TouK Nussknacker

GUI for Flink

Maciek Próchniak

(that’s me)

How it all began?

POC with Apache Flink

Great results :)

Configuration

↓

Client not (really) happy

Scala expression for filters

$$\Downarrow$$

Client still not very happy

Ok, we'll build you a GUI

↓

Finally, approving nod…

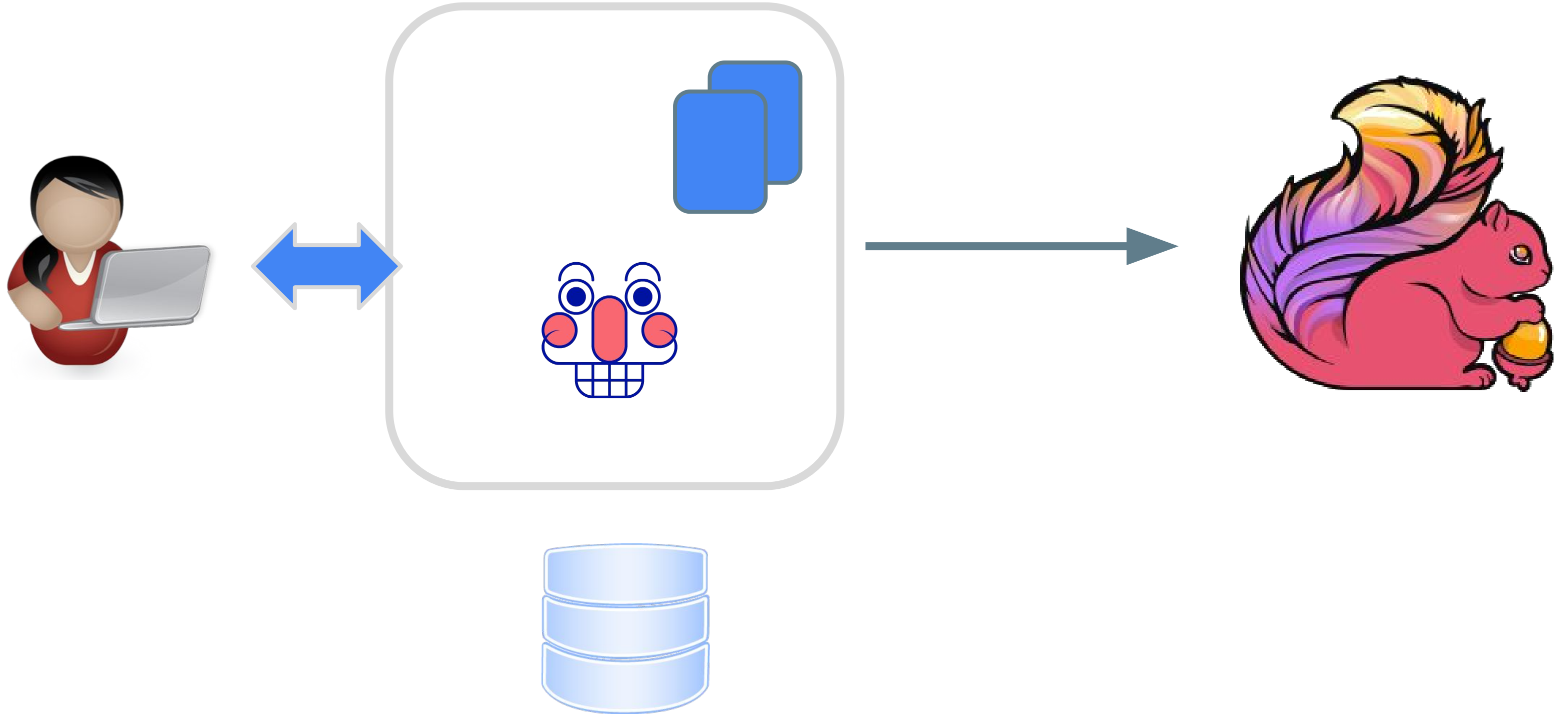
NUSSKNACKER

Main assumptions

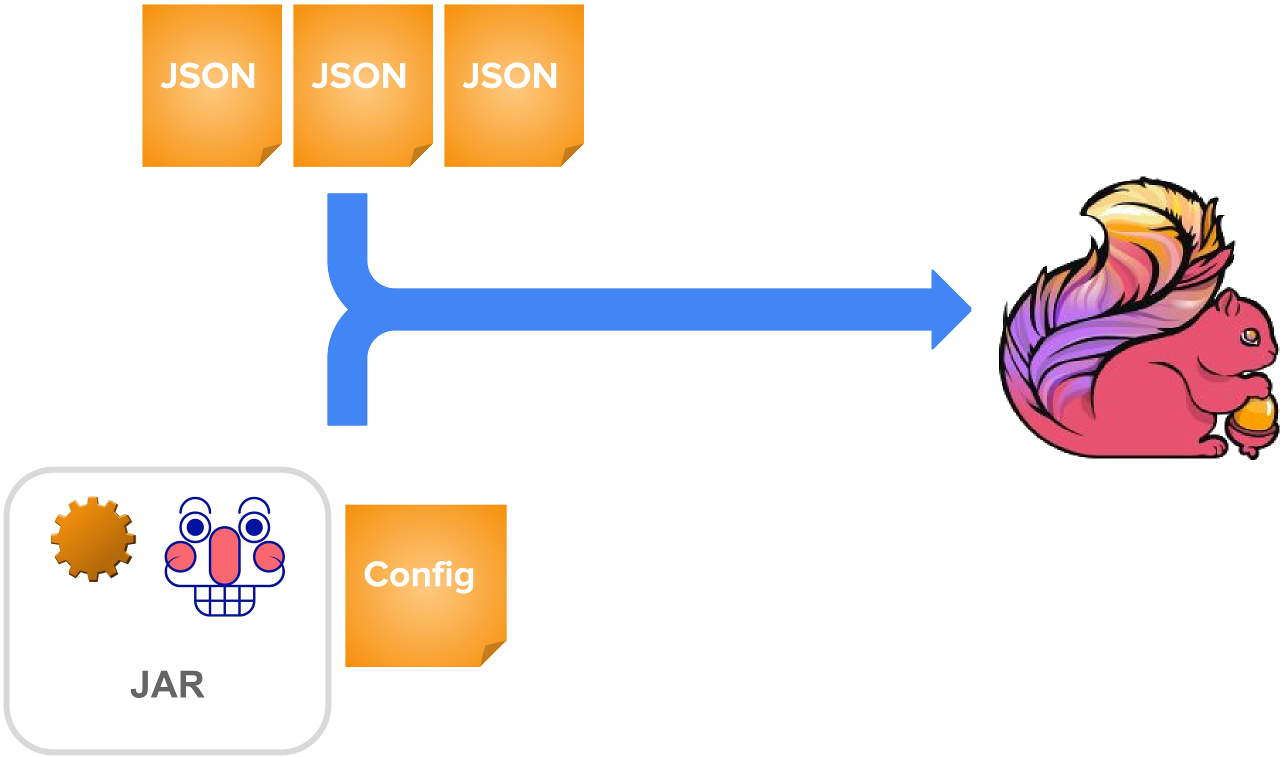
- Model & integration in “**normal” code**
- Expressions accessible for **semi-technicals** - like SQL
- **Facilitate** testing and experimentation

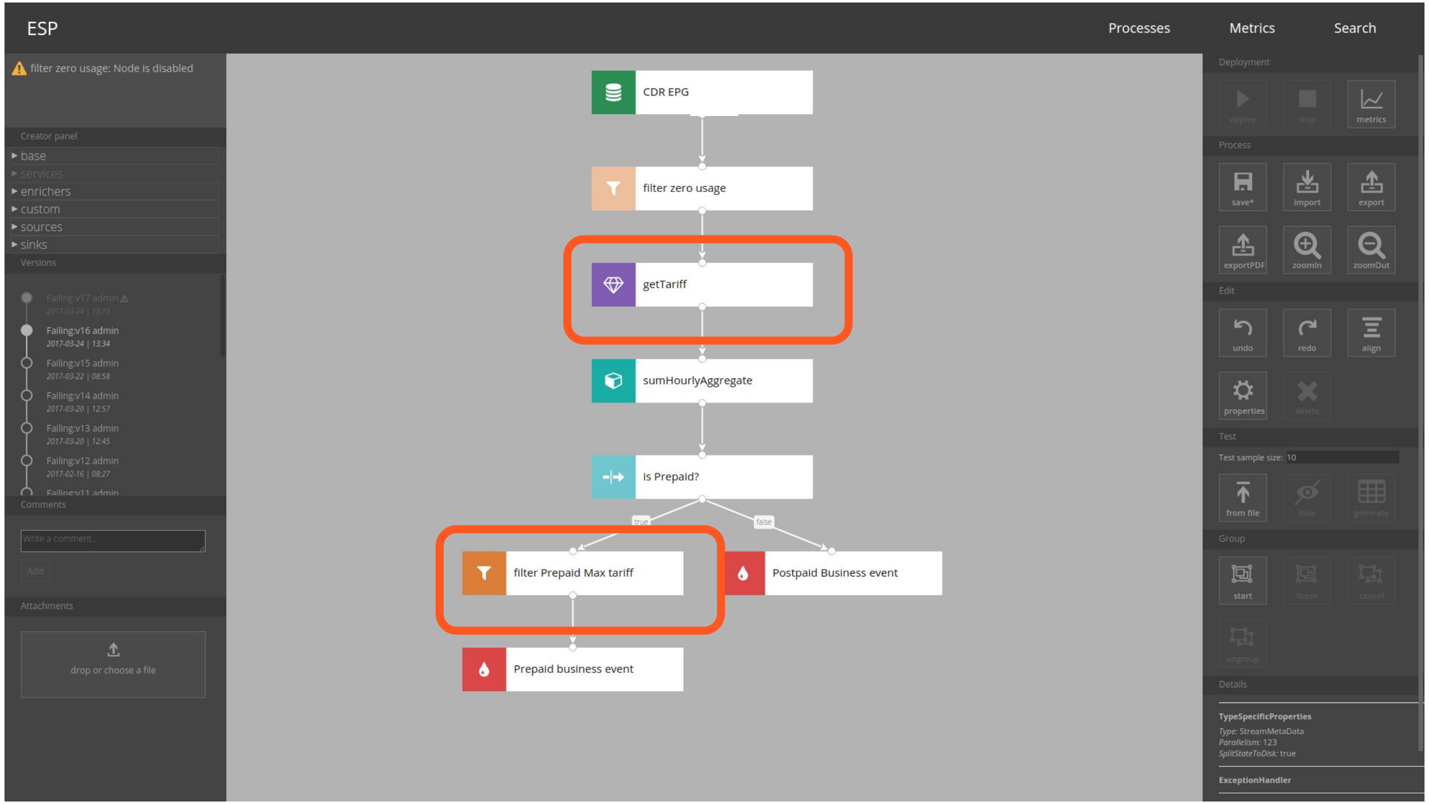
Basics

Architecture



Model + process

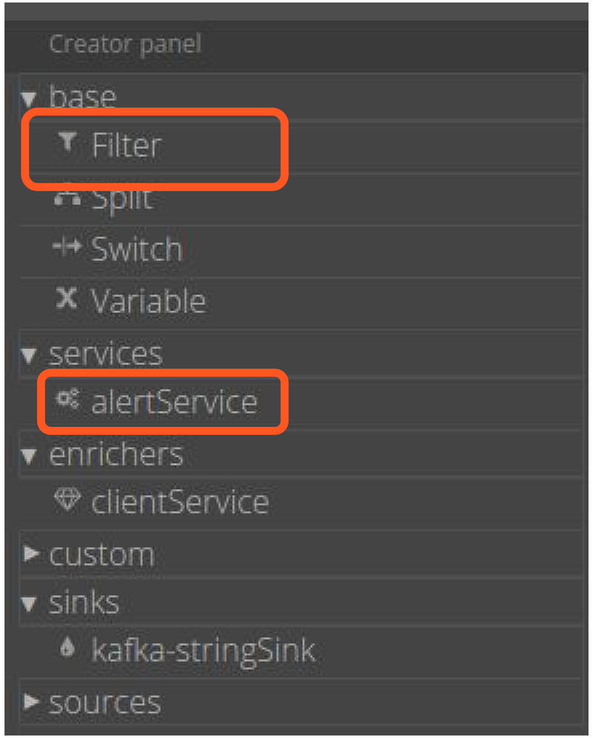




filter zero usage: Node is disabled

Test sample size: 10

TypeSpecificProperties
Type: StreamMetaData
Parallelism: 123
SplitStateToDisk: true


Creator panel
base
Filter
Split
Switch
Variable
services
alertService
enrichers
clientService
custom
sinks
kafka-stringSink
sources

Model

Model

- **Data** - POJOs, case classes
- **Source/Sink** - Flink API + ε
- **Processor/Enricher** - Services

Config creator

```
trait ProcessConfigCreator extends Serializable {

 def services(config: Config) : Map[String, WithCategories[Service]]

 def sourceFactories(config: Config): Map[String, WithCategories[SourceFactory[_]]]

 def sinkFactories(config: Config): Map[String, WithCategories[SinkFactory]]

 ...

}
```

Service/Enricher

```
@MethodToInvoke
def invoke(@ParamName("clientId") clientId: String)
    (implicit ec: ExecutionContext): Future[Client] = {
    ...
}
```

Business rules - expressions

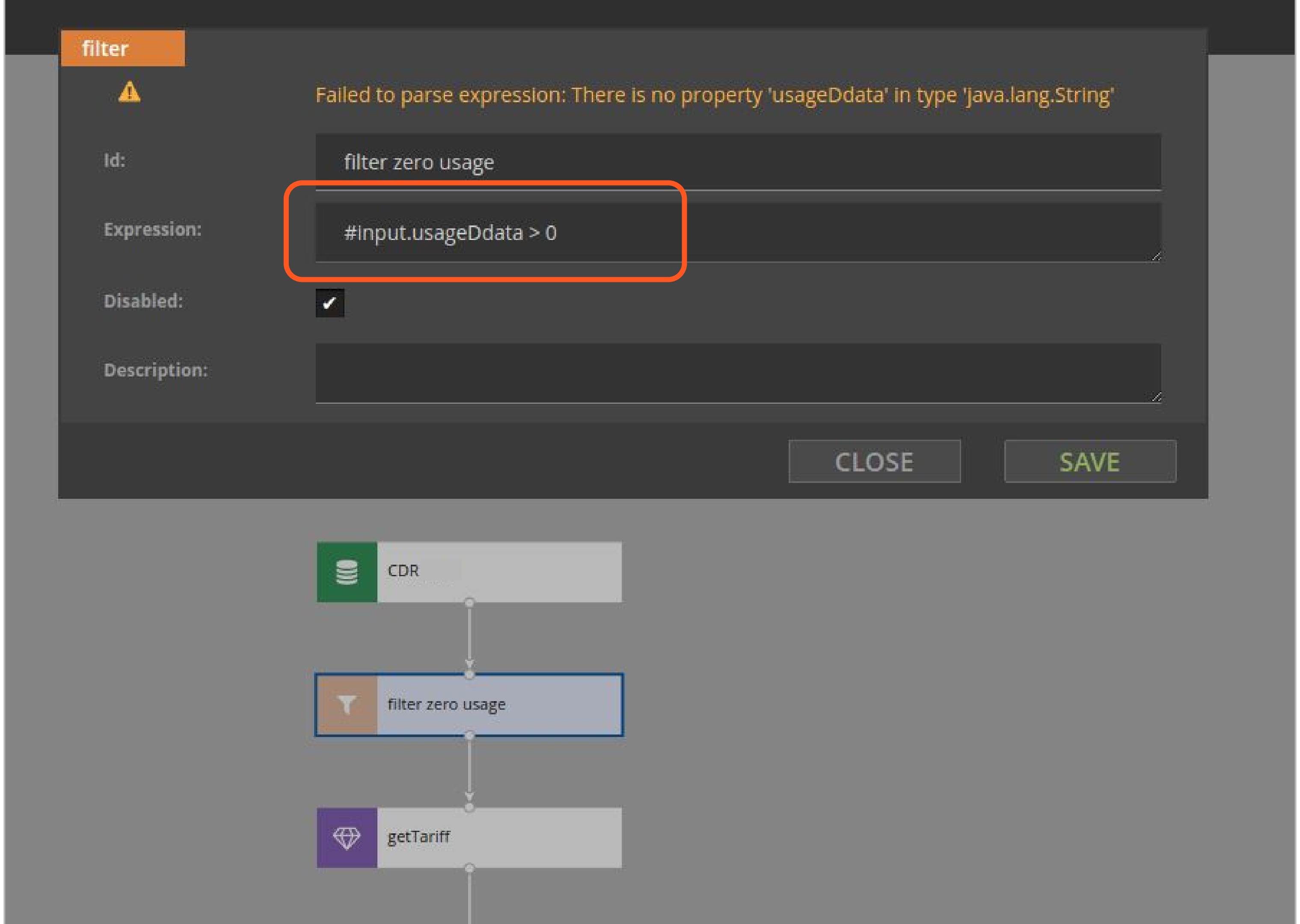
- **Filters**
- **Aggregation** definitions
- **Actions** - mail subject, sms content

How to **define** them...?

Expressions

Spring **E**xpression **L**anguage!

- Accessible enough
- More or less fast enough
- Ability to do basic validation/code-completion


filter

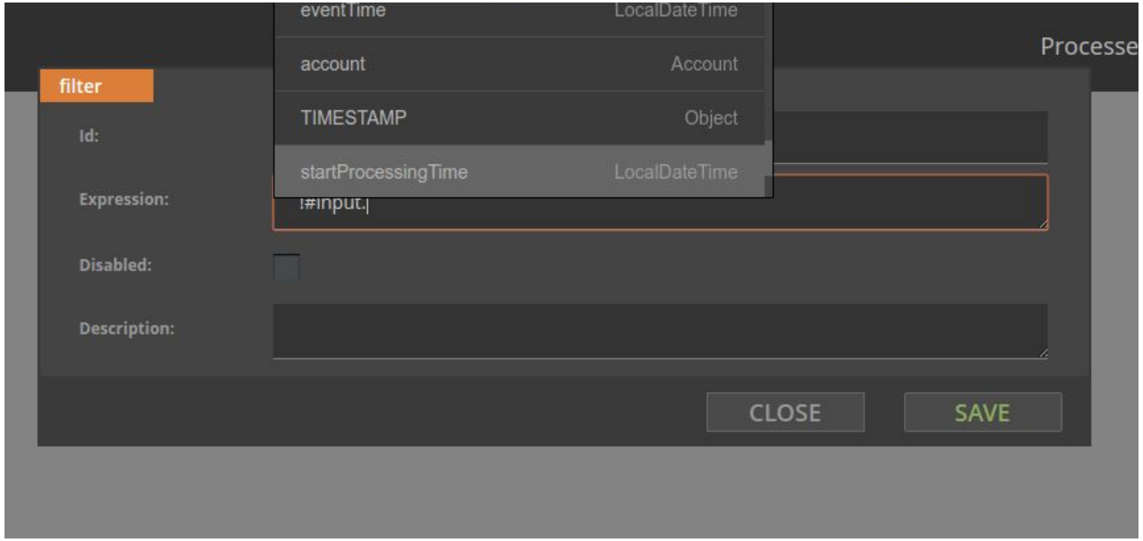
Failed to parse expression: There is no property 'usageDdata' in type 'java.lang.String'

Id: filter zero usage

Expression: #input.usageDdata > 0

Disabled: ✓

Description:



eventTime LocalDateTime
account Account
TIMESTAMP Object
startProcessingTime LocalDateTime
Processe
filter
Id:
Expression:
Disabled:
Description:
CLOSE
SAVE

```
!#input.
```

Limitations of SPEL

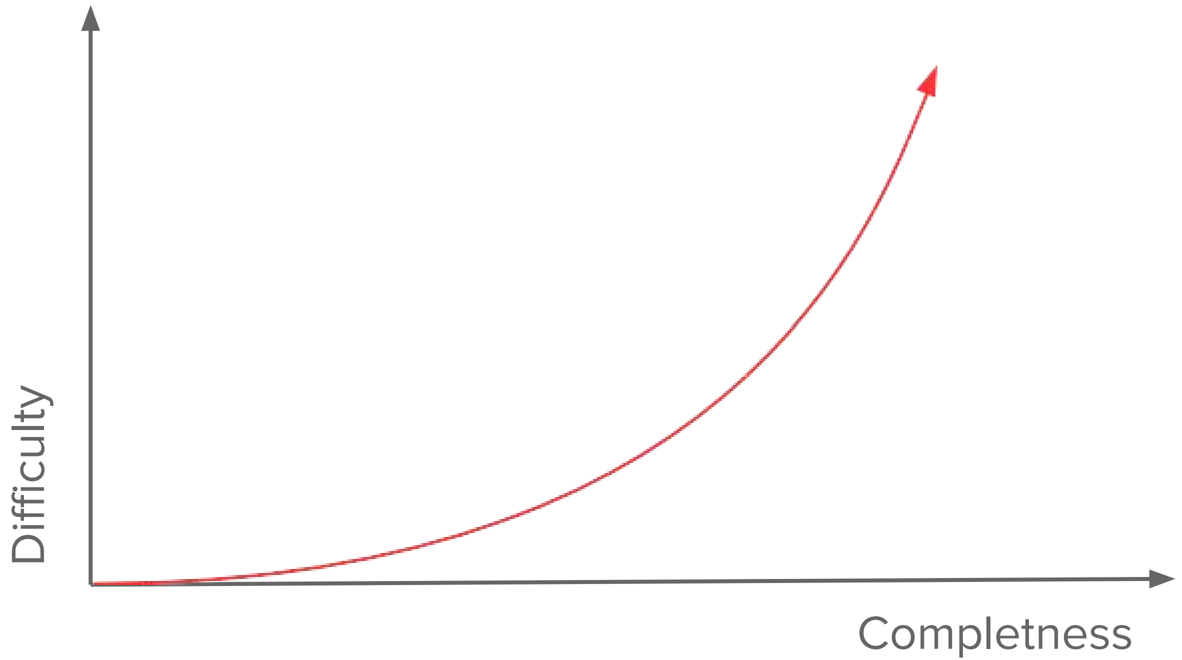
- Speed
- Synchronous
- Type safety??

Advanced?

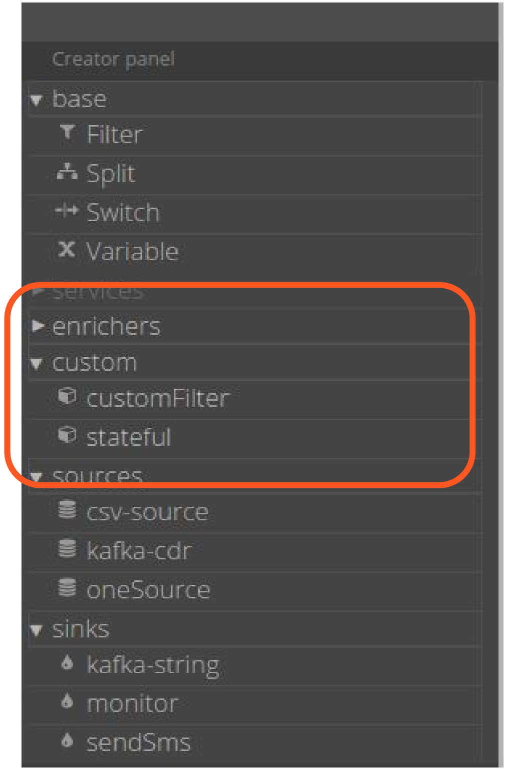
What about...

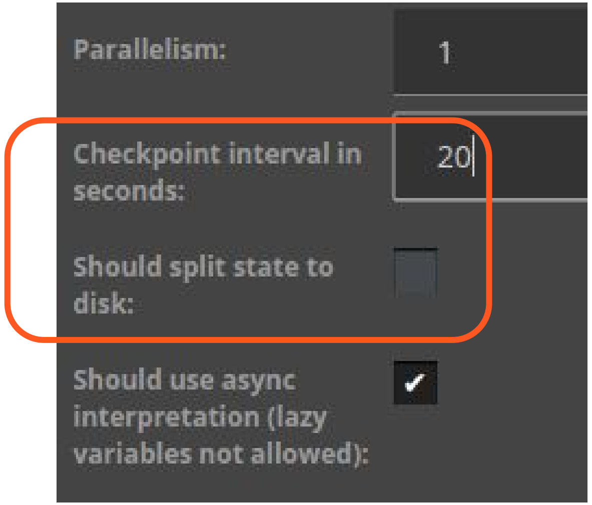
- Windows?
- State?
- Joins?
- Other Flink goodies?

Knowing where to stop



Knowing where to stop





Parallelism: 1

Checkpoint interval in seconds: 20

Should split state to disk: ☐

Should use async interpretation (lazy variables not allowed): ✓

Custom stream transformer - POFO

```
@MethodToInvoke(returnType = classOf[EventCount])
def execute(@ParamName("key") key: LazyInterpreter[String],
            @ParamName("length") length: String) =

  FlinkCustomStreamTransformation((start: DataStream[InterpretationResult]) => {

      start.keyBy(key.syncInterpretationFunction)
        .transform("eventsCounter", new CounterFunction(lengthInMillis))


})
```

Tests/Ops

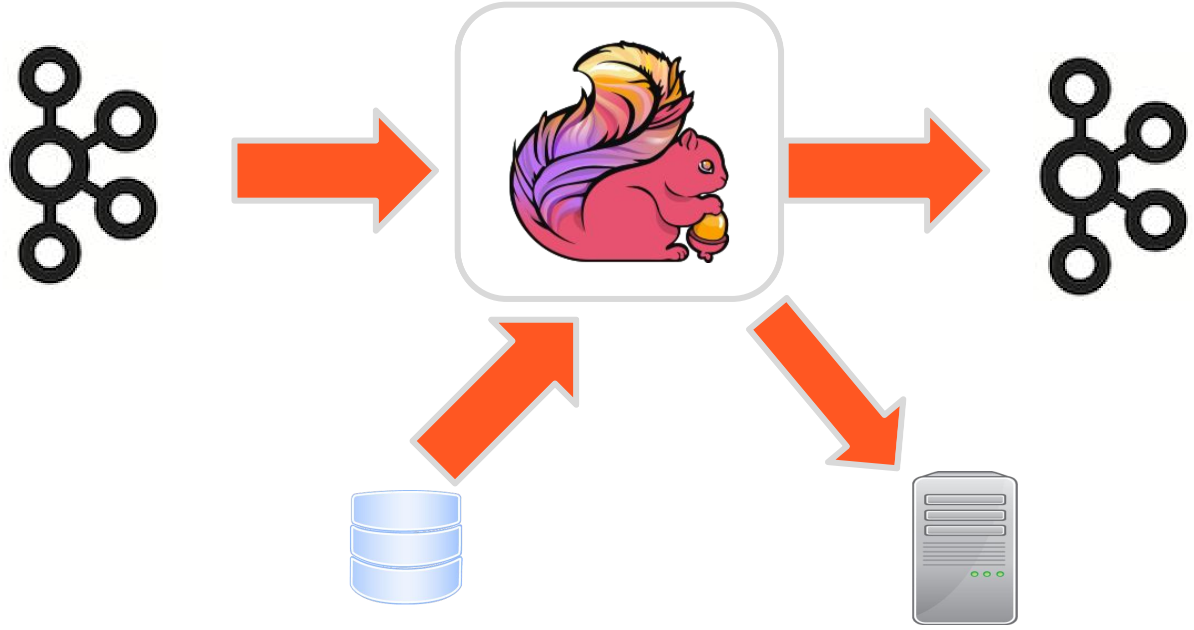
“Convincing” to test

We’ll do it **“our” way**:

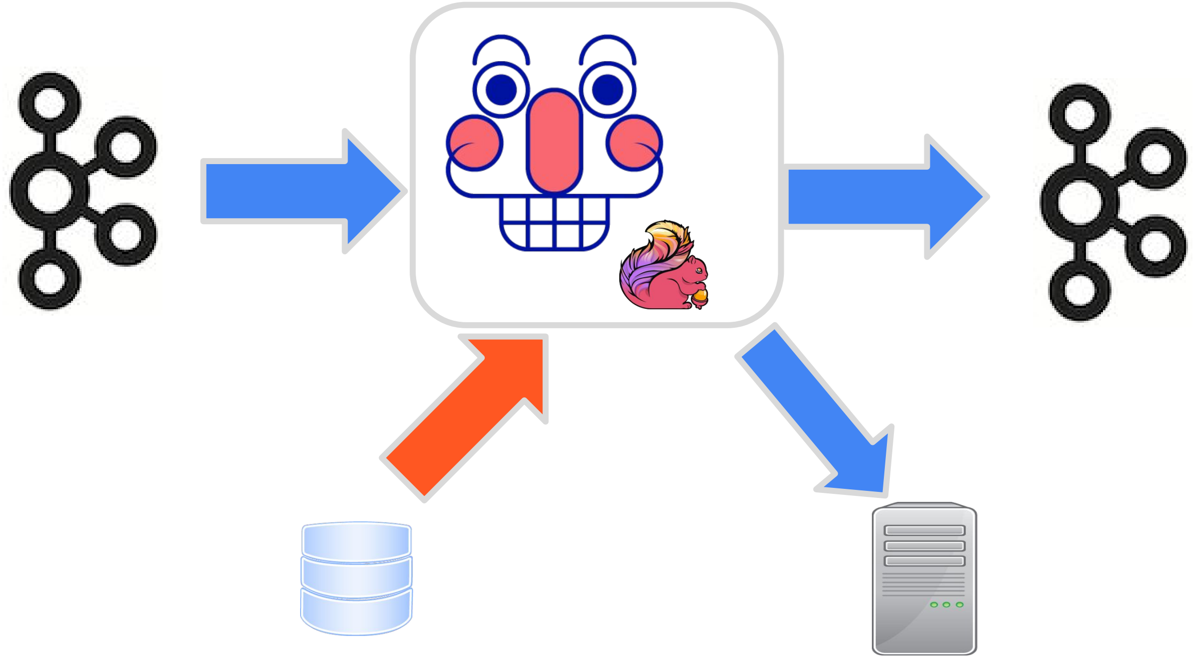
first we run on production,

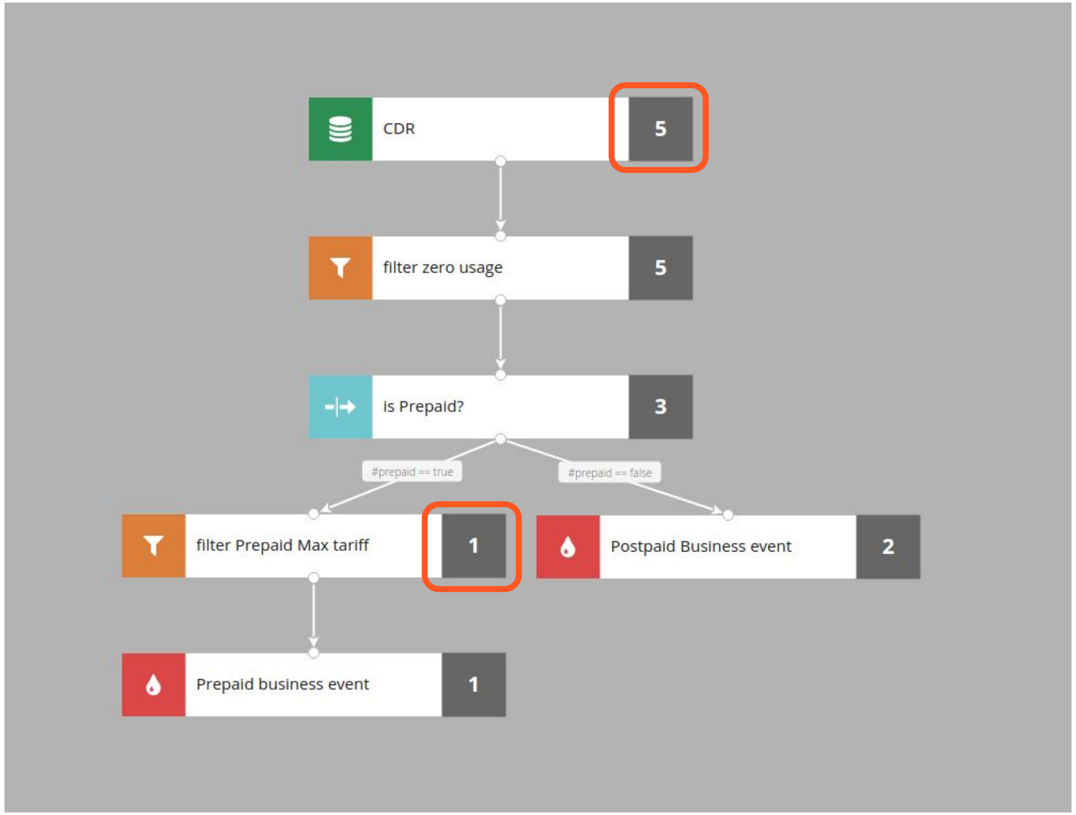
then deploy to test

Tests - running

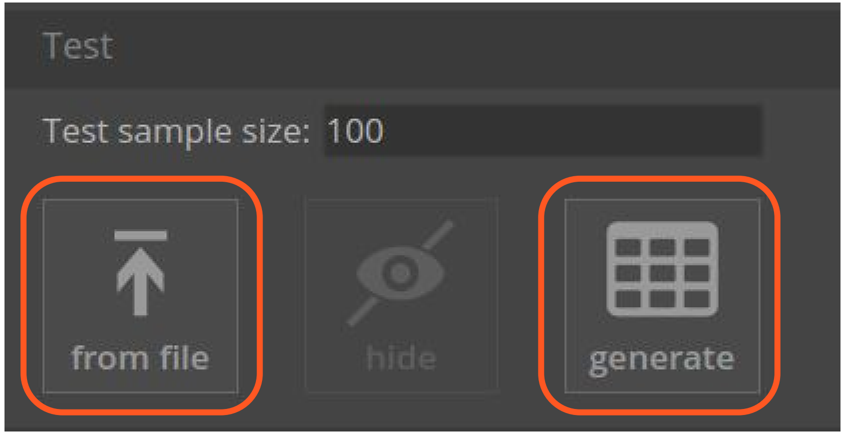


Tests - running




CDR
5
filter zero usage
5
is Prepaid?
3
#prepaid == true
#prepaid == false
filter Prepaid Max tariff
1
Postpaid Business event
2
Prepaid business event
1

Generate test data

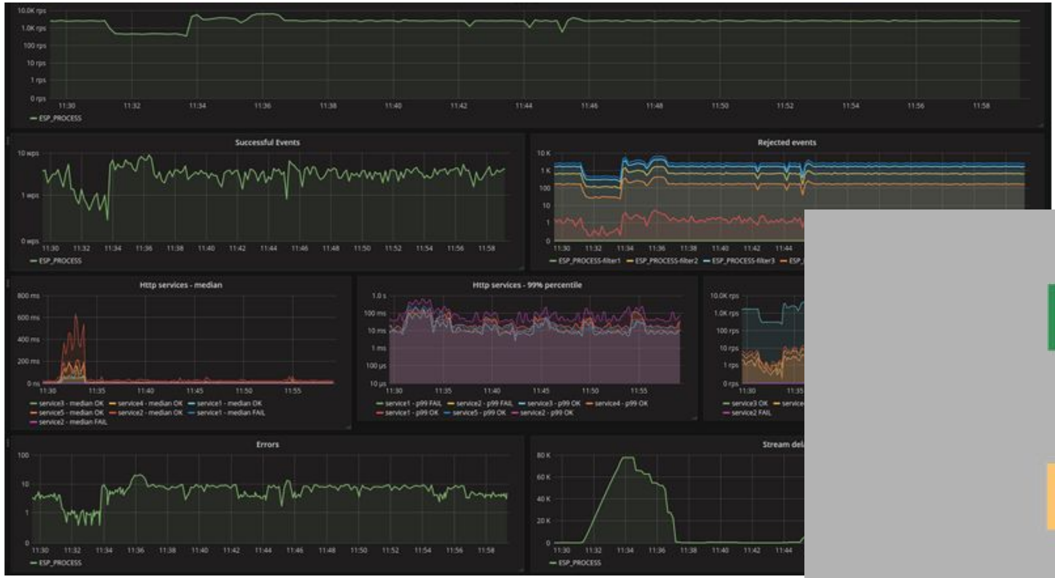

Test

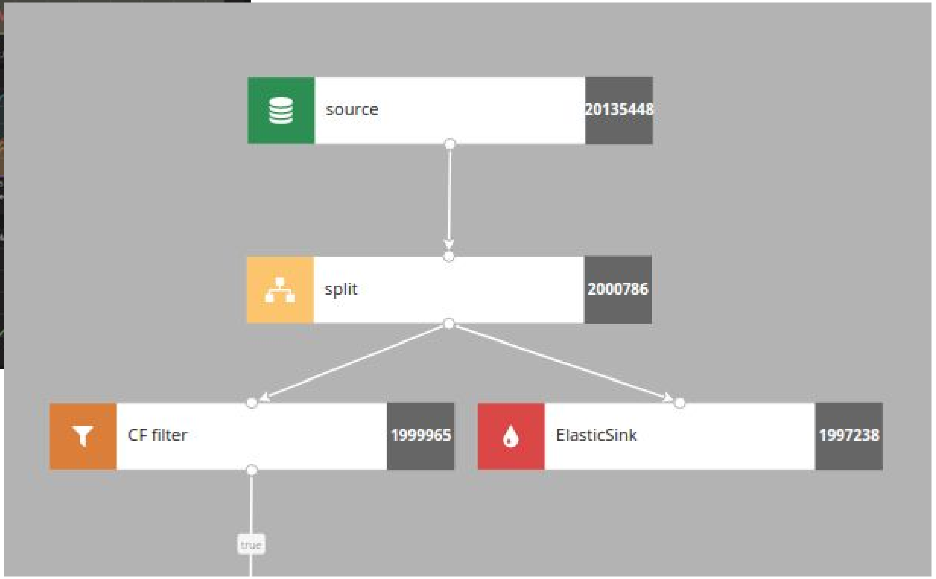
Test sample size: 100

Taking control

- Do we **process** data?
- What is **latency**?
- Where do we **filter** out events?
- Are there **errors**?

Taking control



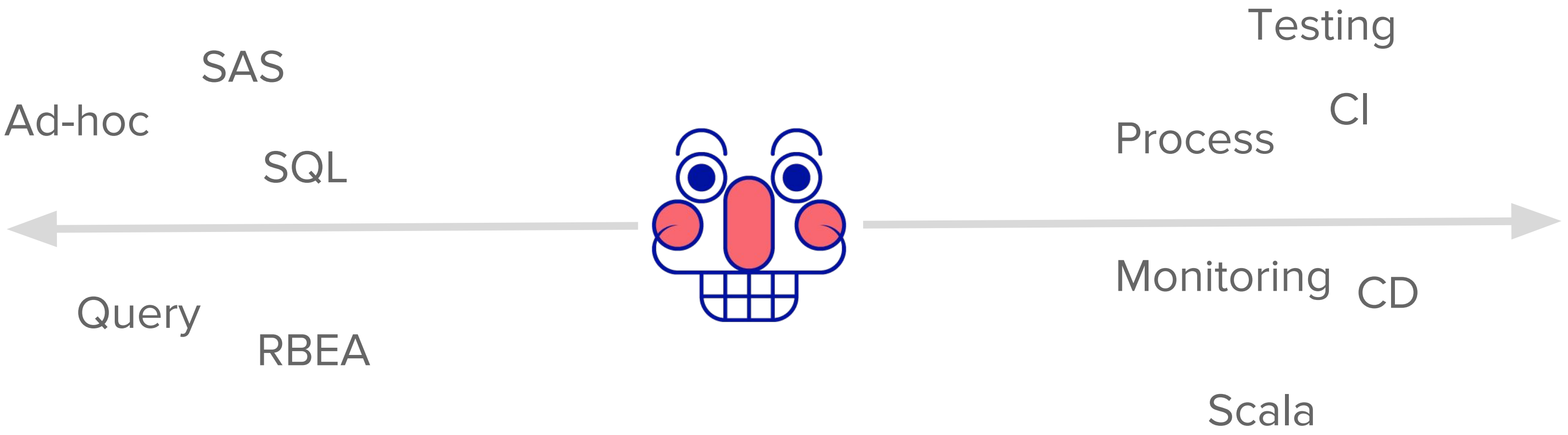


DEMO

Where can I use it?

- Telco
- Banking
- Media
- RTM
- Fraud detection

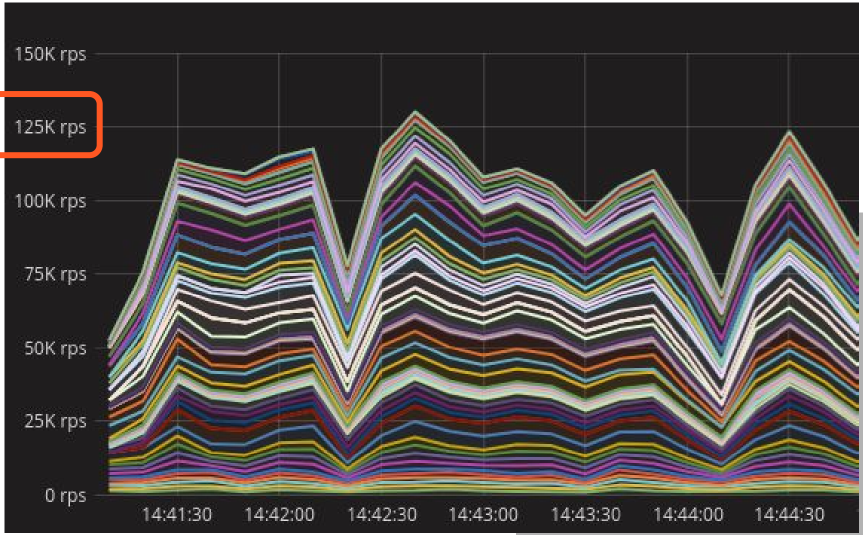
Adhoc jobs vs process

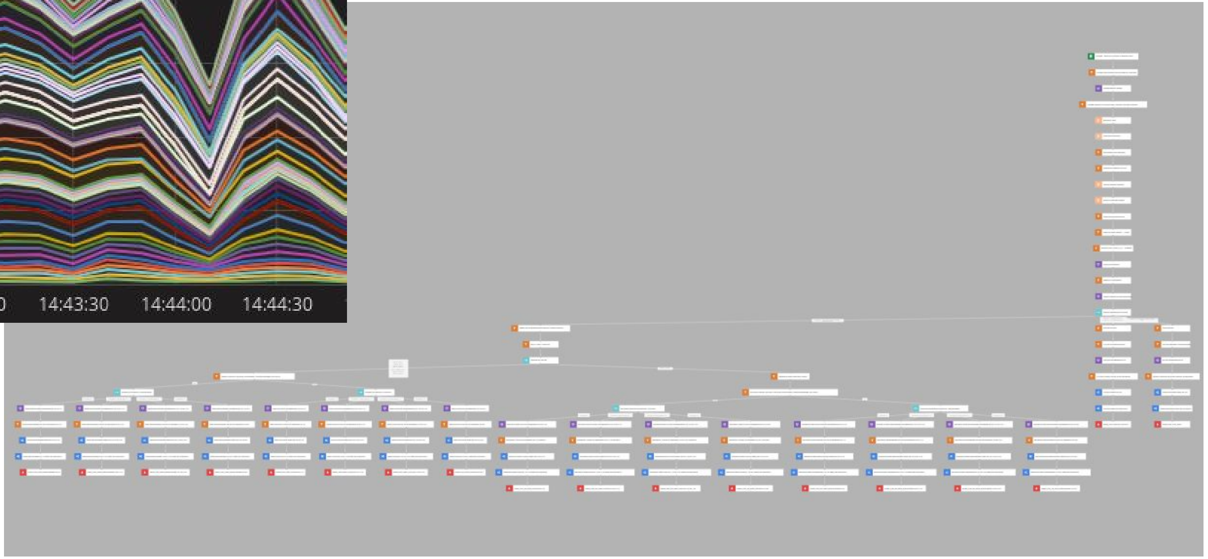


Does it work in production?

- **1 year** in production
- One of largest **Polish telco**’s
- **RTM** + **Fraud** detection

Does it work in production?





ΝΕΑ ΔΗΜΟΚ

Road ahead?

- **Asynchronous** services
- **Pluggable** security
- **Define** model via GUI/Schema registry

Road ahead?

- Integration with **CEP** and **SQL**
- **DAGs** instead of Trees in UI?
- **Governance**?
- Nussknacker as a **Service**?

Try it out!

NUSSKNACKER

https://github.com/touk/nussknacker

Thanks

@mpproch
@touk_pl