Assisting Millions of User in Real-Time

Flink Forward Berlin 2018

The Speakers

Who are these guys?

Kcell

Alexey Brodovshuk

@alexeybrod

Krzysztof Zarzycki

@k_zarzyk

getindata

About Kcell

Kcell JSC is a part of the largest Scandinavian telecommunications holding – TeliaCompany

Largest GSM operator in Kazakhstan

> 10 000 000 subscribers

Great network coverage

4G (40%), 3G (73%), 2G (96%) population

We like innovations

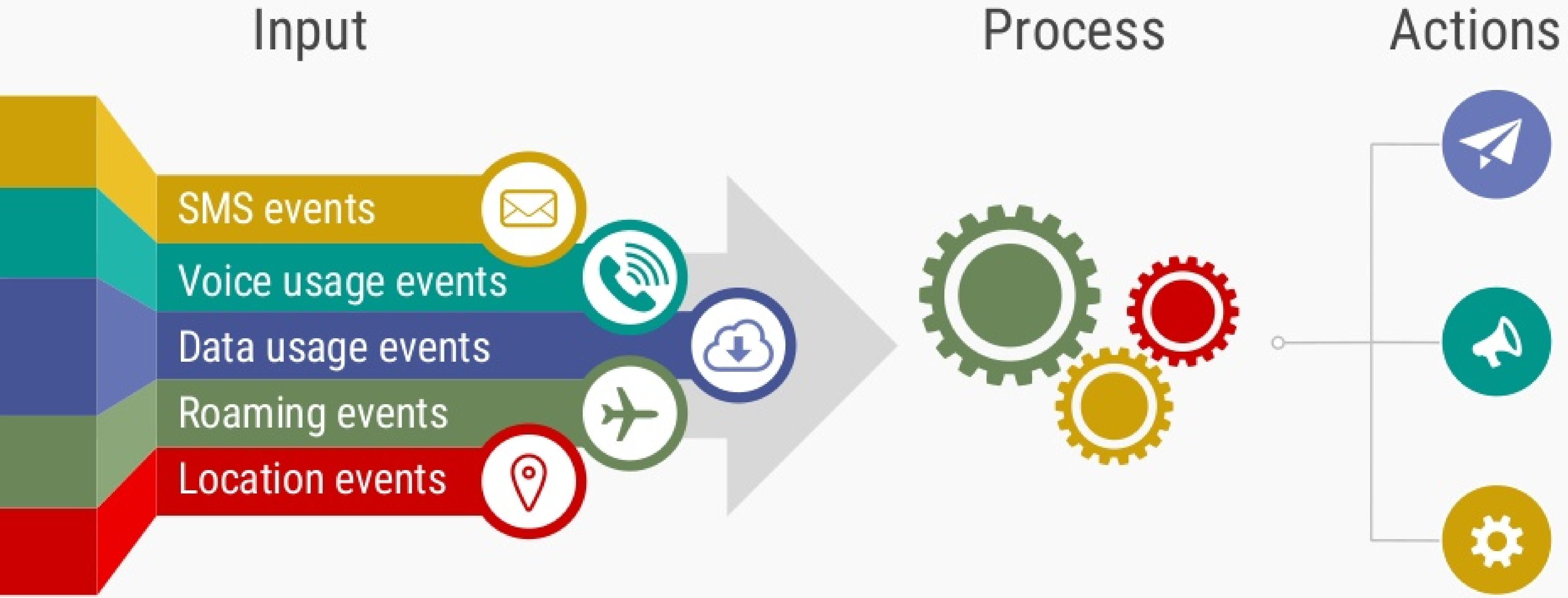
Kcell has a strong software development team and lots of experience in building services and products

Not only telco

There is the ongoing process of company digital transformation

Business needs

Assisting Millions of Users in Real-Time

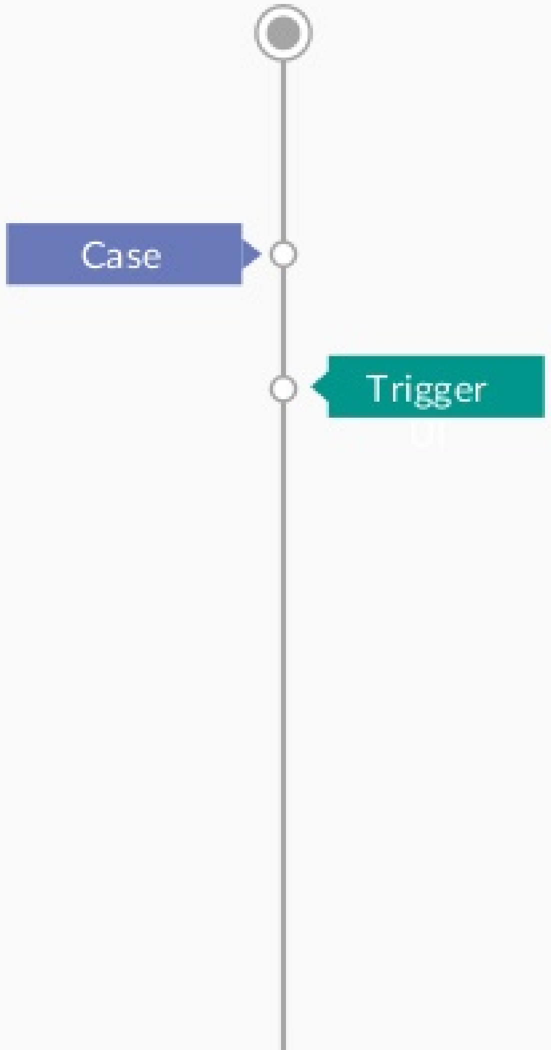


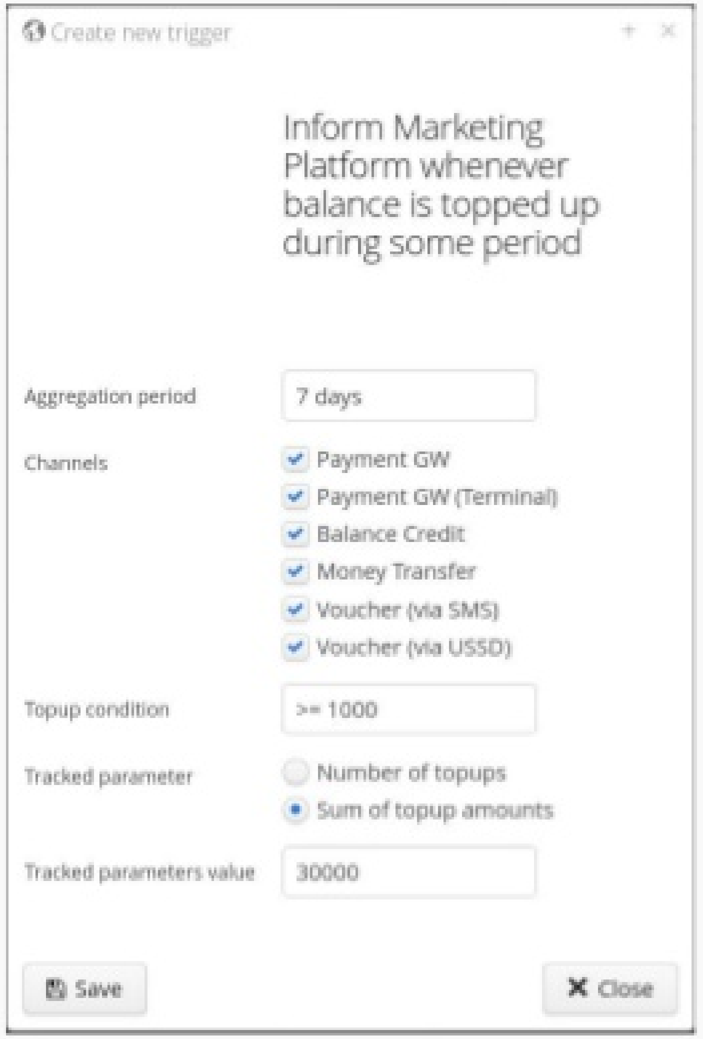
Use Cases

Use case scenarios. Just few of many.

Balance Top Up Case

If subscriber top-ups her balance too often in short period of time. We can offer her a less expensive tariff or auto-payment services.





Inform Marketing Platform whenever balance is topped up during some period

Aggregation period	7 days
Channels	✓ Payment GW ✓ Payment GW (Terminal) ✓ Balance Credit ✓ Money Transfer ✓ Voucher (via SMS) ✓ Voucher (via USSD)
Topup condition	>= 1000
Tracked parameter	○ Number of topups ● Sum of topup amounts
Tracked parameters value	30000

Roaming case

Trigger to Marketing Platform if subscriber visited X country OR/AND registered in Y visited mobile network and his device's type is Z

Roaming

Fraud

Fraud case in roaming

Send an email to the anti-fraud unit if subscriber registered in roaming but his balance at the moment is equal to 0.
This situation is impossible in standard case.

Old System

Why did we start to look for the new solution?

External Vendor Solution

1. **Blackbox Solution**
   Kcell Developers can't fix, tweak or optimize it
2. **Scalability issues**
   Limited to ~2000 events / sec
   Can't support all needed data sources
3. **Not reliable**
   Multiple accidents which took too much time to resolve

Scale

Required system throughput

10M
Subscribers

165K
Events / second

22.5
TB / month

About GetInData

Big Data. Passion. Experience.

Roots at Spotify

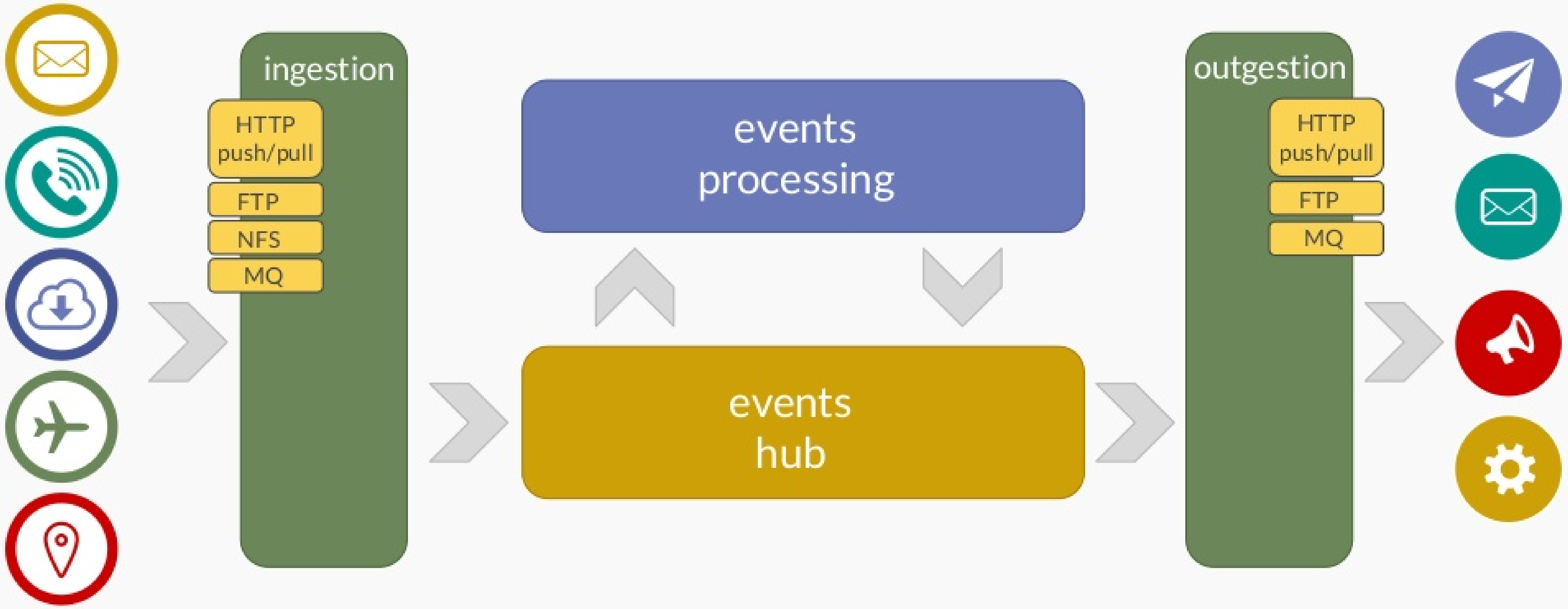
Focus on Big Data from Day 1

Production Experience

Contributions to Apache Flink

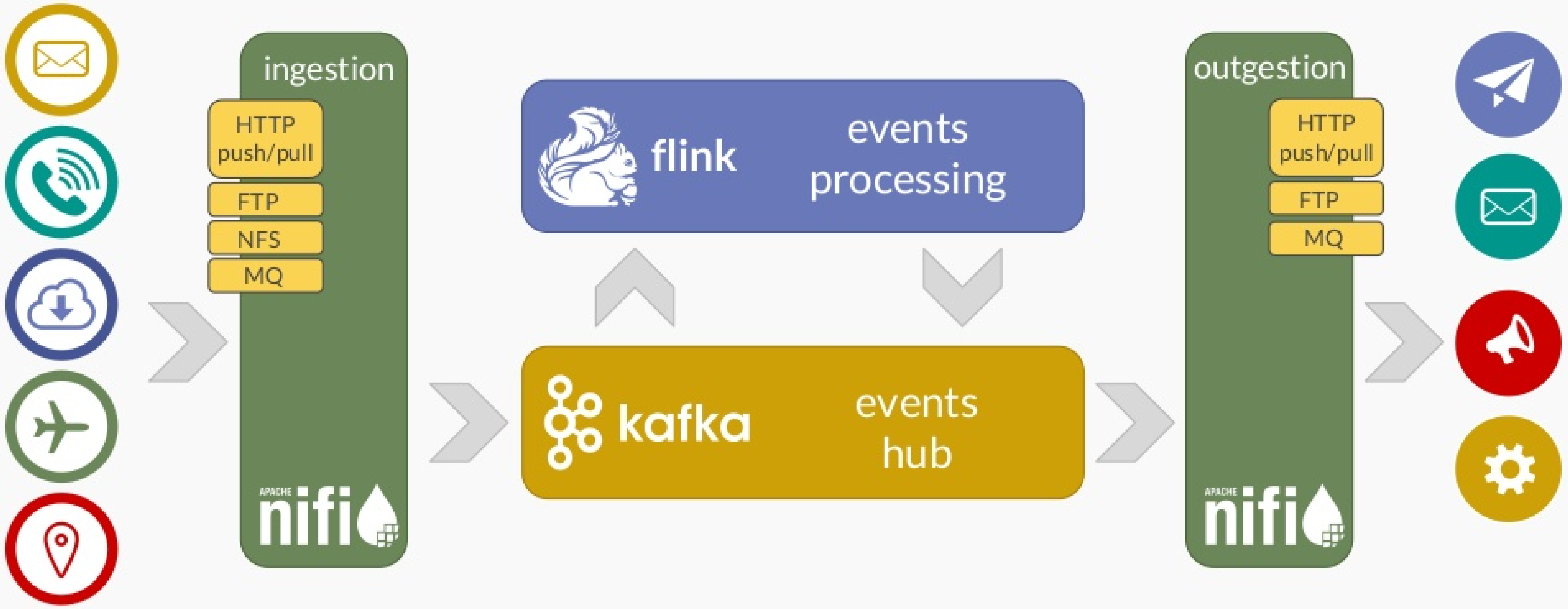
New Solution

Real-time Stream Processing



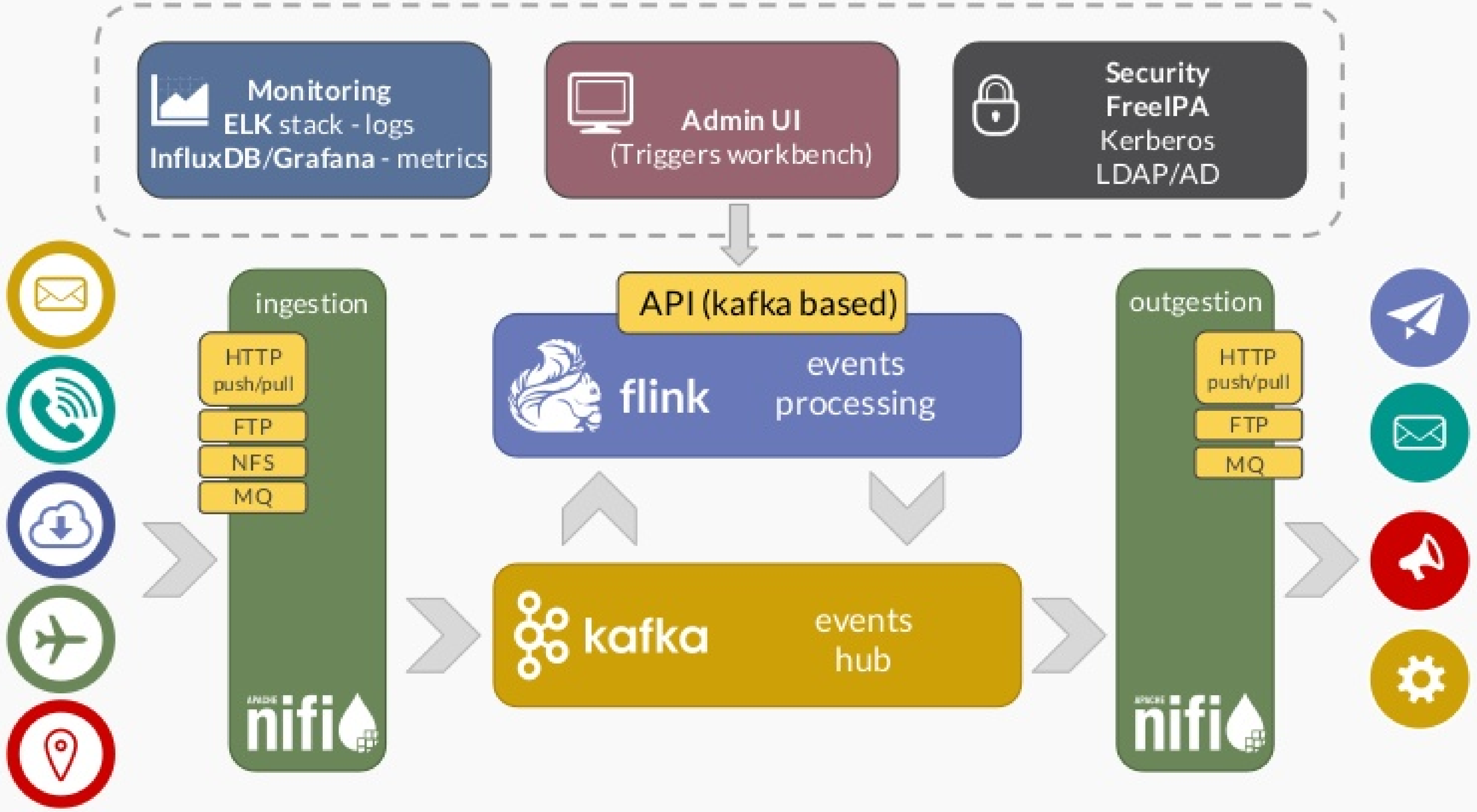
New Solution

Real-time Stream Processing



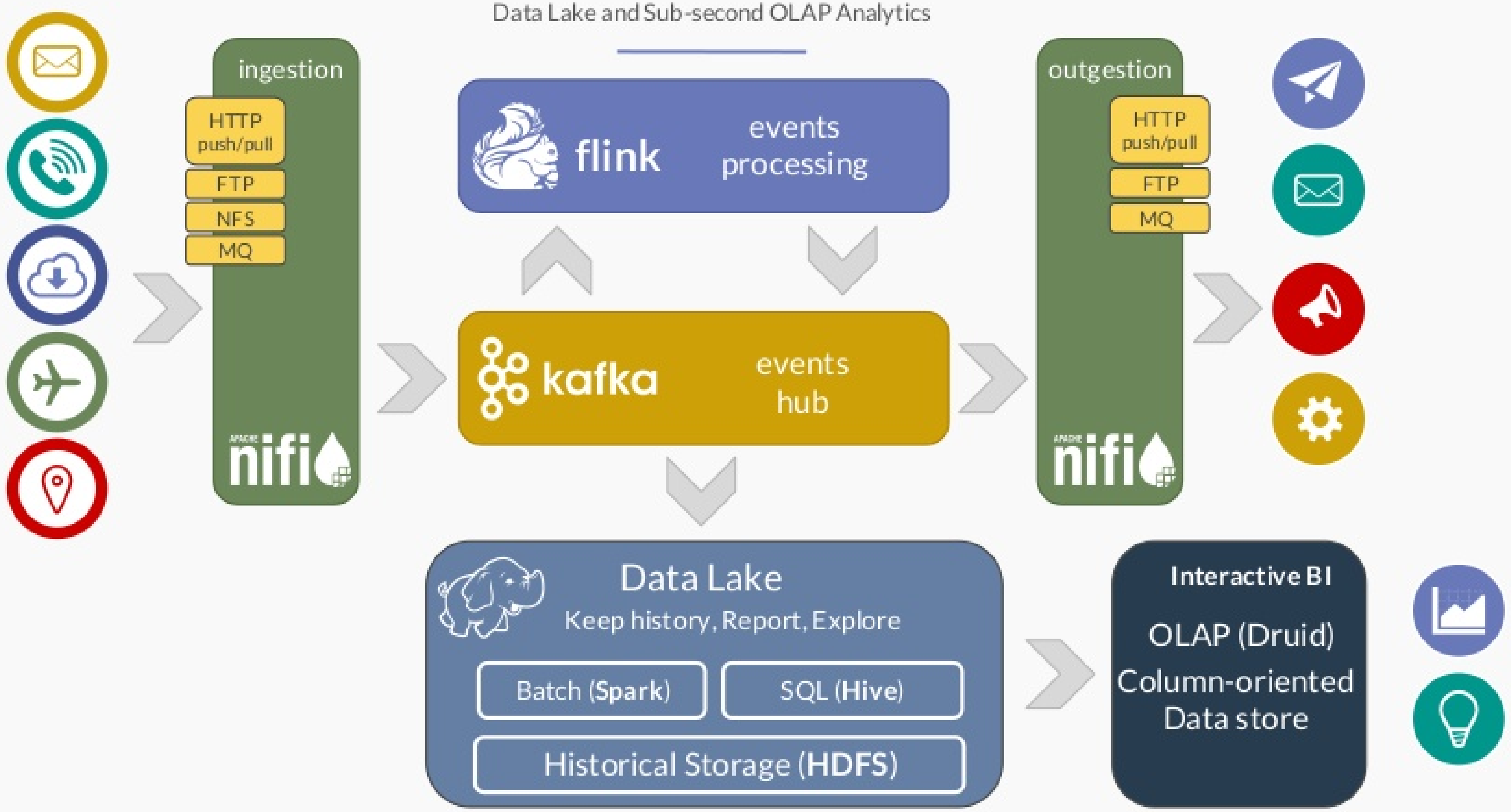
New Solution (Operations)

Web UI, Monitoring, Security



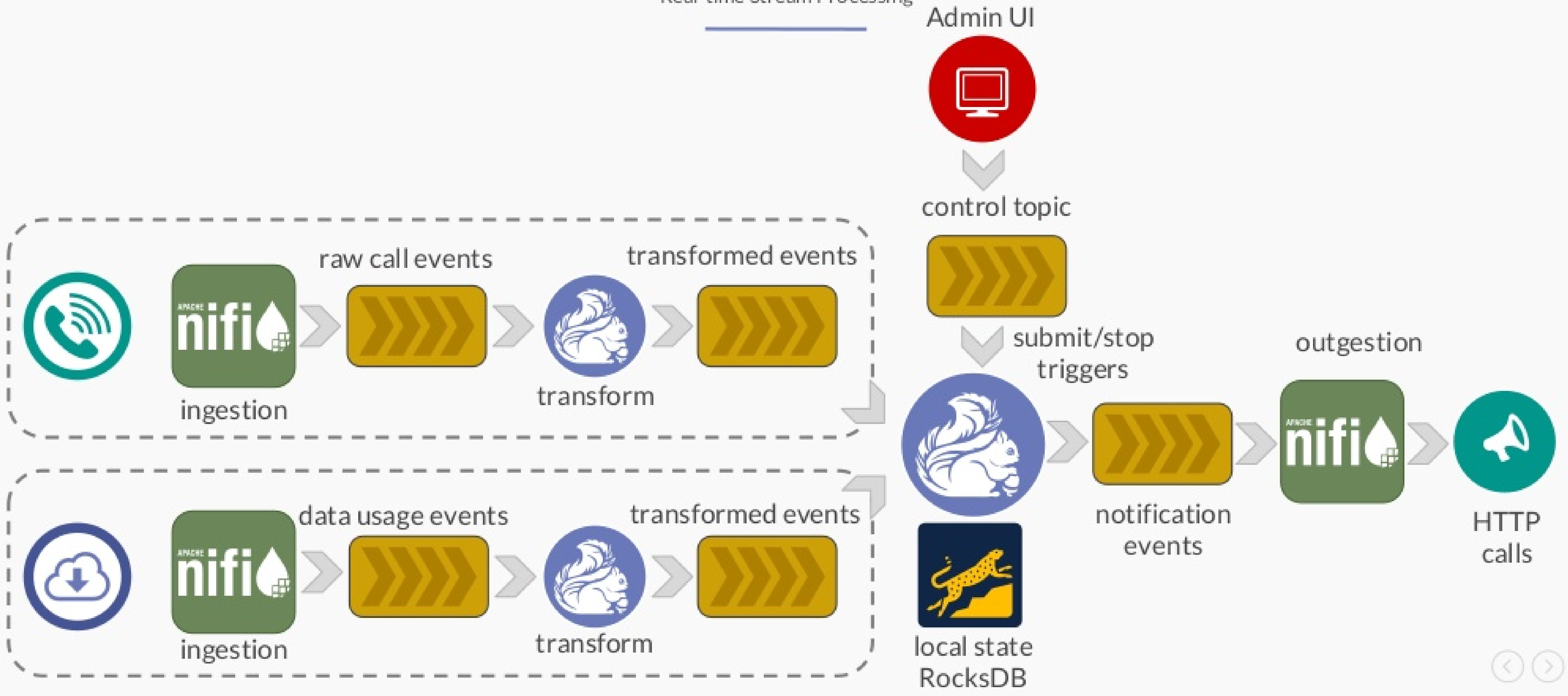
New Solution (Data Lake)

Data Lake and Sub-second OLAP Analytics



Processing Flow

Real-time Stream Processing



Dynamic Rules Design

Some treats for Squirrels

Dynamic Rules Design

Key Points

- We want to run 100s of triggers/business rules
- A typical approach: job per rule
- Won’t work in our case:
  - Run 100s of topologies/jobs = multiplied resources cost
  - Pull data from Kafka 100s of times
  - State (user features) replicated 100s times
  - Starting rule requires deployment of the job

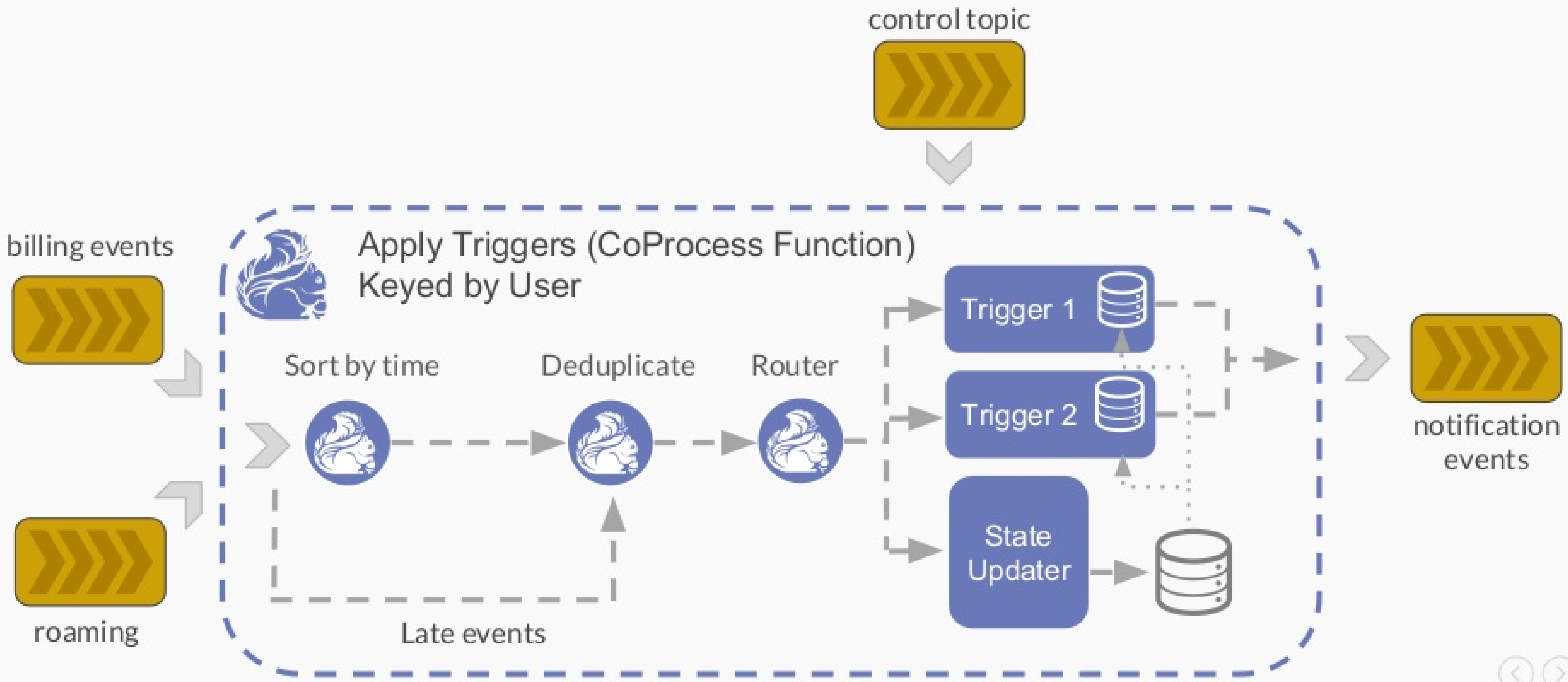
Dynamic Rules Design

Key Points

- Our approach: One job to run all triggers/rules
    - And to consume all the sources
- Trigger “templates” still coded with java
- adding/removing rules without restarting application
- 100s of rules running efficiently

Dynamic Rules Design

The Overview



Dynamic Rules Design

Pros and Cons

Shared resources and costs

- CPU, RAM, state, shuffle
- Pulling data from Kafka

Sharing of state

- Build customer features, that can be seen by all rules

No job restart on start of new rules

- Rules started by business, no IT involved

One bad rule affects whole system

- Watermarks are shared
- Failures are shared

Still need to code rule template in the job

- No way to use SQL, Table API, CEP

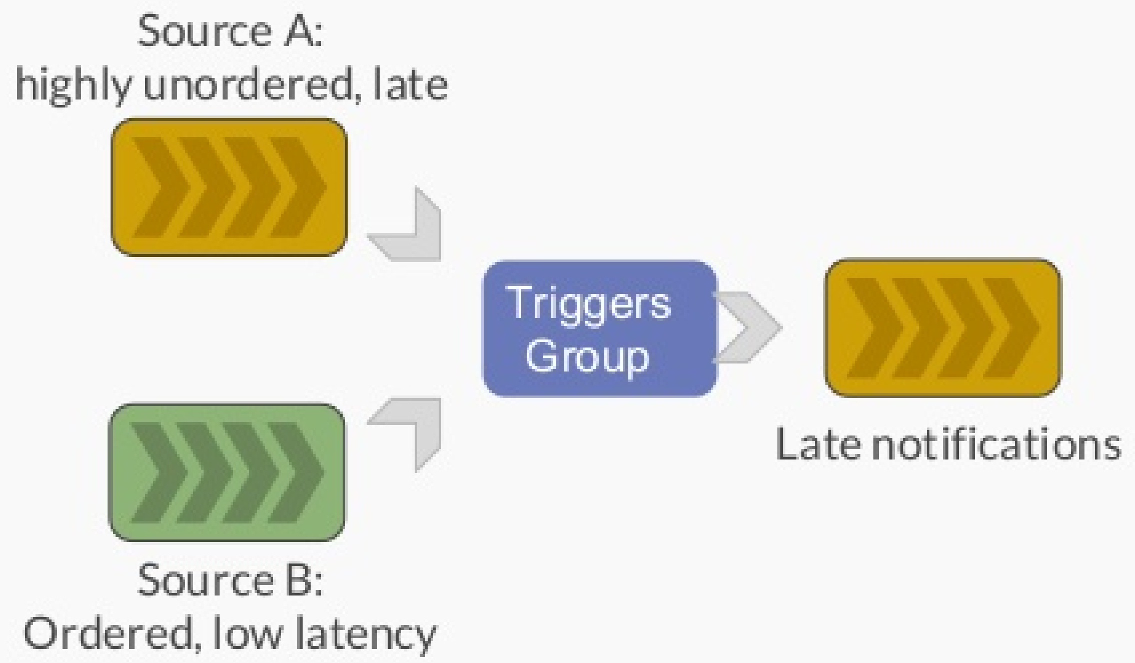
Can be tricky to debug

- Code is shared
- Code paths enabled externally

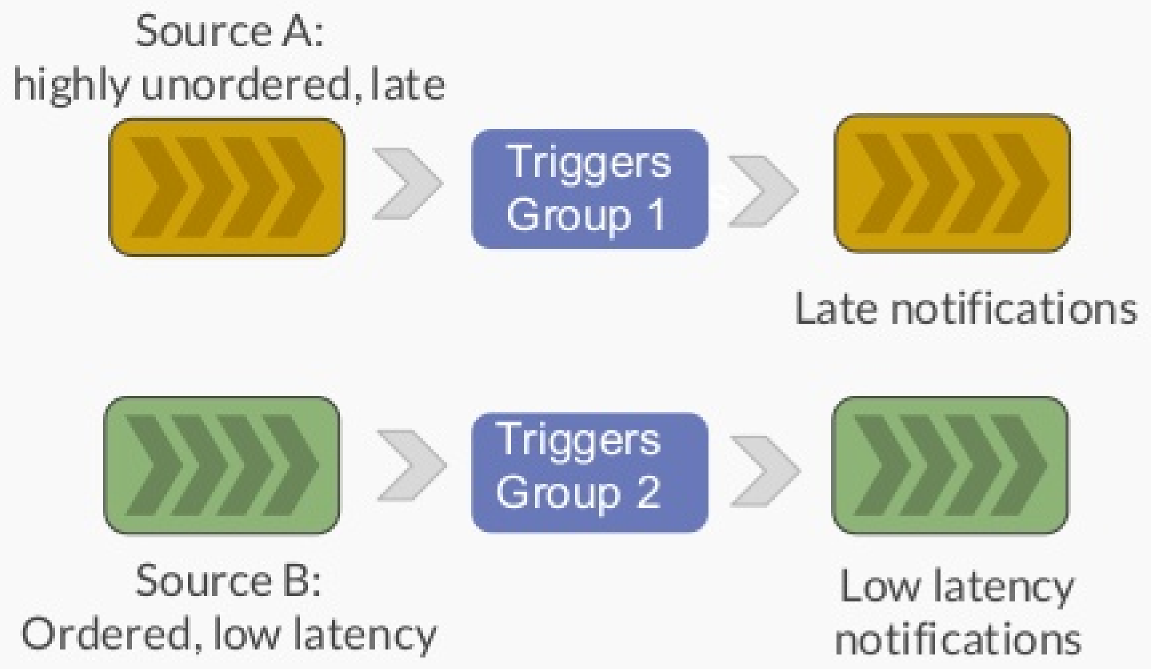
Dynamic Rules Design

Issue: lagging sources slow down all rules

Problem

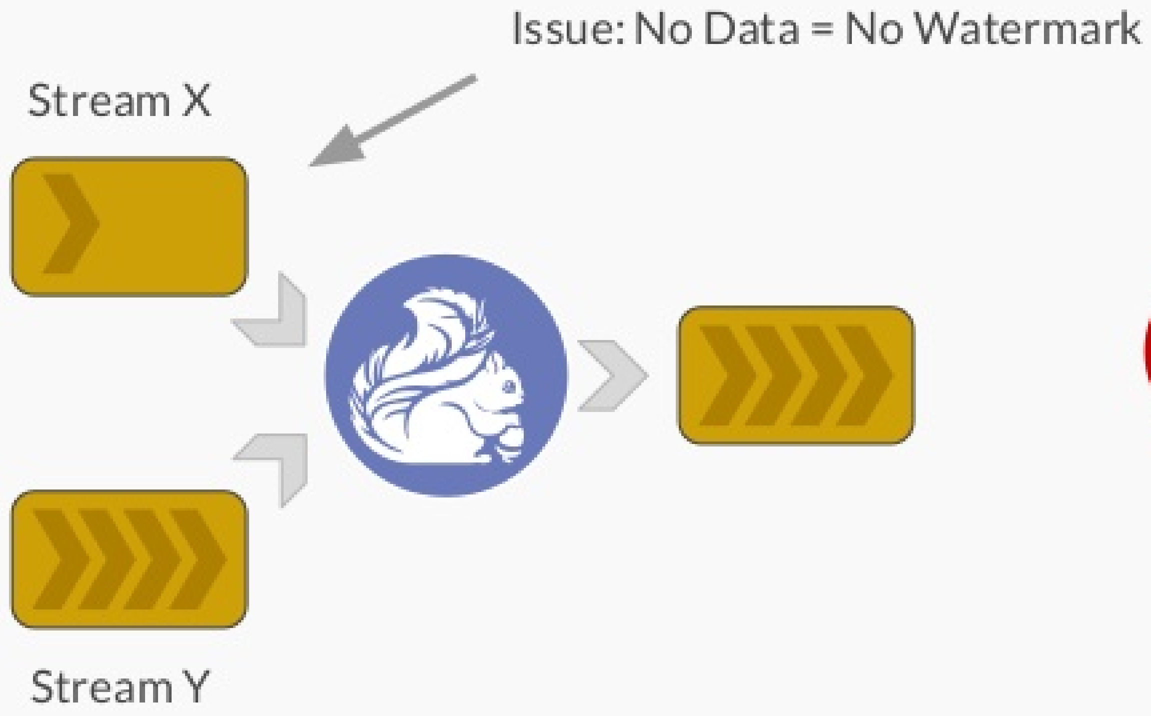


Solution



Join Streams

Join slowly changing stream with event stream



Solution

Ignore Watermark for slow stream
(Watermark = Long.MAX_VALUE)

Flink Changes Wishlist

What could be even better?

Dynamic Topologies

attach new branch to existing topology that receives the same data

Share inputs between topologies

- Graph of topologies that pass data locally in Flink
- Other words: Local Proxying/fan-out of Kafka traffic

Cheaper topologies

Dynamic SQL

Decisions made

Some decisions our team made before or during project implementation

Powerful Real-Time Analytics

Streaming-first approach

Apache Kafka for event hub

Apache Flink

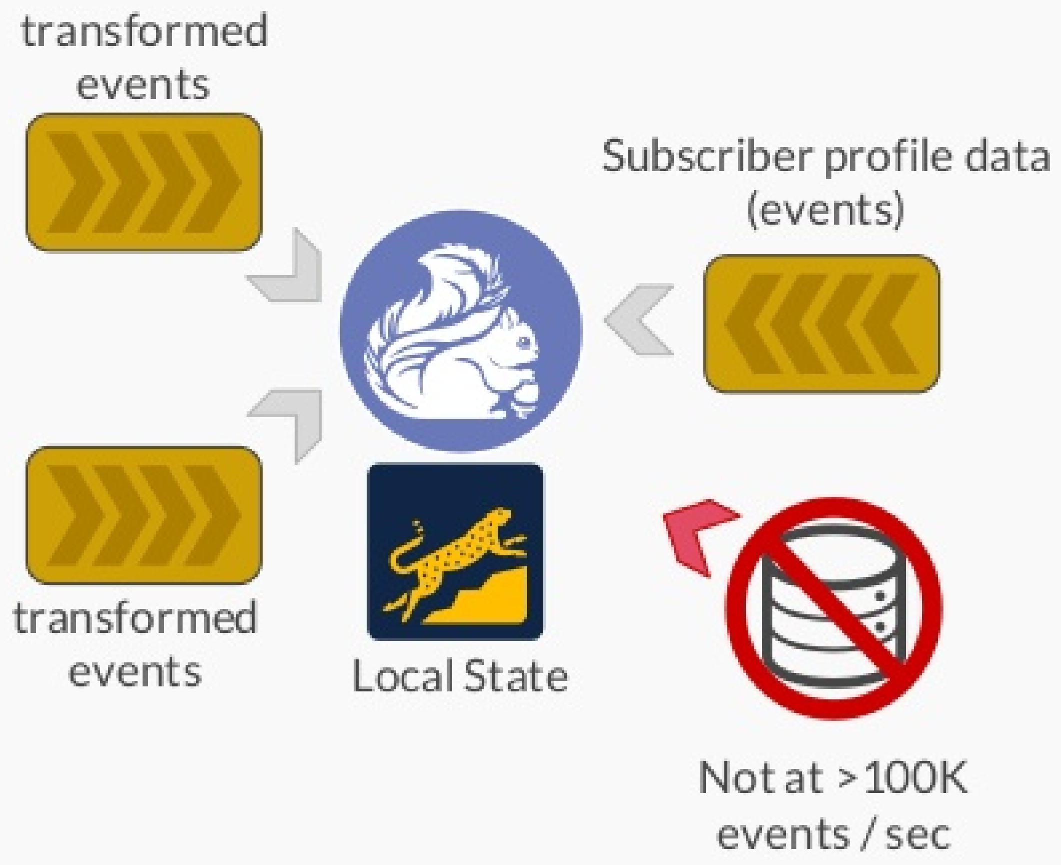
Performance

Apache Avro

Keep state local to the process

Ingest reference data for local joins and enrichment

- No need to query external systems while processing
- Data time correlation correctness

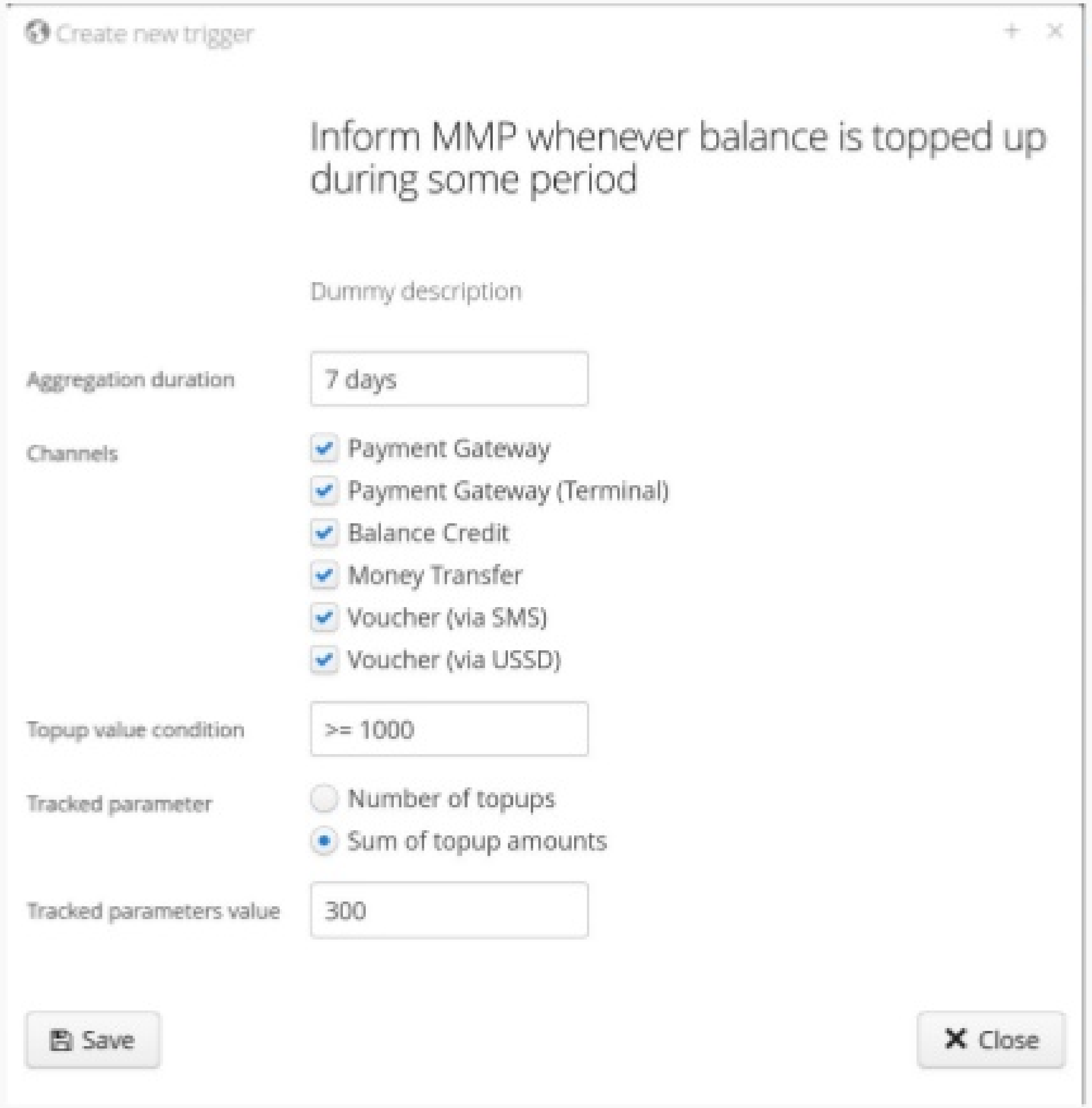


Ease of Use

Nifi for data ingestion (no coding)

- but not for CEP

Web UI for configuring triggers



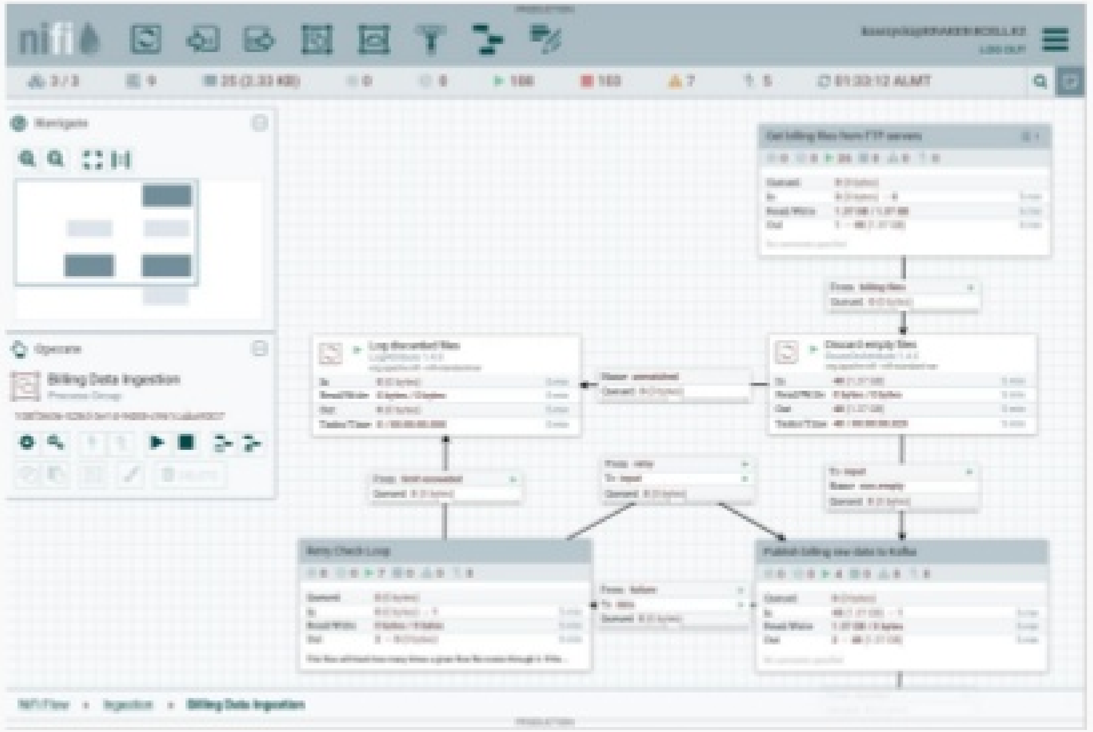
Create new trigger

Inform MMP whenever balance is topped up during some period

Dummy description

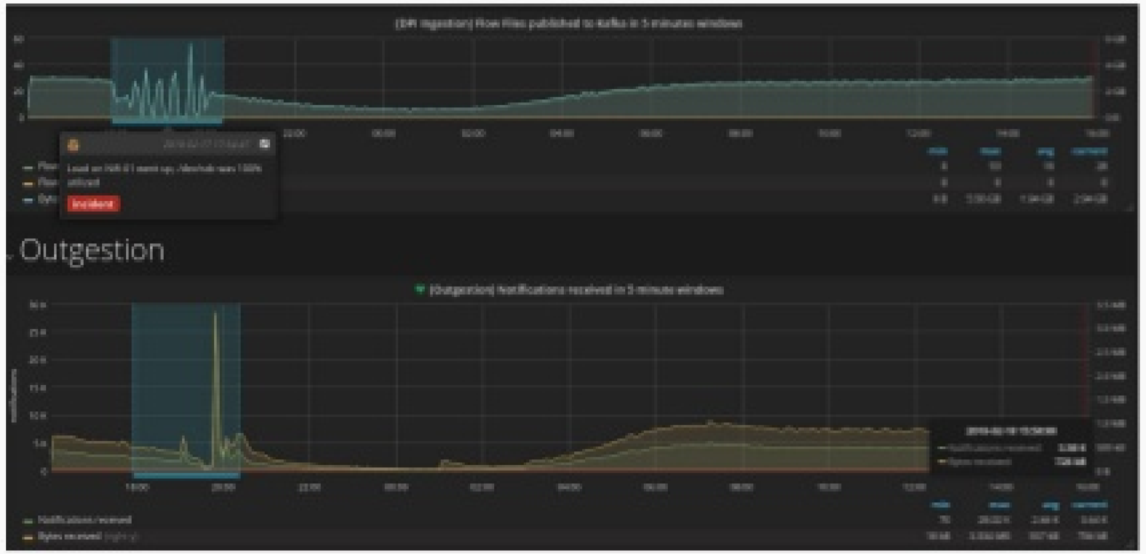
Aggregation duration	7 days
Channels	[✓] Payment Gateway [✓] Payment Gateway (Terminal) [✓] Balance Credit [✓] Money Transfer [✓] Voucher (via SMS) [✓] Voucher (via USSD)
Topup value condition	>= 1000
Tracked parameter	Number of topups [✓ Sum of topup amounts]
Tracked parameters value	300

Save   Close



Reliability and battle-tested techniques

Flink on YARN, with HDFS
HA for redundancy and running ~24/7
InfluxDb & Grafana for monitoring & alerting
ELK for logs collection and aggregation



Security

Kerberos and AD thanks to FreeIPA
Apache Ranger for authorization

Extensiveness

One platform for the whole Enterprise

Batch (adhoc) queries too

- Spark, Hive/Presto

Online analytics

- OLAP

Cost-Efficiency

Open-source technologies

HDP as a licence-free distribution

Just start with a bunch of servers

Our Collaboration

Two heads are better than one

Joint development team
Not a vendor solution
Development as one team

Testing
Separate testing environment
Automated Unit/E2E tests

Code quality
Code review and automated tools for code quality control

Knowledge sharing
Constant knowledge exchange in areas of expertise

Agile Practices
Distant geographic locations, but everyday standups

Deliver
DevOps/Automation

Go live quickly!
<4 months to first production case running 24/7!

Future Work

We have already done a lot. But more great things are coming.

More Data Sources
More Triggers
Geolocation data
Equipment logs

Data Lake
Make it a company-wide, self-service go-to place for data analysis

Customer 360 view
Call center, clickstream, communication… all in one place ready for behavioral analysis

Data Monetization
Monetize valuable insights from our combined rich data sources.

2018 Q4 | 2019 Q1 | 2019 Q2 | Bright Future

Real-time BI
Intraday view on business and operations

Predictive maintenance
Network Optimization
To lower operational costs
And make better investments

Machine Learning
We plan to include machine learning and other tools that would enhance our platform even more

And many more...

Questions?

Flink Forward Berlin 2018

Contact us:

zarz@getindata.com
alexey.brodovshuk@gmail.com