Stream Processing for the Practitioner:

Blueprints for Common Stream Processing Use Cases with Apache Flink®

Konstantinos Kloudas, Software Engineer @ data Artisans
slides by Aljoscha Krettek, Software Engineer @ data Artisans

dataArtisans

About Data Artisans

PLATFORM

Original creators of Apache Flink®

Open Source Apache Flink + dA Application Manager

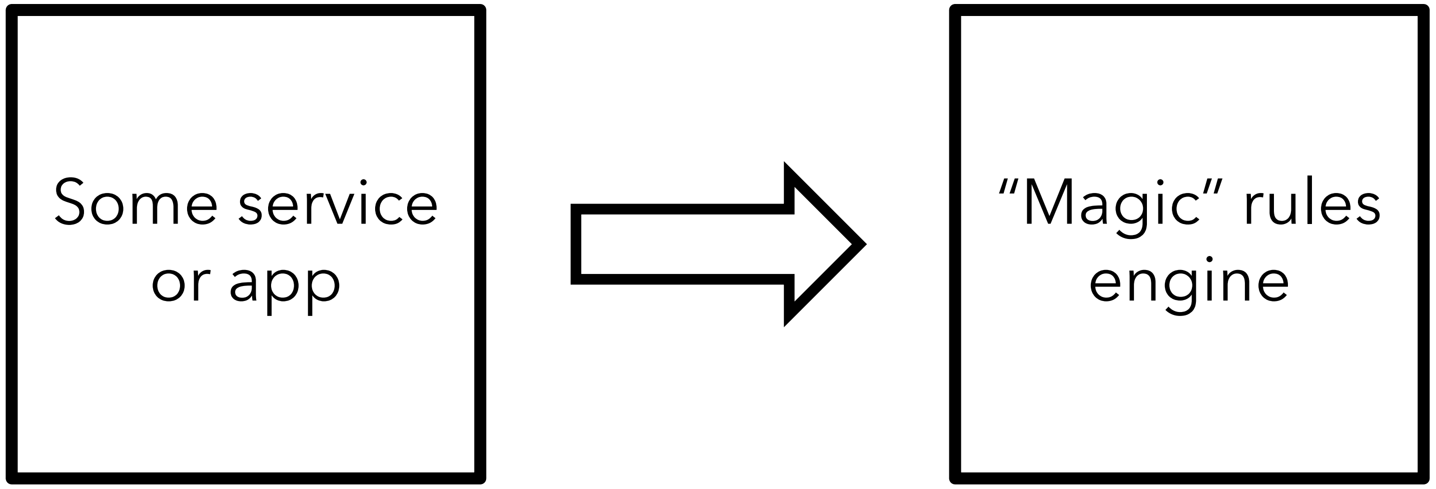
What is stream processing and why is it useful?

Use Case: "Suspicious Behaviour" Detection

Some service or app

- Dropbox, Google Suite, Box
  - Sharing, accessing, and modifying data produces events that we can/want to analyse
- Banking
  - We monitor all transactions, know the user data

Use Case: "Suspicious Behaviour" Detection



- Spits out alerts when suspicious stuff is happening
  - "More than 10 failed login attempts"
  - "Sharing more than 100 files within 1 Hour"
  - "Impossible Travel"/"Magic carpet travel"
  - "Continuously increasing withdrawal amounts"

Use Case: "Suspicious Behaviour" Detection

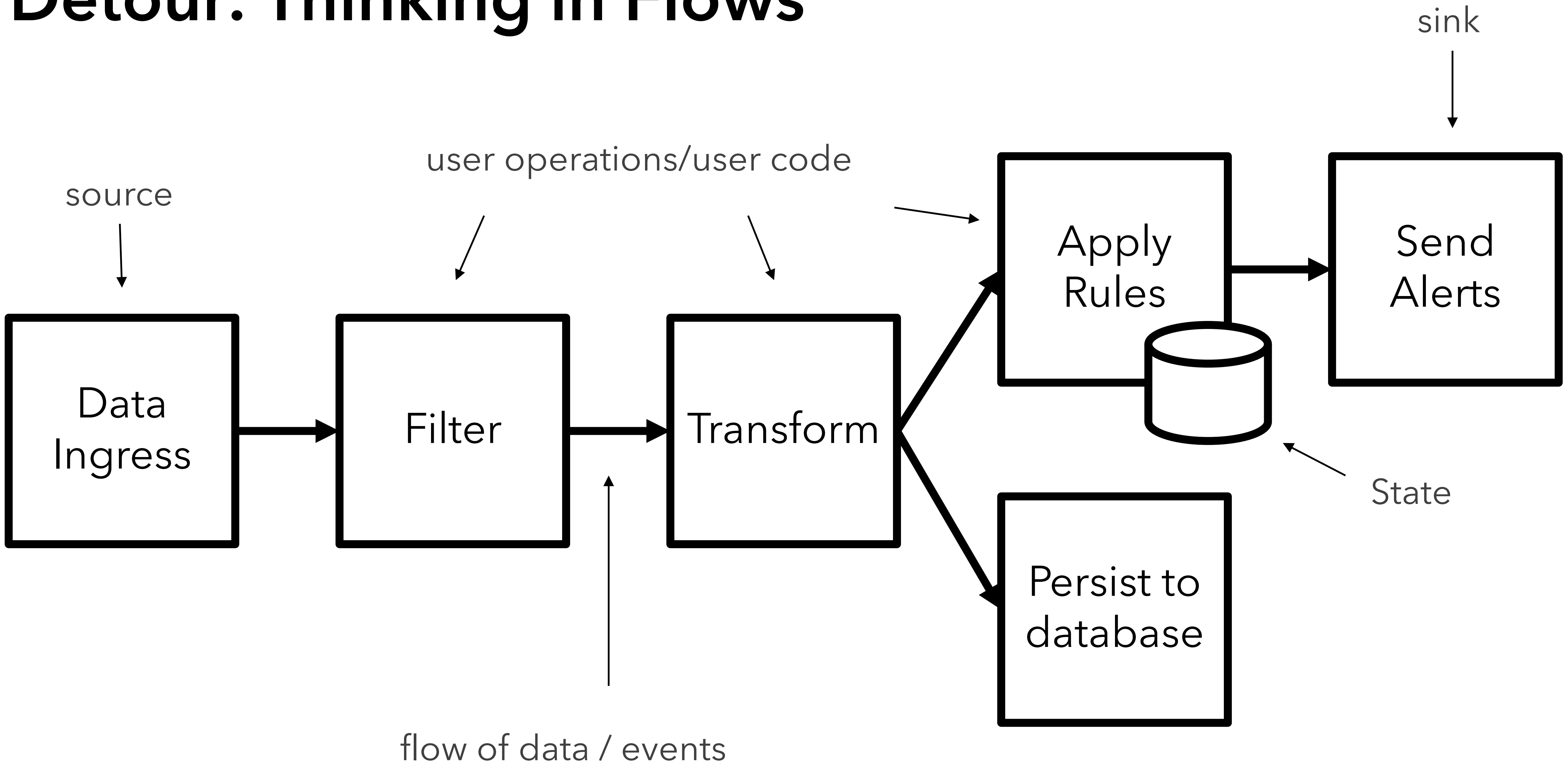
Ok, let me just use a single machine for this!

- What if the load becomes too high? → I'll use a distributed batch processor. (Hadoop MapReduce, Spark, and the like)

- This runs nightly? Isn't the latency very high then? I don't want to wait a day for my alerts 😳

I need alerts in real time → Apache Flink® is a real-time, distributed, stateful, and fault-tolerant stream processor

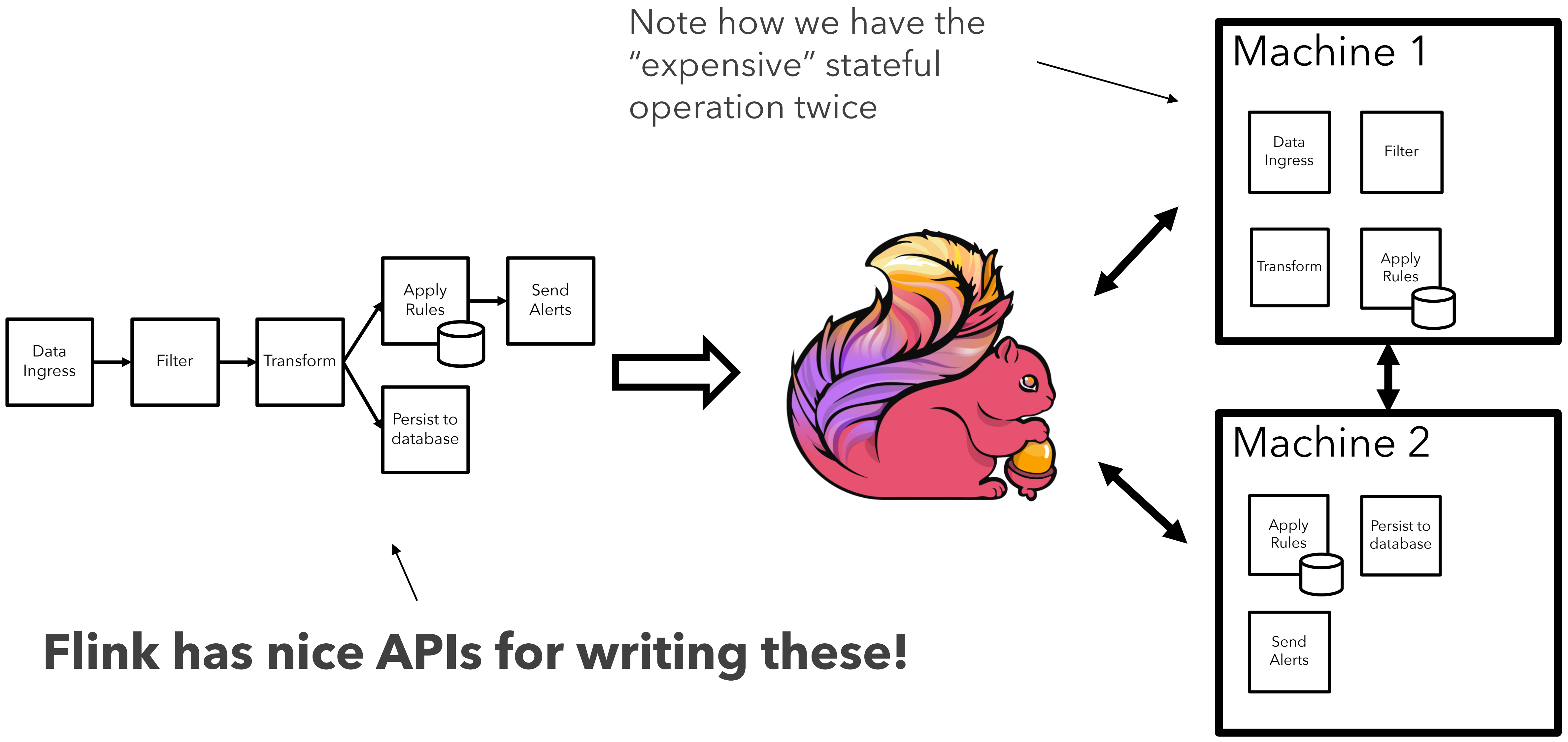
Detour: Thinking in Flows



Why flows and operations?

- This is how the physical world works
- Operations can be **composed** and are **reusable**
- Allows a system (Flink) to take these operations and execute them on different machines
- A system can execute the same operation multiple times on several machines to deal with high workloads

Apache Flink® distributed stateful stream processor



Flink has nice APIs for writing these!

Apache Flink® distributed stateful stream processor

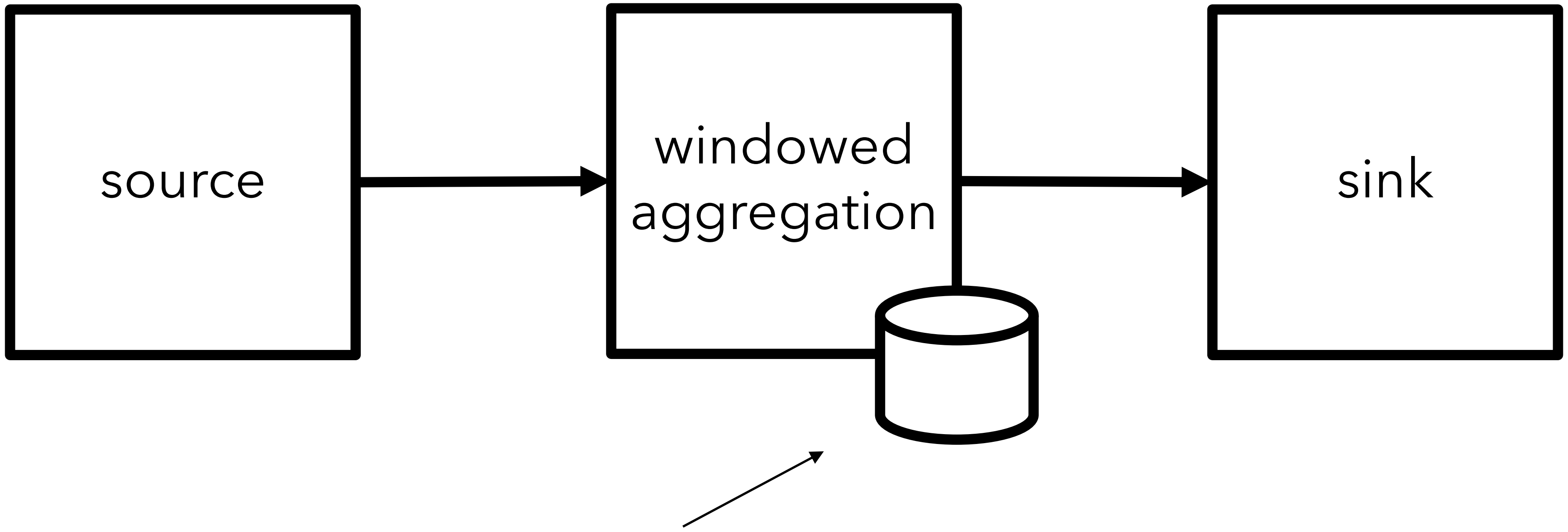
- Questions that a good stream processing system needs to have answers for:
  - What happens when machines fail or when user operations fail?
  - What happens if I move my stateful operations/flows/jobs?
  - What happens if I need to change the schema of the state that operations keep?
  - How can I update framework code while keeping my program state?
  - Same for user code?

Common stream processing blueprints

Blueprint: Aggregation of timestamped data

- Use cases
  - Give me the number of tweet impressions per tweet for every hour/day/…
  - Calculate the average temperature over 10 minute intervals for each sensor in my warehouse
  - Aggregate user interaction data for my website to display on my internal dashboards

Blueprint: Aggregation of timestamped data



state: contents of all the in-flight windows

Blueprint: Aggregation of timestamped data

Some things to look out for.

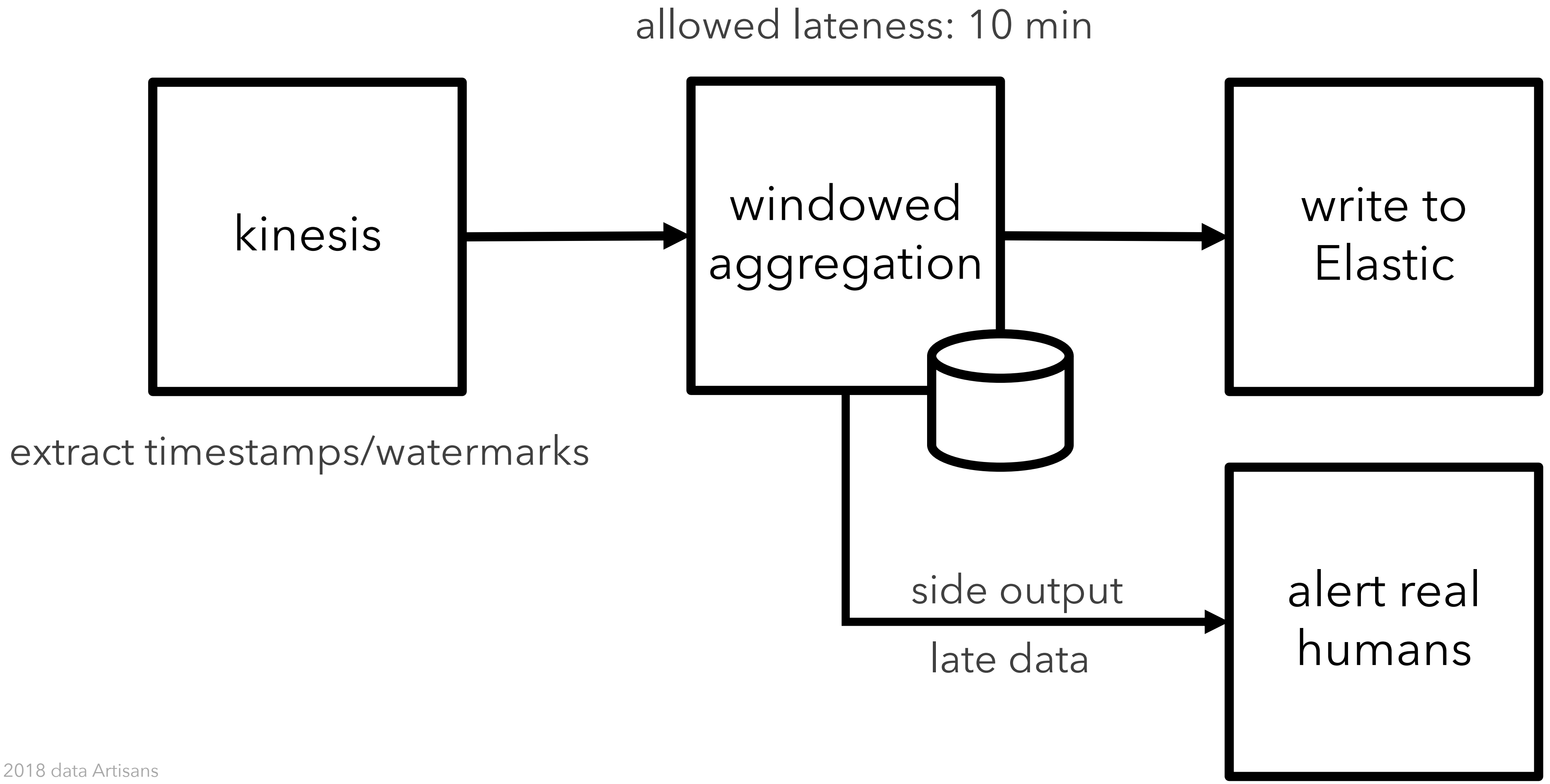
- Do I want to window by event-time or processing time?
- If using event-time, how do I know when my window is “done”?
- What happens if data arrives out of order with respect to their timestamp?
- If using event-time, when is data considered late?
- What should happen with late data?

Blueprint: Aggregation of timestamped data

Flink features to look at.

- Windowing API
- Timestamp assigners/watermark extractors for defining event-time and defining “readiness”
- *Allowed lateness* for defining when data is late
- Side output of late data as a special flow path

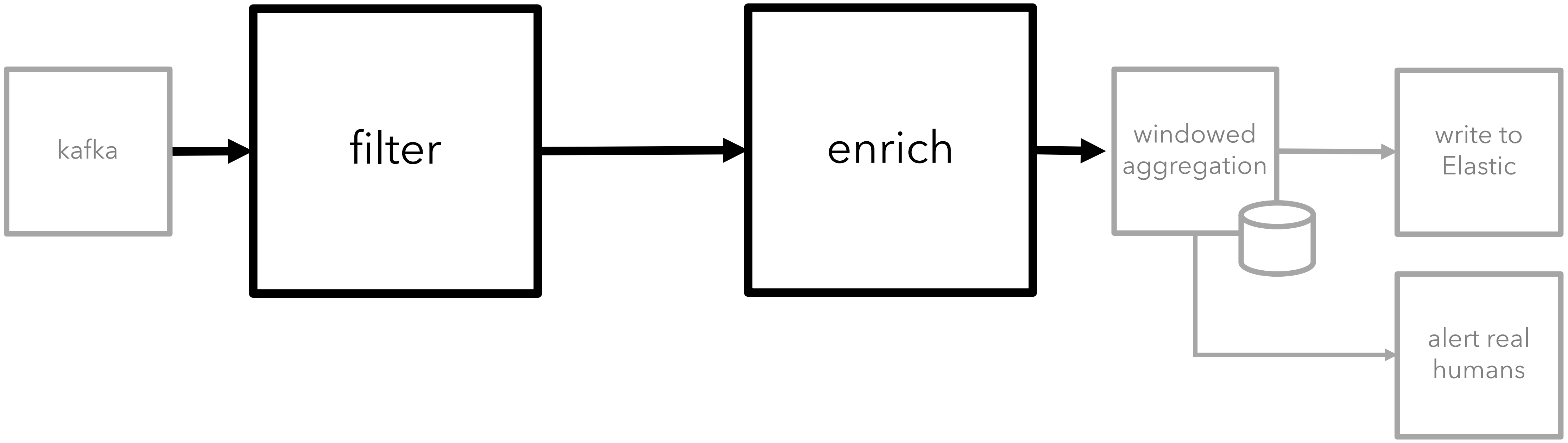
Blueprint: Aggregation of timestamped data



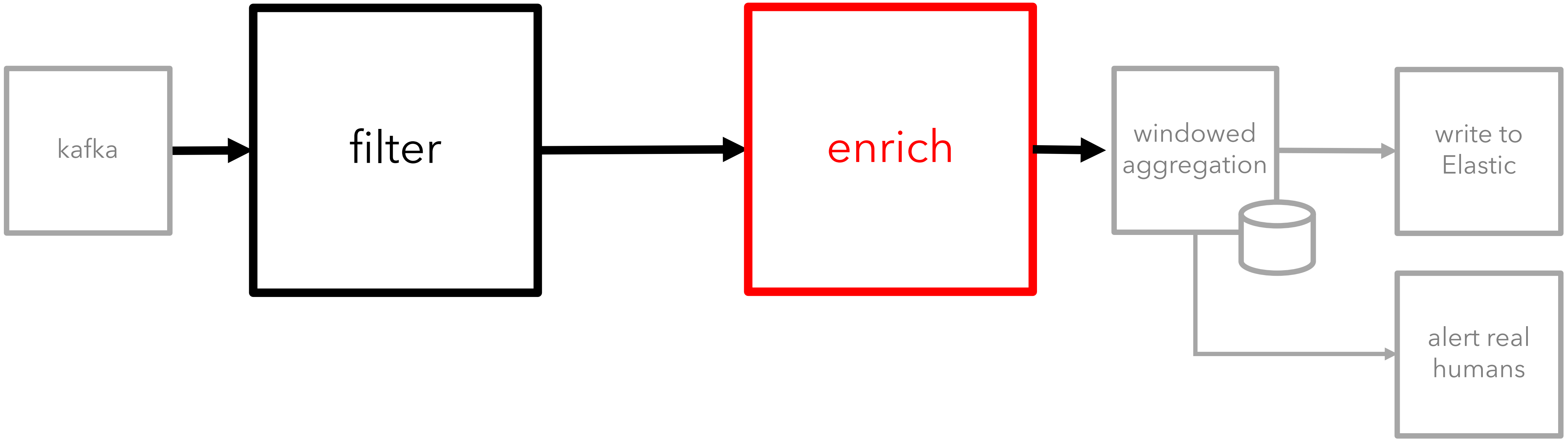
Blueprint: Enriching data with “side input”

- Use cases
  - Enrich user events with known user data
  - Add geolocation information to geotagged events

Blueprint: Enriching data with “side input”



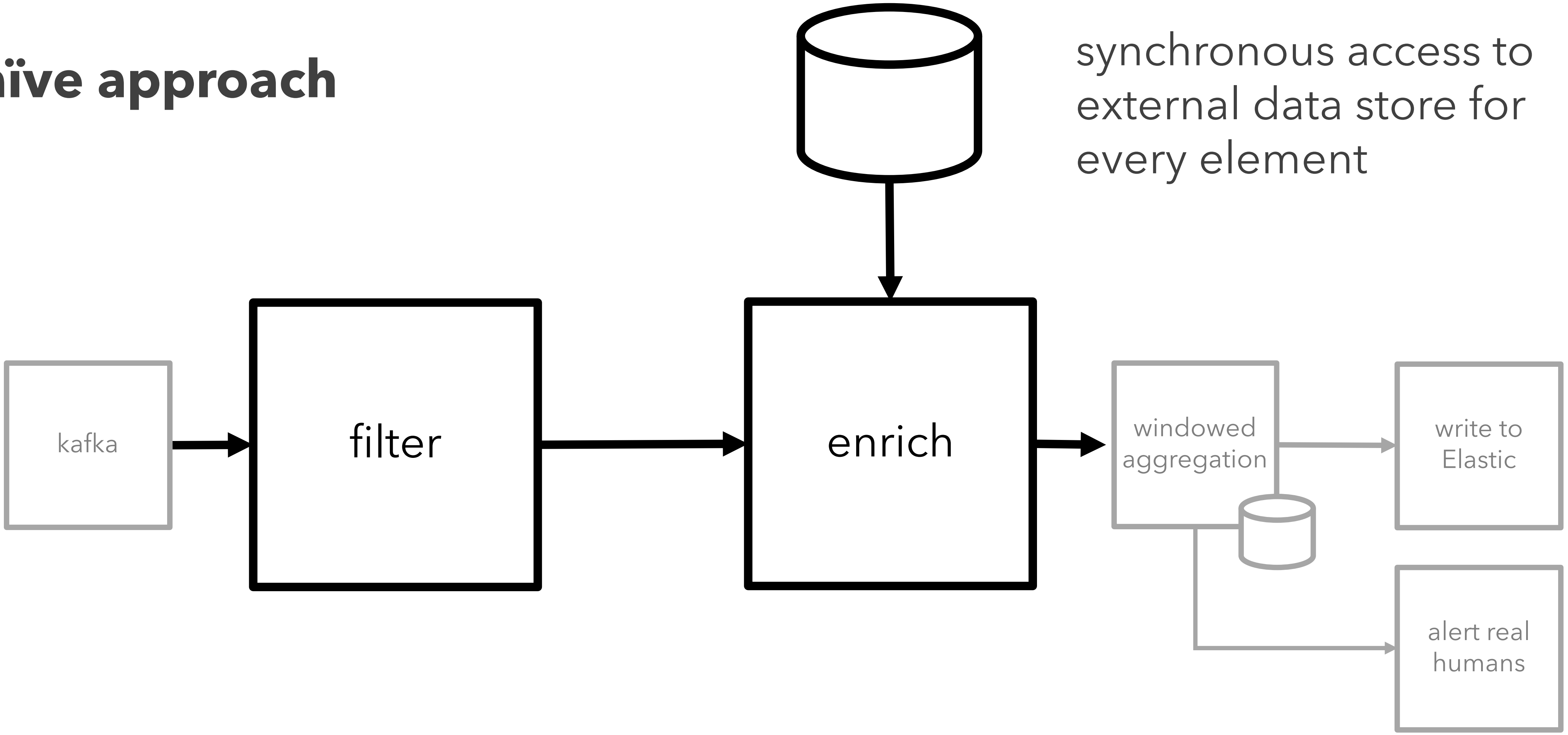
Blueprint: Enriching data with “side input”



Blueprint: Enriching data with “side input”

Naïve approach

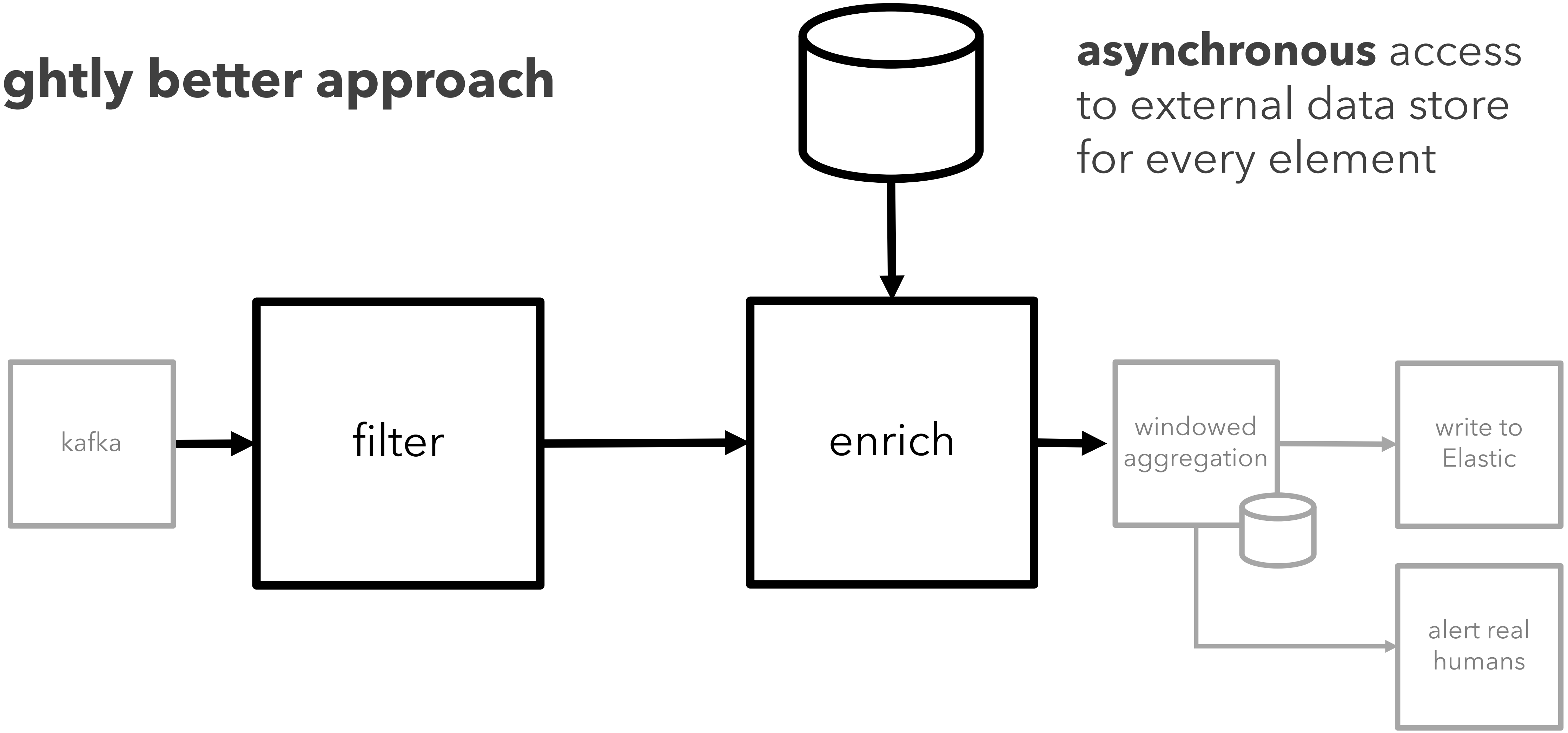
synchronous access to external data store for every element



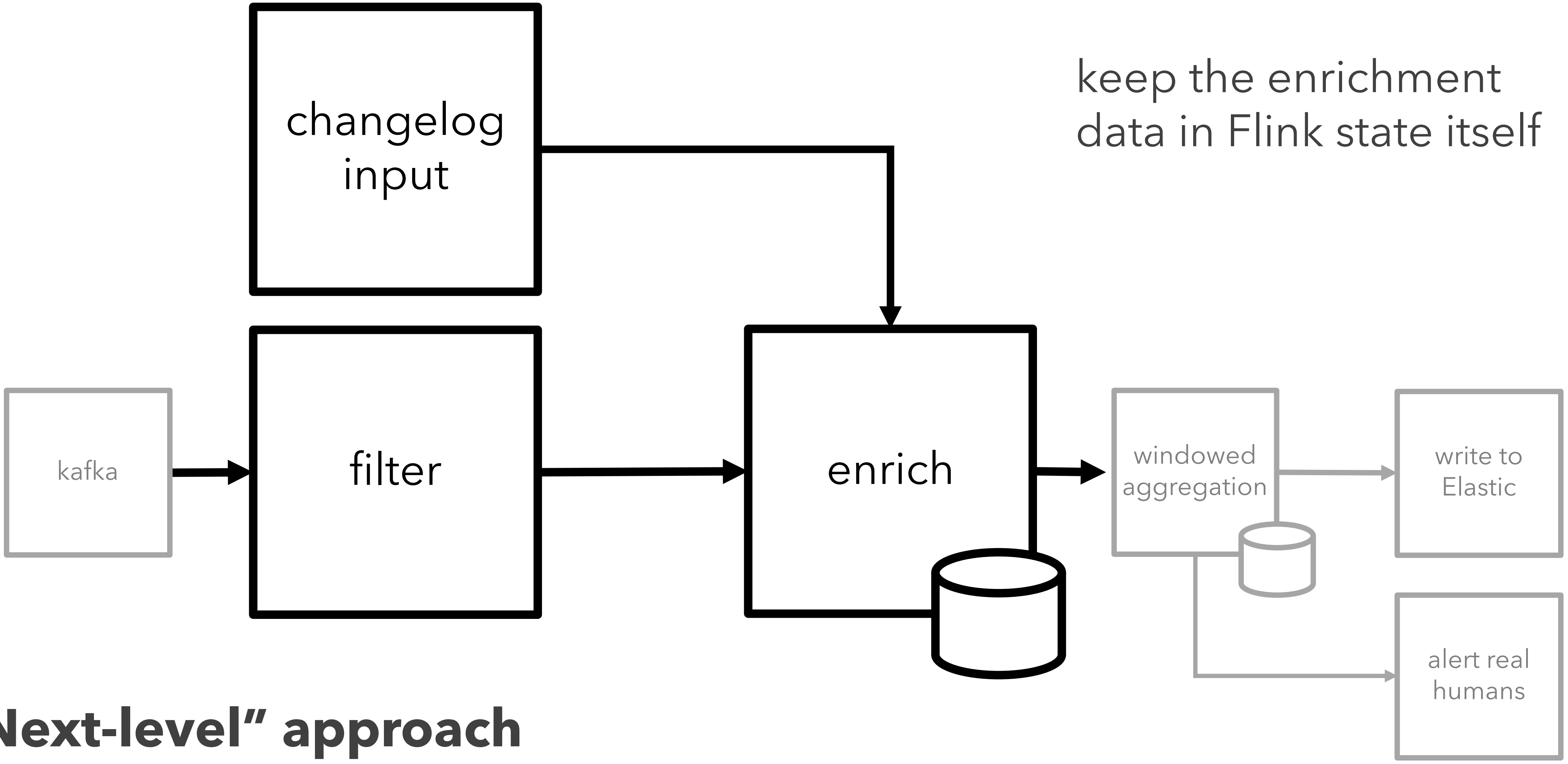
Blueprint: Enriching data with "side input"

Slightly better approach

asynchronous access to external data store for every element



Blueprint: Enriching data with "side input"



keep the enrichment data in Flink state itself

"Next-level" approach

Blueprint: Enriching data with “side input”

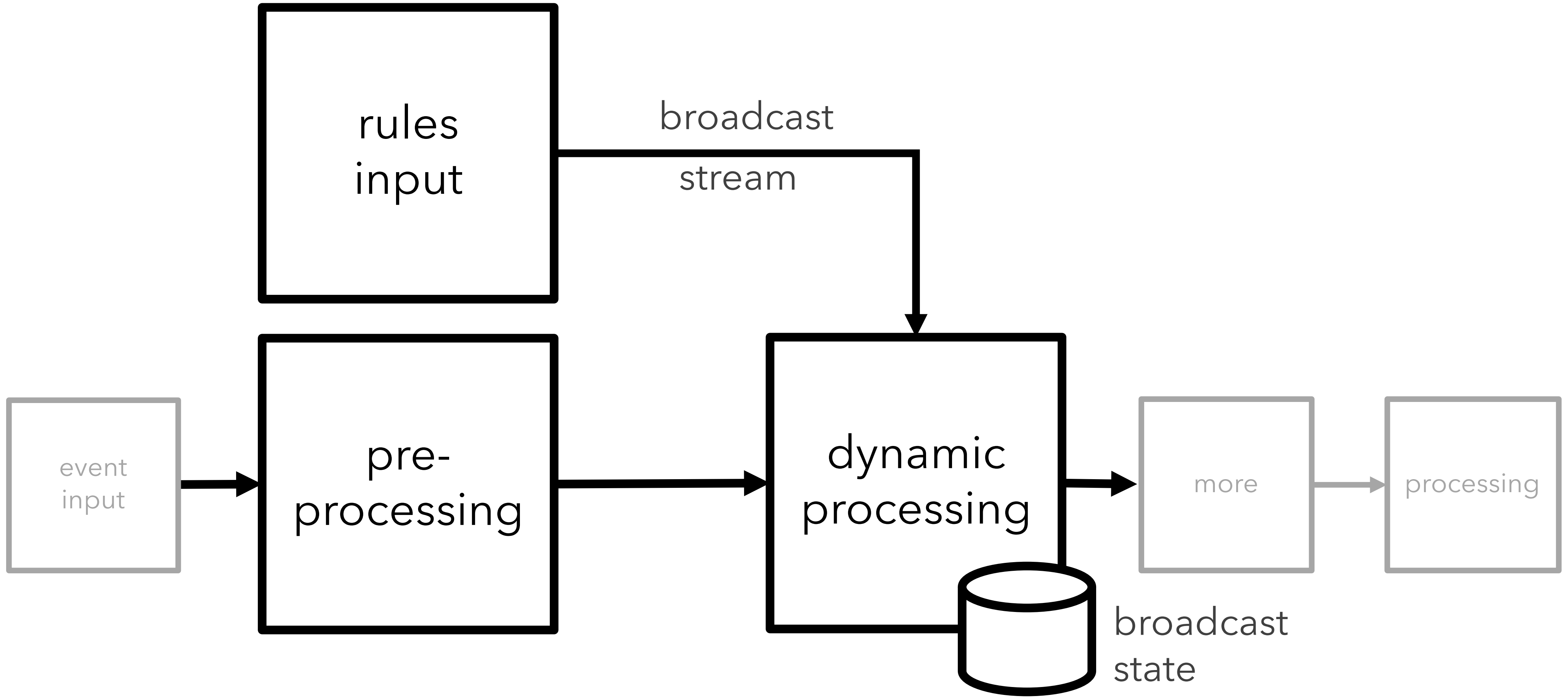
Flink features to look at.

- Regular user functions/operations
- Async I/O operation for more efficient data store accesses
- Two-input operations and stateful operations

Blueprint: Dynamic processing

- Use cases
  - Update of processing rules via DSL, think dynamic fraud-detection rules/policies
  - Live-update of machine learning models

Blueprint: Dynamic processing



Blueprint: Dynamic processing

Flink features to look at.

- ProcessFunction
- Broadcast streams and broadcast state

Closing

Learnings

- For immediate results you probably need a stream processor
- Start thinking in terms of data flows and reusable operations
- Getting state fault-tolerance, and event-time right is tough, check what your stream processor has as answers for those questions
- Flink has your use cases covered

FLINK FORWARD

organized by dataArtisans

The Apache Flink® Conference

Stream Processing | Event Driven | Real Time

3 SEPTEMBER 2018: TRAINING
4-5 SEPTEMBER 2018: CONFERENCE

BERLIN, GERMANY

Register at berlin.flink-forward.org

Early bird prices available until June 22

@dataArtisans #flinkforward

Thank you!

aljoscha@apache.org
kkloudas@apache.org
@dataArtisans
@ApacheFlink

We are hiring!

data-artisans.com/careers

dataArtisans

Hardened at scale

UBER

Streaming Platform Service
billions messages per day
A lot of Stream SQL

NETFLIX

Streaming Platform as a Service
3700+ container running Flink,
1400+ nodes, 22k+ cores, 100s of jobs,
3 trillion events / day, 20 TB state

Alibaba Group

1000s jobs, 100.000s cores,
10 TBs state, metrics, analytics,
real time ML,
Streaming SQL as a platform

ING

Fraud detection
Streaming Analytics Platform

Powered by Apache Flink

Alibaba Group
ING
NETFLIX
UBER
HUAWEI
COMCAST

King
CISCO
accenture
Expedia
ebay
Capital One

DELL EMC
yelp
lyft
here
DTRB
zalando

MediaMath
amadeus
LINE
Telefónica
bouygues TELECOM
Lightbend

bol.com
SK telecom
OVH.com
relayr.
BetterCloud
otto group