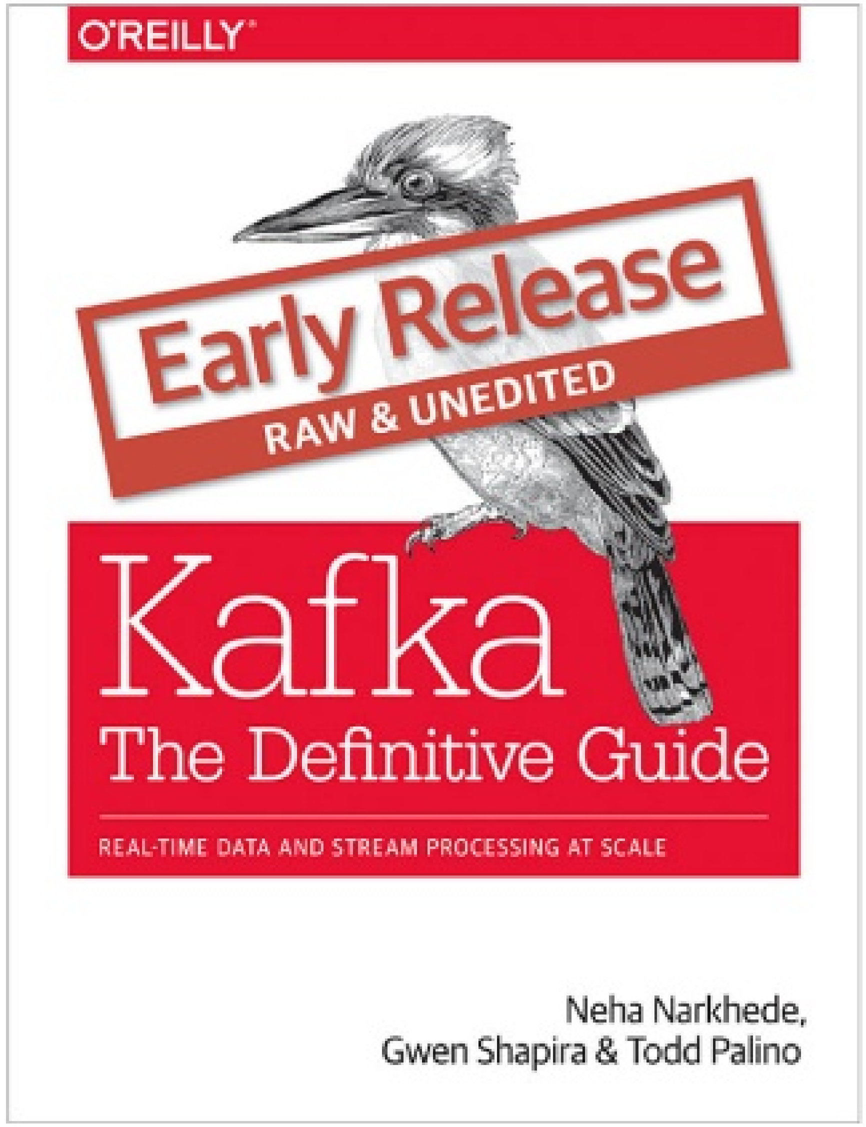
Stream All Things

Patterns of Modern Data Integration

About Me

- Working @confluent
- Wrote a book or two
- Product manager
- Apache Kafka PMC member
- Used to be an engineer, consultant, DBA
- @gwenshap
- Github.com/gwenshap



Evolution of Data Integration

Data Warehouse

- Difficult modeling
- Difficult ingest
- Missing data
- Limited throughput
- Batch only

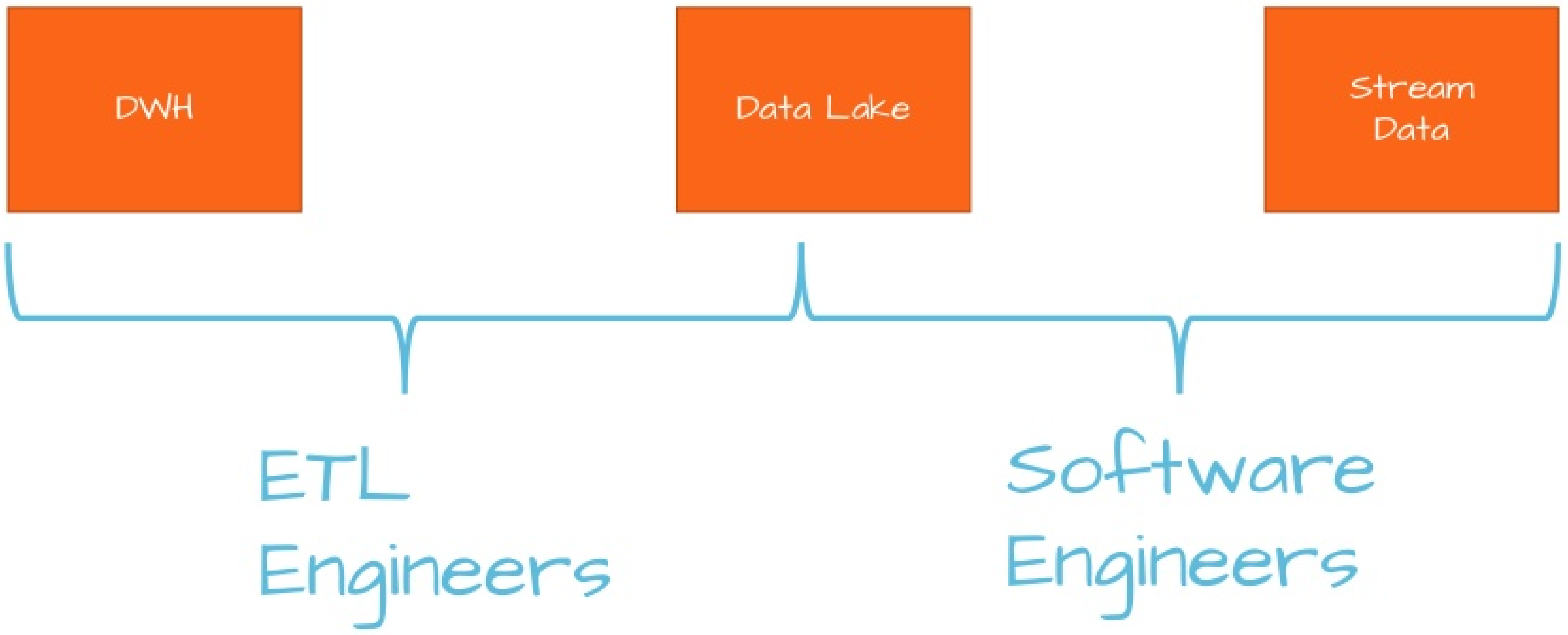
Data Lake

- Still batch only
- Data quality is suspect
- Data model left as exercise for each developer
- Where is my data?

Data Stream

- Kept the scale
- Reduced latency
- Totally new model
- Data producers are 1st class participants
- Microservices? Apps?
- Still figuring the whole thing out

Evolution of Data Integration



Evolution of IT

2000	2017
Software engineers	
Data modelers	
DBAs	
Sysadmins	
Storage admins	Software engineers
Network admins	
ETL engineers	
QA	
....	

The line between application development and ETL is blurring.

BOLD CLAIM:

ALL YOUR DATA

IS

EVENT STREAMS

Few Patterns

1. Stream all things (in one place)

2. Keep Compatible and Process On

3. Ridiculously Parallel Data Integration

4. Streaming Data Enrichment

Example: Large Hotel Chain

BOLD CLAIM:

ALL YOUR DATA

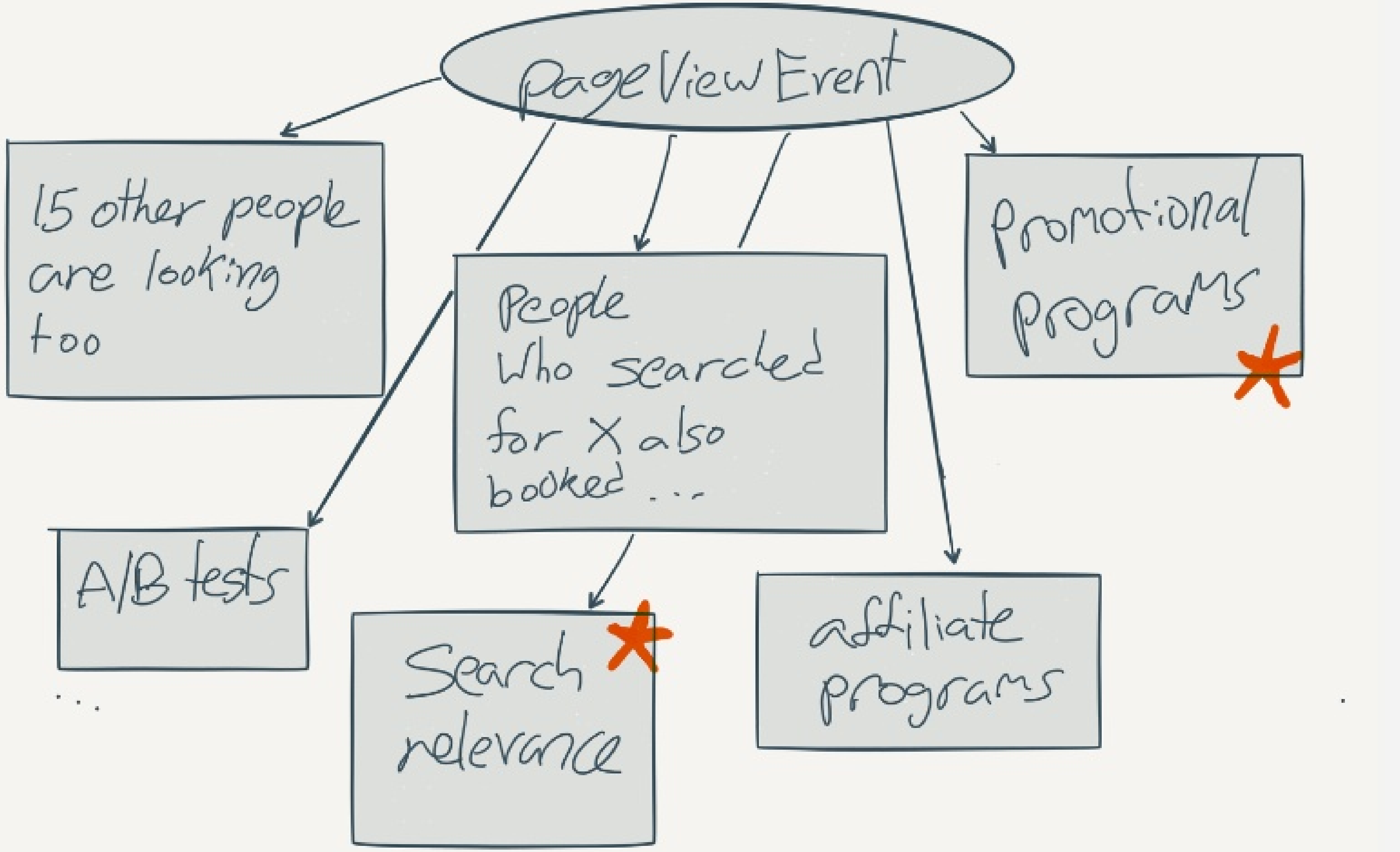
IS

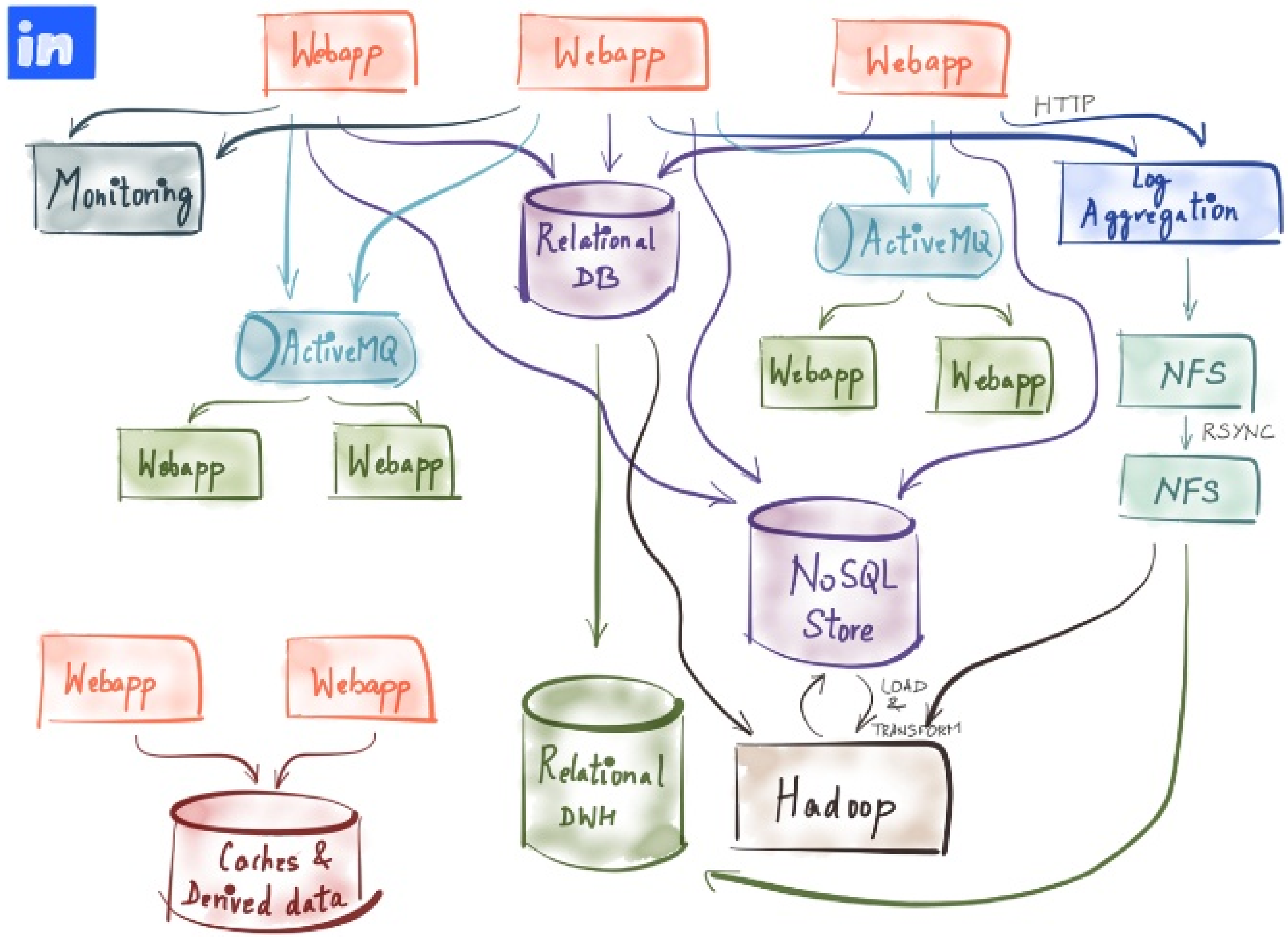
EVENT STREAMS

pageViewEvent

PageViewEvent

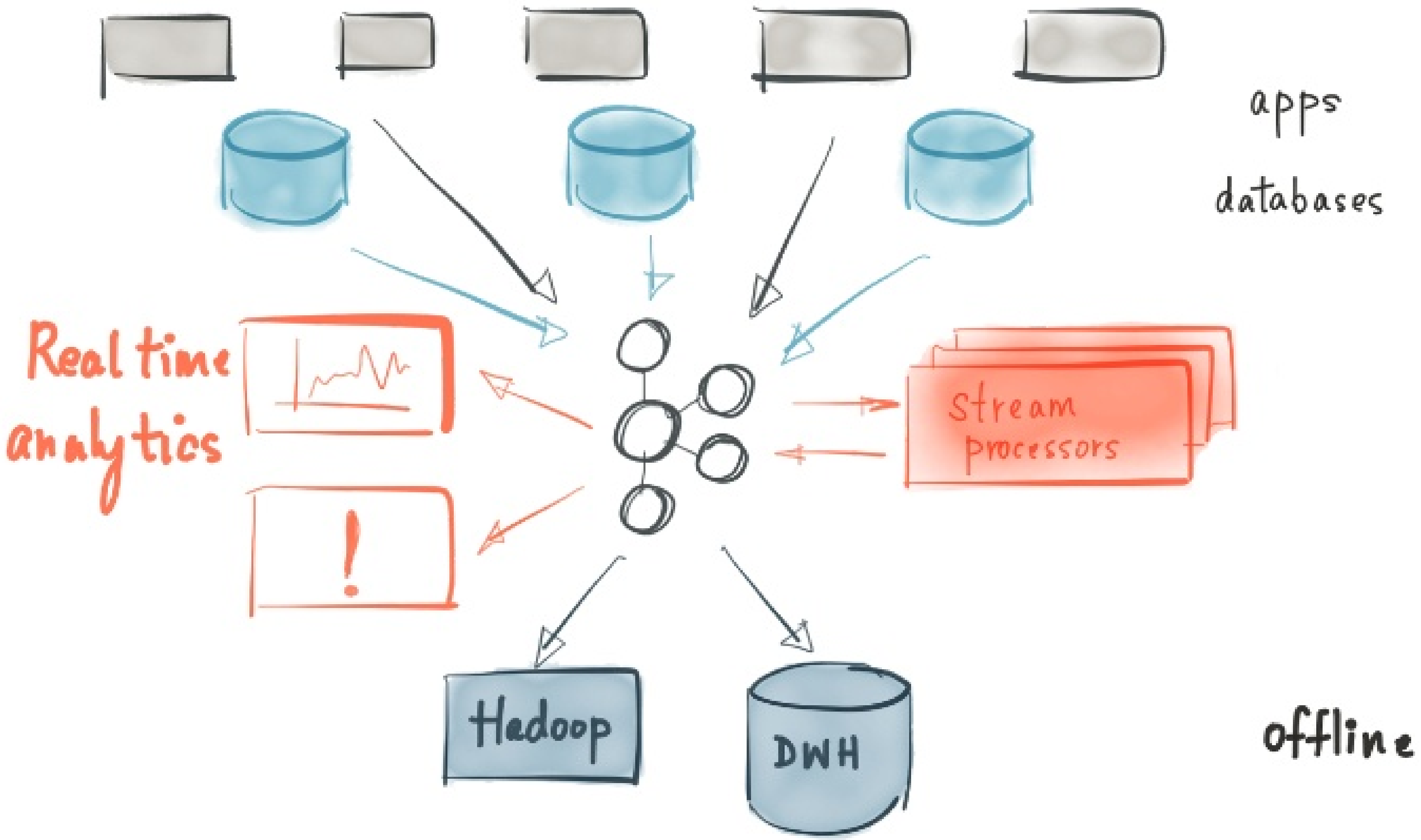
```
{
  sessionId: 676fc8983gu563,
  timestamp: 1413215458,
  viewType: "propertyView",
  propertyId: 7879,
  loyaltyId: 6764532
  origin: "promotion",
  ...... lots of metadata....
}
```


pageViewEvent
15 other people are looking too
People who searched for X also booked ...
promotional programs
A/B tests
...
Search relevance
affiliate programs

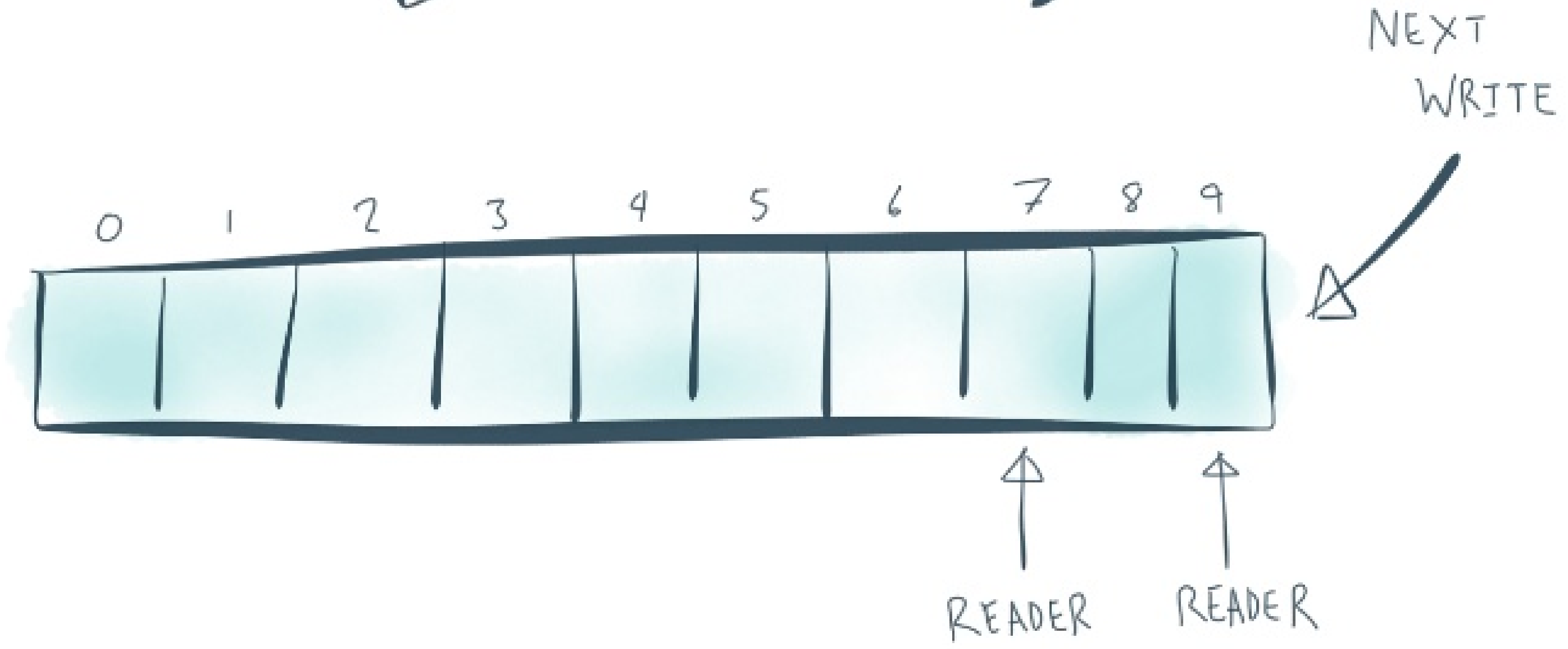

Webapp
Webapp
Webapp
HTTP
Monitoring
Relational DB
ActiveMQ
Log Aggregation
ActiveMQ
Webapp
Webapp
NFS
RSYNC
NFS
Webapp
Webapp
NoSQL Store
LOAD & TRANSFORM
Webapp
Webapp
Relational DWH
Hadoop
Caches & Derived data

confluent

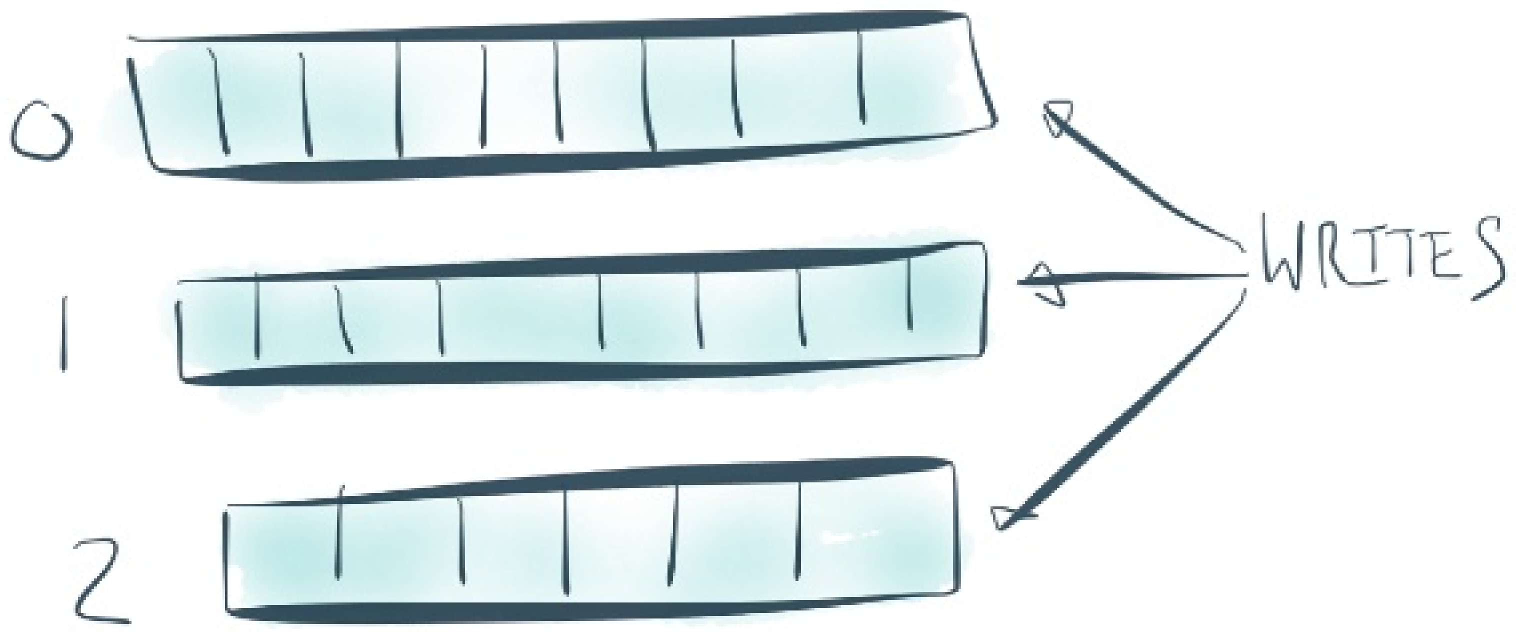
Pattern #1 – Put all streams in One Kafka



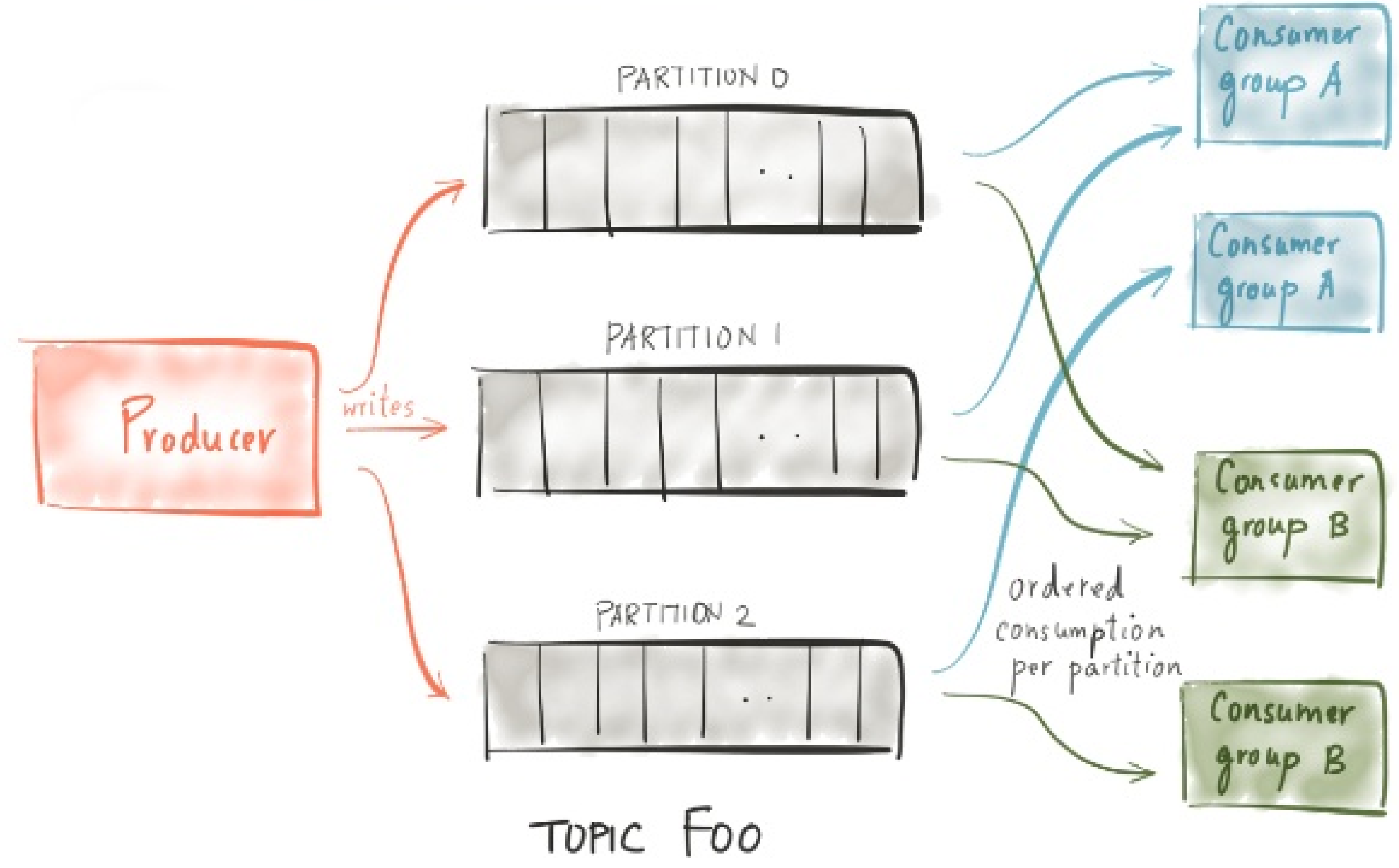
LOGS

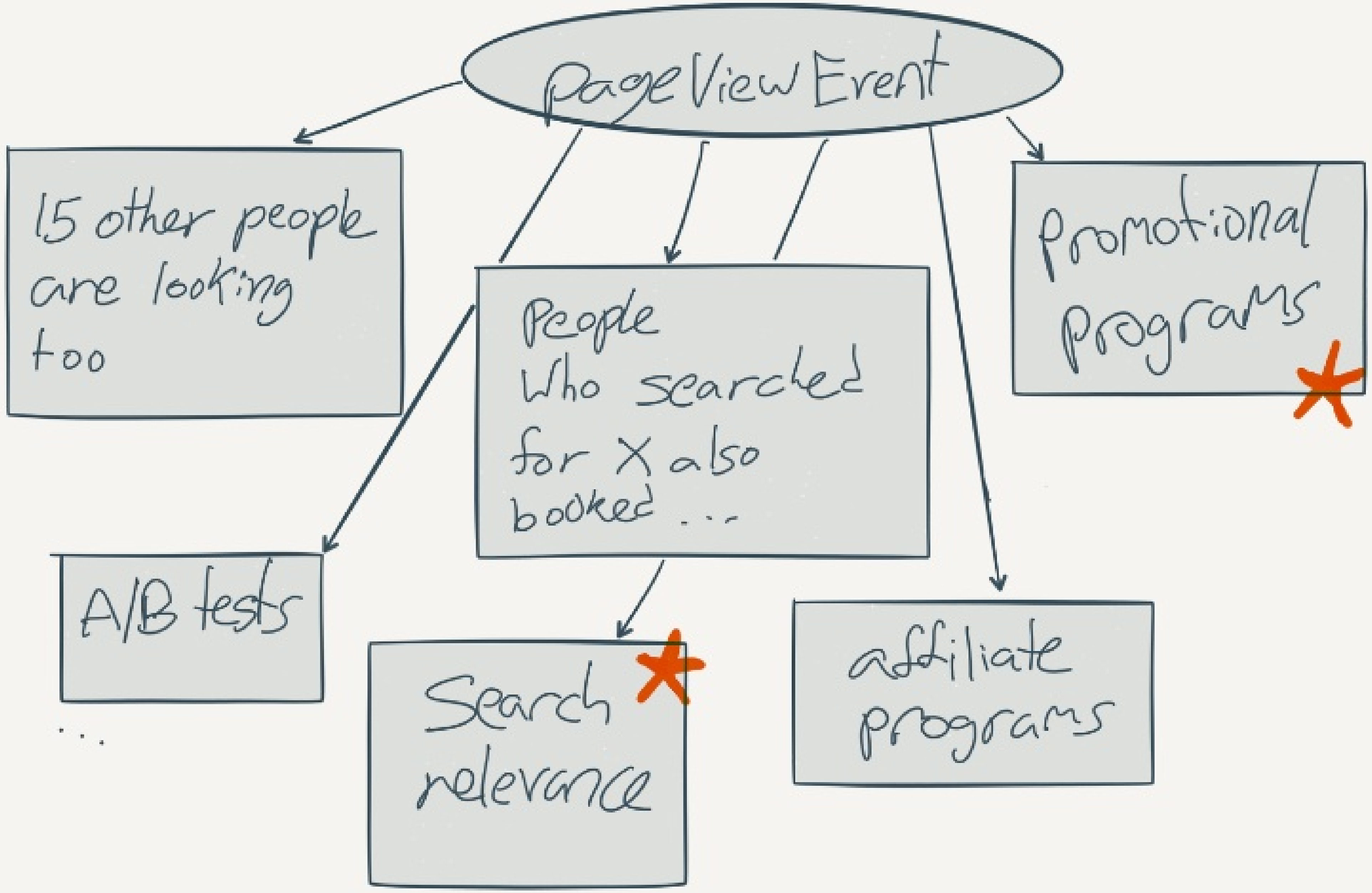


KAFKA TOPIC = PARTITIONED LOG



Scalable Consumption



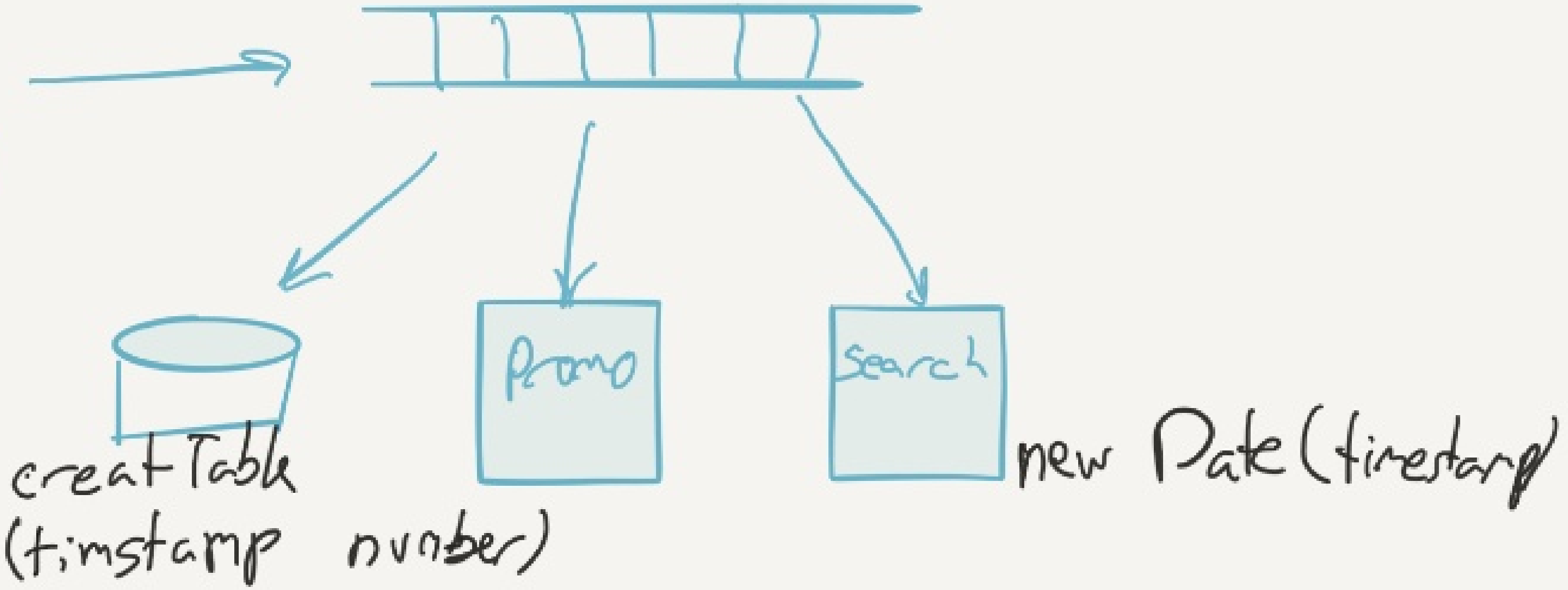

PageViewEvent
15 other people are looking too
People who searched for X also booked ...
Promotional programs
A/B tests
Search relevance
affiliate programs
...

Common Story

PageView

```
{
  timestamp: 1415532168

  ....  }
```



Common Story

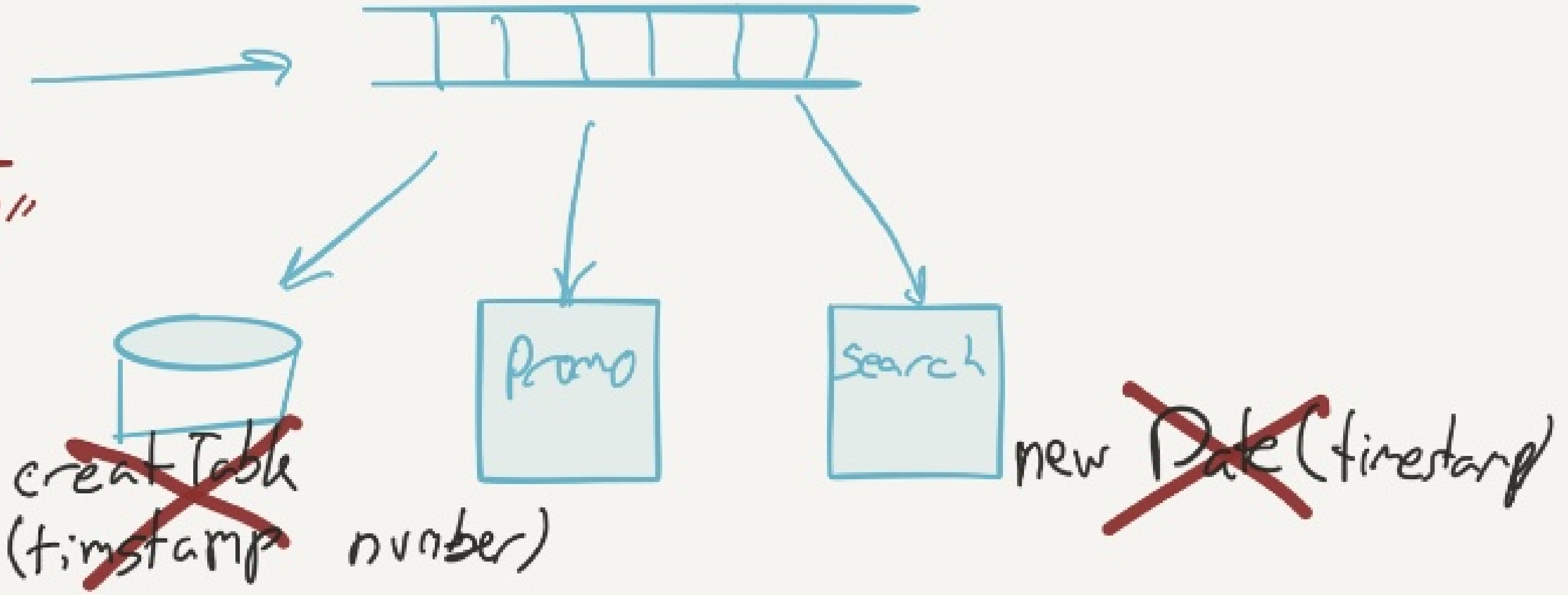
PageView

{

timestamp: ~~1415532118~~

"Jan 20, 2017"

. . . . }



Pattern #2

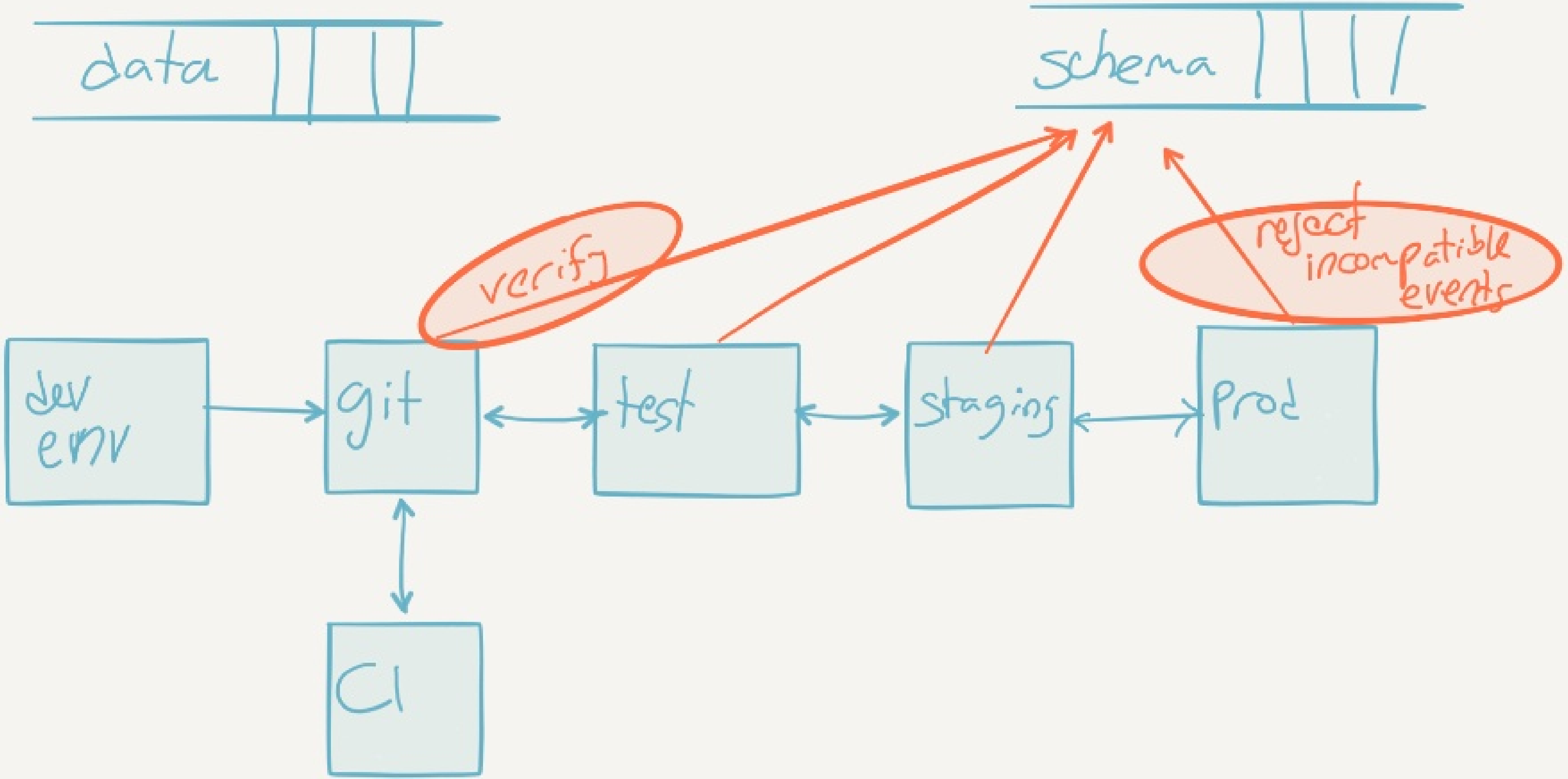
KEEP
COMPATIBLE
AND
PROCESS
ON

APIs between services are Contracts

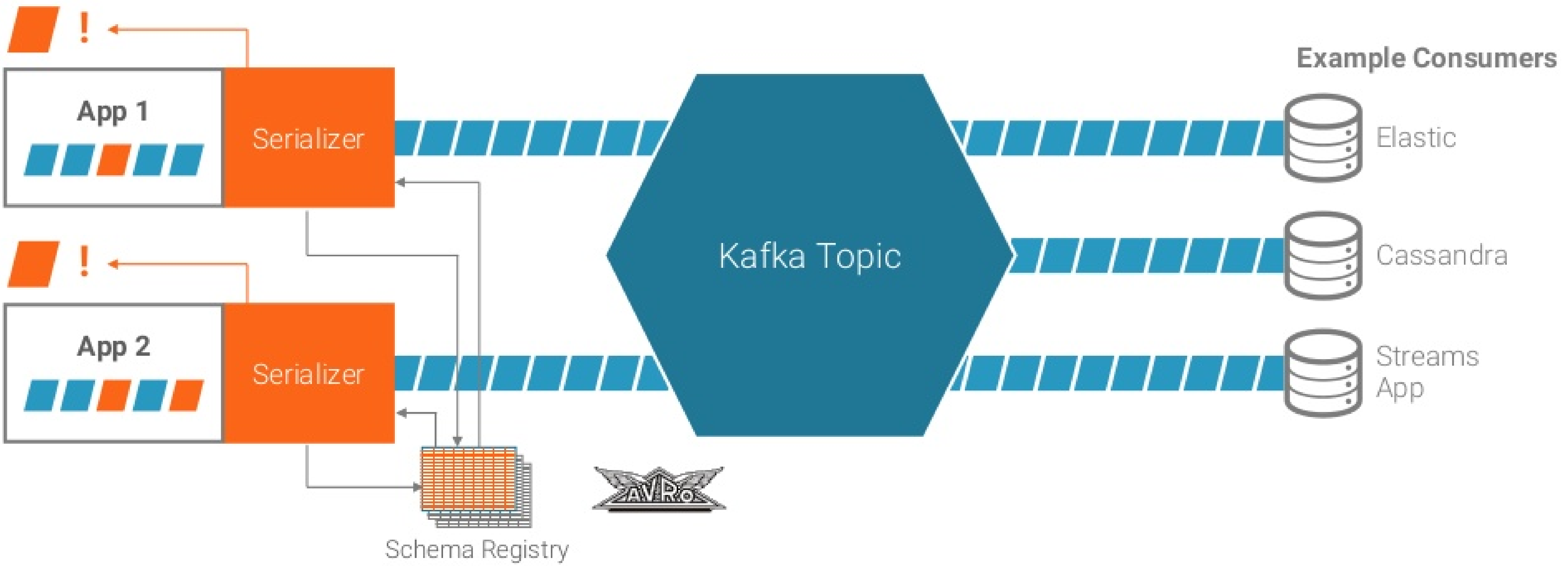
In Stream Processing World – Event Schemas ARE the API

…except they stick around for a lot longer

keep events compatible



Schema Registry



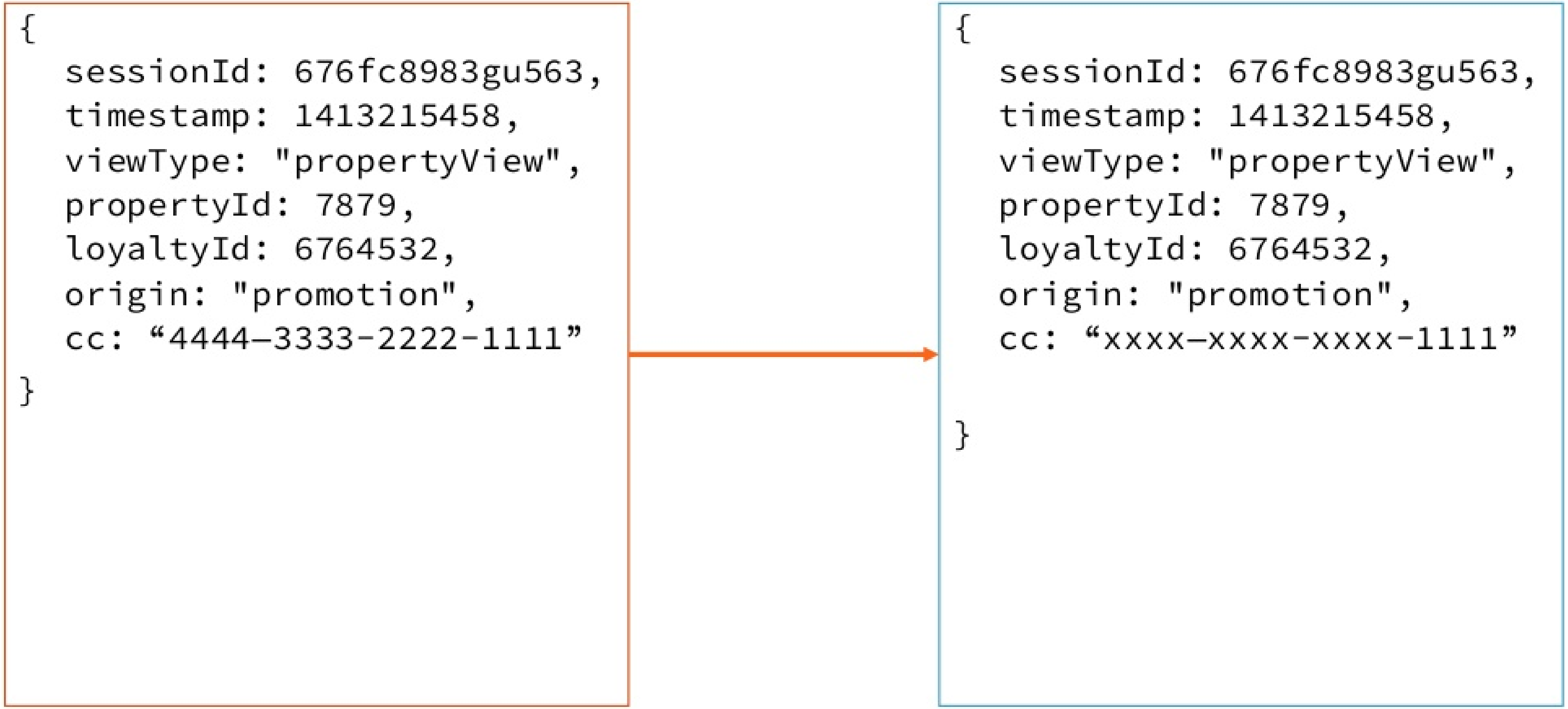
Define the expected fields for each Kafka topic

Verify Schema Compatibility at every stage of dev cycle

Automatically **handle** schema changes (e.g. new fields)

Supports multi-datacenter environments

Pattern #3 – Ridiculously parallel data integration



```
{
  sessionId: 676fc8983gu563,
  timestamp: 1413215458,
  viewType: "propertyView",
  propertyId: 7879,
  loyaltyId: 6764532,
  origin: "promotion",
  cc: “4444–3333-2222-1111”
}
```

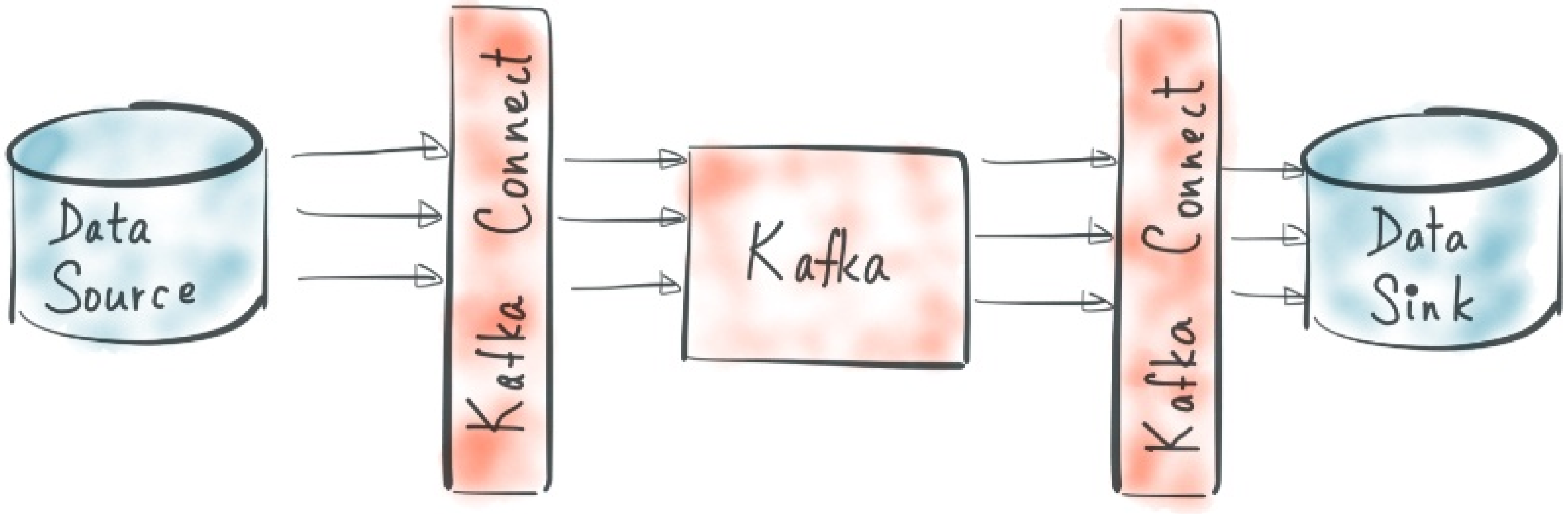
```
{
  sessionId: 676fc8983gu563,
  timestamp: 1413215458,
  viewType: "propertyView",
  propertyId: 7879,
  loyaltyId: 6764532,
  origin: "promotion",
  cc: “xxxx–xxxx-xxxx-1111”

}
```

confluent

Hipster Stream Processing

Is there a data store involved? KafkaConnect can help



Ecosystem of Connectors

Databases

JDBC
mongoDB
DATASTAX
Cassandra
ORACLE GOLDEN GATE
Just One Data
Couchbase
amazon DynamoDB
InfluxDB
Kudu
APACHE HBASE
RethinkDB

Datastore/File Store

hadoop
apache Ignite
FTP
syslog-ng
hazelcast

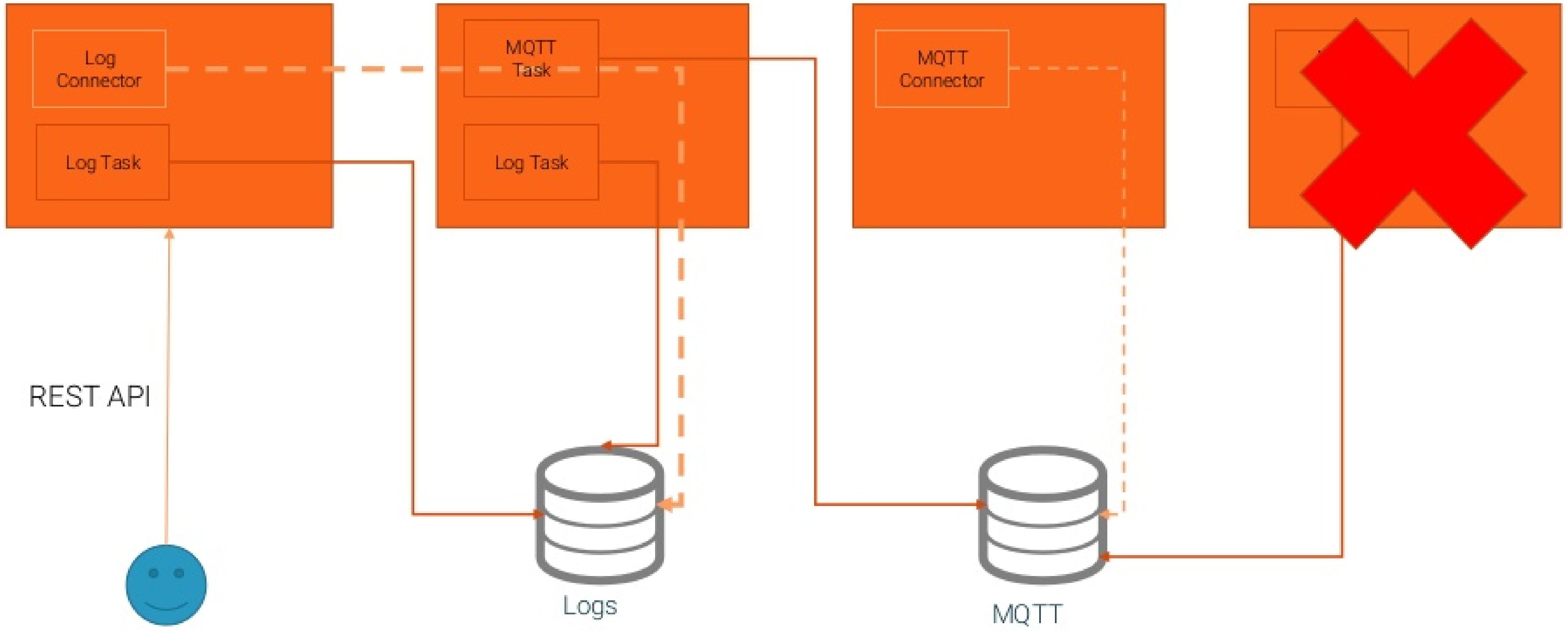
Analytics

VERTICA An HP Company
elastic
mixpanel
Solr
APACHE KUDU

Applications / Other

amazon web services S3
Bloomberg
syncsort
twitter
ATTUNITY
MQTT.ORG

How Connect Works?



Connect + Single Message Transformations

Mix:

Ridiculously parallel database integration

With

Ridiculously parallel simple transformations

confluent

Promotion!

We want to send Platinum members

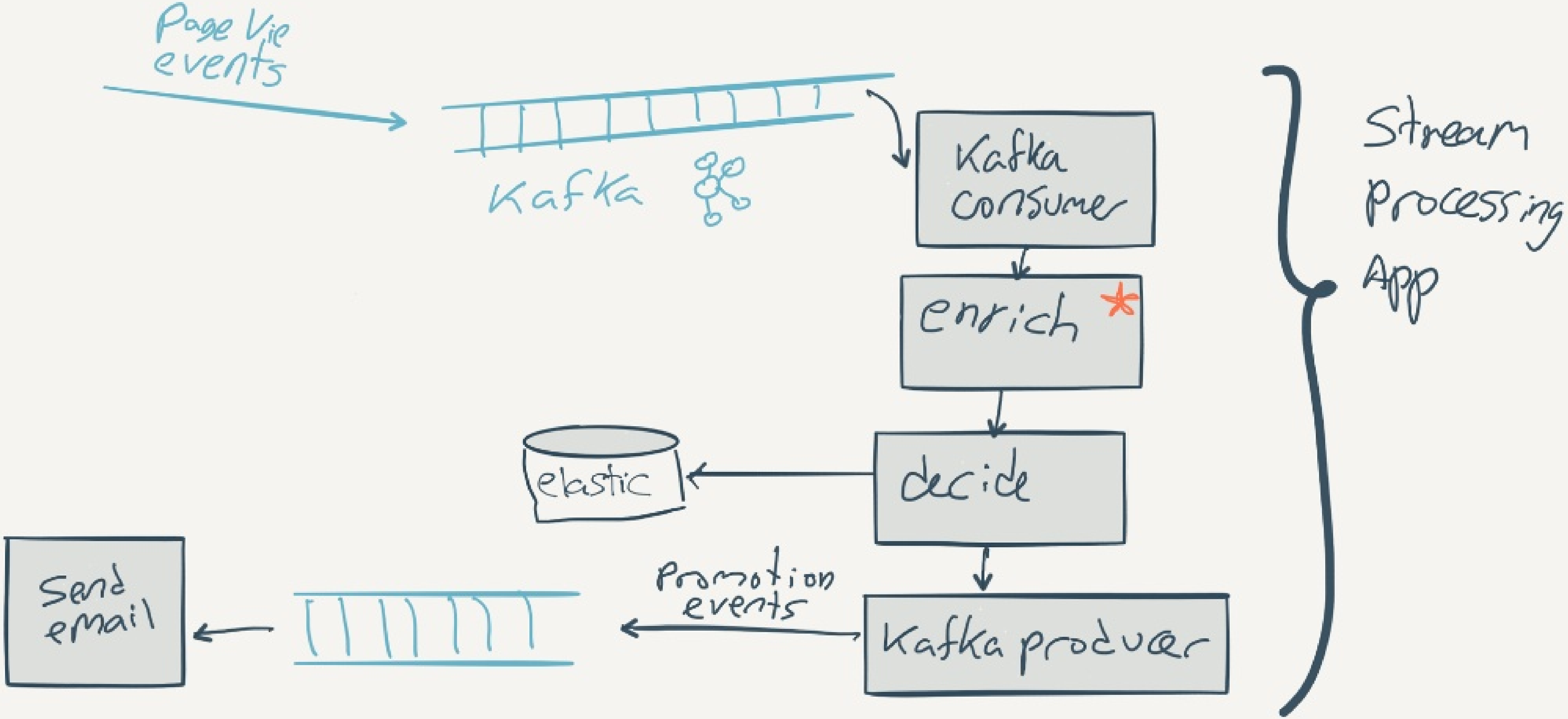
Who are looking at beach properties

An email about

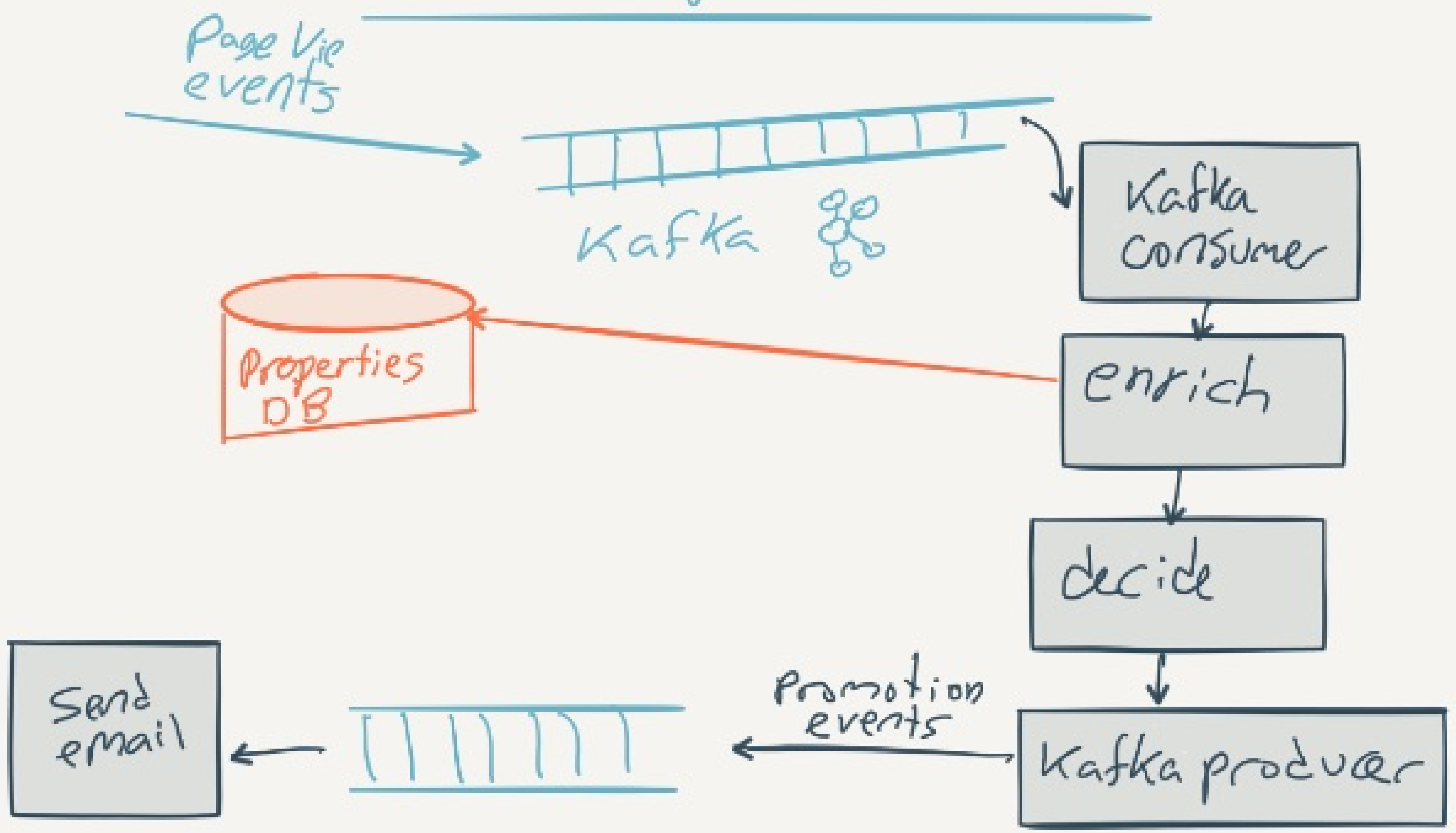
Discount package in new Hawaii hotel

Pattern #4 – Enriching events

From Event	To Enriched Event
sessionID	sessionID
timestamp	timestamp
loyaltyID	loyaltyID
propertyID	Level
	isPlatium
	propertyID
	Location
	isBeach

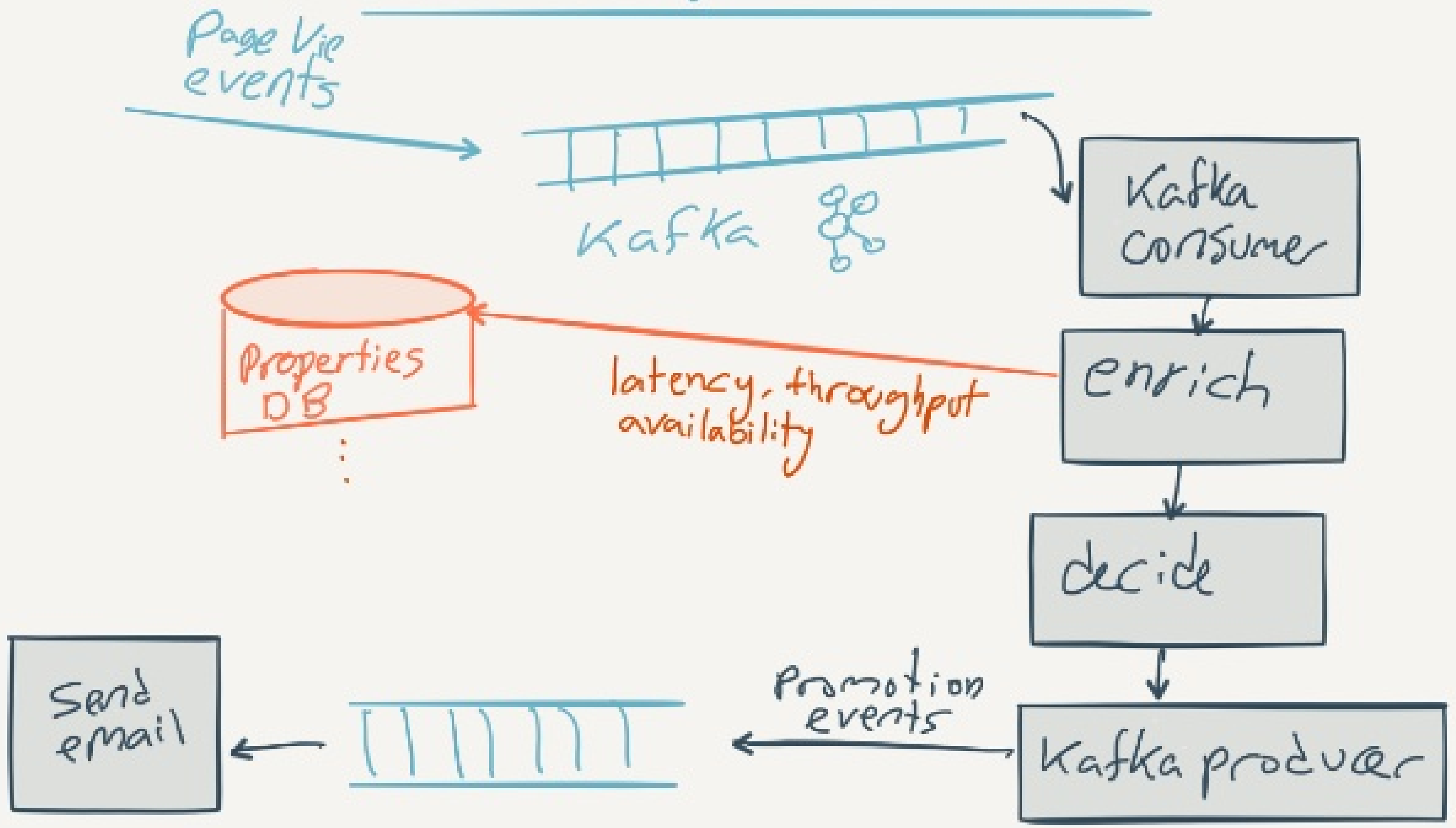

Page Vie events
Kafka
Kafka consumer
enrich *
decide
elastic
Kafka producer
Promotion events
Send email
Stream Processing App

Attempt #1

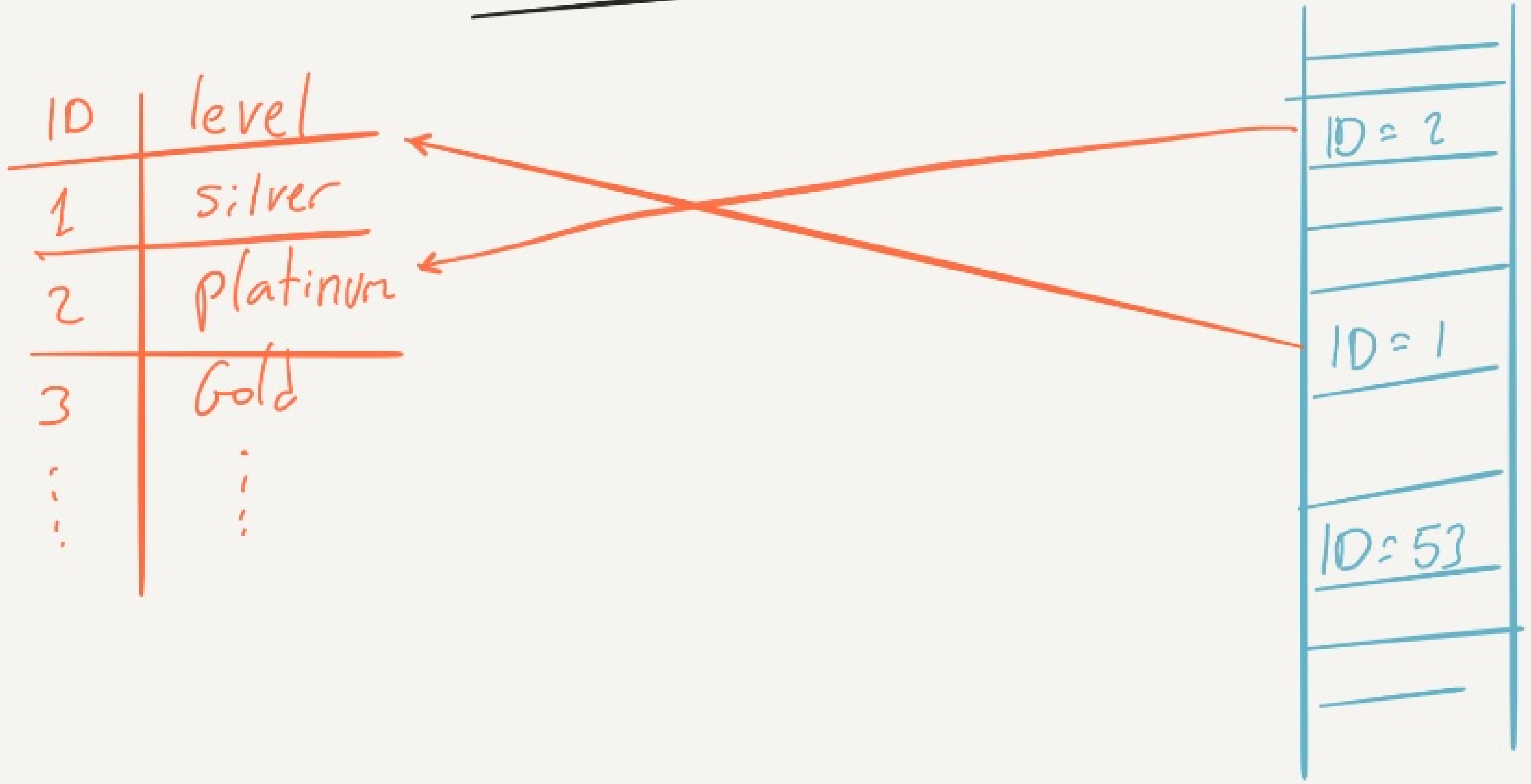


Page Vie events
Kafka
Kafka consumer
enrich
Properties DB
decide
Kafka producer
Promotion events
Send email

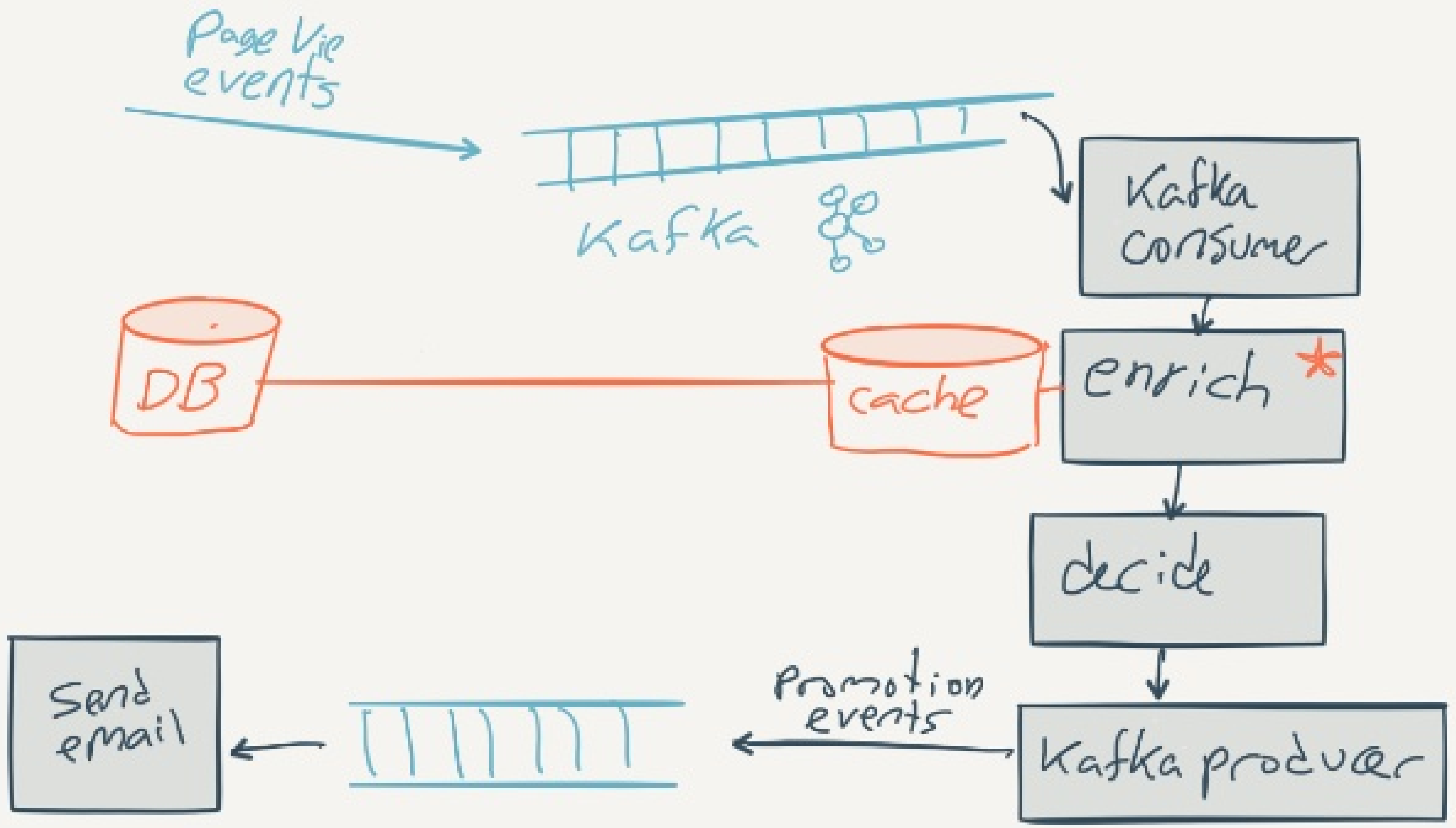
Attempt #1 - Issues

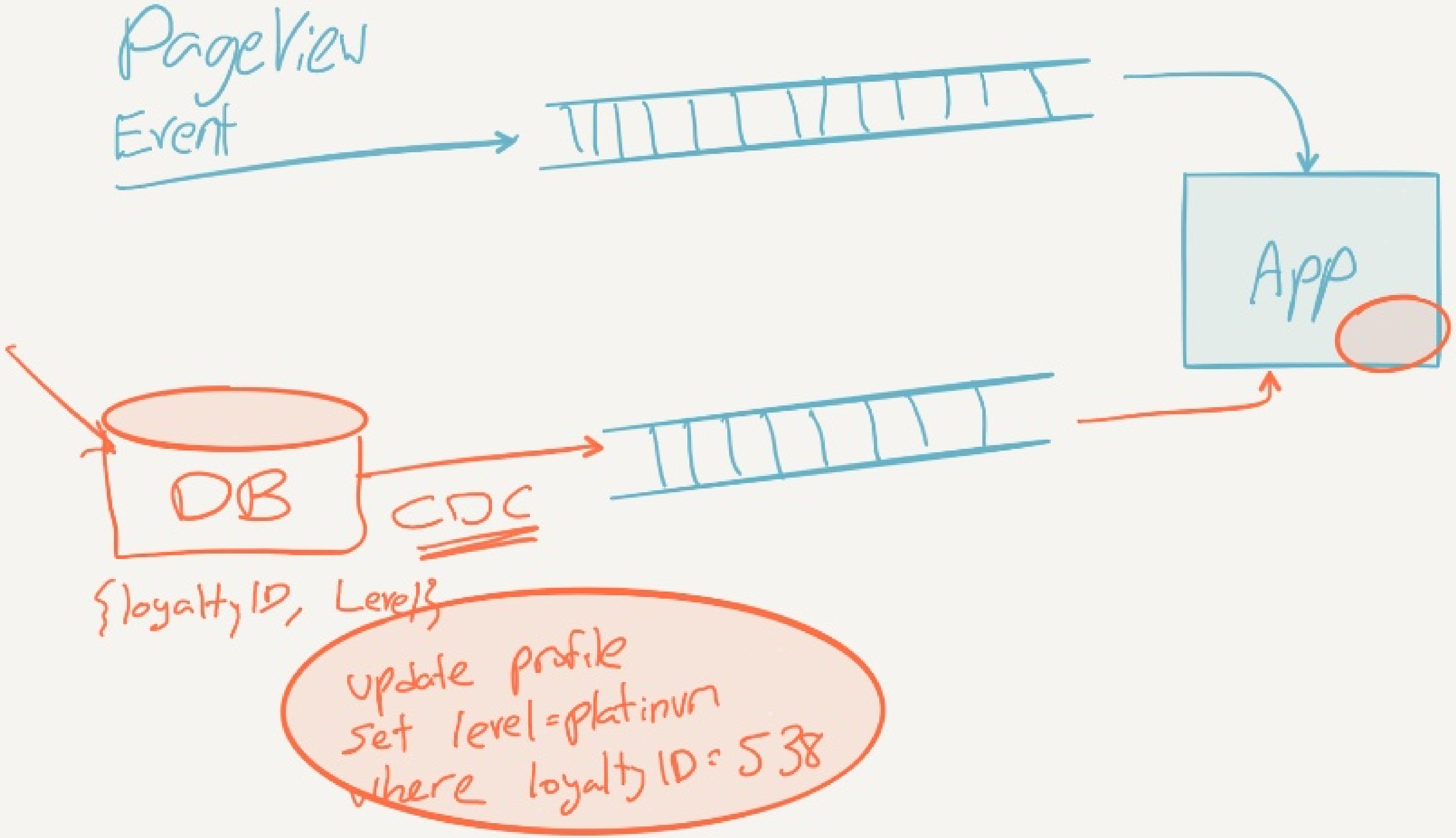


Stream/table join



Attempt #2


Page Vie events
Kafka
Kafka consumer
DB
cache
enrich *
decide
Kafka producer
Promotion events
Send email

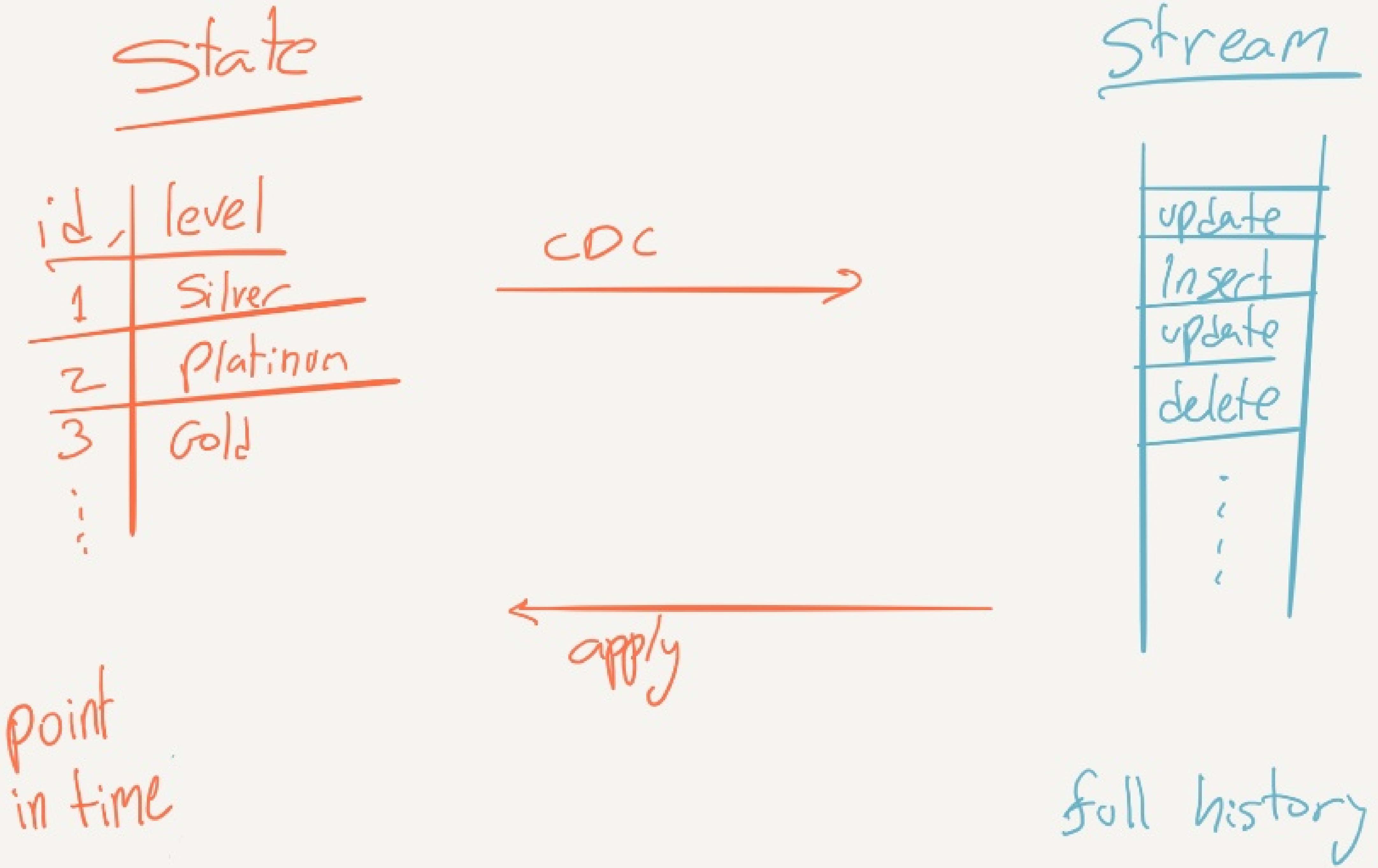

PageView Event
App
DB
CDC
{loyaltyID, Level}
update profile
set level=platinum
where loyaltyID=538

We use Kafka, Connect and CDC Connectors

To turn state (in DB)

Into stream of events (in Kafka)

And back into state (in application cache)


State
id
level
1
Silver
2
Platinum
3
Gold
CDC
apply
Stream
update
Insert
update
delete
point in time
full history

In reality…

- Maintaining the distributed cache isn't trivial
- How do we persist state? Failover? Recover?
- How do we co-partition for joins?

KafkaStreams makes it easy:

```
Ktable loyaltyTbl = builder.table(…, "loyalty-cdc-topic")
PageViewStream.leftJoin(loyaltyTbl, <operation>)
```

Example:
https://www.confluent.io/blog/distributed-real-time-joins-and-aggregations-on-user-activity-events-using-kafka-streams/

Few Patterns

1. Stream all things (in one place)
2. Keep Compatible and Process On
3. Ridiculously Parallel Single Message Transformations
4. Streaming Data Enrichment

confluent

Thank You!