#SPRINGONE @S1P

EVENT-DRIVEN ARCHITECTURES FOR SPRING DEVELOPERS

Viktor Gamov
Developer Advocate, Confluent
@gamussa

Gary Russell
Engineer, Pivotal
@gprussell

Why Event Streaming?


The world is changing.
#ONASSIGNMENT

The New Business Reality

Past

Technology was **a support function**

Innovation required for **growth**

Running the business on yesterday's data was "good enough"

Today

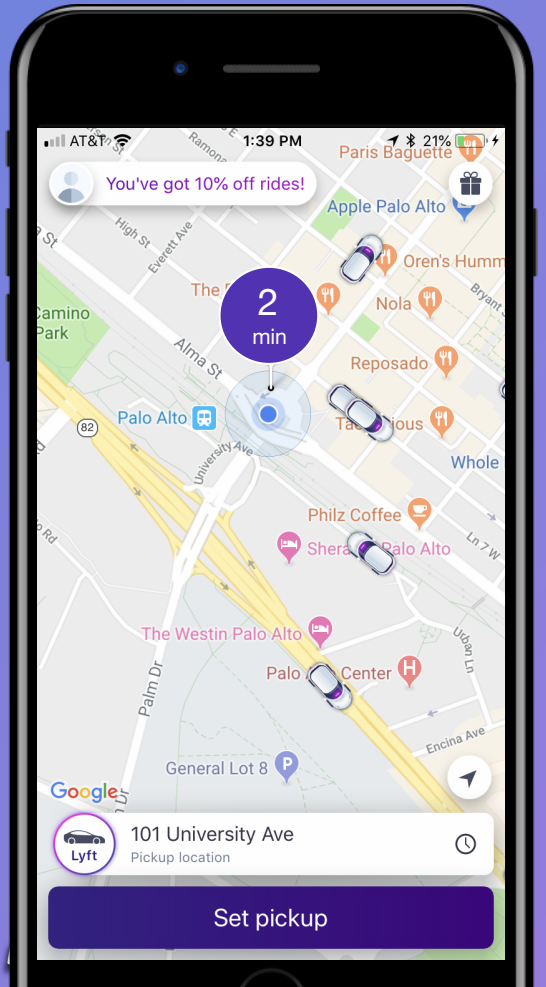
Technology **is the business**

Innovation required for **survival**

Yesterday's data = failure.

Modern, real-time data infrastructure is required.

Taxis become Software



confluent

Transportation

Then	Now
Hardware product	Hardware, Software, and Global Internet Service
Up-front purchase	On-demand
Opaque	Real-time visibility
No data	Built on a foundation of data

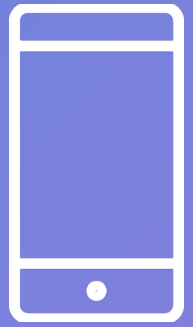
What enables this transformation?

Cloud

Rethink
Data Centers

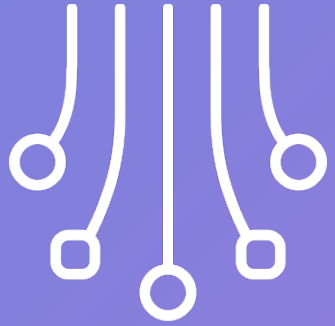
Machine Learning

Rethink
Decision Making



Mobile

Rethink
User Experience



Event Streaming

Rethink
Data

Do you see me?

Or: Would you blindly cross the street with traffic information that is 5 minutes old?

Transportation

ETA

Real-time sensor diagnostics

Driver-rider match

Banking

Fraud detection

Trading and risk systems

Mobile applications / customer experience

Retail

Real-time inventory

Real-time POS reporting

Personalization

Entertainment

Real-time recommendations

Personalized news feed

In-app purchases

This is a fundamental paradigm shift...

Cloud	Future of the datacenter	Infrastructure as code
Event Streaming	Future of data	Data as continuous stream of events

The Event Streaming Paradigm

Two Problems in Application Infrastructure

What's the state of the world?

Solution:
Databases

What's happening in the world?

Solution:
Messaging, RPC, ETL, etc.

ETL/Data Integration

−

Batch

Expensive

Time Consuming

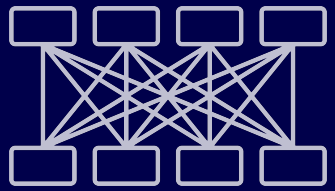
+

High Throughput

Durable

Persistent

Maintains Order



Messaging

−

+

Fast (Low Latency)

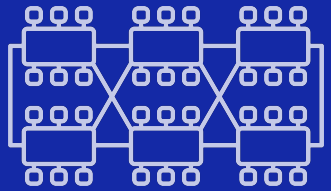
−

Difficult to Scale

No Persistence

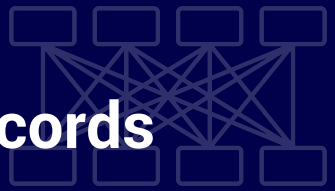
Data Loss

No Replay



ETL/Data Integration

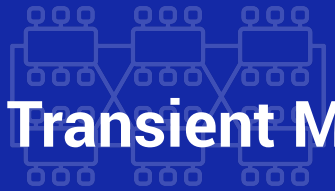
−	+
Batch	High Throughput
Expensive	Durable
Time Consuming	Persistent
	Maintains Order



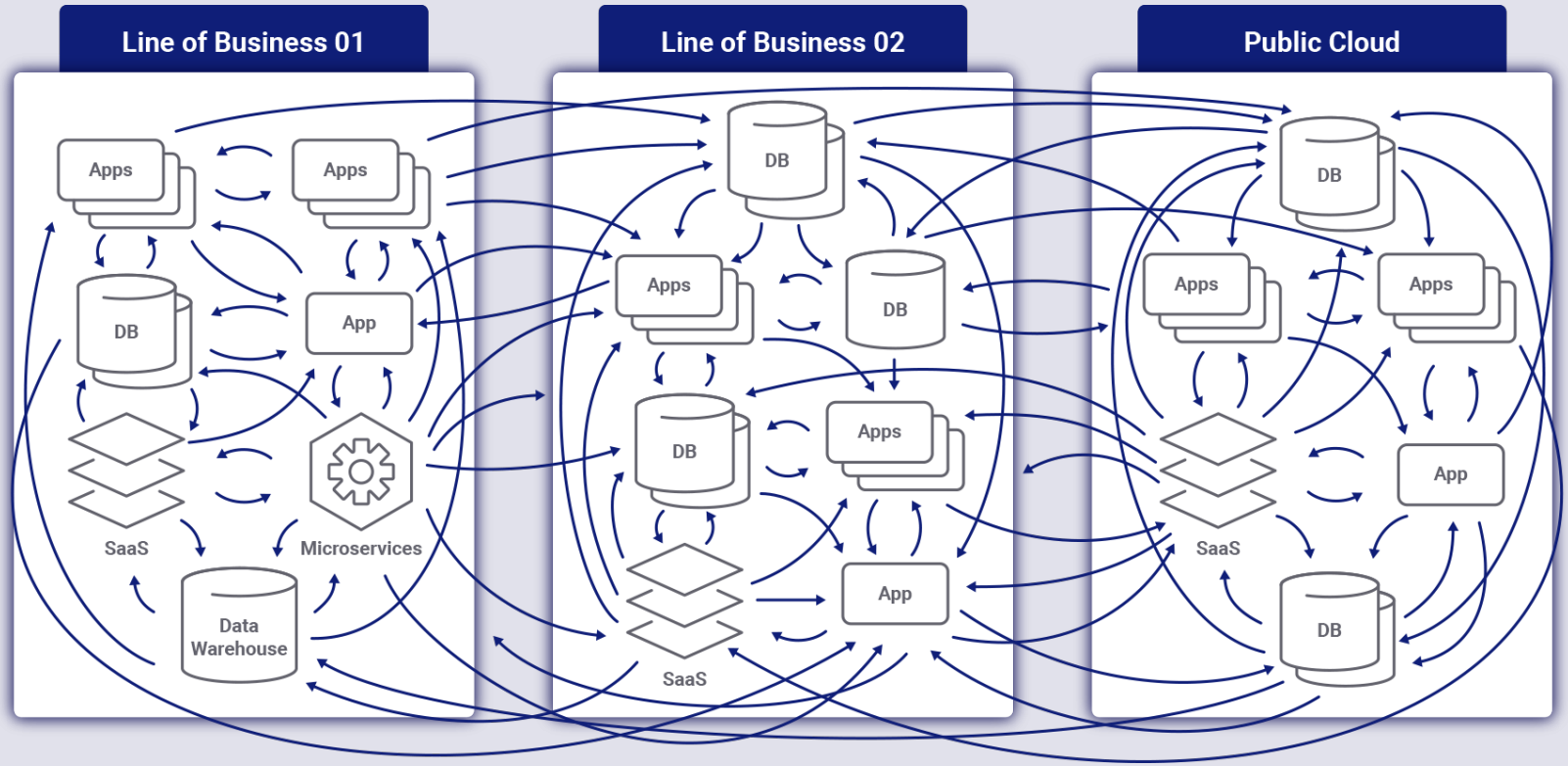
Stored records

Messaging

+	−
Fast (Low Latency)	Difficult to Scale
	No Persistence
	Data Loss
	No Replay



Transient Messages


Line of Business 01
Line of Business 02
Public Cloud
Apps
Apps
DB
App
SaaS
Microservices
Data Warehouse
DB
Apps
DB
DB
Apps
App
SaaS
DB
Apps
Apps
App
SaaS
DB

ETL/Data Integra

Messaging

−

Batch

Expensive

Time Consuming

−

Difficult to Scale

No Persistence

Data Loss

No Replay

Event Streaming Paradigm

+	+
High Throughput	Fast (Low Latency)
Durable	Replay
Persistent	
Maintains Order	

Stored records

Transient Messages

Event Streaming Paradigm

To **rethink data** as neither stored records nor transient messages, but instead as a **continuously updating Stream of Events**

An Event
records the fact that something happened

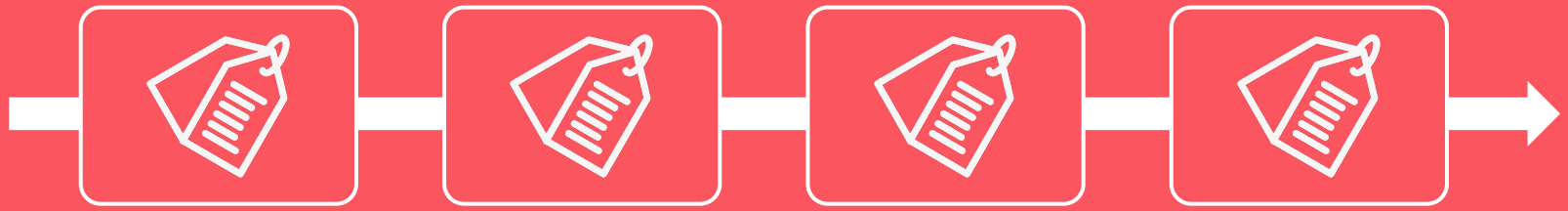
A good
was sold

An invoice
was issued

A payment
was made

A new customer
registered

A Stream
represents history as a sequence of Events



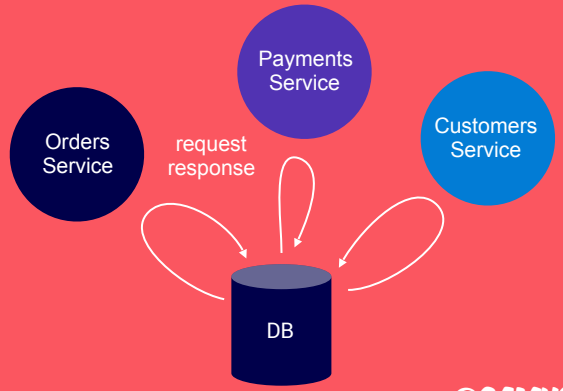
Events change the way we think

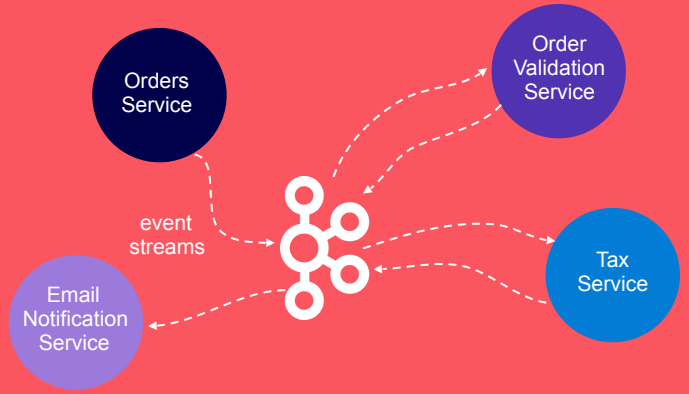
Monolithic Approach

- a database
- a variable
- a singleton
- an RPC

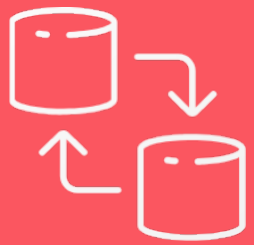
Event-First Approach

- an event
- a stream
- a ‘data’ flow
- a stream processor





An Event Streaming Platform gives you three key functionalities

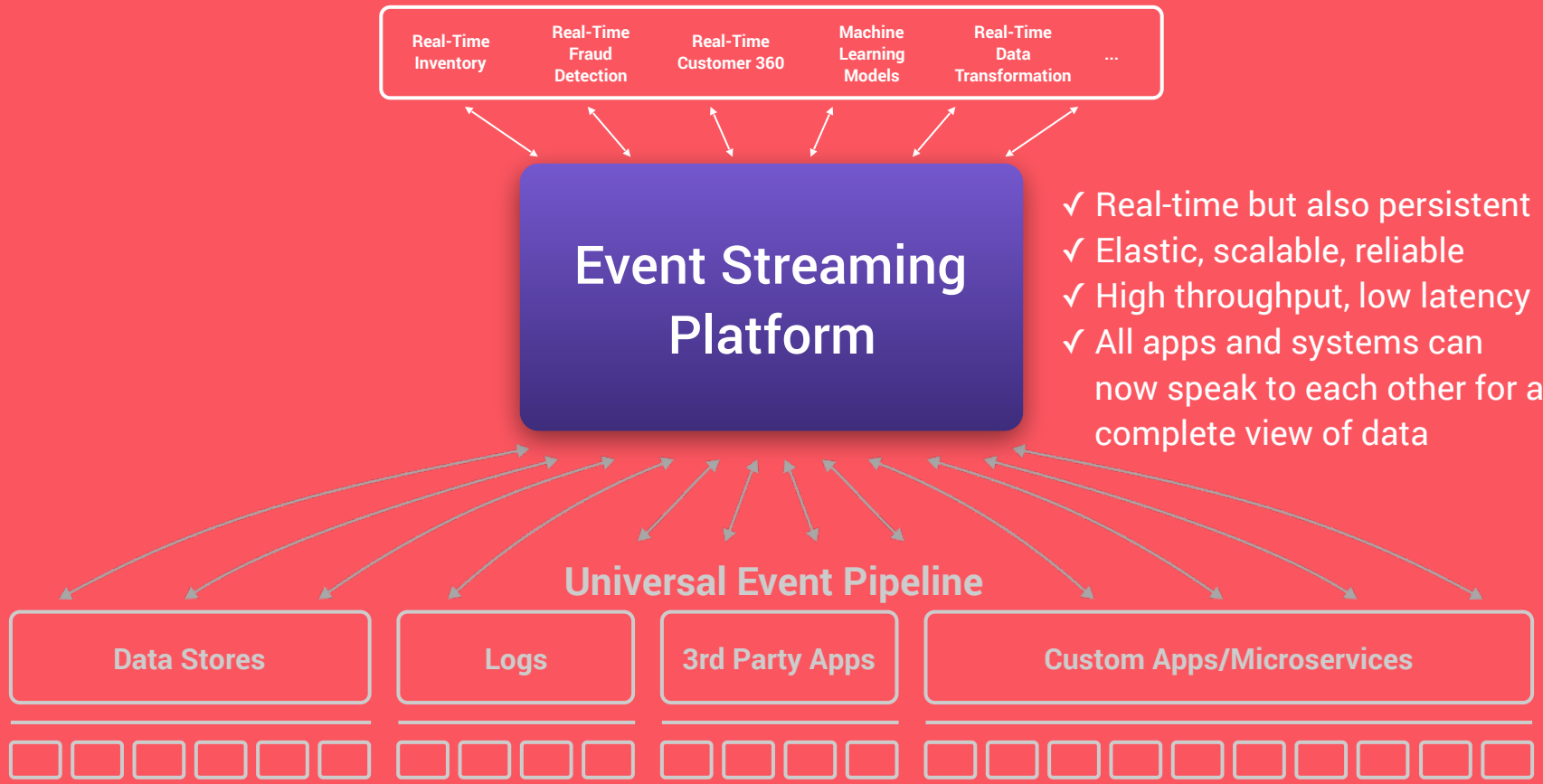


Publish & Subscribe to Events

Store Events

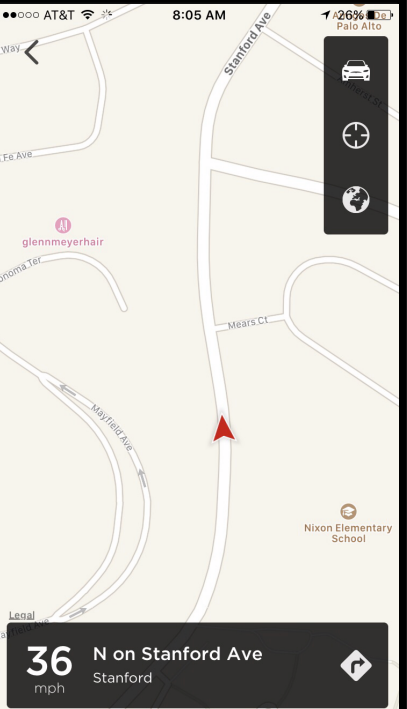
Process & Analyze Events

Event-Driven Apps, with Historical Context



- ✓ Real-time but also persistent
- ✓ Elastic, scalable, reliable
- ✓ High throughput, low latency
- ✓ All apps and systems can now speak to each other for a complete view of data

Why Combine Real-time With Historical Context?



Event-Driven App
(Location Tracking)

Only Real-time Events

Messaging Queues and Event Streaming Platforms can do this

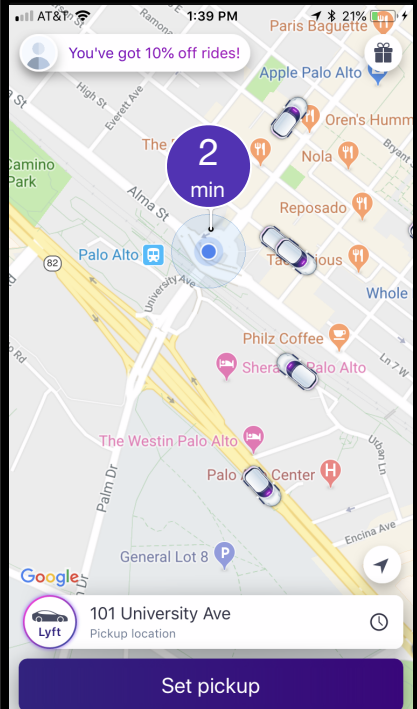
Where is my driver?

Contextual Event-Driven App
(ETA)

Real-time combined with stored data

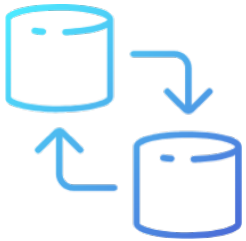
Only Event Streaming Platforms can do this

When will my driver get here?

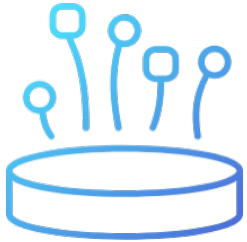


How to Build
Event Streaming Architectures
With Kafka

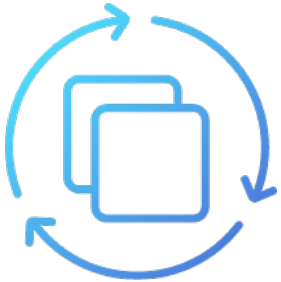
APACHE kafka® is a distributed event streaming platform



Publish & Subscribe to Events



Store Events



Process & Analyze Events

confluent

01
Stream your data in real-time as Events

02
Store your Event Streams

03
Process & Analyze your Events Streams

01

Stream y
in real-ti

ps, microservices

ka producer client from your favorite language

python™ GO ... and many more

https://gamov.dev/kafka-clients-demo

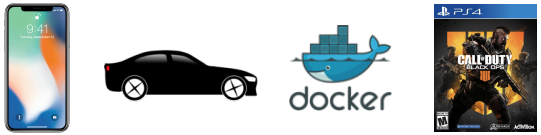
other systems

Connect plus a Connector for your system

amazon web services S3 elastic ORACLE GOLDEN GATE mongoDB ... and many more

confluent.io/hub

From apps, microservices: producer example



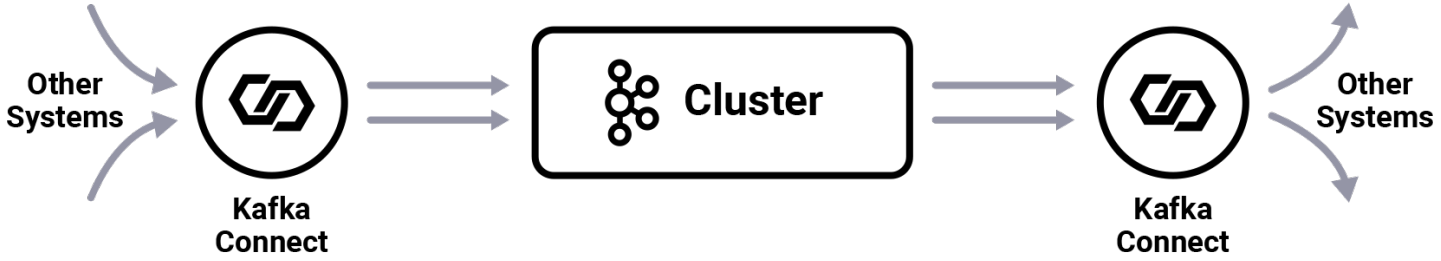
... and more

```
@Service
class Producer @Autowired
constructor(private val kafkaTemplate: KafkaTemplate<String, User>) {

  @Value("\${topic.name}")
  lateinit var TOPIC: String;

  internal fun sendMessage(user: User) {
    this.kafkaTemplate.send(this.TOPIC, user.getName(), user)
    println(String.format("Produced user -> %s", user))
  }
}
```

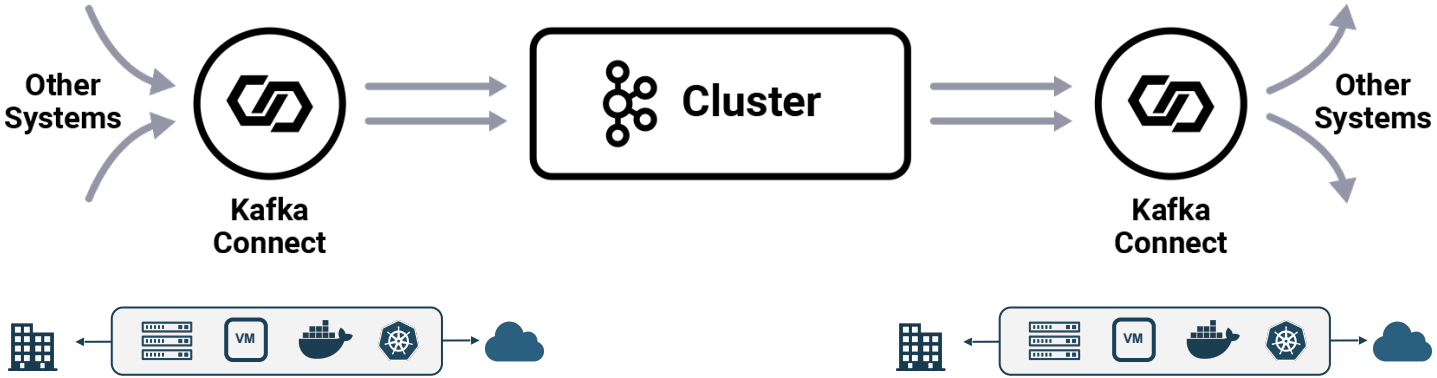
From/to other systems: Kafka Connect



Tip: Great option to gradually move workloads to Kafka while keeping production running!

Kafka Connect

- Deployed standalone (development) or as a **distributed cluster (production)**
- Elastic service that works on bare-metal, VMs, containers, Kubernetes, ...
- The individual ‘Connector’ determines delivery guarantees, e.g., exactly-once



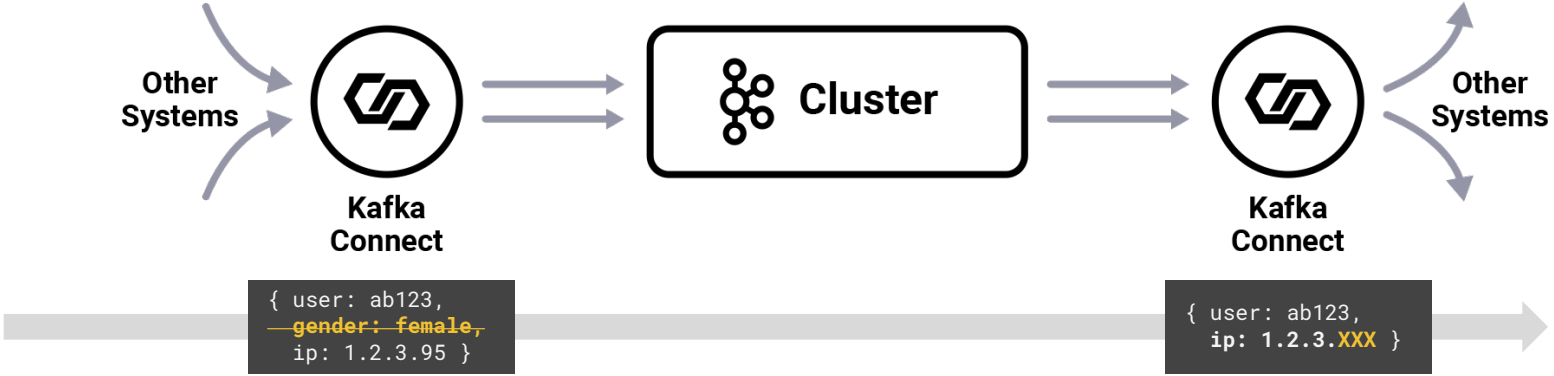
Single Message Transforms for real-time ETL

Ingress: modify an Event before storing

- Obfuscate sensitive information, e.g. PII
- Add origin of event for lineage tracking
- Remove unnecessary data fields
- … and more

Egress: modify an Event on its way out

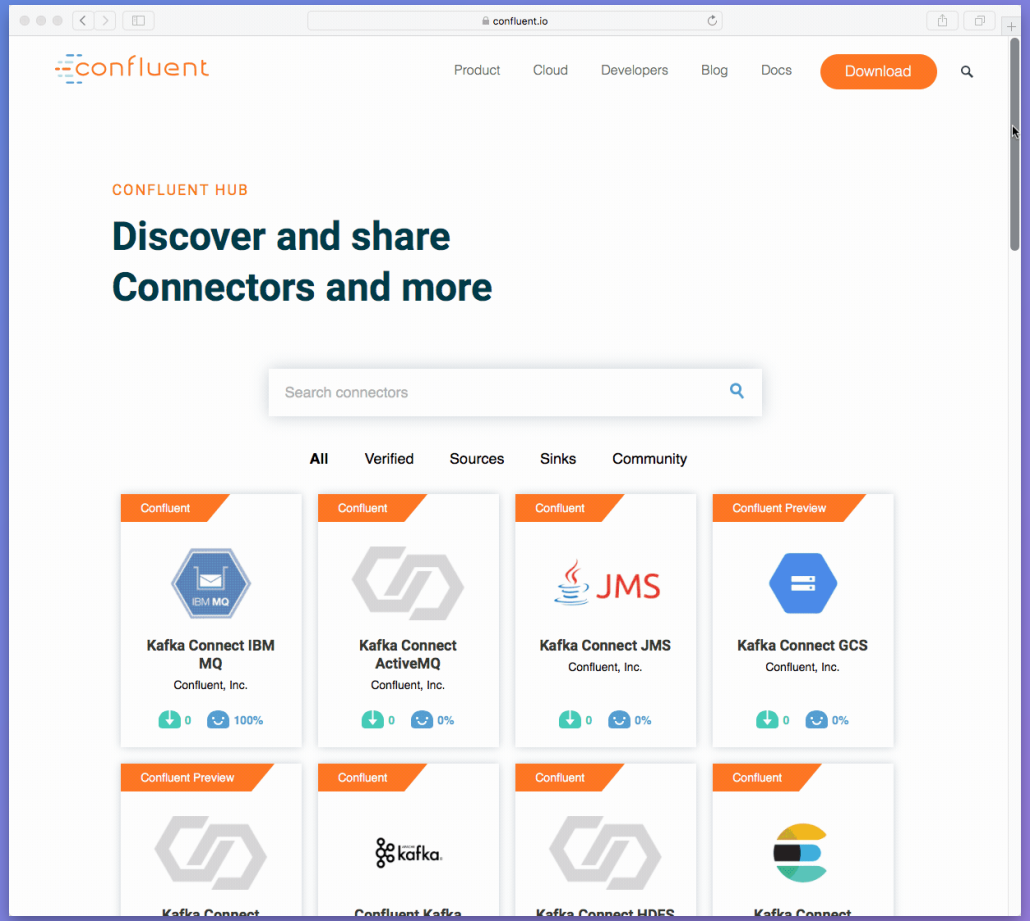
- Route high-priority events to faster stores
- Direct events to different Elasticsearch indexes
- Cast data types to match destination
- … and more



Confluent Hub

- ✓ Discover Connectors, SMTs, and converters
- ✓ Documentation, support, etc.
- ✓ Easy installation

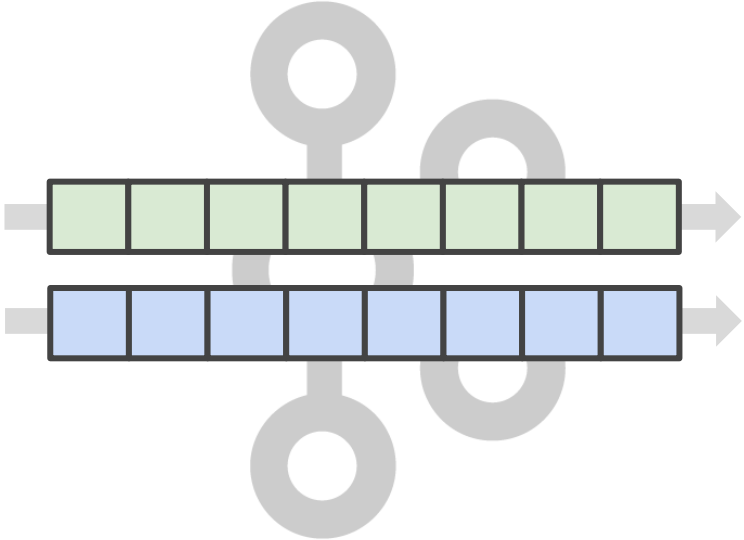
confluent.io/hub



02

Store your Event Streams

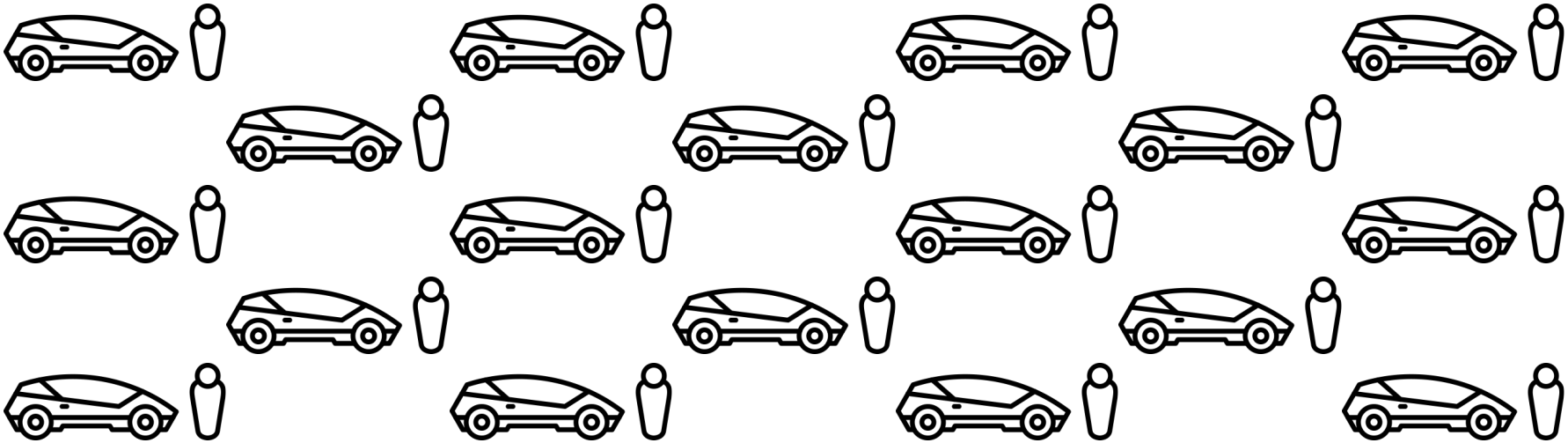
Kafka Cluster



Storage is

- ✓ Distributed
- ✓ Scalable
- ✓ Reliable
- ✓ Durable
- ✓ Performant

Kafka scales from S to XXL



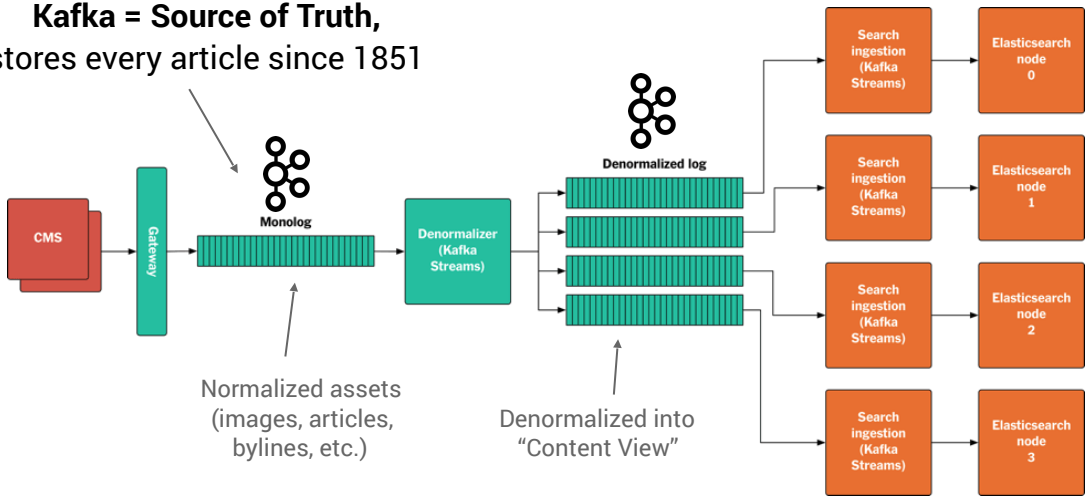
Messages / sec	Topics	Partitions	Brokers
1,000,000	50,000	100,000	1,500

https://www.confluent.io/blog/publishing-apache-kafka-new-york-times/

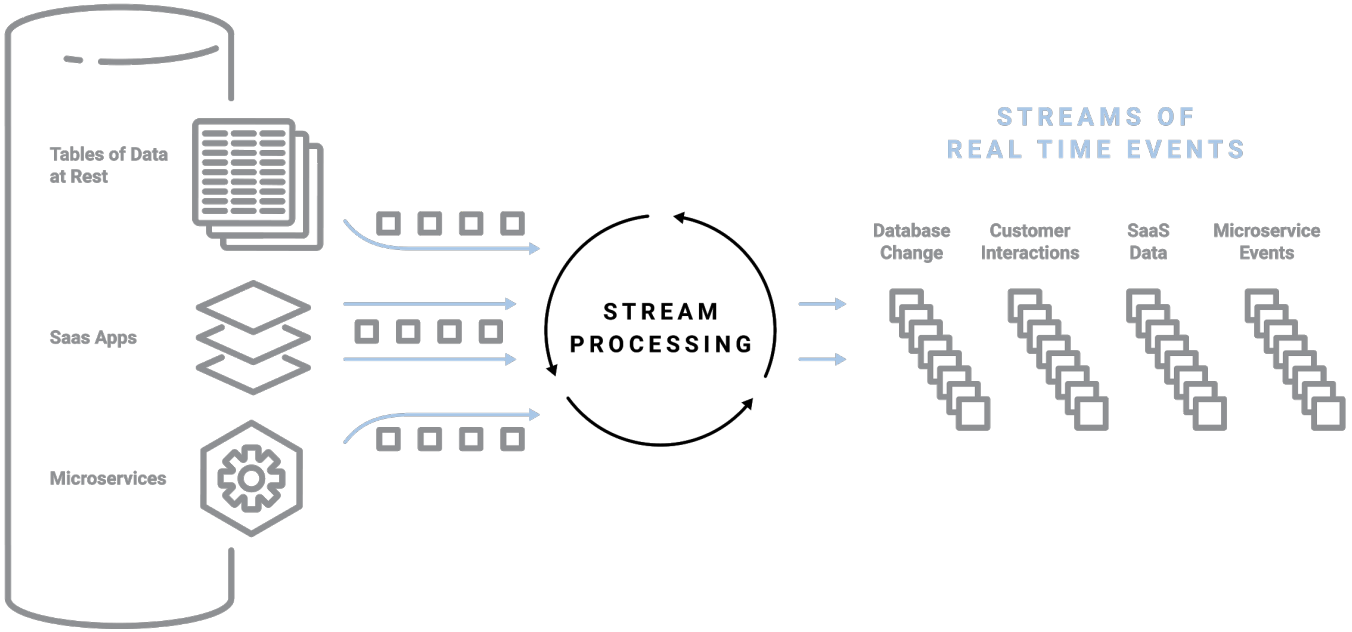
Store your Events as long as you want

The New York Times

Kafka = Source of Truth,
stores every article since 1851



~~Achievement~~ Data Unlocked:
All Your Data Now Available as Streams of Events



03

Process & Analyze your Events Streams

With Streaming SQL

KSQL

With apps, microservices

or

... and more
Kafka consumer clients

With separate frameworks

... and more

Stream Processing with KSQL

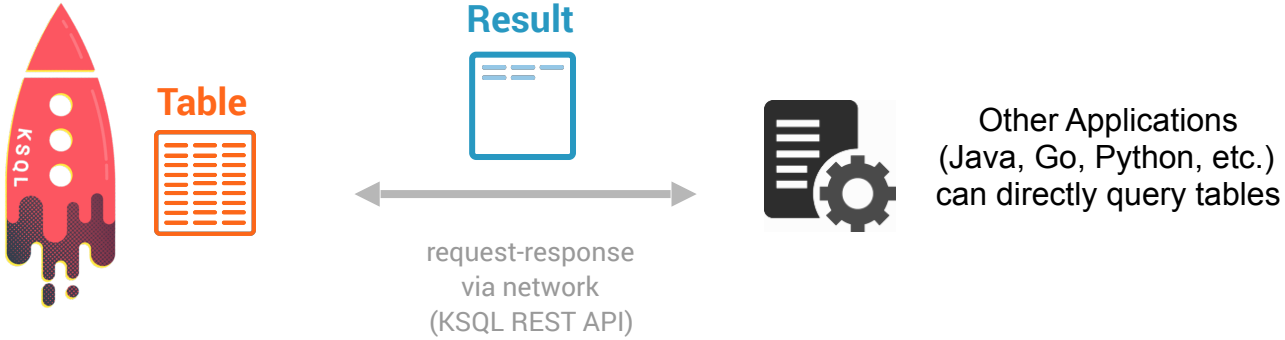
Process event streams to create new, *continuously updated* streams or tables


```
CREATE TABLE OrderTotals AS SELECT * FROM ... EMIT CHANGES
```

Upcoming feature (KLIP-8)

Stream Processing with KSQL

Query tables in Kafka from other apps, similar to a relational database




```
SELECT * FROM OrderTotals WHERE region = 'Europe'
```

An example use cas

Creating an event-dri s database

Max
streaming c

https://github.com/garyrussell/s1p-2019

```
@StreamListener
fun processCurrency(input: KStream<String, Double>) {

  val groupByKey: KGroupedStream<String, Double> = input.groupByKey()

  val countKTable = groupByKey.count()
  val sumKTable = groupByKey.reduce { value1, value2 → value1 + value2 }

  val avgRate = sumKTable.join(countKTable) { sum, count → sum / count }
  avgRate.toStream().to("avg-rates")
}
```

```
@StreamListener
fun processCurrency(input: KStream<String, Double>) {

  val groupByKey: KGroupedStream<String, Double> = input.groupByKey()

  val countKTable = groupByKey.count()
  val sumKTable = groupByKey.reduce { value1, value2 → value1 + value2 }

  val avgRate = sumKTable.join(countKTable) { sum, count → sum / count }
  avgRate.toStream().to("avg-rates")
}
```

```
@StreamListener
fun processCurrency(input: KStream<String, Double>) {

  val groupByKey: KGroupedStream<String, Double> = input.groupByKey()

  val countKTable = groupByKey.count()
  val sumKTable = groupByKey.reduce { value1, value2 -> value1 + value2 }

  val avgRate = sumKTable.join(countKTable) { sum, count -> sum / count }
  avgRate.toStream().to("avg-rates")
}
```

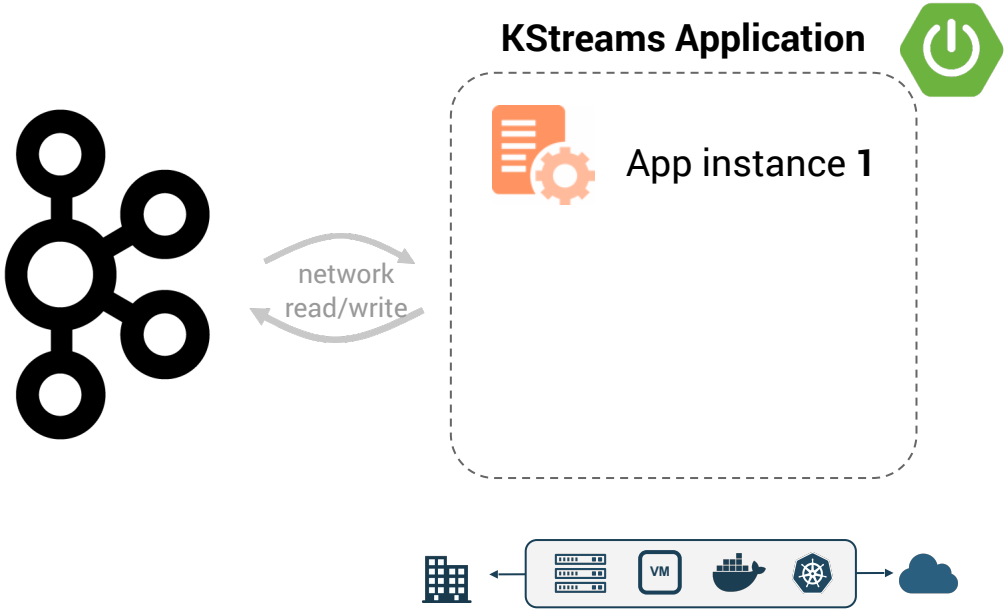
```
@StreamListener
fun processCurrency(input: KStream<String, Double>) {

  val groupByKey: KGroupedStream<String, Double> = input.groupByKey()

  val countKTable = groupByKey.count()
  val sumKTable = groupByKey.reduce { value1, value2 → value1 + value2 }

  val avgRate = sumKTable.join(countKTable) { sum, count → sum / count }
  avgRate.toStream().to("avg-rates")
}
```

Where your Kafka Streams apps live



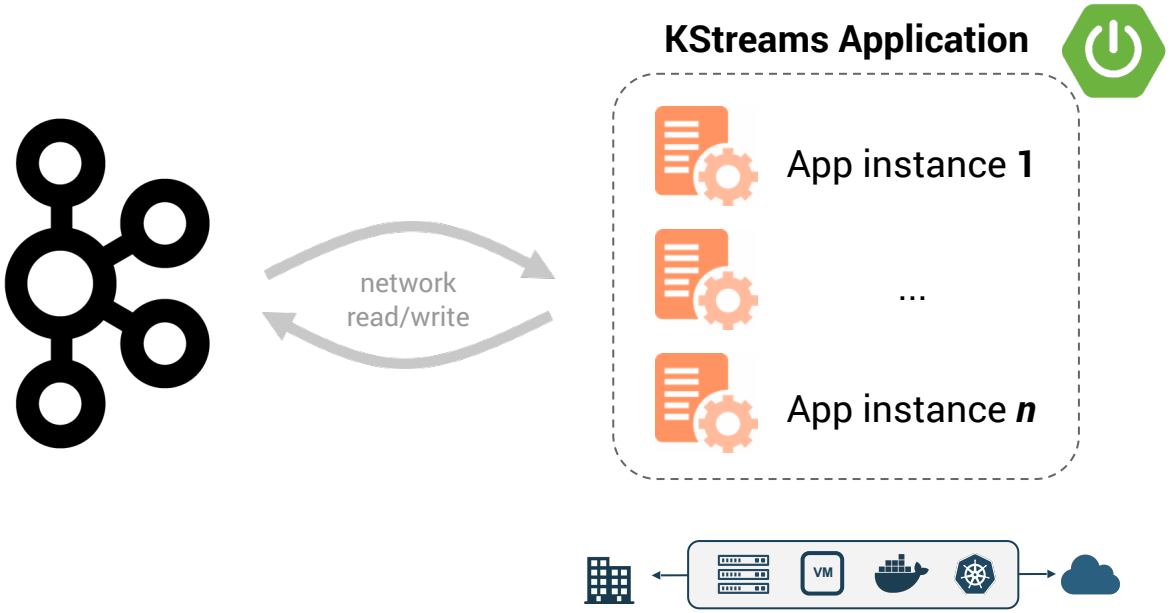

```
fun processCurrency(input: KStream<String,
Double>) {

  val groupByKey: KGroupedStream<String,
Double> = input.groupByKey()

  val countKTable = groupByKey.count()
  val sumKTable = groupByKey.reduce
{ value1, value2 → value1 + value2 }

  val avgRate = sumKTable.join(countKTable)
{ sum, count → sum / count }
  avgRate.toStream().to("avg-rates")
}
```

Where your Kafka Streams apps live

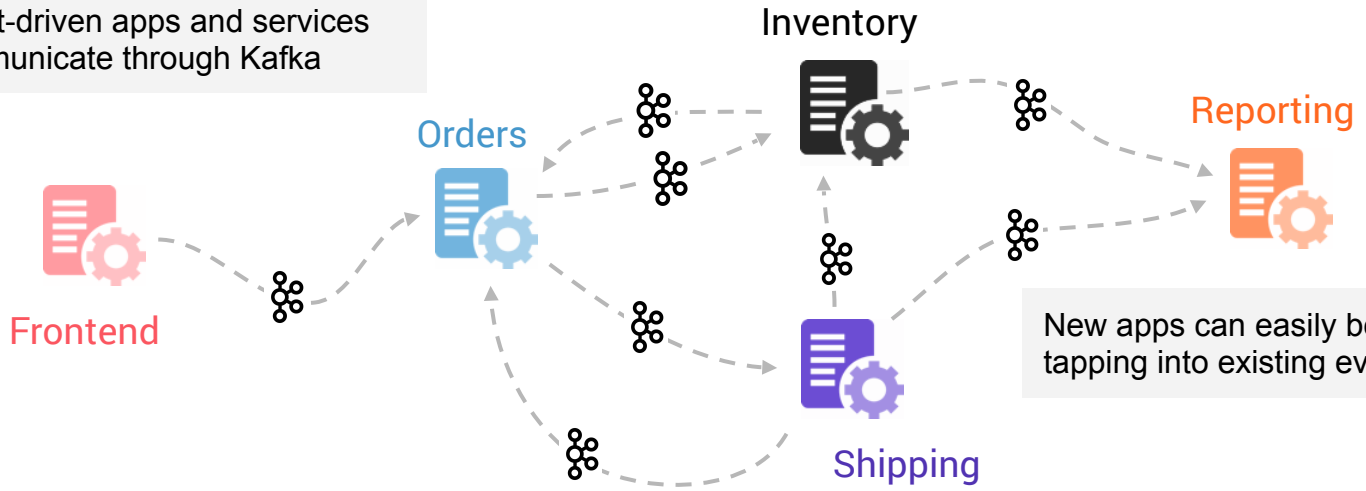


- ✓ Elastic & Scalable
- ✓ Fault-tolerant
- ✓ Exactly-once
- ✓ Kafka security
- ✓ Aggregations
- ✓ Windowing
- ✓ Streams & Tables

Stream Processing with Kafka Streams apps

Process event streams to create new, *continuously updated* streams or tables

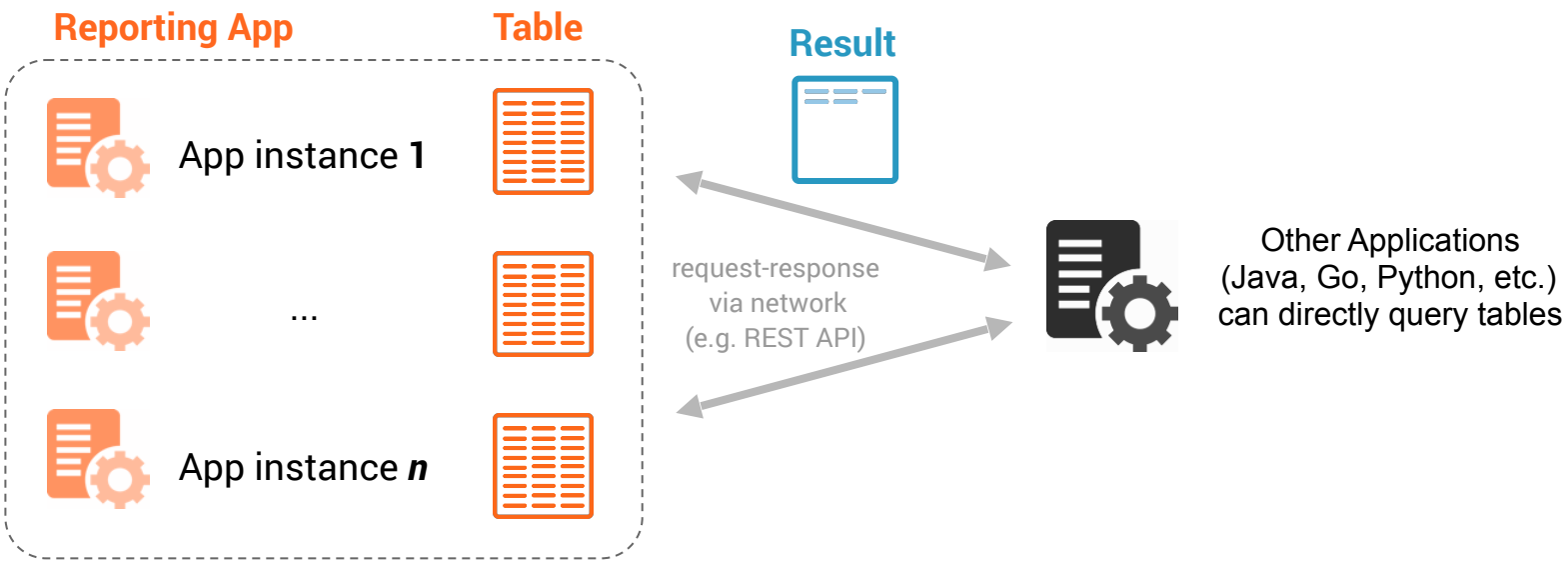
Event-driven apps and services communicate through Kafka



New apps can easily be added by tapping into existing event streams

Stream Processing with Kafka Streams apps

Query your application's tables and state from other apps



- Spring For Apache Kafka 2.3 - What's New?

Producers

- Option for Producer per thread
- `AggregatingReplyingKafkaTemplate`

- Spring For Apache Kafka 2.3 - What's New?

Consumers

- Consumer `RecordInterceptor`
- Relative Seeks in `ConsumerSeekAware`
- Configurable delay between `poll()`s
- Micrometer Timers
- Backoff between redeliveries after delivery failures
- `RetryingDeserializer`

Pivotal

- Spring For Apache Kafka 2.3 - What's New?

Streams

- `RecoveringDeserializationExceptionHandler`
- Transformers
  - Add headers (SpEL)
  - Invoke Spring Integration flows

Pivotal

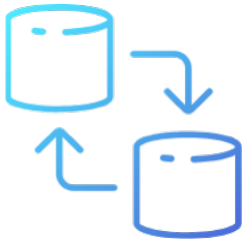
- Spring For Apache Kafka 2.3 - What's New?

Miscellaneous

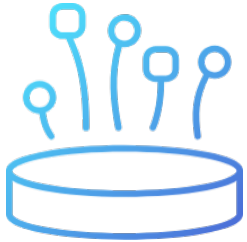
- Delegating `Serializer/Deserializer`

APACHE kafka®

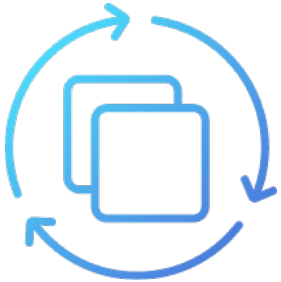
is a distributed event streaming platform



Publish & Subscribe to Events



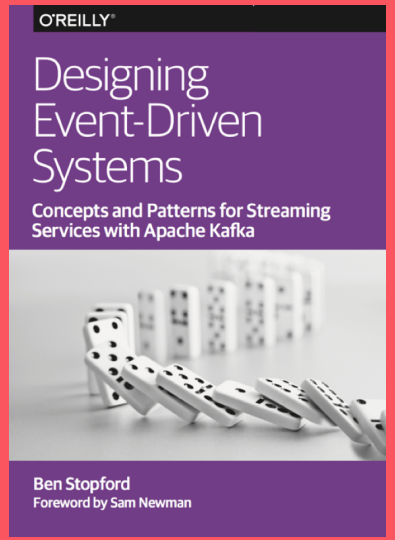
Store Events

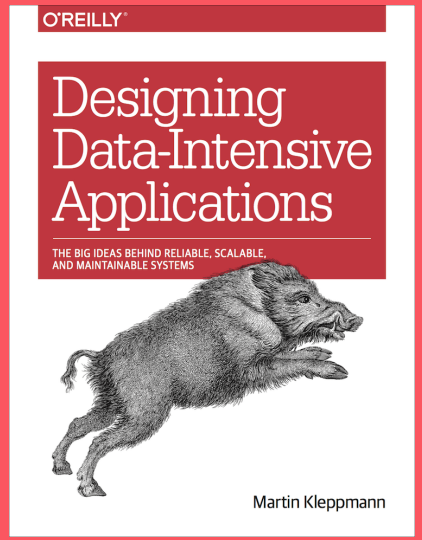


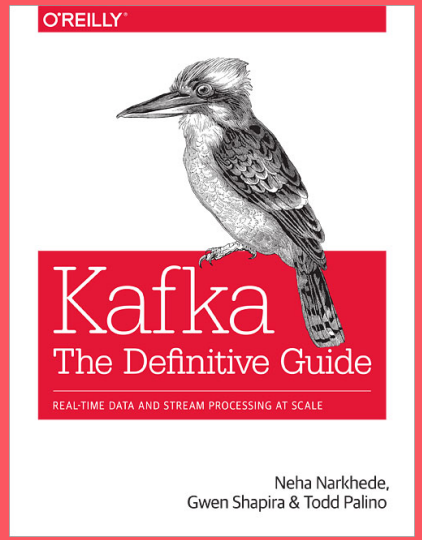
Process & Analyze Events

Where to go from here

for more details on event-driven architectures with Kafka







THANK YOU

@gamussa
viktor@confluent.io
@gprussell
grussell@pivotal.io

---

cnfl.io/meetups cnfl.io/blog cnfl.io/slack