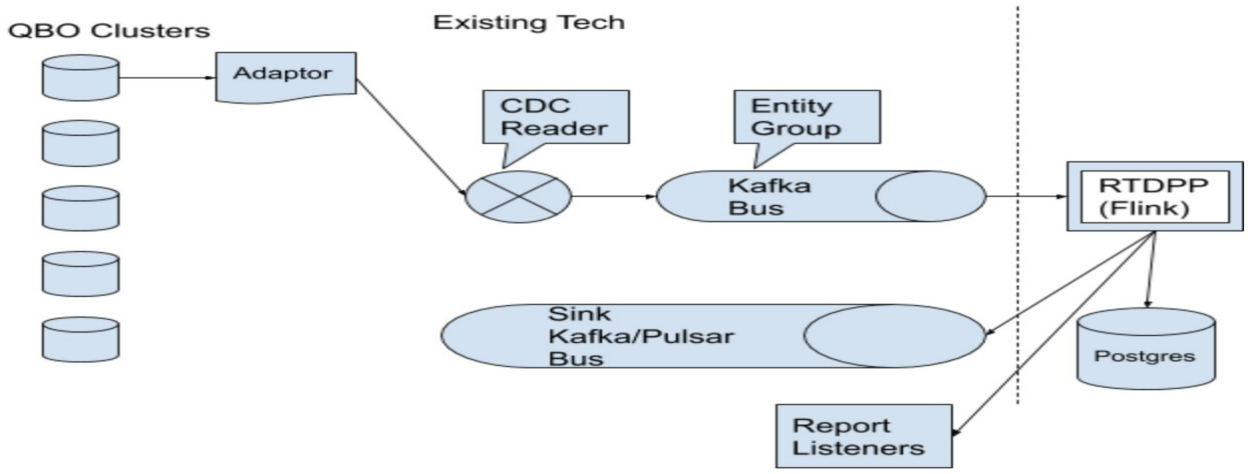


The diagram above shows Apache Flink's job execution architecture. Most big data frameworks work on Lambda architecture, which has separate processors (code) for batch and streaming data. For querying and obtaining results, the codebases need to be merged, which is a pain. Kappa architecture solves this issue as it has only one view − real-time, hence merging of codebase is not needed.

This doesn't mean Kappa architecture replaces Lambda. The use-case and the application decide the choice of architecture.

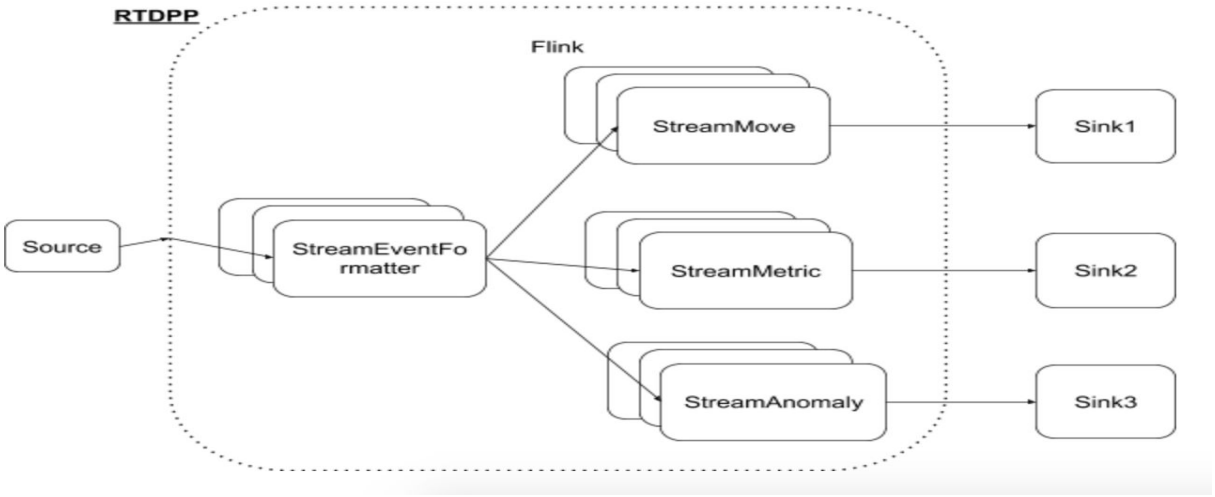
Design

Below is the explosion of the Flink based system that we are building. Rest of the systems have been replaced by source and sink. At a high-level, we are moving data from QBO Monolith which uses Oracle as RDS to Aurora Postgres for use by Business Decisions Service, which is a standalone service to serve Reporting and Business Insights use-cases.



The system has three Flink stream transformers, as below. The incoming event stream will be split into three and passed to these parallel stream transformers.

1. StreamMove - Move data - stateless transformer
2. StreamMetric - Computer Metrics - stateful transformer
3. StreamAnomaly - Detection of anomalies - stateful transformer



For our project, real-time data movement will be our only focus and not Sharding, or efficient schema and other technical problems which are also problem areas in this solution space.

Assumptions

Event processing requires Bootstrap (ingestion of data for the first time), which typically happens once a day. In a target state, such bootstrap would happen at the point of new signup (new company creation). For our system to work, current data must be moved, before streaming data can be used.

Event format

Format of the event would be a list of attributes (key/values a.k.a tags). Some attributes will be mandatory. Below is the list,

ID of event

Source of event

TenantId (CompanyId)

Type of Entity

Entity Id

Other attributes

e.g. Invoice amount

A sample raw event is,

{"header": {"source": "AWS-US-EAST", "seqno": 14590086726641, "fragno": 14590090486121, "schema": "QBO_DATA", "serverid": 1, "table": "TXDETAILS_1", "timestamp": 1581498219001, "eventtype": "UPDATE", "shardid": "QBO_CDC_E2E", "eventid": "000000145900904861 21", "query": "", "primarykey": "COMPANY_ID,TX_ID,SEQUENCE", "eventno": 1, "changedcolumns": "CREATE_USER_ID"}, "payload": {"beforerecord": {"COMPANY_ID": "4620816365395475810", "TX_ID": "4", "SEQUENCE": "0", "LINE_ORDER": null, "TX_TYPE_ID": null, "TX_DATE": null, "ACCOUNT_ID": null, "CUSTOMER_ID": null, "VENDOR_ID": null, "EMPLOYEE_ID": null, "DEPT_ID": null, "PAYMENT_METHOD_ID": null, "KLASS_ID": null, "MEMO_TEXT": null, "ITEM_ID": null, "PITEM_ID": null, "WAGE_BASE": null, "YTD_AMT": null, "YTD_HOURS": null, "INCOME_SUBJECT_TO_TAX": null, "OVERRIDE_AMT": null, "OVERRIDE_WAGE_BASE": null, "OVERRIDE_ISTT": null, "HOURS": null, "STATE": null, "LIVE_STATE": null, "LIVE_JURISDICTION_ID": null, "WORK_JURISDICTION_ID": null, "AS_OF_DATE": null, "PTO_PAY_OUT": null, "IS_PAYROLL_LIABILITY": null, "IS_PAYROLL_SUMMARY": null, "PAYROLL_LIABILITY_DATE": null, "PAYROLL_LIAB_BEGIN_DATE": null, "QUANTITY": null, "RATE": null, "AMOUNT": null, "SPLIT_PERCENT": null, "DOC_NUM": null, "IS_PURCHASE": null, "IS_SALE": null, "IS_CUSTOMER_PAYMENT": null,

"IS_PAYMENT_TO_VENDOR": null, "IS_BILL": null, "IS_PROFITIBILITY_EXPENSE": null, "IS_PROFITIBILITY_INCOME": null, "IS_CLEARED": null, "IS_DEPOSITED": null, "IS_DEPOSITED2": null, "DEPOSIT_ID": null, "IS_NO_POST": null, "TAXABLE_TYPE": null, "IS_AR_PAID": null, "IS_AP_PAID": null, "OPEN_BALANCE": null, "PAYROLL_OPEN_BALANCE": null, "IS_STATEMENTED": null, "IS_INVOICED": null, "STATEMENT_ID": null, "INVOICE_ID": null, "STATEMENT_DATE": null, "INVOICE_DATE": null, "DUE_DATE": null, "OTHER_ACCOUNT_ID": null, "OTHER_KLASS_ID": null, "IS_BILLABLE": null, "REIMB_TXN_ID": null, "MARKUP": null, "SERVICE_DATE": null, "SALES_DETAIL_TYPE": null, "SOURCE_TXN_ID": null, "SOURCE_TXN_SEQUENCE": null, "PAID_DATE": null, "OFX_TXN_ID": null, "OFX_MATCH_FLAG": null, "OLB_MATCH_MODE": null, "OLB_MATCH_AMOUNT": null, "OLB_RULE_ID": null, "DD_ACCOUNT_ID": null, "DD_LINE_STATUS": null, "INVENTORY_COSTING_FOR_SEQ": null, "CREATE_DATE": null, "CREATE_USER_ID": "100", "LAST_MODIFY_DATE": null, "LAST_MODIFY_USER_ID": null, "EDIT_SEQUENCE": null, "ADDED_AUDIT_ID": null, "AUDIT_ID": null, "AUDIT_FLAG": null, "EXCEPTION_FLAG": null, "IS_PENALTY": null, "IS_INTEREST": null, "NET_AMOUNT": null, "TAX_AMOUNT": null, "TAX_CODE_ID": null, "TAX_RATE_ID": null, "CURRENCY_TYPE": null, "EXCHANGE_RATE": null, "HOME_AMOUNT": null, "HOME_OPEN_BALANCE": null, "IS_FOREX_GAIN_LOSS": null, "SPECIAL_TAX_TYPE": null, "SPECIAL_TAX_OPEN_BALANCE": null, "TAX_OVERRIDE_DELTA_AMOUNT": null, "INCLUSIVE_AMOUNT": null, "CUSTOM_ACCOUNT_TAX_AMT": null, "JOURNAL_CODE_ID": null, "DISCOUNT_ID": null, "DISCOUNT_AMOUNT": null, "TXN_DISCOUNT_AMOUNT": null, "EXTERNAL_TXN_MATCH_MODE": null, "SUBTOTAL_AMOUNT": null, "LINE_DETAIL_TYPE": null, "WITHHOLDING_RATE_ID": null, "REMAINING_QUANTITY": null, "REMAINING_AMOUNT": null, "SRC_QTY_USED": null, "SRC_AMT_USED": null, "LINE_MANUALLY_CLOSED": null, "PROGRESS_TRACKING_TYPE": null, "ITEM_RATE_TYPE": null, "CUSTOM_FIELD_VALS": null, "REGION_CUSTOMS_CODE": null, "LAST_MODIFIED_UTC": null}, "afterrecord": {"COMPANY_ID": "4620816365395475810", "TX_ID": "4", "SEQUENCE": "0", "LINE_ORDER": null, "TX_TYPE_ID": null, "TX_DATE": null, "ACCOUNT_ID": null, "CUSTOMER_ID": null, "VENDOR_ID": null, "EMPLOYEE_ID": null, "DEPT_ID": null, "PAYMENT_METHOD_ID": null, "KLASS_ID": null, "MEMO_TEXT": null, "ITEM_ID": null, "PITEM_ID": null, "WAGE_BASE": null, "YTD_AMT": null, "YTD_HOURS": null, "INCOME_SUBJECT_TO_TAX": null, "OVERRIDE_AMT": null, "OVERRIDE_WAGE_BASE": null, "OVERRIDE_ISTT": null, "HOURS": null, "STATE": null, "LIVE_STATE": null, "LIVE_JURISDICTION_ID": null, "WORK_JURISDICTION_ID": null, "AS_OF_DATE": null, "PTO_PAY_OUT": null, "IS_PAYROLL_LIABILITY": null, "IS_PAYROLL_SUMMARY": null, "PAYROLL_LIABILITY_DATE": null, "PAYROLL_LIAB_BEGIN_DATE": null, "QUANTITY": null, "RATE": null, "AMOUNT": null, "SPLIT_PERCENT": null, "DOC_NUM": null, "IS_PURCHASE": null, "IS_SALE": null, "IS_CUSTOMER_PAYMENT": null, "IS_PAYMENT_TO_VENDOR": null, "IS_BILL": null, "IS_PROFITIBILITY_EXPENSE": null, "IS_PROFITIBILITY_INCOME": null, "IS_CLEARED": null, "IS_DEPOSITED": null, "IS_DEPOSITED2": null, "DEPOSIT_ID": null, "IS_NO_POST": null, "TAXABLE_TYPE": null, "IS_AR_PAID": null, "IS_AP_PAID": null, "OPEN_BALANCE": null, "PAYROLL_OPEN_BALANCE": null, "IS_STATEMENTED": null, "IS_INVOICED": null, "STATEMENT_ID": null, "INVOICE_ID": null, "STATEMENT_DATE": null, "INVOICE_DATE": null, "DUE_DATE": null, "OTHER_ACCOUNT_ID": null, "OTHER_KLASS_ID": null, "IS_BILLABLE": null, "REIMB_TXN_ID": null, "MARKUP": null, "SERVICE_DATE": null, "SALES_DETAIL_TYPE": null, "SOURCE_TXN_ID": null, "SOURCE_TXN_SEQUENCE": null, "PAID_DATE": null, "OFX_TXN_ID": null, "OFX_MATCH_FLAG": null, "OLB_MATCH_MODE": null, "OLB_MATCH_AMOUNT": null, "OLB_RULE_ID": null, "DD_ACCOUNT_ID": null, "DD_LINE_STATUS": null, "INVENTORY_COSTING_FOR_SEQ": null, "CREATE_DATE": null, "CREATE_USER_ID": "9130349800290446", "LAST_MODIFY_DATE": null, "LAST_MODIFY_USER_ID":

null, "EDIT_SEQUENCE": null, "ADDED_AUDIT_ID": null, "AUDIT_ID": null, "AUDIT_FLAG": null, "EXCEPTION_FLAG": null, "IS_PENALTY": null, "IS_INTEREST": null, "NET_AMOUNT": null, "TAX_AMOUNT": null, "TAX_CODE_ID": null, "TAX_RATE_ID": null, "CURRENCY_TYPE": null, "EXCHANGE_RATE": null, "HOME_AMOUNT": null, "HOME_OPEN_BALANCE": null, "IS_FOREX_GAIN_LOSS": null, "SPECIAL_TAX_TYPE": null, "SPECIAL_TAX_OPEN_BALANCE": null, "TAX_OVERRIDE_DELTA_AMOUNT": null, "INCLUSIVE_AMOUNT": null, "CUSTOM_ACCOUNT_TAX_AMT": null, "JOURNAL_CODE_ID": null, "DISCOUNT_ID": null, "DISCOUNT_AMOUNT": null, "TXN_DISCOUNT_AMOUNT": null, "EXTERNAL_TXN_MATCH_MODE": null, "SUBTOTAL_AMOUNT": null, "LINE_DETAIL_TYPE": null, "WITHHOLDING_RATE_ID": null, "REMAINING_QUANTITY": null, "REMAINING_AMOUNT": null, "SRC_QTY_USED": null, "SRC_AMT_USED": null, "LINE_MANUALLY_CLOSED": null, "PROGRESS_TRACKING_TYPE": null, "ITEM_RATE_TYPE": null, "CUSTOM_FIELD_VALS": null, "REGION_CUSTOMS_CODE": null, "LAST_MODIFIED_UTC": null}}}

This event will first be converted to the format we described. Metric and Anomaly event will key based on TenantId and Type of Entity.

As mentioned, this system works only when bootstrap for a company has happened. Metric and Anomaly transformers would start working from the first event.

Scale

The system will scale on two dimensions - tenant and entity type.

Information on Event Producer (QBO)

What	Description
Data-source characteristics	Total DB Size: 70tb #QBO Clusters (Shards): 45 Size of the largest Table (TxDetails): xxTB, xxBillion Rows, x% index, y% data Top3 Tables: TxDetails, TxHeaders, Cashbasis Max number of columns in a table: 60
Test Size	Sample size: 55000 (QBO Advanced cohort of companies)
Throughput	35-40 million events/weekday 15 million events/ weekend Peak - 600 events/sec Low - 100 events/sec
Use-cases	Analytical queries which join dimension with fact tables to build Reports, Insights, etc Add new columns to these fact tables Create indexes on fact tables

Other Design Considerations

Throughput

It takes 20-30 ms to persist a typical event payload into Postgres (persistent store configured for Flink). This requires us to calibrate the number of hosts needed to scale. In a target state pods (Kubernates) could be imagined. For the purposes of our project, our laptop would be used to demonstrate date movement.

We will measure the number of events the system is capable of processing per second.

Sharding:

In our source, 3 million companies in Oracle are split across 45 Shards based on CompanyID. When this moves into our target service (Postgres), sharding is a consideration. Sample approach to consider is to Shard by Company creation date, Size of Company and such. For the purpose of this exercise Sharding is out of scope as it is a topic in itself.

Anomaly Detection:

Anomalies in our system may be of various kinds. Some anomalies are when transactions are placed into wrong Accounting categories. Values for fields in the event payload (ex. IS_INVOICED, IS*) can help to detect anomalies.

In some cases DOS attacks can be detected using metrics. Ex. A case where a Sample Company running large reports at a high per-second rate. There have been instances where malicious users have used multiple trial companies to generate high load on the systems. Such cases can be detected by StreamMetric stateful transformer (by counting Report runs per user in a given time window)

Expected setup

We will use Intuit QBO pre-production environments as source and QBO Kafka Event Producer for event sourcing. Flink will be setup on each of our laptops, configured with Postgres as a persistent store and consuming events from QBO Kafka event source. Data movement from QBO to Postgres will be shown.

Open Issues:

On the producing end, CDC pushes events to different topics, ex: Txheaders topic, TxDetails topic, etc. What we need is a way to consume events across the Kafka topics. Related events are grouped together in a single topic.

- Transactions topic (TxDetails, TxHeaders, CashBasis, Links)

- Accounts (Account_1, AccountInternals_1)
- Tax (Taxcodes, taxrates, taxcoderates, agency, authority, etc)

Each event is an Insert/Delete/Update as seen by a database. These events are preceded by a "start" event which marks the start of a DB transaction and succeeded by an "end" event which marks the end of a DB transaction.

Consumers must handle committing events atomically and referential integrity for facts.

Options to handle events atomically include buffering events in memory until the "end" event is received and persisting to DB from the list held in memory. DB transactions may also be used where a DB transaction is opened upon a "start" event, followed by execution of DMLs for each event until an “end” event upon which a DB commit occurs.

Handling referential integrity for facts. Ex - We get an invoice event but the corresponding customer is not present.

Appendix - know streaming scale in production

Link