Data pipelines with PostgreSQL & Kafka

Oskari Saarenmaa

PostgresConf US 2018 - Jersey City

Agenda

1. Introduction
2. Data pipelines, old and new
3. Apache Kafka
4. Sample data pipeline with Kafka & PostgreSQL
5. SQL for everything
6. Data processing inside a Kafka cluster
7. Q & A

This presentation was created by Aiven Ltd - https://aiven.io.
Product and vendor logos used for identification purposes only.

Speaker

- CEO, co-founder @ Aiven, a cloud DBaaS company
- Previously: database consultant, software architect
- PostgreSQL user since 1999 (rel 6.4)
    - Contributed bug fixes and features to core
    - Worked on extensions and tooling in the PG ecosystem

@OskariSaarenmaa

UpCloud

DigitalOcean

Microsoft Azure

Google Cloud Platform

amazon web services

Aiven

- Independent Database as a Service provider
- Based in Helsinki and Boston
- 8 database systems available in 70+ regions around the world
- First to offer PostgreSQL 10 as a service!

https://aiven.io

@aiven_io

Data pipelines

From Wikipedia, “pipeline (computing)”:

> **“**pipeline is a set of data processing elements connected in series, where the output of one element is the input of the next one. The elements of a pipeline are often executed in parallel or in time-sliced fashion. ….. The name 'pipeline' comes from a rough analogy with physical plumbing.**”**

- Modern data pipelines are used to ingest & process vast volumes of data in real time
- Real time processing of data as opposed to traditional ETL / batch modes

Common components of a data pipeline

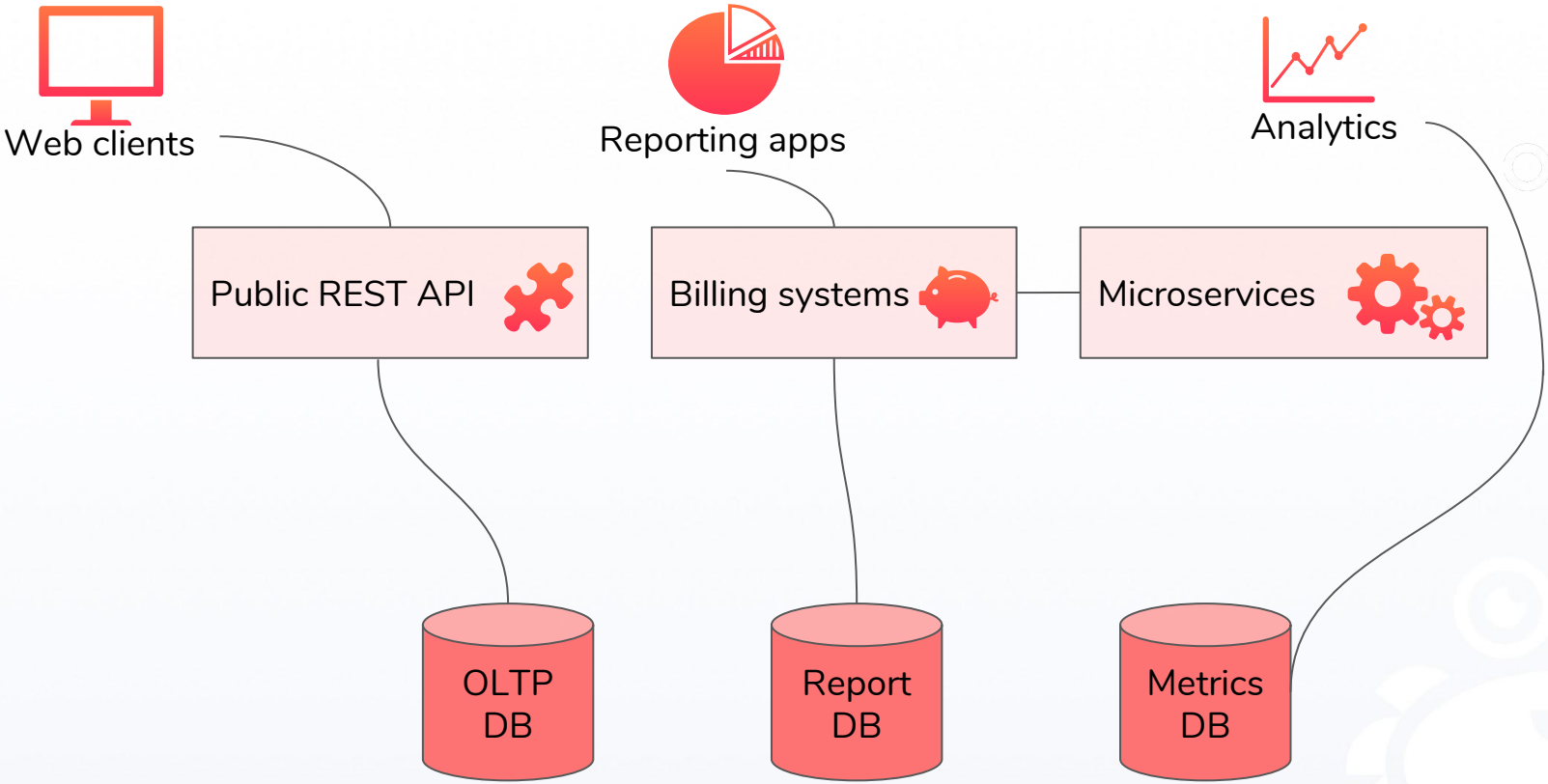
Typical parts of a data pipeline

- Data ingestion
- Filtering
- Processing
- Querying the data
- Data warehousing
- Reprocessing capabilities

Typical requirements

- Scalability
  - Billions of messages and terabytes of data 24/7
- Availability and redundancy
  - Across physical locations
- Latency
  - Real-time, batch?
- Platform support

“Traditional” data flow model

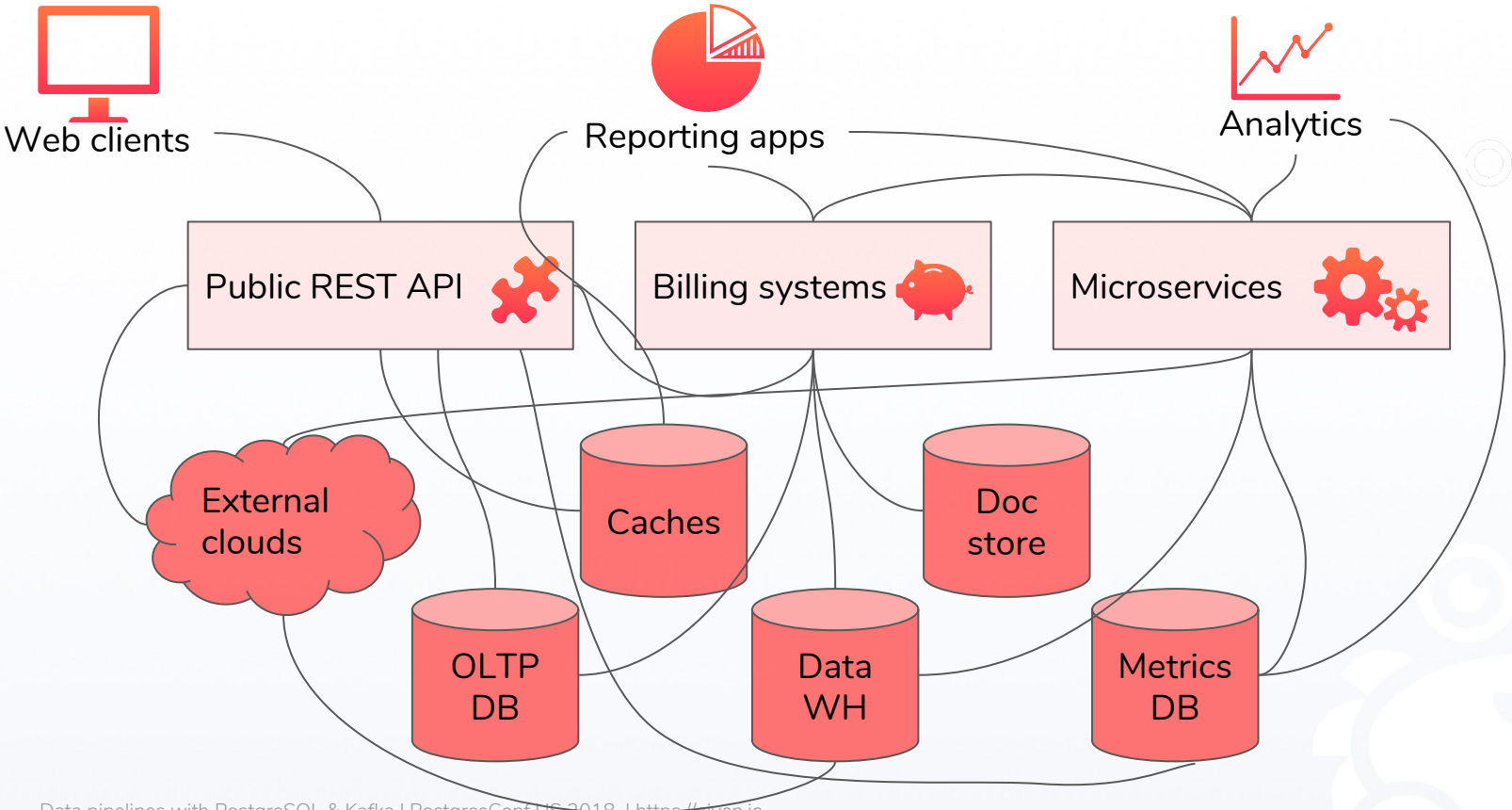


“Traditional” data flow model

“Hmm, let’s just connect a couple of our tools, it’s simple, right?”

```
$ curl api.example.com | filter.py | psql
```

“Traditional” data flow model



Apache Kafka

Apache Kafka is an open source stream processing platform.

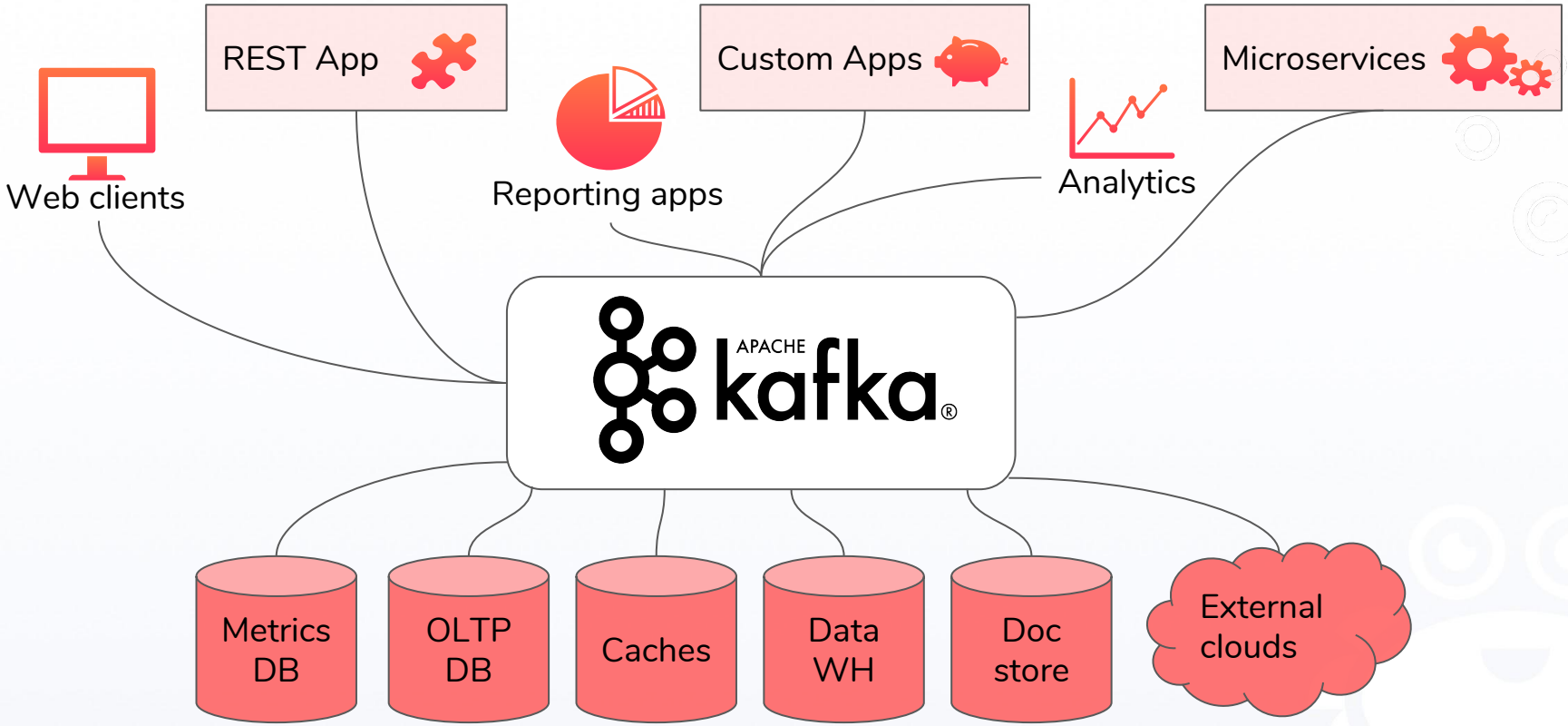
> "The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds."

Originally developed by LinkedIn, open sourced in 2011, now a top-level Apache project. Nowadays used by e.g. New York Times, Pinterest, Zalando, Airbnb, Shopify, Spotify and many others for event streaming. [See https://kafka.apache.org/powered-by for more.]

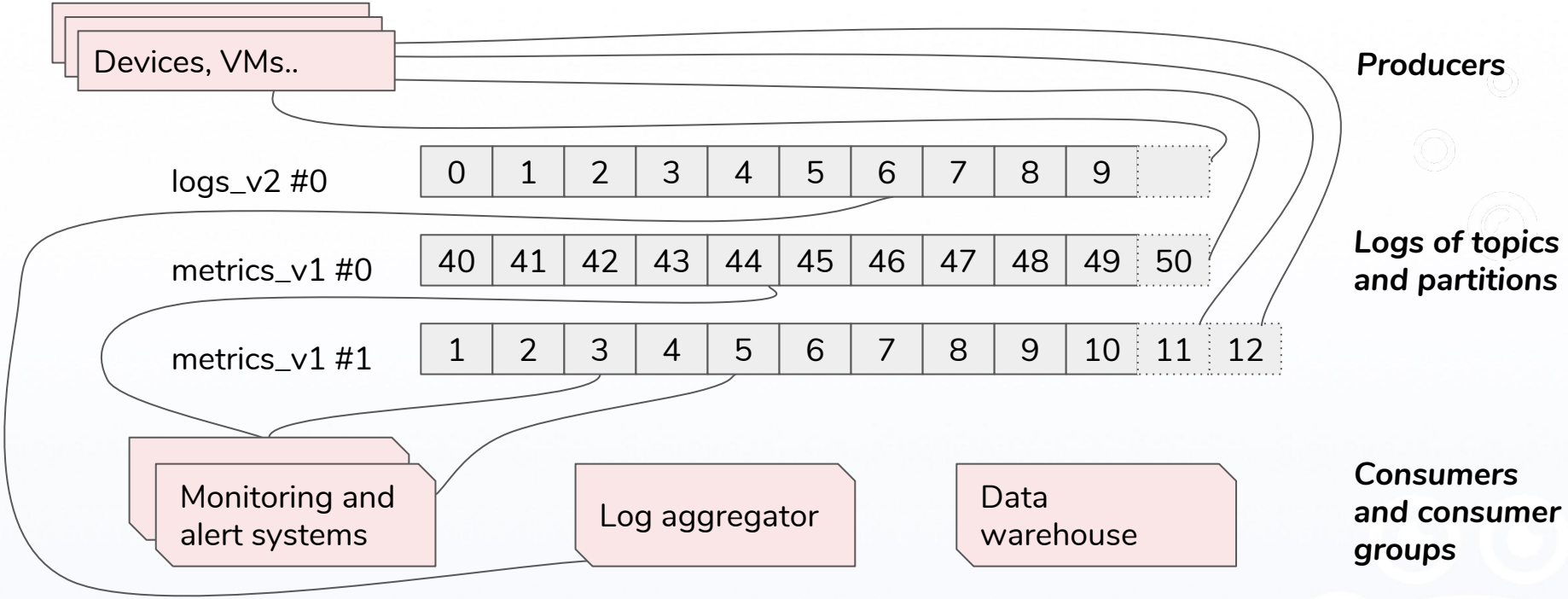
Kafka excels as a centerpiece for event delivery, where a range of applications can produce and consume real-time event streams.

https://kafka.apache.org/

Kafka-centric data flow model



Kafka concepts



A key abstraction in Kafka is its commit log, where each consumer maintains maintains its own position in the log. This allows clean decoupling of the producing and consuming processes.

Operating Apache Kafka

Covers data pipeline requirements

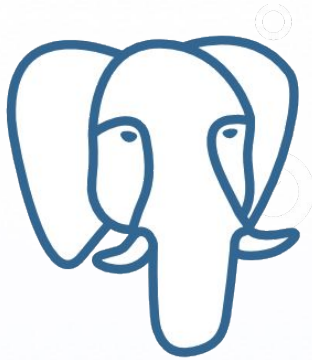
- Scales to billions of messages per day
- Supports rack and data center aware replication
- Cross-region replication using MirrorMaker
- Real-time streaming
- Decoupling of message consumption & producing
- Client libraries & tools available for all popular languages

But maintenance can be a burden

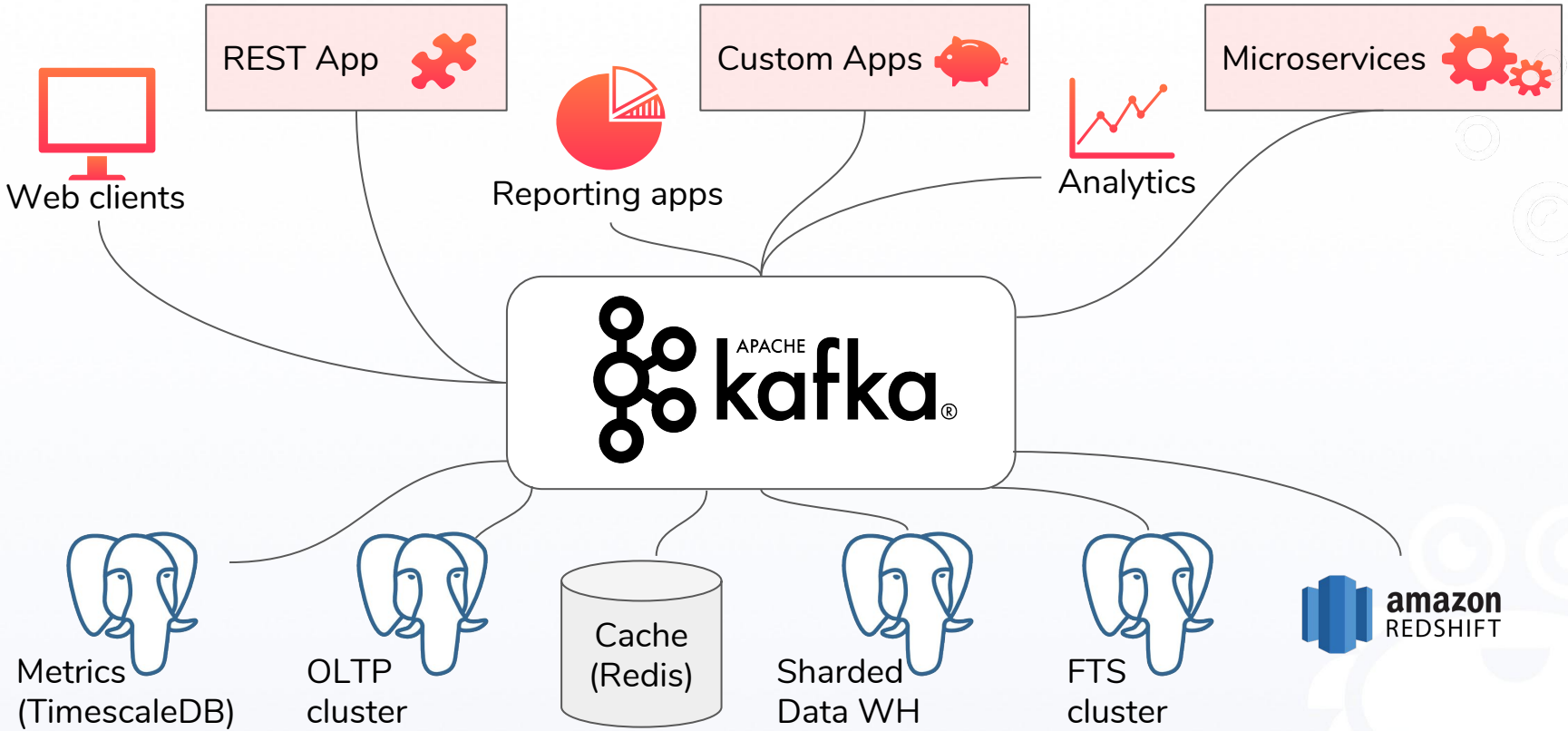
- Depends on ZooKeeper
- Rebalancing of leaders and partitions
  - In case of failure
  - When scaling up or down
- Broker hangs
- Consider using a managed Kafka service, available from multiple vendors including us

Databases in the pipeline

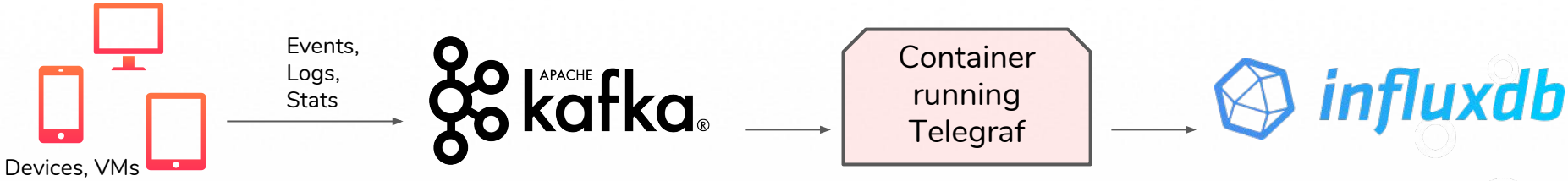
- Specialized database technologies available for different usecases
- Consider the same requirements as for the streaming platform:
  - \- Scalability, reliability, platform support
- PostgreSQL can often play this role
  - \+ Robust tooling for developers, data scientists, DBAs
  - \+ Easy to run ad-hoc queries with JOINs
  - \+ `EXPLAIN`
  - \- Limited horizontal scalability for now



Kafka-centric data flow model



Metrics pipeline using Telegraf, Kafka and InfluxDB



telegraf.conf

```
[tags]
service = “pgdemo”

[outputs]
[[outputs.kafka]]
brokers = [“kafka.example.com”]
topic = “pgmetrics_v1”

[inputs]
[[inputs.cpu]]
[[inputs.postgresql_extensible]]
address = “dbname=_stats user=_stats”
```

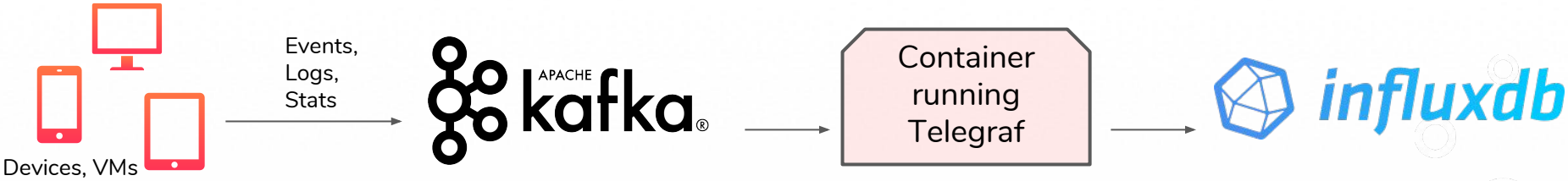
telegraf.conf

```
[tags]
service = “pgdemo”

[inputs]
[[inputs.kafka_consumer]]
brokers = [“kafka.example.com”]
topics = [“pgmetrics_v1”]
consumer_group = “telegraf_pg_input”
data_format = influx

[outputs]
[[outputs.influxdb]]
urls = [“http://influx.example.com:8086”]
```

Metrics pipeline using Telegraf, Kafka and InfluxDB

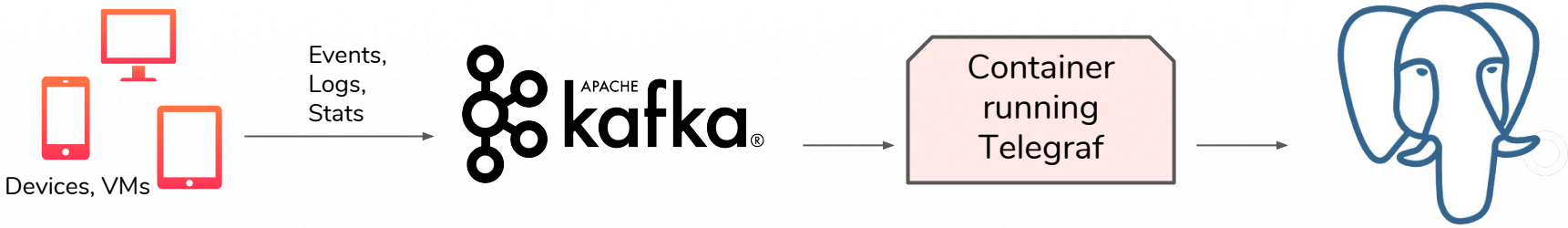


InfluxDB as time-series database backend

+ Purpose-built for handling time-series data
+ Very efficient data-storage format on-disk
+ Telegraf (part of the TICK stack) includes input and output plugins for a wide array of systems

- No replication in Open Source version: no HA, no horizontal scaling
- Inefficient memory usage
  - Improved in upcoming v1.6
- Requires learning new tooling

Metrics pipeline using Telegraf, Kafka and PG



telegraf.conf

```
[tags]
service = “pgdemo”

[outputs]
[[outputs.kafka]]
brokers = [“kafka.example.com”]
topic = “pgmetrics_v1”

[inputs]
[[inputs.cpu]]
[[inputs.postgresql_extensible]]
address = “dbname=_stats user=_stats”
```

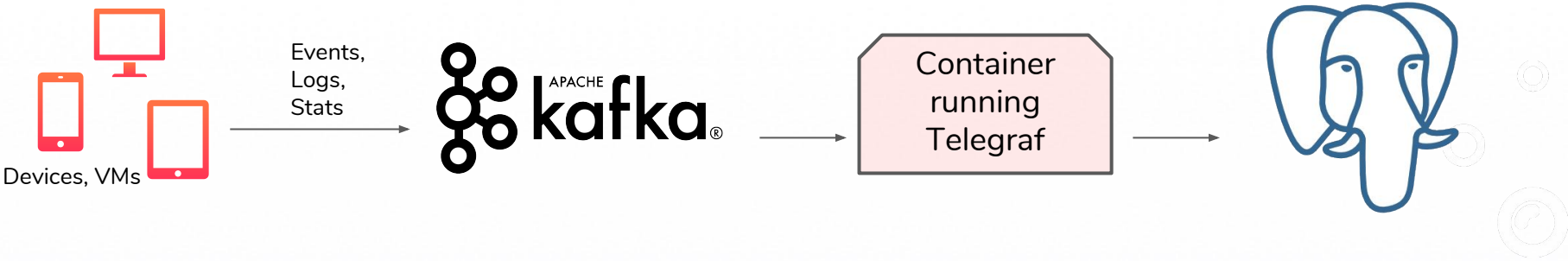
telegraf.conf

```
[tags]
service = “pgdemo”

[inputs]
[[inputs.kafka_consumer]]
brokers = [“kafka.example.com”]
topics = [“pgmetrics_v1”]
consumer_group = “telegraf_pg_input”
data_format = influx

[outputs]
[[outputs.postgresql]]
address = “postgres://pgstat.example.com/metrics”
tags_as_jsonb = false
fields_as_jsonb = false
```

Metrics pipeline using Telegraf, Kafka and PG



PostgreSQL as time-series database backend

- \+ The reliable RDBMS you may already know
- \+ Run pure SQL queries on your data
- \+ Ecosystem full of robust clients and tools
- \+ Includes HA features out of box
- \+ Now also has open source extension for more efficient handling of time-series data

```
metrics=> \d
[..]
-- how's my database backup compression working?
metrics=> SELECT host, avg(value)
            FROM pghoard_compressed_size_ratio
            WHERE time > now() - INTERVAL '1 day'
            GROUP BY 1 ORDER by 2 DESC LIMIT 10;
-- hmm, which hosts never have idle CPUs?
metrics=> SELECT host,
                  PERCENTILE_CONT(0.9) WITHIN GROUP
                  (ORDER BY usage_user)
            FROM cpu
            WHERE time > now() - INTERVAL '1 day'
            GROUP BY 1 ORDER by 2 DESC LIMIT 10;
-- Let's see the query plan
metrics=> EXPLAIN ...
```

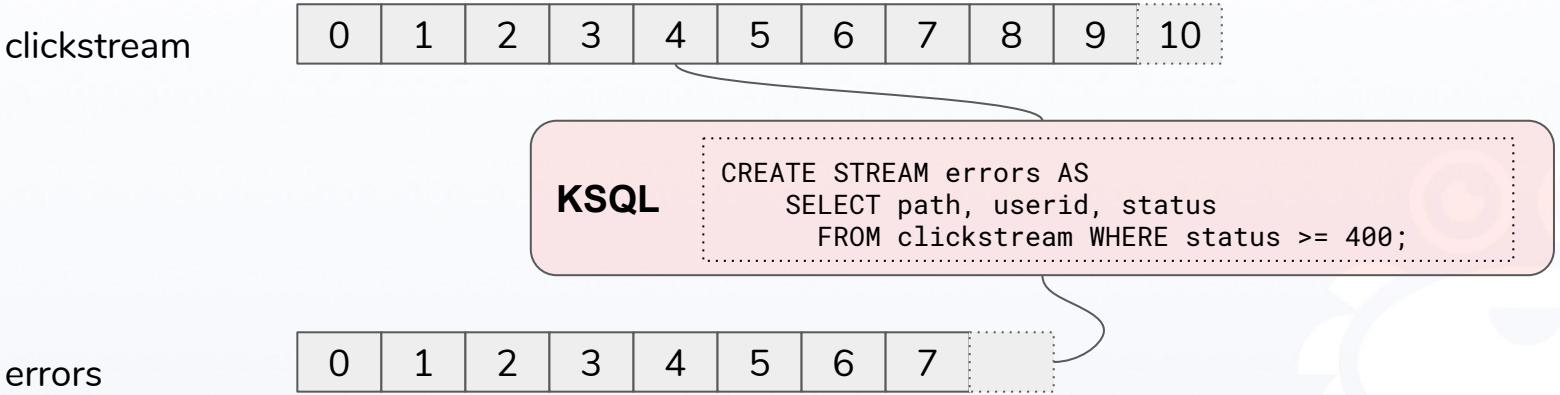
SQL for everything

SQL - Structured Query Language, first described in the 70s

- Originally used in relational databases
- Sometimes used to describe both the data model and the language
- Started appearing as the query language for “NoSQL” databases
  - Apparently NoSQL now means “not only SQL”
- Widely criticized, but used everywhere and outlasted everything else
- Now finding its way into data streaming usecases, e.g. KSQL for Kafka

KSQL: SQL engine for Kafka

- KSQL allows performing data transformation directly inside Kafka using SQL syntax
- Standalone service using Kafka APIs typically running as its own cluster next to Kafka
- Kafka, KSQL and the JDBC sink connectors make it possible to perform complex operations on data inside the Kafka cluster and push the results directly to PostgreSQL or other JDBC-compliant databases without any “application code”.



Processing data inside a Kafka cluster

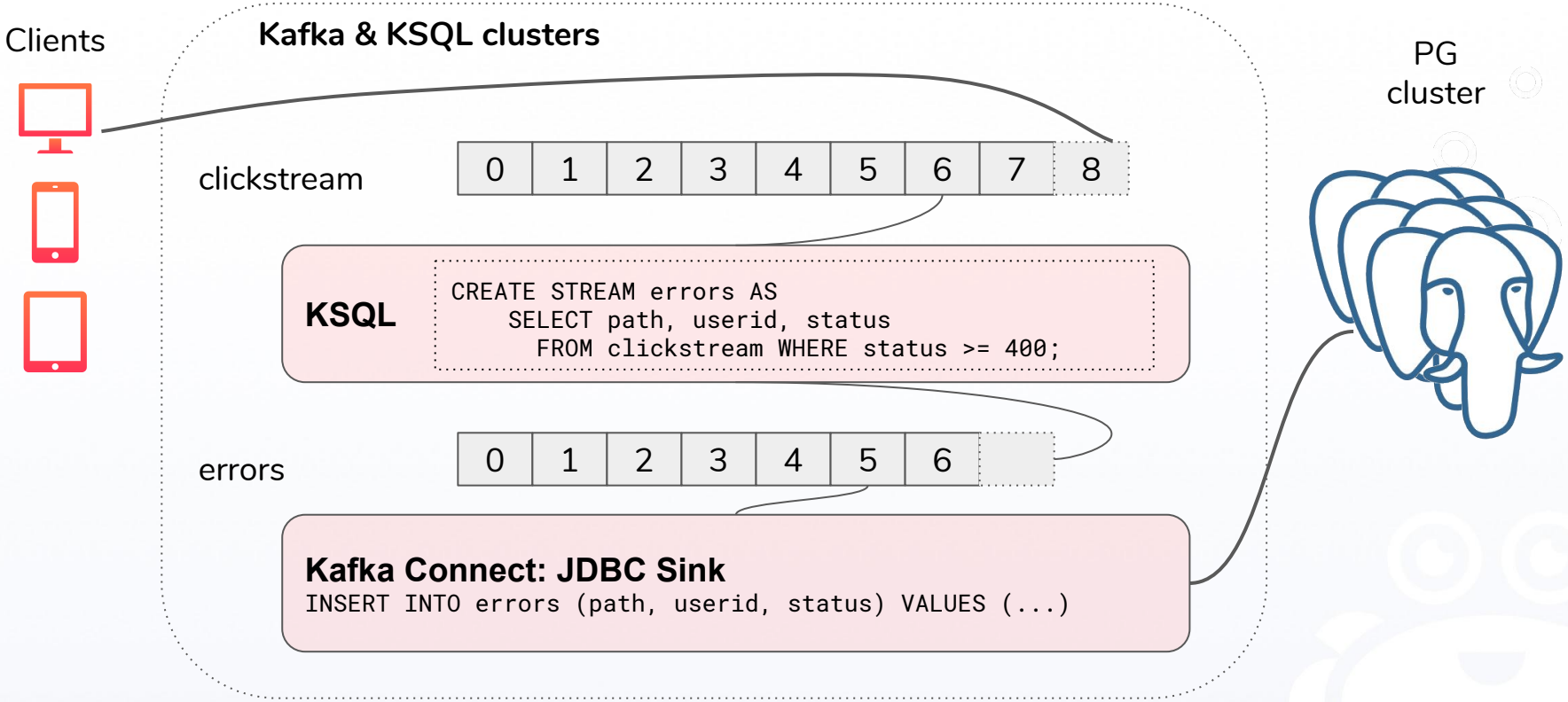
KSQL - “continuous queries”

- Transforming data in the background instead of performing heavy operations when data is accessed
- Continuous queries are implemented by various NoSQL and streaming systems and are similar to automatically updating materialized views in PostgreSQL

Kafka Connect - “background workers”

- Framework for running source (import) and sink (export) connectors directly inside the Kafka cluster, similar to PostgreSQL’s background workers
- Reuse existing, community built and shared connectors

Clickstream pipeline with KSQL and PGSQL



Summary

- Kafka as the central component of a data pipeline helps clean up messy architectures
- Kafka's connectors make it easy to reuse code and allow building data pipelines with configuration only
- PostgreSQL is a robust RDBMS that can handle OLTP, DWH, time-series workloads among other things, sometimes outperforming specialized NoSQL systems
- Kafka and PostgreSQL can be used to ingest and process billions of events and hundreds of terabytes of data with open source tools
- SQL is here to stay, start applying it in new usecases, e.g. time-series and event streaming

Questions?

Cool t-shirts for the first ones to ask a question!

P.S. try out Aiven and get a cool t-shirt even if you didn't ask a question

aiven

Thanks!

https://aiven.io

@aiven_io

@OskariSaarenmaa