Data streams and fraud detection

Maciej Próchniak

August 2018

Data streams and fraud detection

There are many types of frauds occurring in the telco business. Let's look at a few examples:

- **Premium usage fraud.** In this case, we want to detect excessive premium usage coming from one phone which could be stolen or bought using false documents. We have to be able to calculate the total amount of charges from this account during the last few hours.
- **SMS spam.** Sometimes regular phone accounts are misused to send spam messages - which is against the contract. We want to detect suspicious amounts of messages coming from one account - e.g. 1000 per 4 hours. This case is a bit tricky because we may get easily false positives - so probably we don't want to calculate the total number of messages but count the unique recipients.
- **SIM cloning.** It's possible to cheat on SIM cards and have many physical cards connecting to the network as one number. In this case, one of the ways of detecting such fraud is to track a localization of the caller - if we see that a person moved from Berlin to Paris in 10 minutes, we know that it's not possible.
- **Termination frauds of various kinds.** One of the most common is SIMBox - where fraudster illegally setups kind of VoIP to bypass high interconnect rates. This type of fraud can be pretty tricky to detect - we have to understand how to separate normal usage from suspicious and to relate to a specific location - where the appliances are installed.

In each case, we want to detect suspicious activity as fast as possible and decide on appropriate actions.

When dealing with certain types of frauds, such as SMS flooding or premium services abuses, each minute of unstopped operation can cause a significant loss of money. Proactive automated actions such as blocking the service should be taken. In other cases lowering the credit limit or alerting the revenue assurance team which can tackle the problem can be enough.

In general, mobile operators have two ways of dealing with frauds:

- Simple credit limits in their billing systems. These are be enforced in real time immediately. However, billing systems are not easily changed.
- More elaborate models and rules performed on data warehouses. Here, rules can be much more flexible, but they usually operate with significant delay - an hour can be considered fast in this context.

Both methods are not sufficient. Revenue assurance teams monitor the suspicious behaviour of customers continuously and want to be able to prototype and check various types of fraud detection algorithms.

For example, while standard limits for SMS amounts may be 100 per hour, analyst detects that in specific county many frauds take place. They want to lower the limit of messages in this particular area for a week to see if it helps.

This type of change should not require development - analysts should be able to configure the system themselves, deploy new rules and monitor how they are working. They are professionals so they also need to test them beforehand - blocking regular clients should be avoided at all costs!

Is it possible? It turns out it is. TouK has successfully delivered modern fraud detection frameworks for one of the largest polish mobile operators. Below we'll look at its key components. The first one is Apache Flink - leading stream processing engine. The second one is TouK Nussknacker - unique GUI for letting analysts and business people design, test and monitor processes deployed on Flink.

Apache Flink

Apache Flink is one of the leading stream processing platforms. It's main features are:

- Scalability and throughput - it can process 10k events per second per one core, which means that even small cluster (2-3 machines) can handle 100k of events per second.
- Very low latency - our experiences show that typical event processing latency is less than 100ms.
- Ability to easily define and efficiently process various aggregations - e.g. time windows.

While not as widely known as Apache Spark, Flink is by no means unknown or immature. It has many complex and large deployments worldwide, the most significant are Alibaba, Netflix or Uber.

Flink is also used in the financial sector - ING and Capital One are widely known examples, and telco - by Bouygues Telecom, Play and Ericsson. You can read about other use cases on https://flink.apache.org/poweredby.html.

Nussknacker

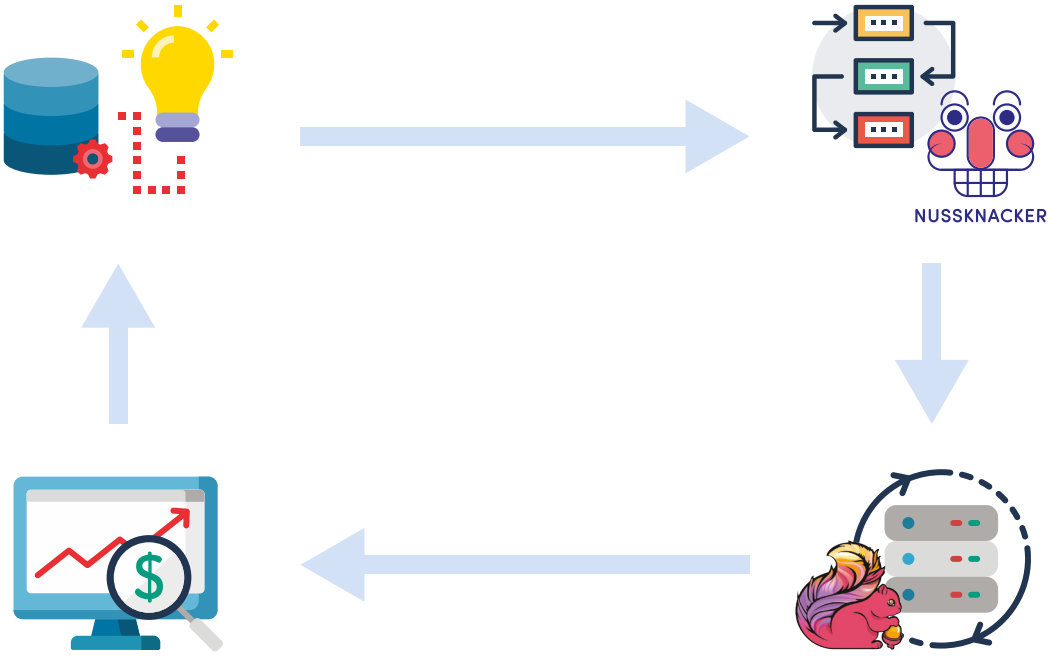
One of the most significant challenges in introducing stream processing platforms is giving analysts and business people (Revenue Assurance team in fraud case) the ability to change logic, test new behaviour, monitor and analyse performance.

These are especially important for companies that do not have an internal software development team. In such cases, each change that requires development from an external vendor is both costly and time-consuming.

TouK Nussknacker is a unique tool that gives analysts access to the power of streaming with Apache Flink. It has a friendly, simple user interface which was built with the following general principles:

- Business power users- e.g. knowing how to write a SQL query, but not necessarily how to code - can author and configure processes themselves.
- Some coding and configuration may still be needed to develop specific model and integrations that are unique to a particular deployment.
- Testing and prototyping are easy - the users have to feel empowered to try out their ideas, and they have to be able to deploy and control the process in the production environment.

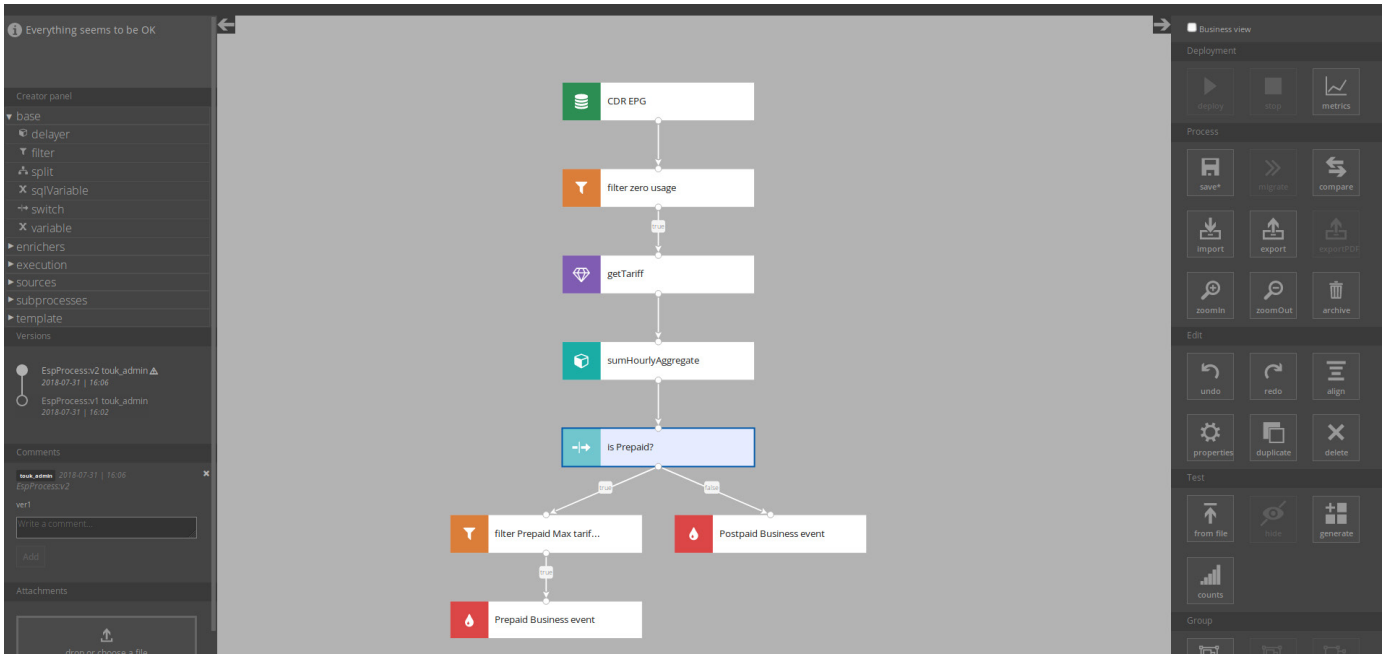
What we want to enable is closing feedback loop shown on diagram below.



We want to enable analysts to perform all steps

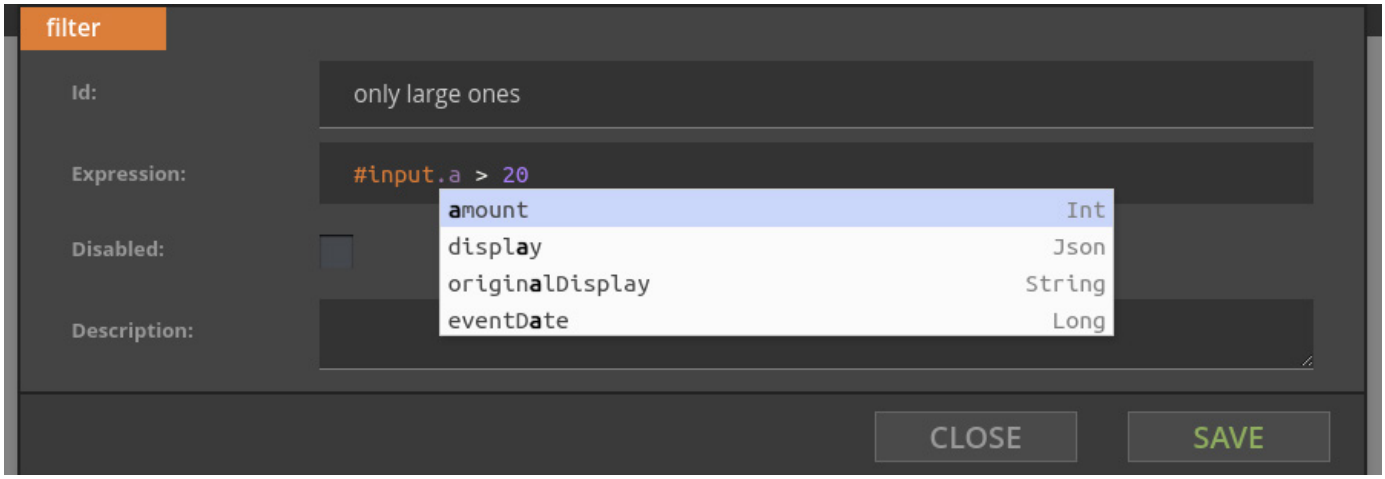
1. designing interesting events
2. prototyping and testing logic in Nussknacker sandbox
3. running on real data on high-performance Flink cluster
4. analyzing outcomes

Below you can see how sample process in Nussknacker looks like:



The diagram defining data flow consists of some generic parts - like filters, variable definitions, aggregate definitions - and also of parts specific for given use case or deployment - e.g. enriching events with additional customer data or defining specific actions - in marketing scenarios these would be contacts with customer, while in fraud detection - alerts or blocking accounts.

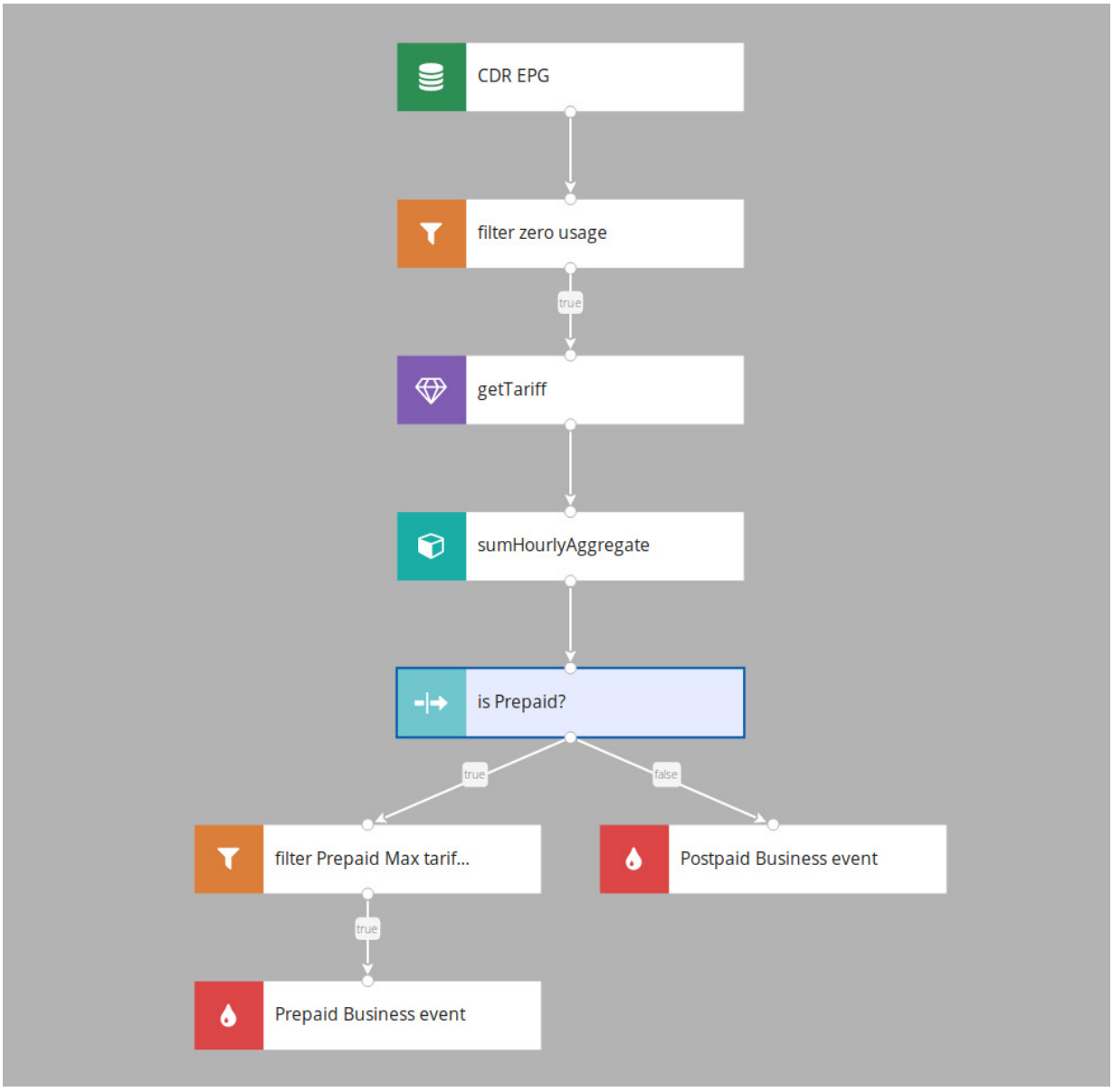
Let's have a look at sample filter or aggregate definitions:



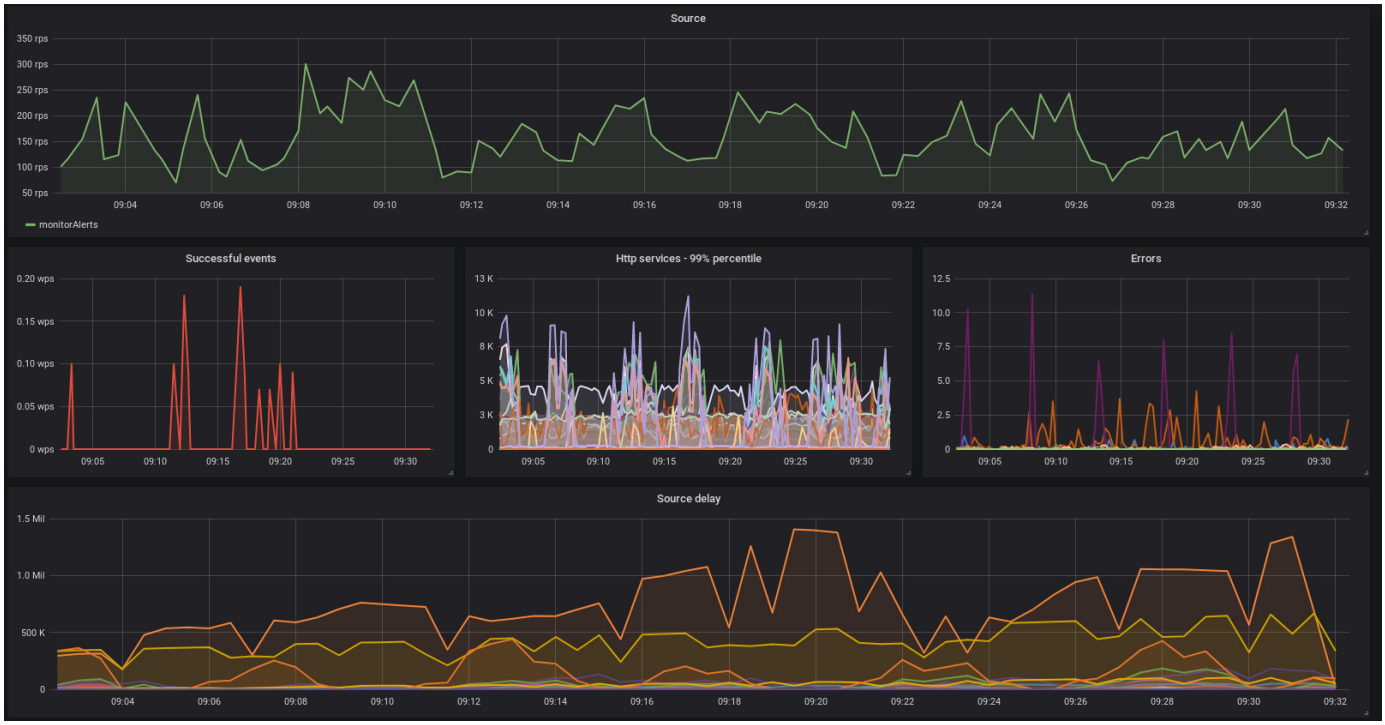
The rules are defined with simple expression language, in many ways similar to SQL. It's expressive enough to handle most cases, while it doesn't require coding skills.

Designing a diagram is just the beginning. Even more important is to test it properly. Nussknacker can help with it in two ways:

- Sandbox testing. We make it easy to prepare test cases based on real data. The user can then run a process in a sandbox environment and track how it behaves.
- Acceptance testing. It's done in a dedicated environment, which has all relevant data but does not trigger real actions. The architecture of the platform (especially Apache Kafka features) make it easy to clone production data, and Nussknacker monitoring solutions make it easier to decide if a process is ready for production. If it is - migration is just one click away.



After deploying the process to a production environment, monitoring is crucial. We observe how the process behaves in two ways. The first one is monitoring based on metrics. Each process has its dashboard:



where you can see:

- Throughput - how many events per second are processed
- Latency - how long does it take from event to processing
- Errors, detected frauds and so on.

This dashboard gives a general view - does process perform well, does it have any problems.

To dig deeper, we use another tool - ElasticSearch and Kibana. We analyze interesting events and perform basic analysis and search - are conditions ok? Did we found real frauds or were there false positives?

Banking

The use case described above comes from the telco, but similar setup can be used in finance and other areas.

Further developments are also possible - for example using ML models to filter more cases.

For example, ING Bank uses Flink for fraud detection: https://berlin-2017.flink-forward.org/kb_sessions/fast-data-at-ing-building-a-streaming-data-platform-with-flink-and-kafka/. Such use cases are a good fit for Nussknacker as they involve business rules that change dynamically.

Other possible use cases

Flink and Nussknacker can also be used in other areas. One of the interesting use cases - also from telco - is monitoring Quality of Service in real time.

For example, Bouygues Telecom uses Flink to detect connection and quality of service problems:

https://2016.flink-forward.org/kb_sessions/a-brief-history-of-time-with-apache-flink-real-time-monitoring-and-analysis-with-flink-kafka-hb/.

Again, this is not limited to telco - services like VoD are also an interesting area (Netflix also uses Flink: https://www.datanami.com/2018/04/30/how-netflix-optimized-flink-for-massive-scale-on-aws/).

Conclusion

Stream processing gives excellent possibilities, but it's not easy. Apache Flink gives you all the power you need while Nussknacker brings this power closer to the business people.

Learn more:

https://touk.pl/esp/

https://touk.pl/portfolio.html

https://github.com/TouK/nussknacker/

https://touk.github.io/nussknacker/