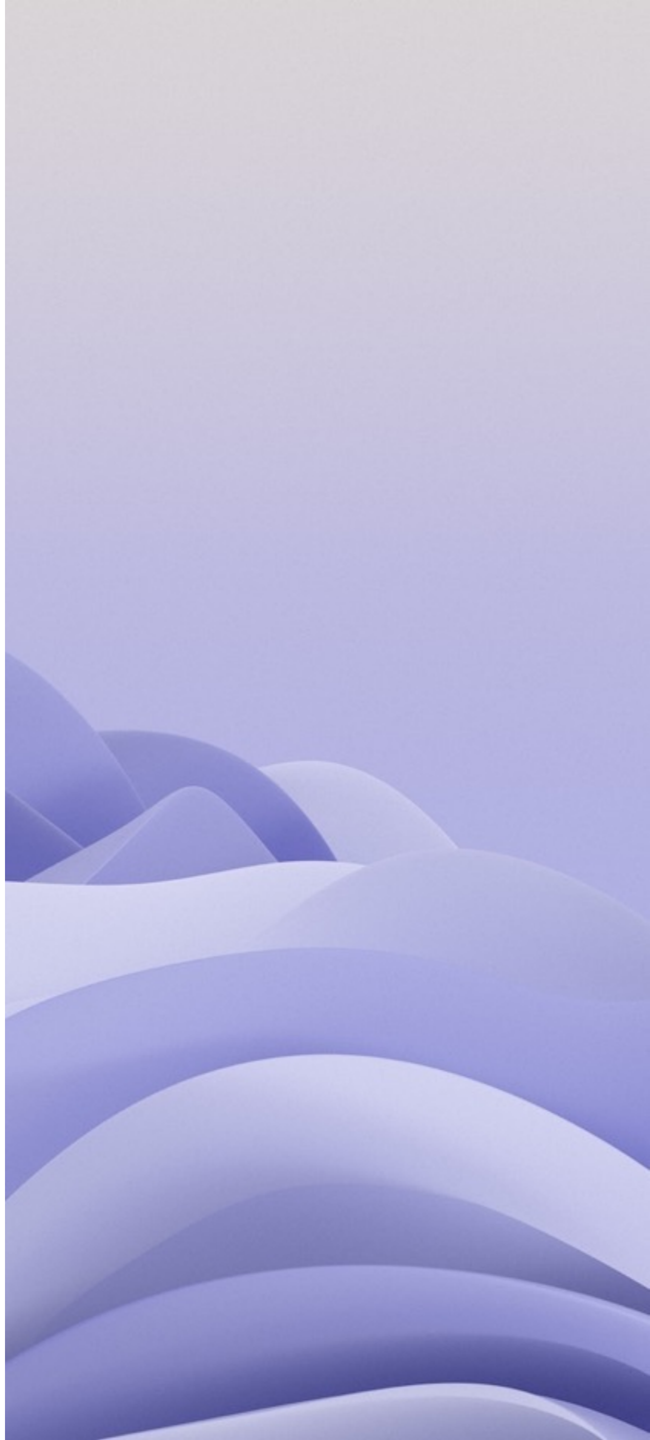
Tuesday 31st October 2023

Big Data

Data Mining, warehouse, viz, nosql

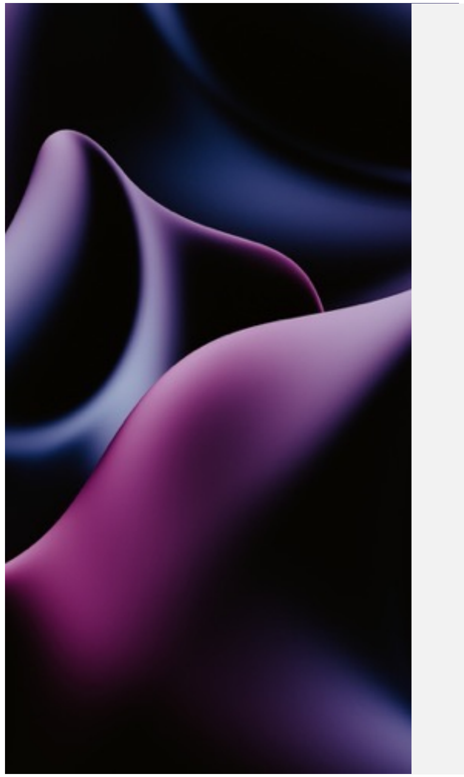
applied to health data



Agenda

Data streamin
introduction

Request/Response
Vs
Event-Driven



Kafka

Data streaming

sægus

SMART SHIFT · FOR REAL

What ?

Real-Time processing

Process data as it arrives

Continuous

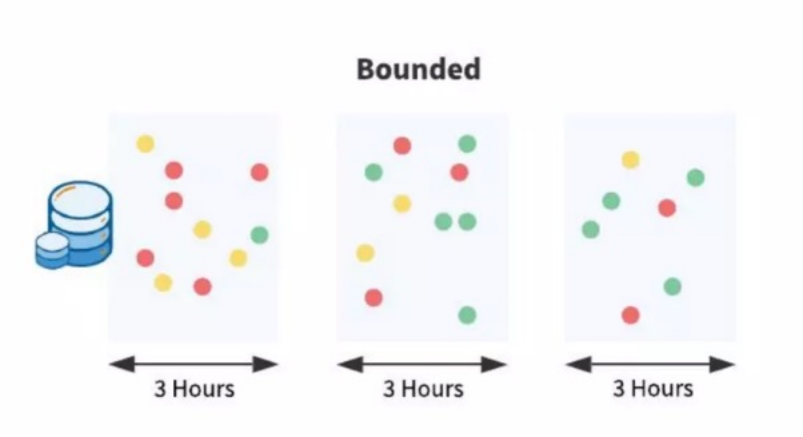
Unending stream, from multiple sources

Timeliness

Quick / instantinsights from incoming data

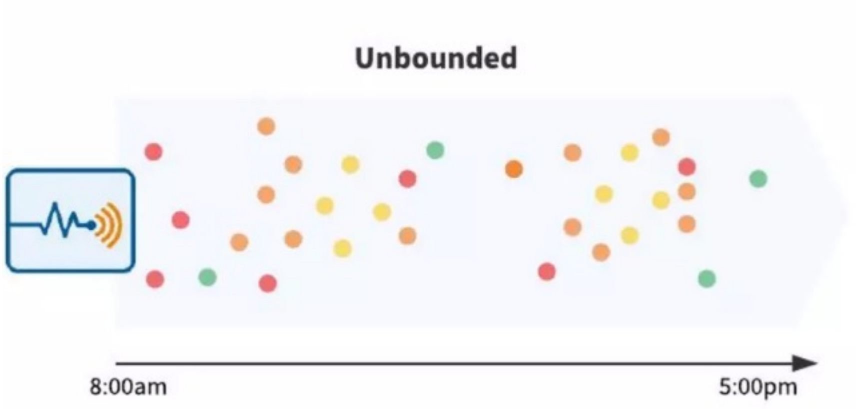
Batch vs Streaming

Batch processing



Processing on accumulated data on a specific bounded period. Ideal for huge n-time-critical data

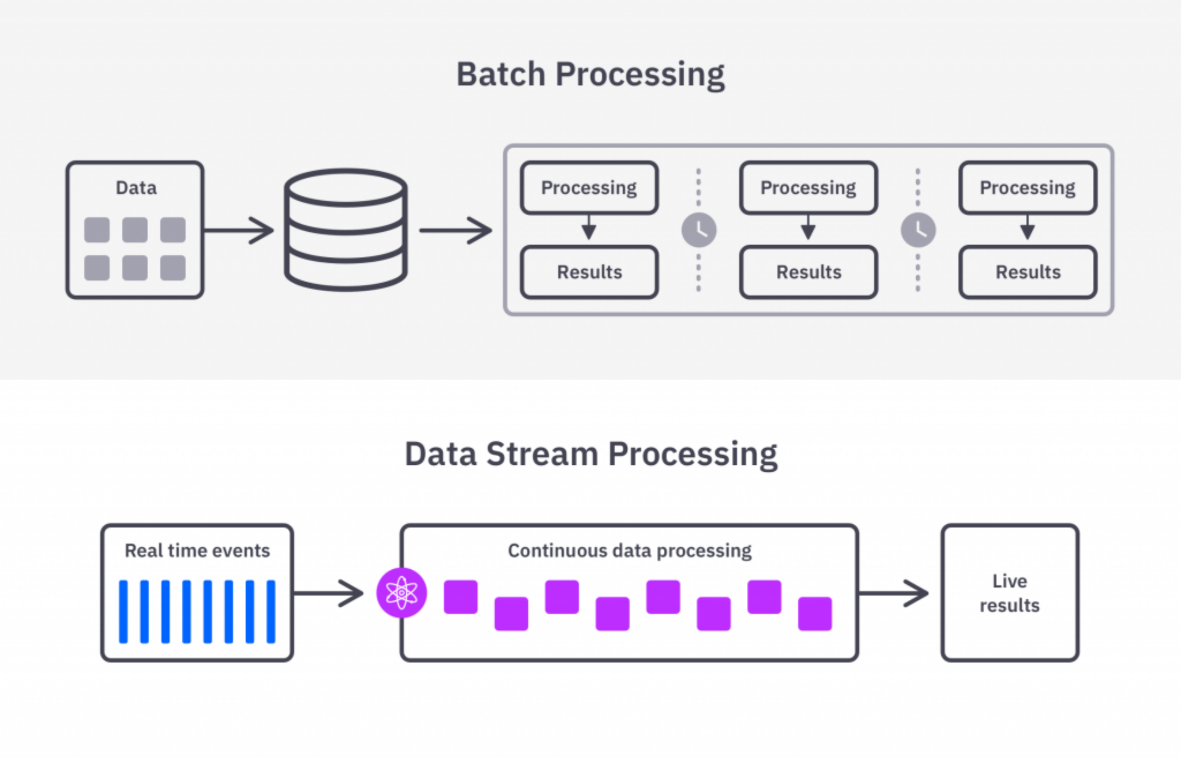
Data Streaming (Real-time)



Continuous and instantaneous processing of cross-devices data as it arrives

Batch vs Streaming

Source: https://addepto.com/blog/stream-data-model-and-architecture/



Characteristics of Data Streams

Size

Small in size (in KBs)

Volume

Unbounded => very large volume because of accumulation

Examples

Iot sensors, log files, financial markets, apps clicks

Velocity

Fast Changing, requires fast reponse

Variety

Unstructured or semi-structured

Challenges

Scalability

Handling Large volumes at high velocity

Veracity

Ensuring data quality and consistency

Interagtion

Integrate existing systems and infras

Examples

Some concrete use-cases

Finance

Fraud Detection – Analyzing patterns and anomalies to detect fraudulent activities, unauthorized transactions

Financial transactions – Analyze market trade and execute trades in real-time

Health

Patients Monitoring – Stream patient’s vital signs thanks to IoT sensors, enabling immediate intervention

Predictive Analytics for disease risk -

Media

Content Personalization

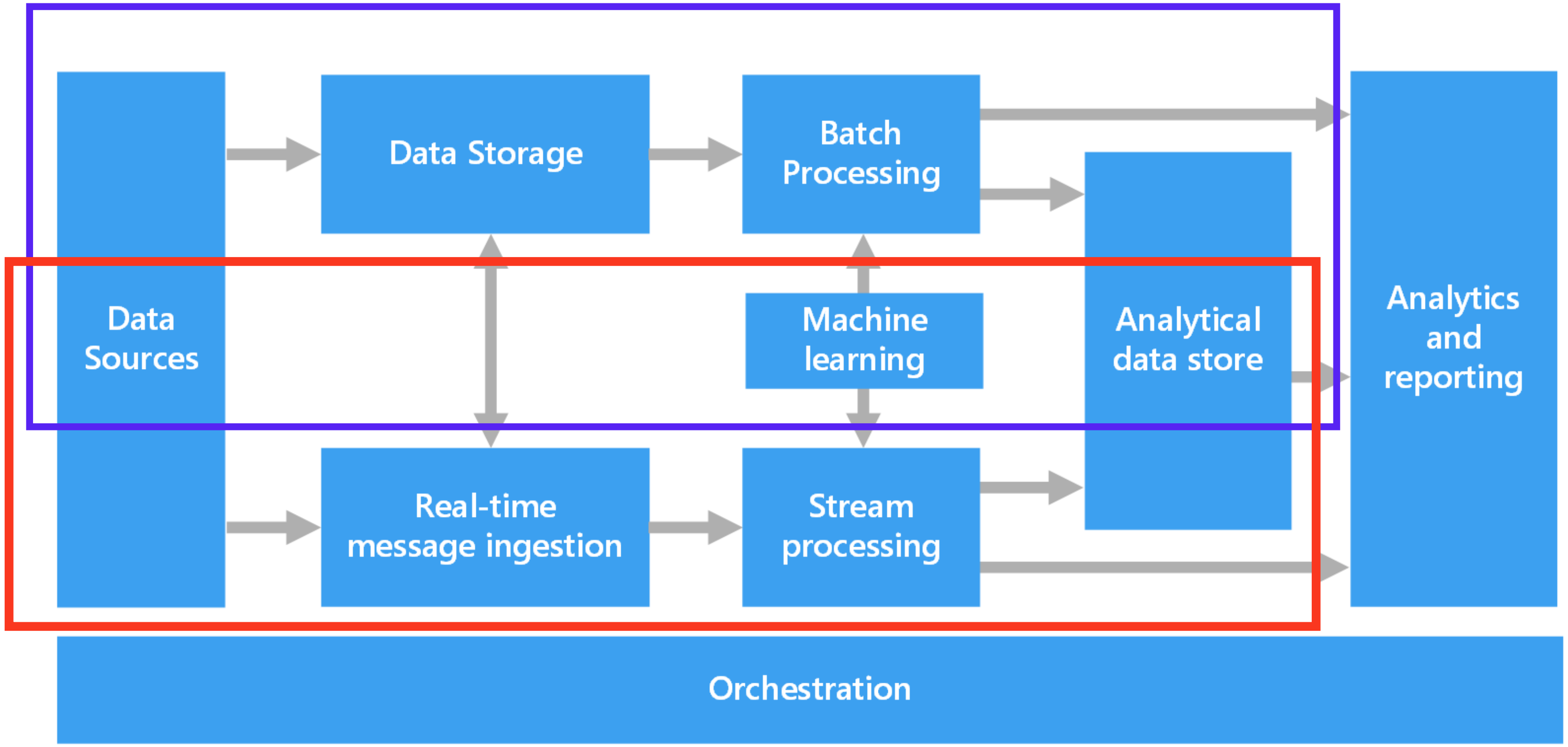
Real time audience analytics for media streaming platforms

Retail

Dynamic Pricing and inventory management

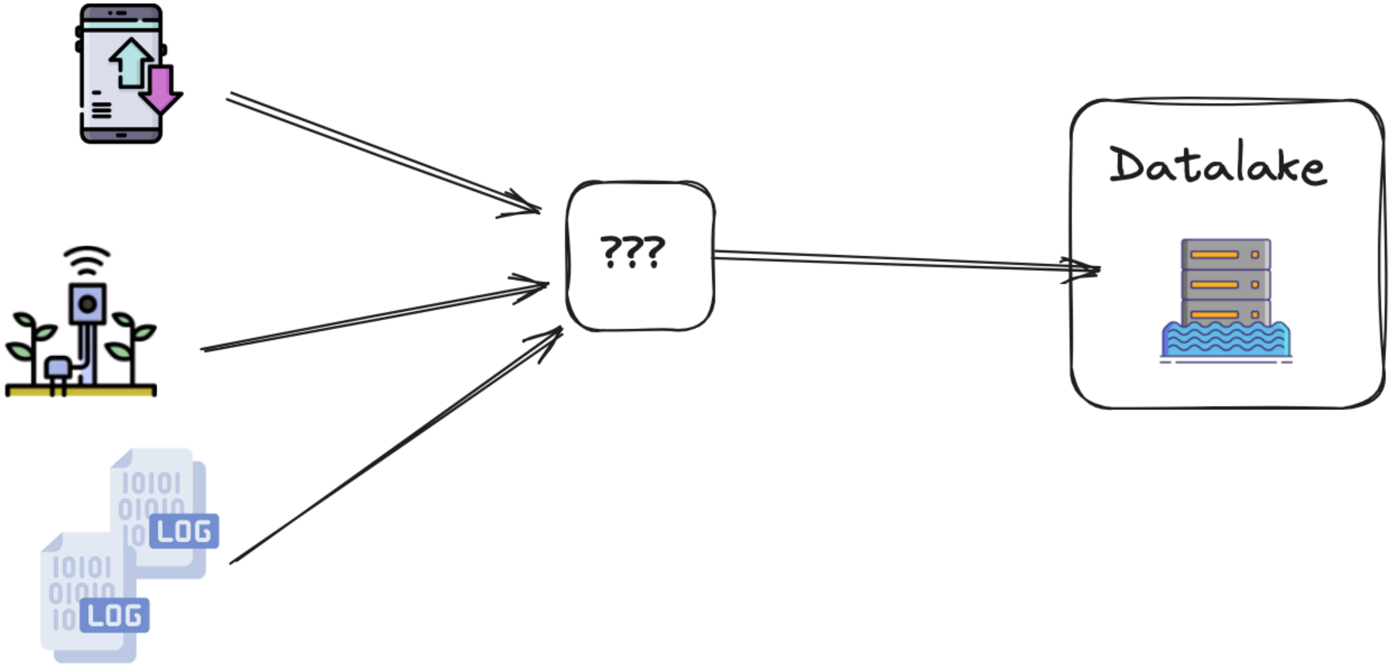
Enhancing customer experience through real-time insights

Reminder : Big Data Architecture



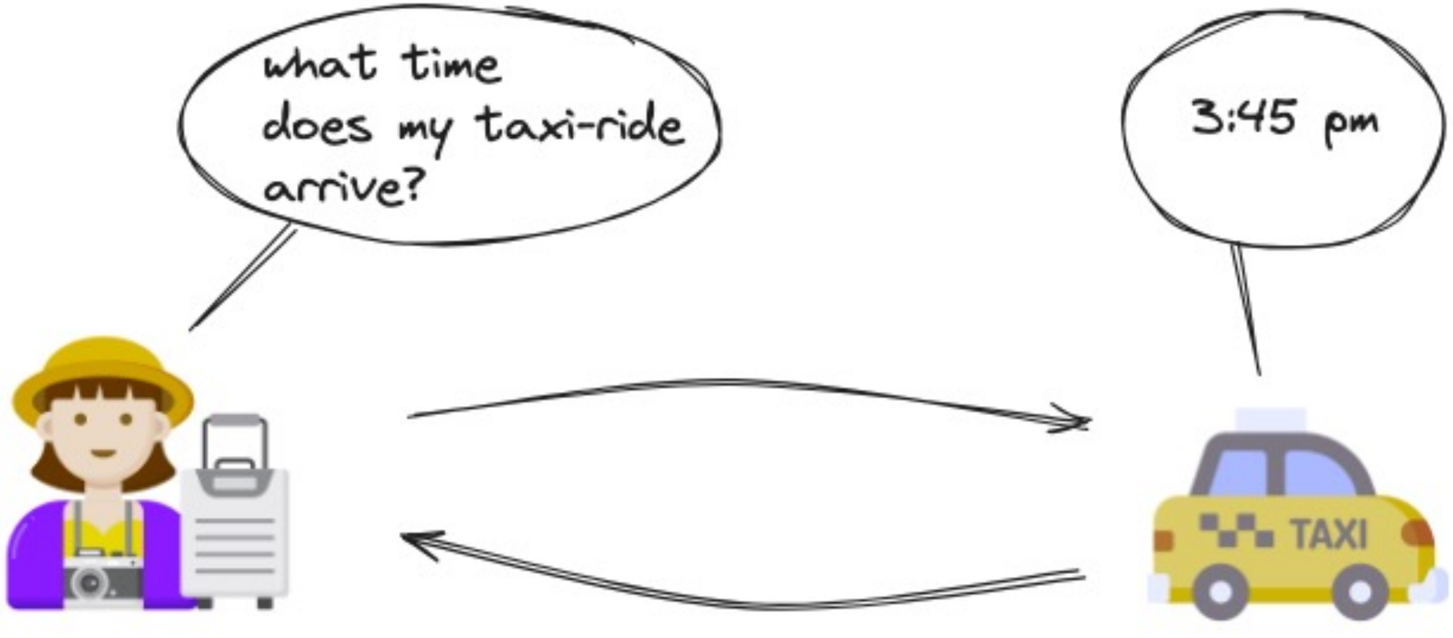
How do you stream data ?

From sources to datalake ?



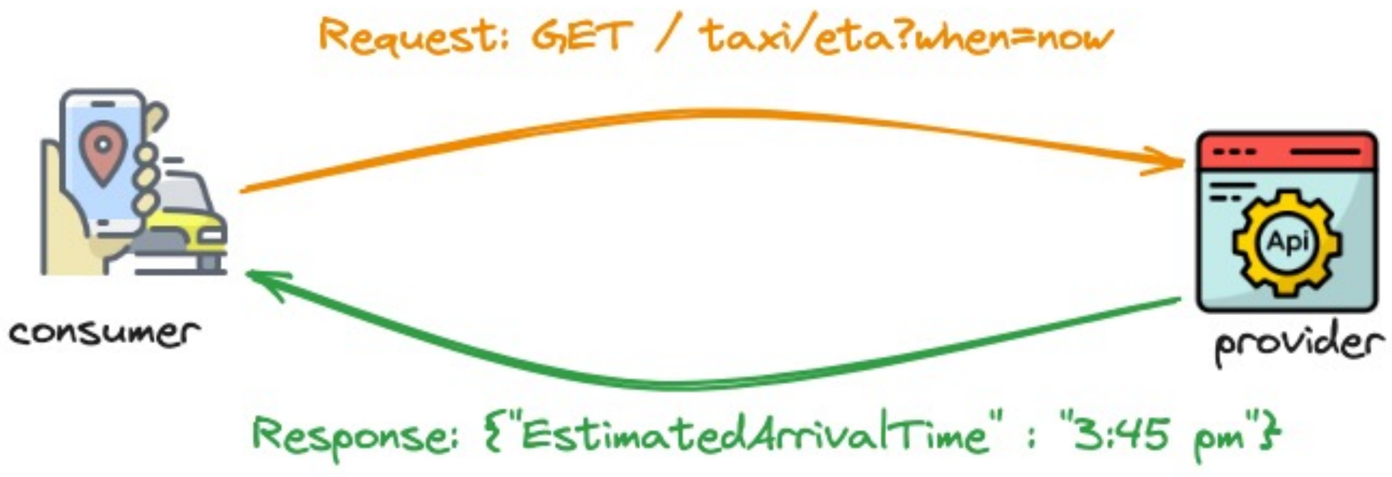
First approach : RESTful API

From sources to datalake ?



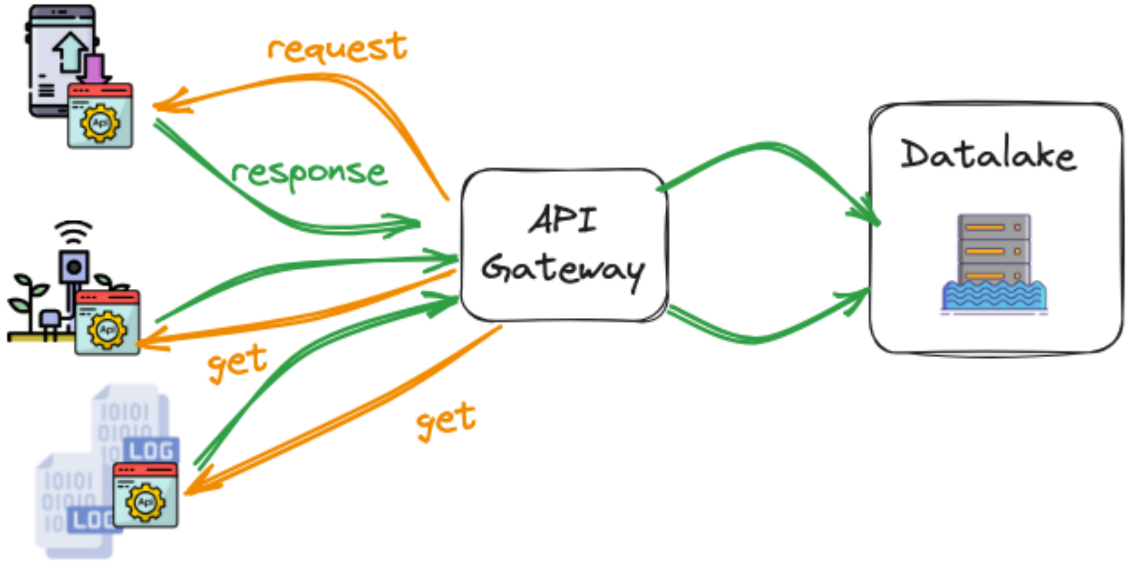
First approach : RESTful API

From sources to datalake ?


Request: GET / taxi/eta?when=now
consumer
Api
provider
Response: {"EstimatedArrivalTime" : "3:45 pm"}

First approach : RESTful API

From sources to datalake ?



Pros :

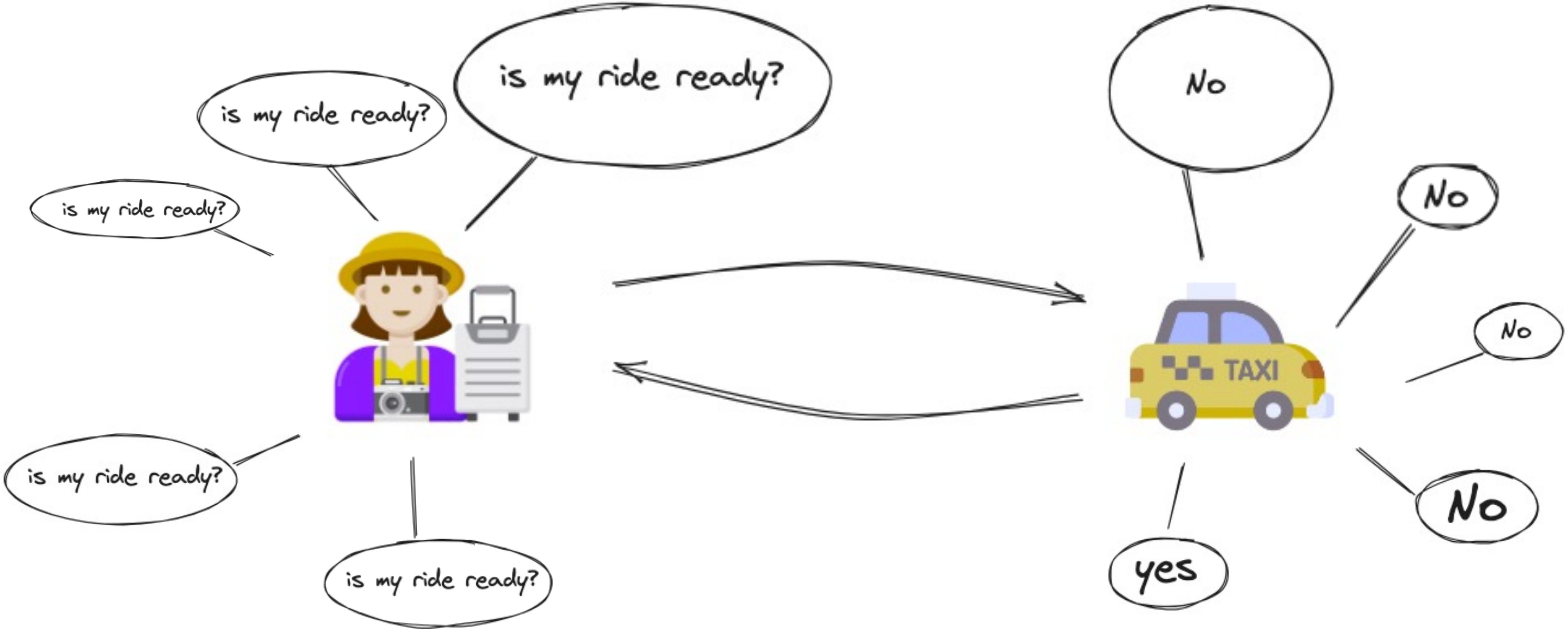
- Simple : Largely used and understood in the industry
- Interoperability : Easy integration for different systems
- Same HTTP norms everywhere

Cons :

- Active and continuous demand
- Latency : periodical « GET » may induce latency
- Scalability : difficult to maintain as data grows

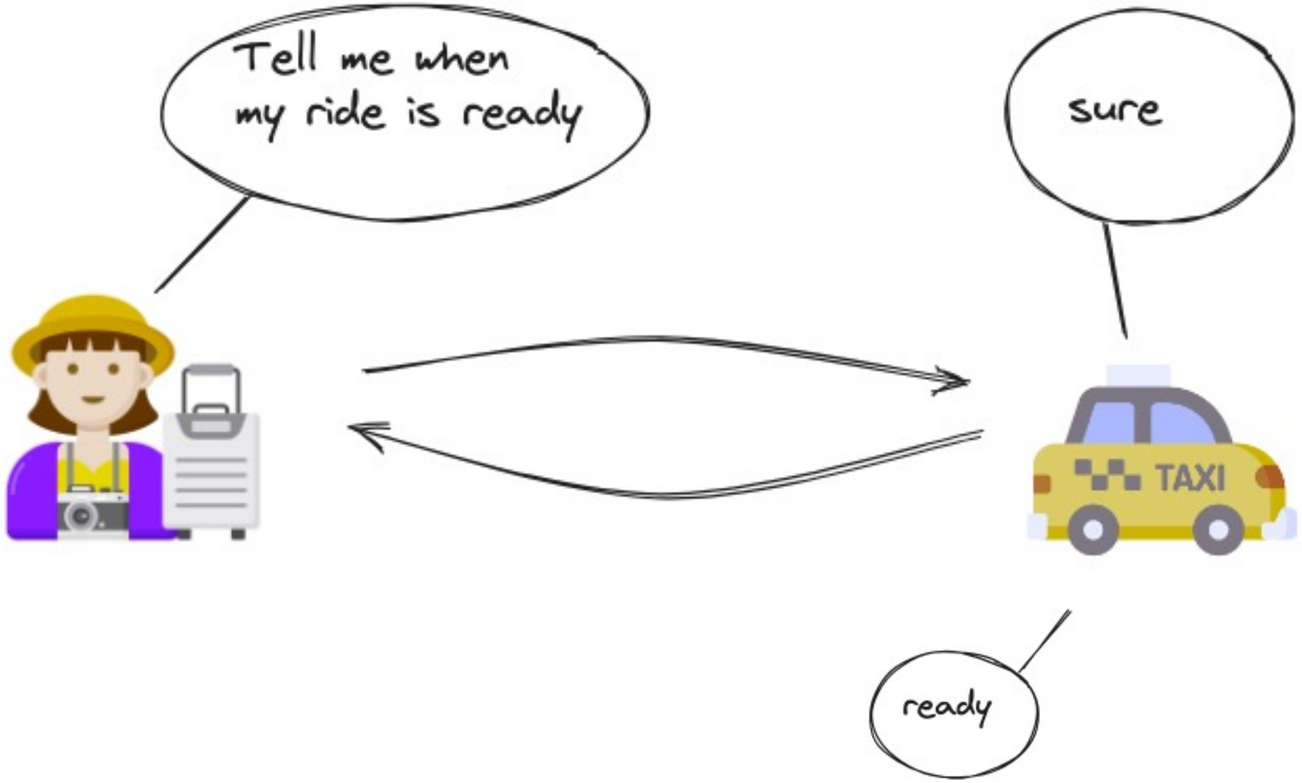
First approach : RESTful API

Not really ”real-time”



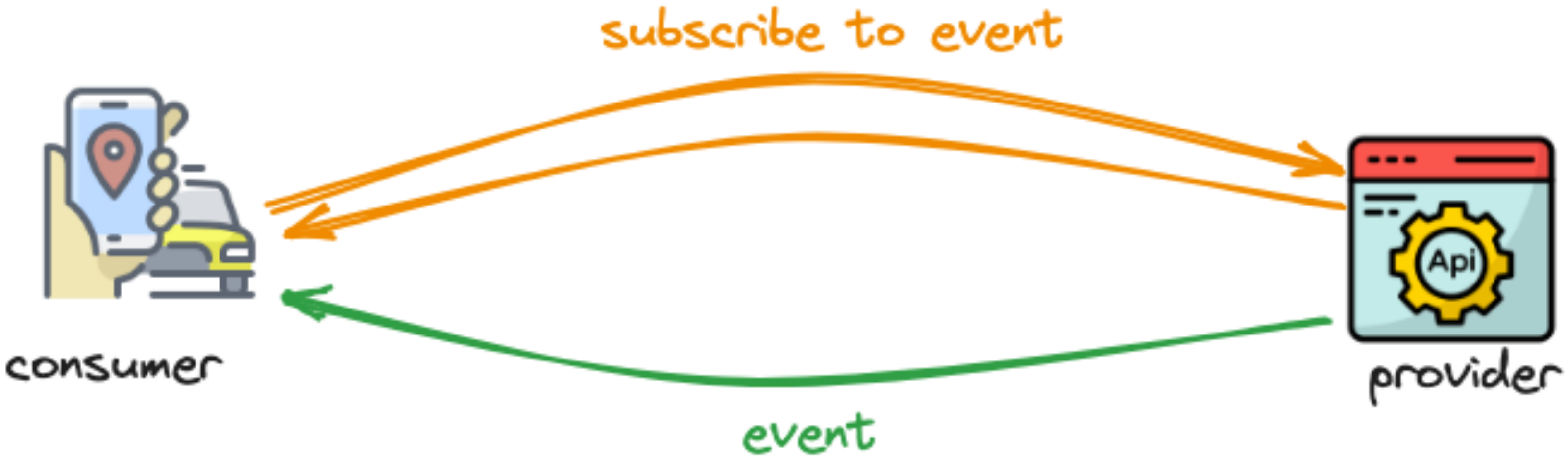
Second approach : Event-driven

Be smarter when asking for your ride



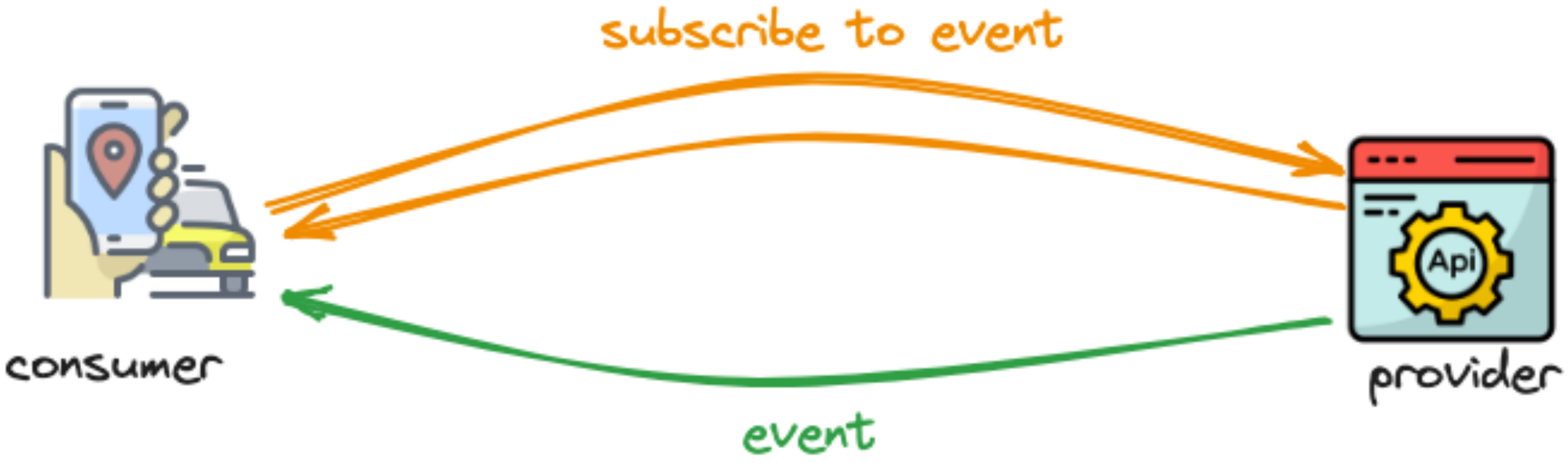
Second approach : Event-driven

Subscription to events



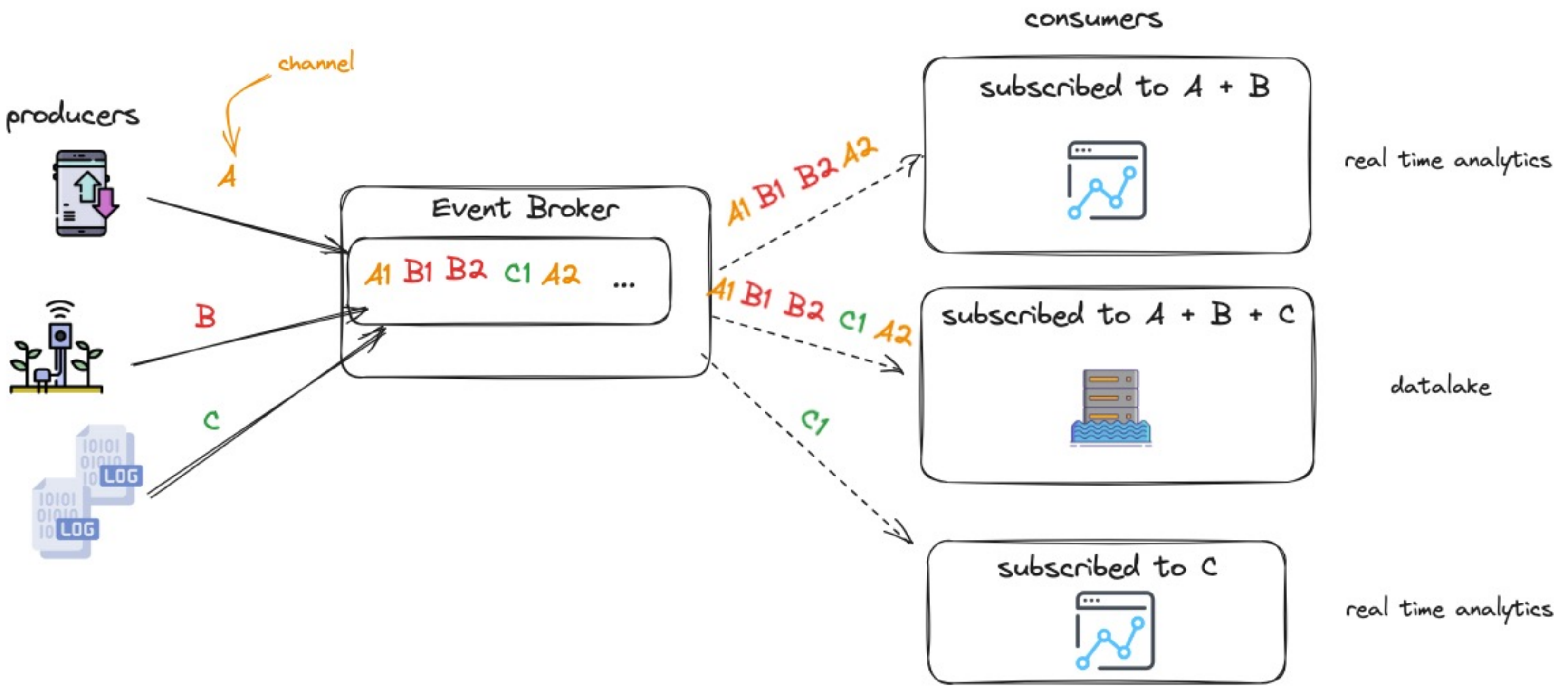
Second approach : Event-driven

Subscription to events



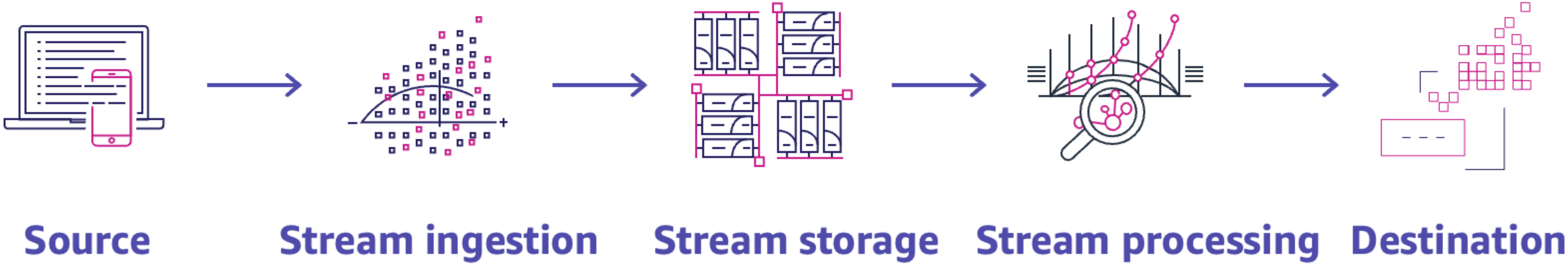
Second approach : Event-driven

Event-driven real-time architecture



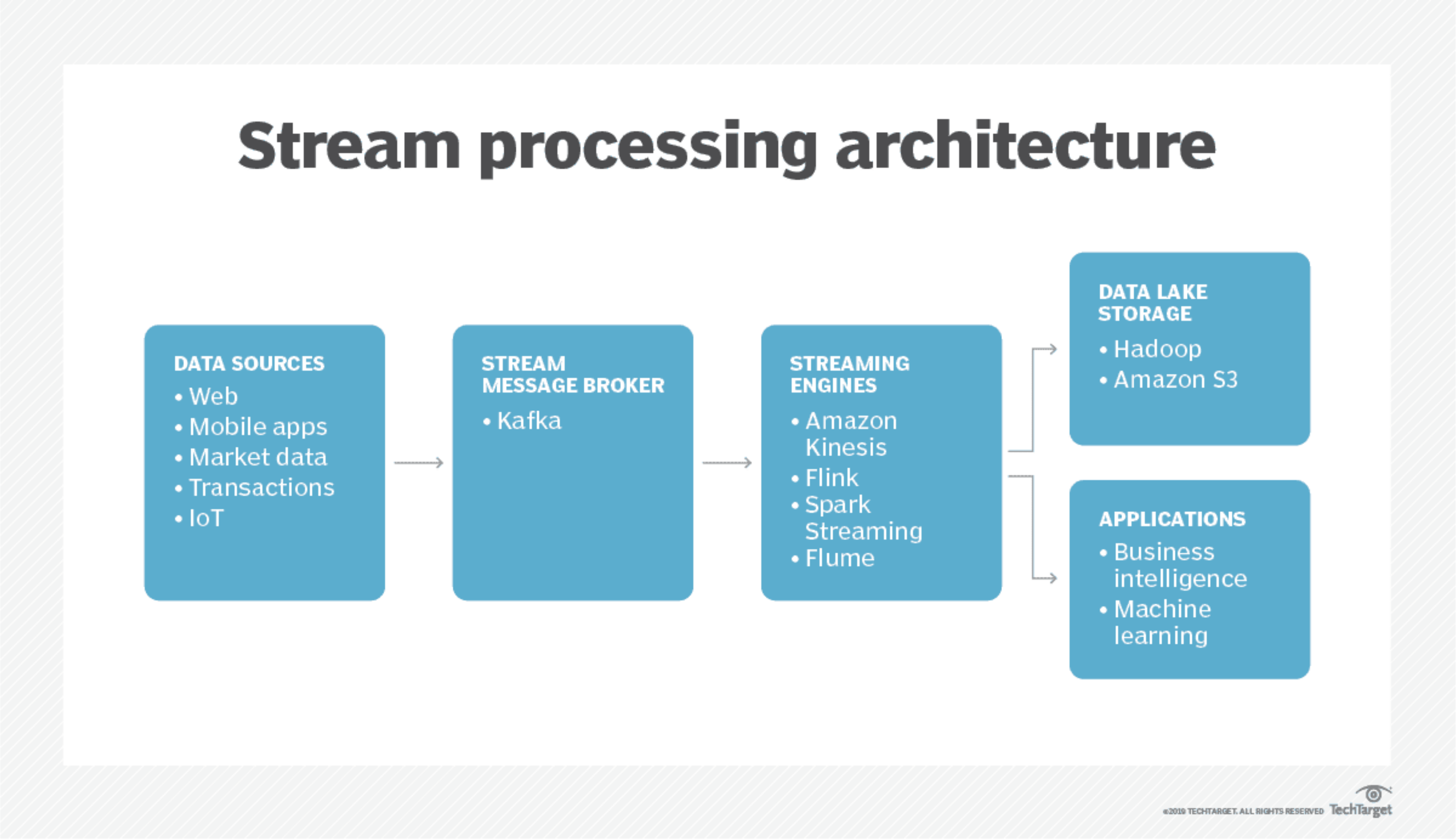
- Asynchronous
- Real-time automatic events **detection**
- Producers and consumers are **independent**
- **Scalability** : easy to add more producer or consumer
- All services are **independants**

Data Streaming Architecture



Source → Stream ingestion → Stream storage → Stream processing → Destination

Data Streaming Architecture



Stream processing architecture

DATA SOURCES
- Web
- Mobile apps
- Market data
- Transactions
- IoT

STREAM MESSAGE BROKER
- Kafka

STREAMING ENGINES
- Amazon Kinesis
- Flink
- Spark Streaming
- Flume

DATA LAKE STORAGE
- Hadoop
- Amazon S3

APPLICATIONS
- Business intelligence
- Machine learning

©2019 TECHTARGET. ALL RIGHTS RESERVED TechTarget

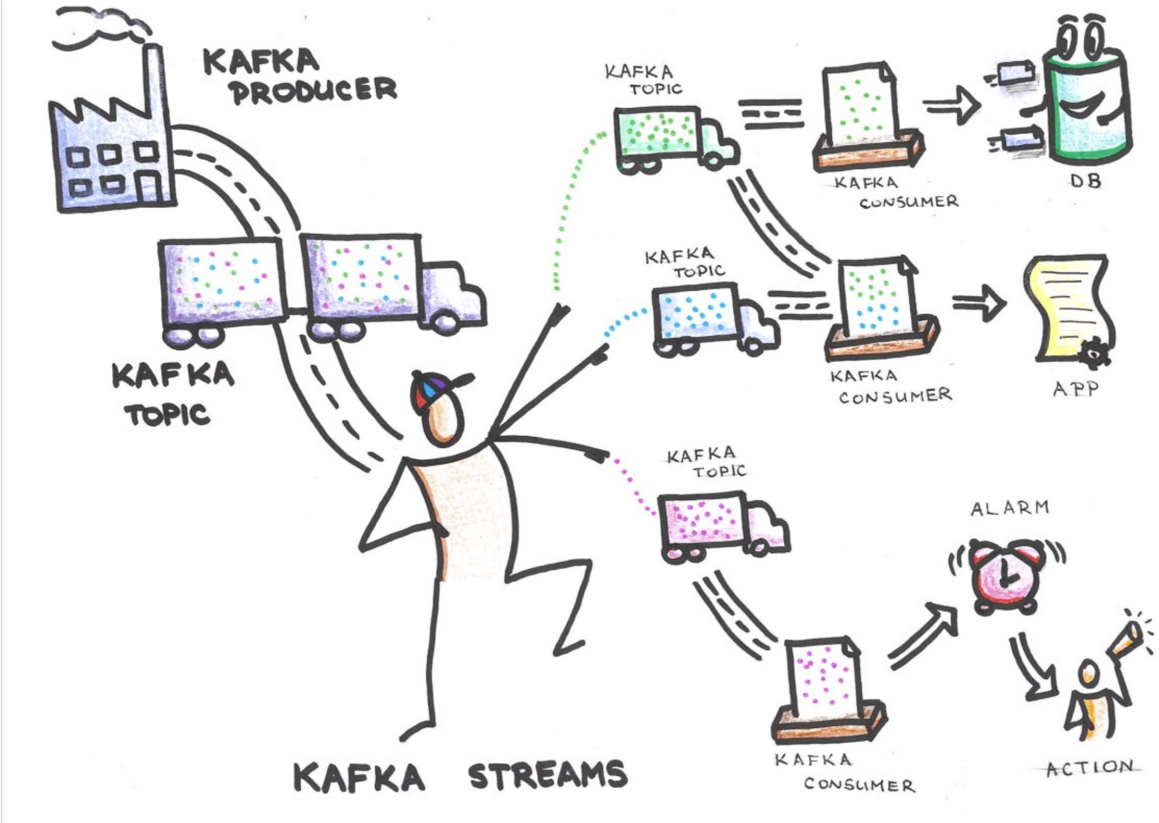
Apache Kafka

sægus

SMART SHIFT · FOR REAL

Introduction to Apache Kafka

Apache Kafka: Real-Time Data Streaming Platform

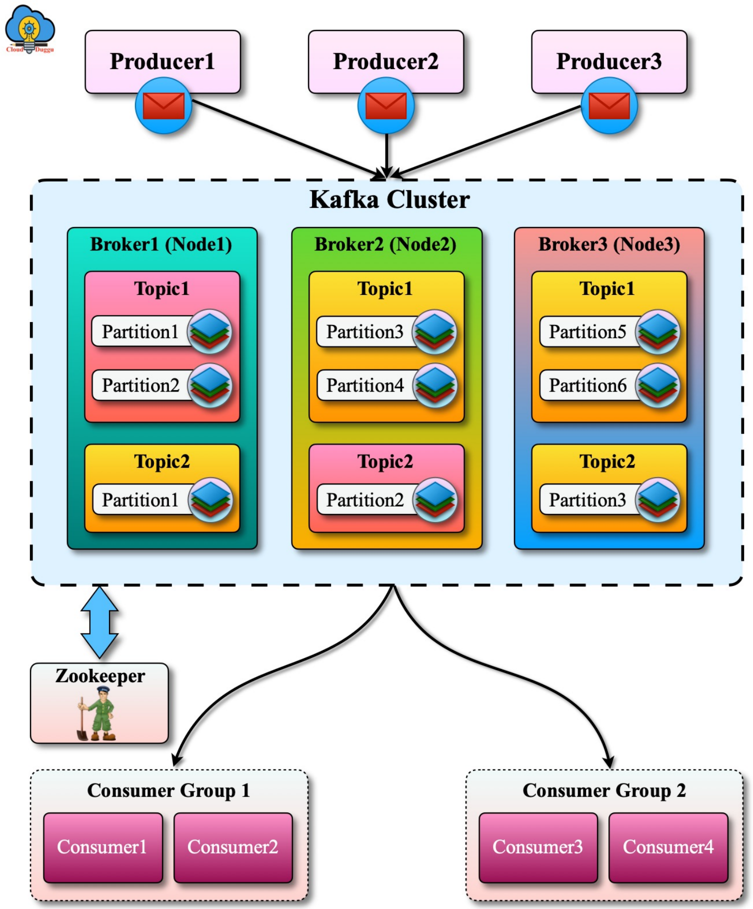


Source : https://croz.net/news/apache-kafka/

- Asynchronous
- Real-time automatic events **detection**
- Producers and consumers are **independent**
- **Scalability** : easy to add more producer or consumer
- All services are **independants**

Kafka Architecture Overview

Understanding Kafka’s Core Components



- Producers : Services that **publish** data to Kafka topics
- Consumers : Services or apps that **subscribe** to topics and process data
- Brokers : Kafka servers that store data and serve clients
- Zookeeper : Manages Kafka cluster metadata and brokers

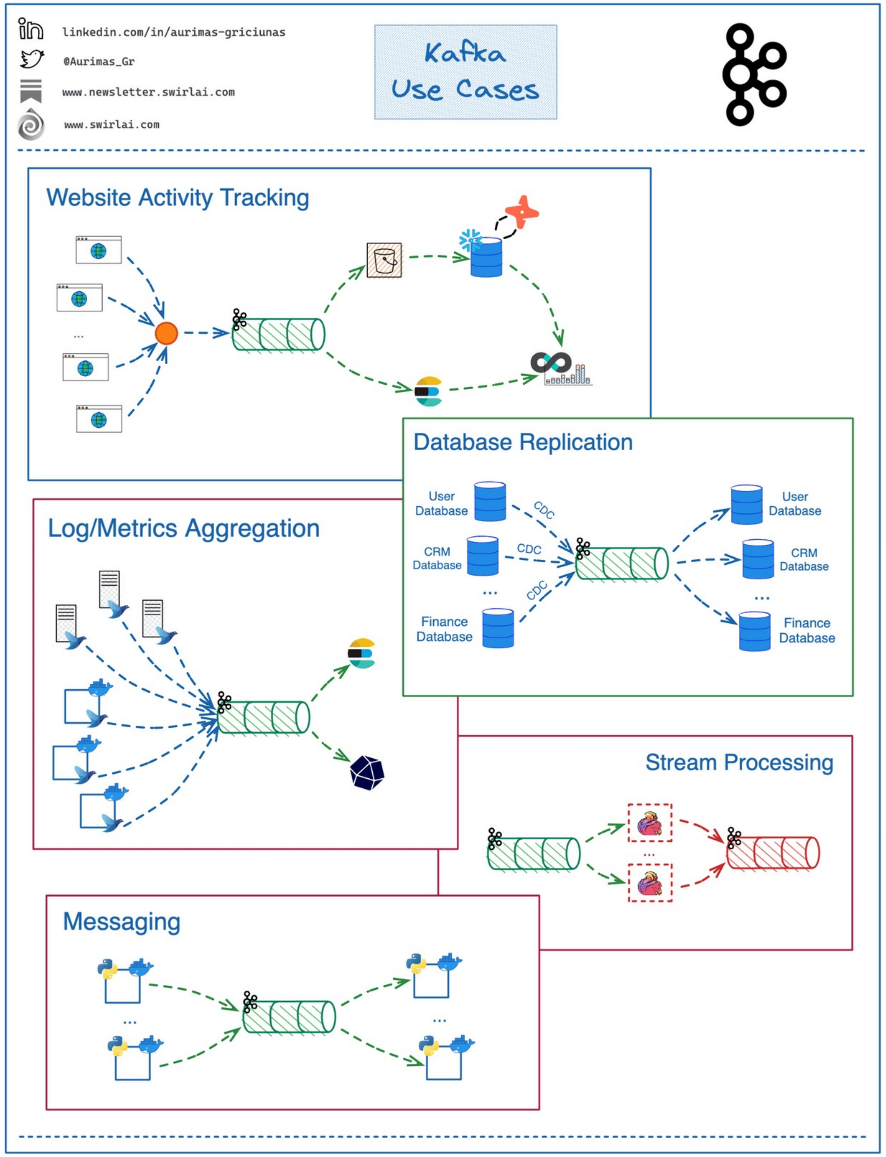
Kafka’s key features

The power of Kafka for Real-Time Data Operations

- Scalable in all dimensions, including event producers, processors, consumers, and connectors
- Durability and Reliability: data is replicated when stored on disk, ensuring data persistence and reliability
- >1M Throughput for both publishing and subscribing
- Real-Time Processing + other technologies integration
- Fault Tolerance
- Distributed Architecture

Kafka

Usecases


linkedin.com/in/aurimas-griciunas
@Aurimas_Gr
www.newsletter.swirlai.com
www.swirlai.com
Kafka Use Cases
Website Activity Tracking
...
Database Replication
User Database
CDC
CRM Database
CDC
...
CDC
Finance Database
User Database
CRM Database
...
Finance Database
Log/Metrics Aggregation
Stream Processing
...
Messaging
...
...