Introduction to Stream Processing with Apache Flink

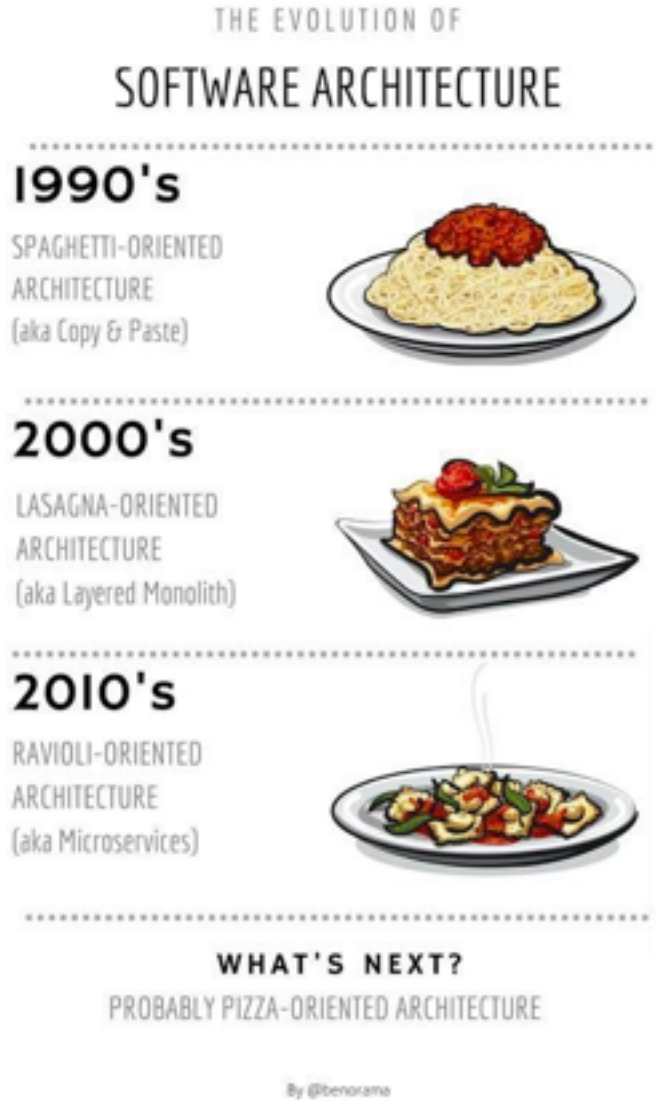
Tugdual Grall
@tgrall

{“about” : “me”}

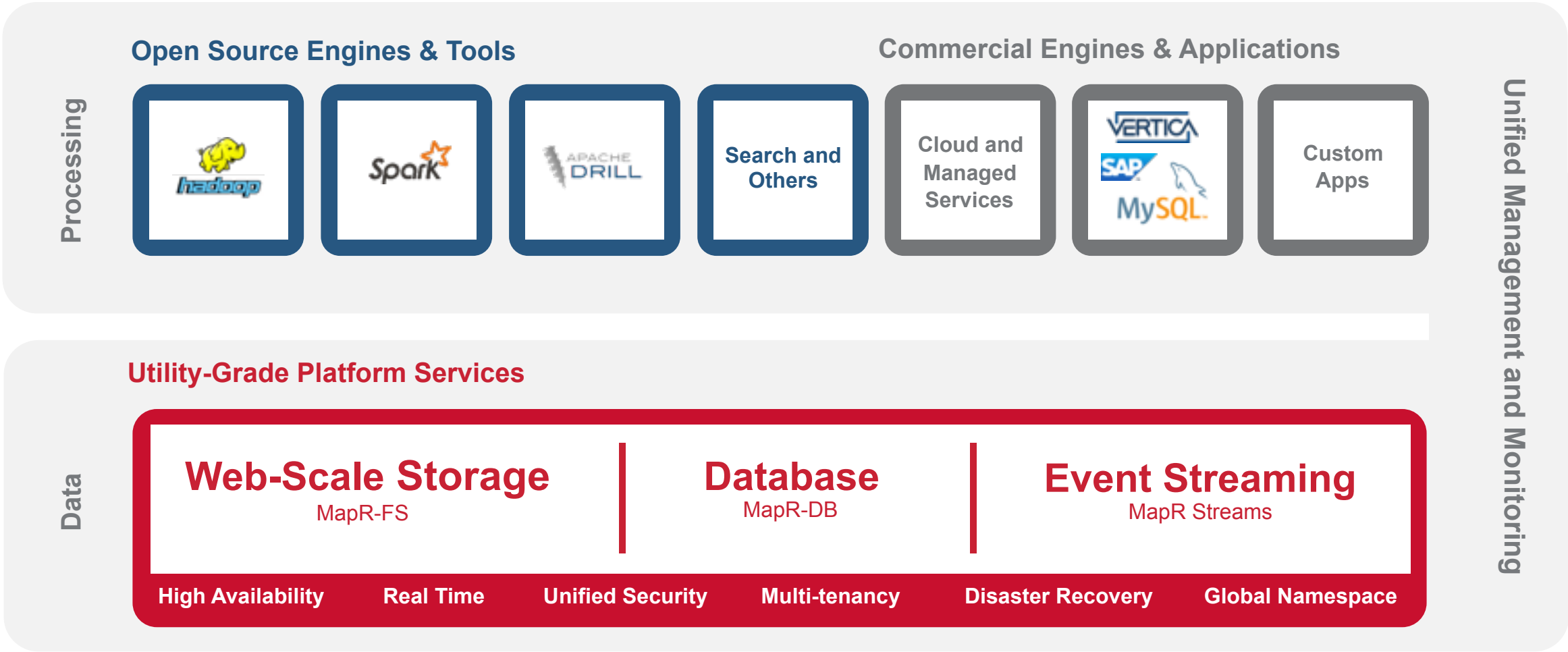
Tugdual “Tug” Grall

- **MapR : Technical Evangelist**
- ***MongoDB, Couchbase, eXo, Oracle***
- ***NantesJUG co-founder***

- ***@tgrall***
- http://tgrall.github.io
- tug@mapr.com / tugdual@gmail.com



MapR Converged Data Platform

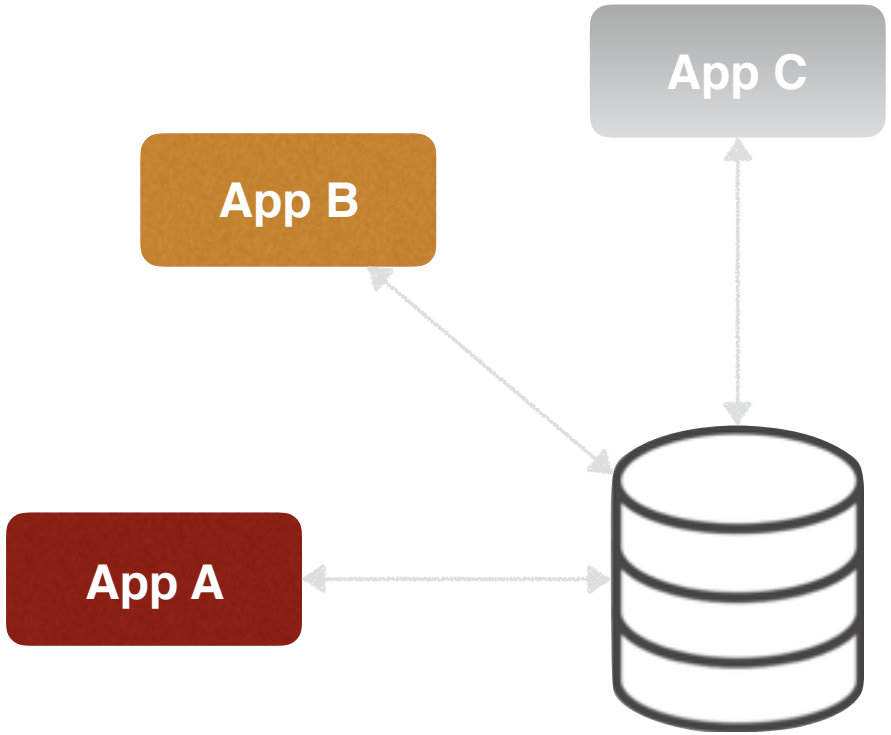


Streaming

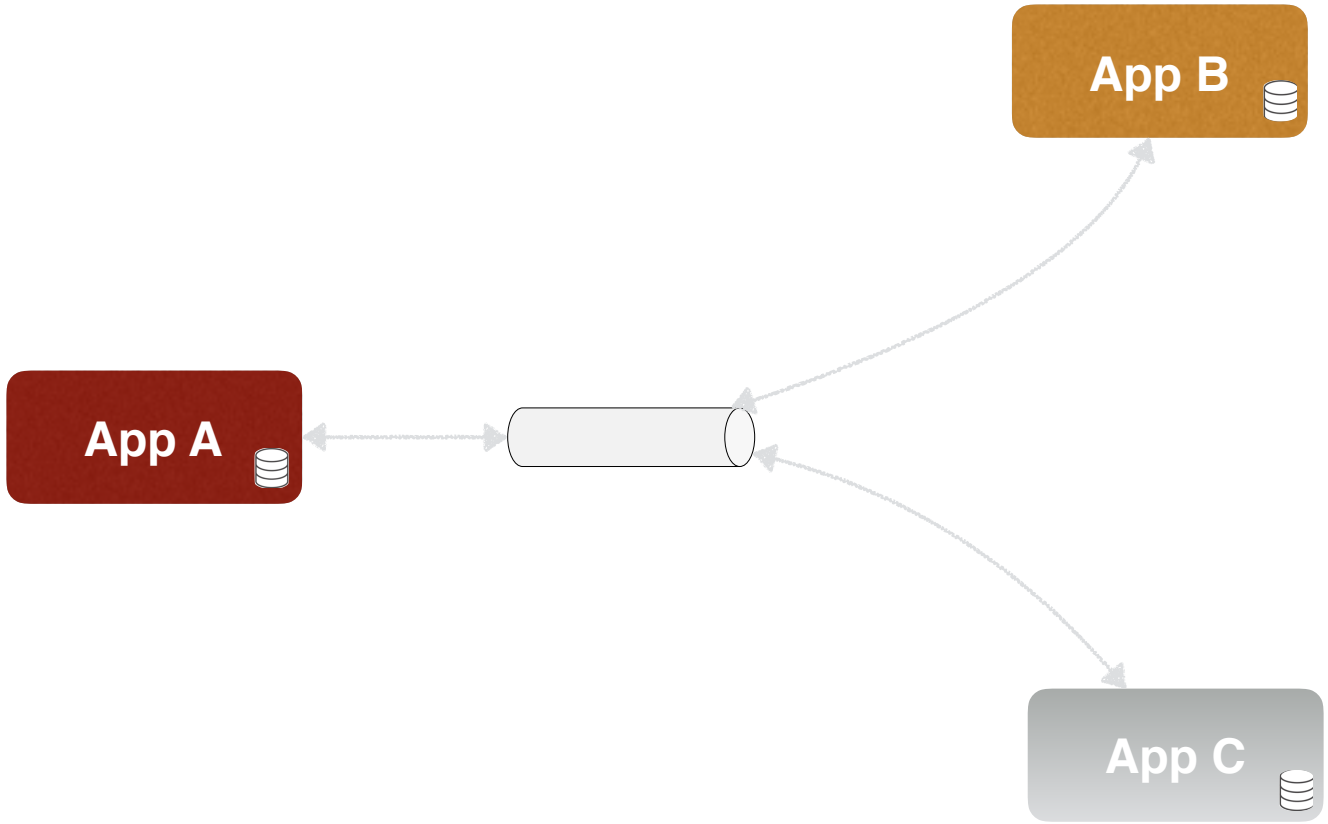
Streaming technology is enabling the obvious:
continuous processing on data
that is continuously produced

Hint: you already have streaming data

Decoupling



State managed centralized

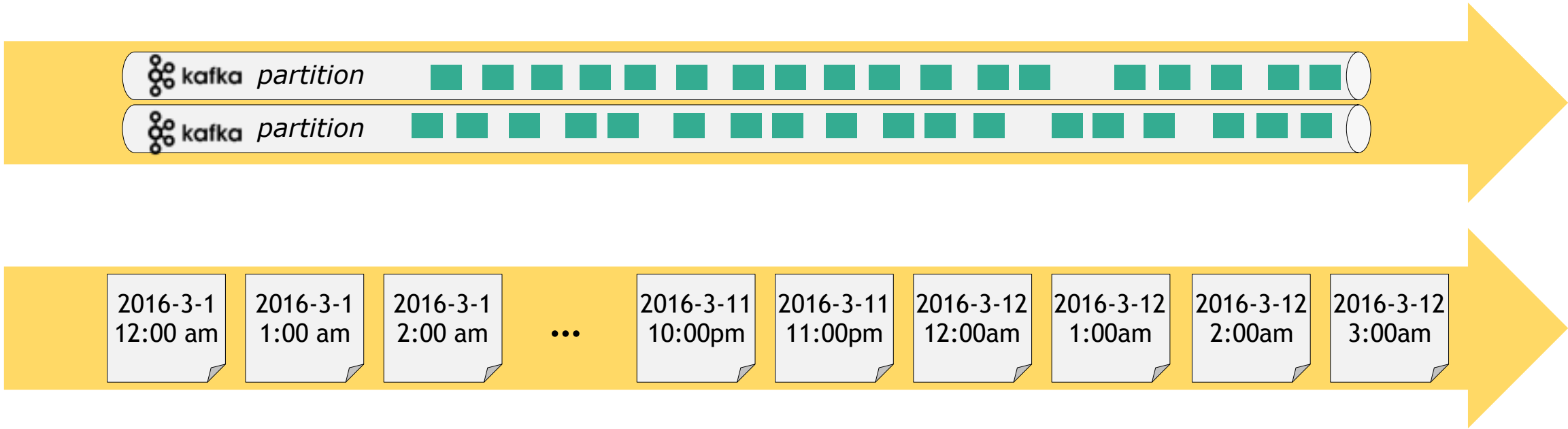


Applications build their own state

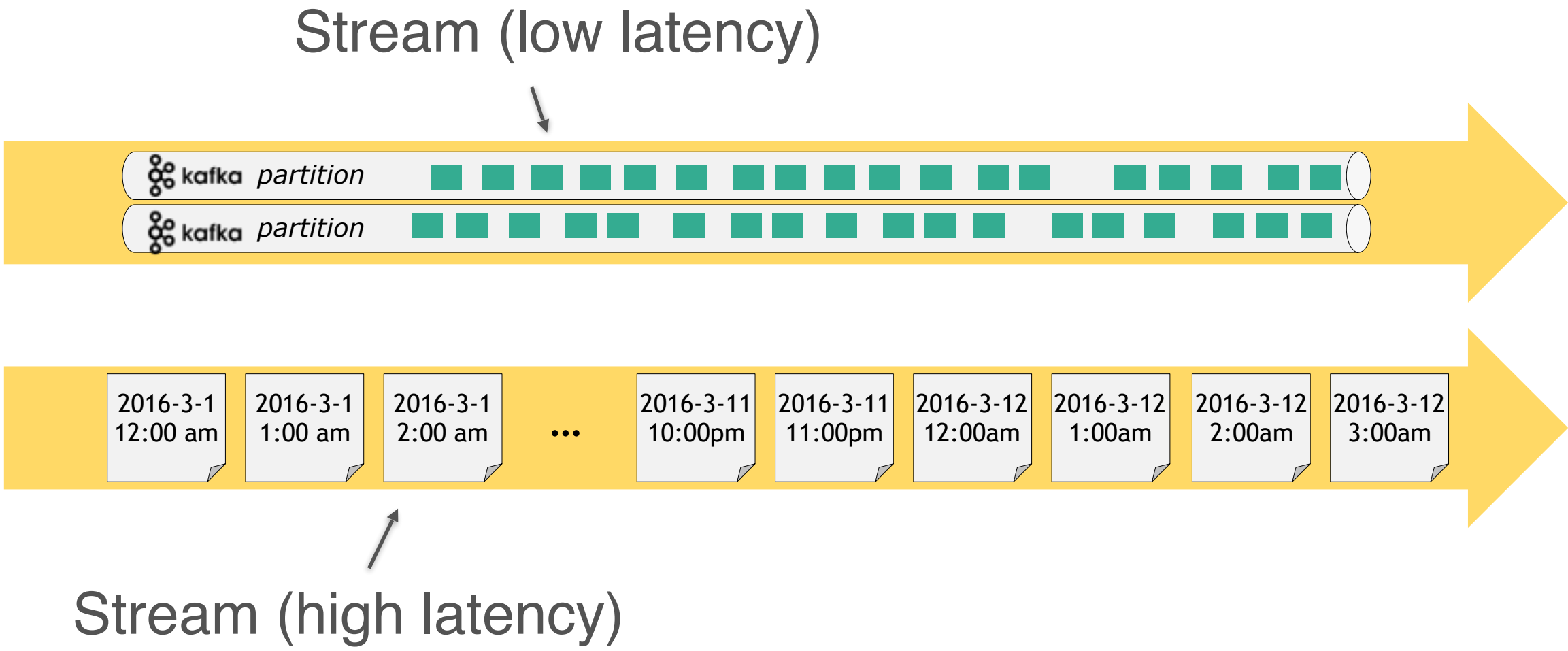
Data Pipelines

Event Stream

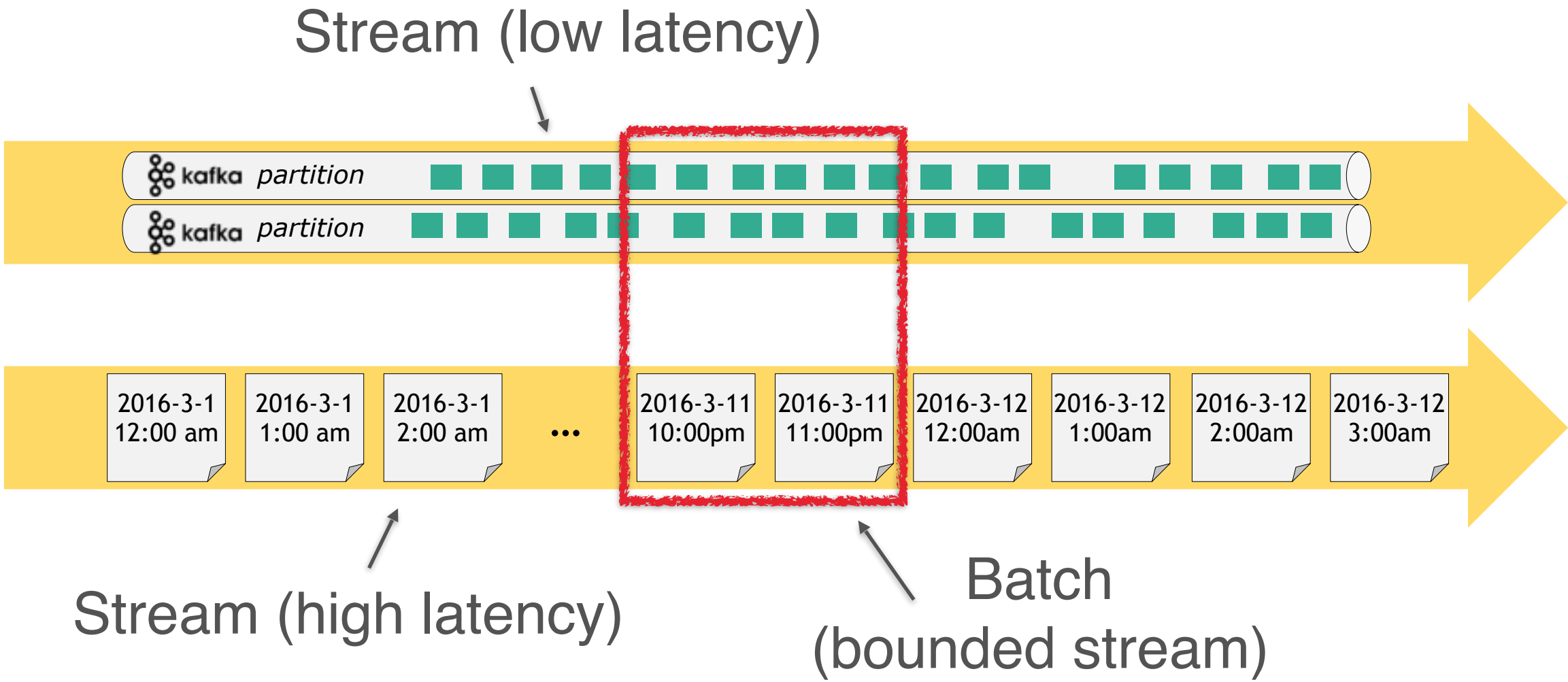
Streaming and Batch



Streaming and Batch



Streaming and Batch



Processing

- Request / Response

Processing

- Request / Response
- Batch

Processing

- Request / Response
- Batch
- Stream Processing

Processing

- Request / Response
- Batch
- Stream Processing
  - Real-time reaction to events
  - Continuous applications
  - Process both real-time and historical data

dataArtisans

Flink Architecture

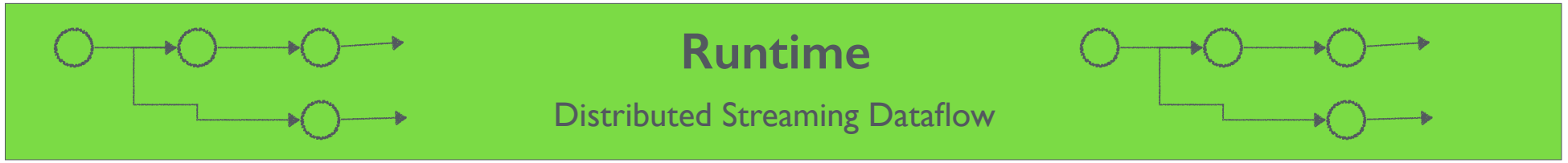
Flink Architecture

Deployment

Local	Cluster	Cloud
Single JVM	Standalone, YARN, Mesos	AWS, Google

Flink Architecture

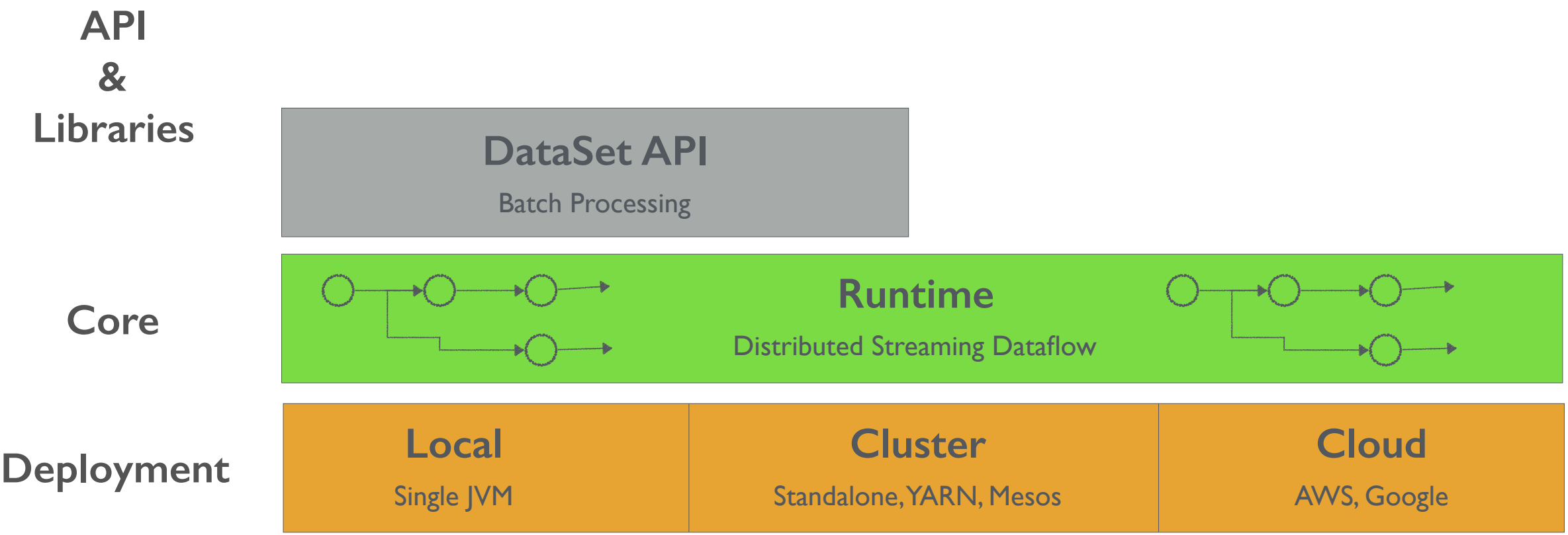
Core



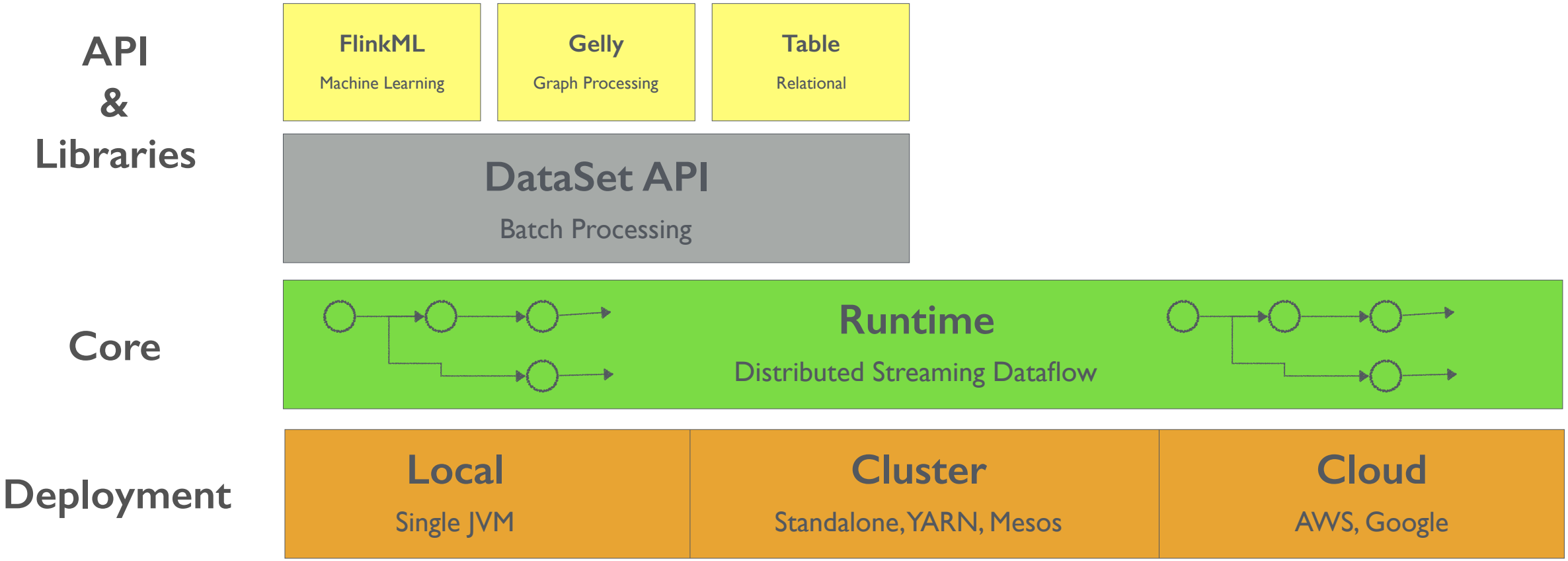
Deployment

Local	Cluster	Cloud
Single JVM	Standalone, YARN, Mesos	AWS, Google

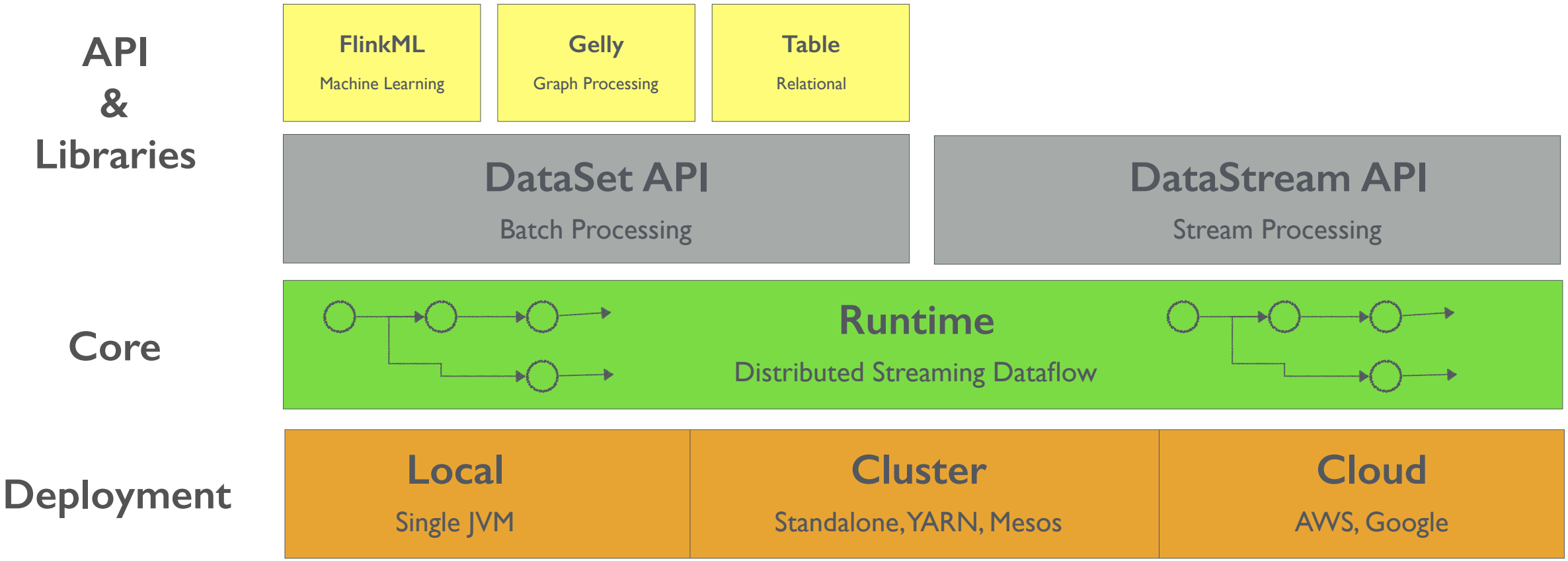
Flink Architecture



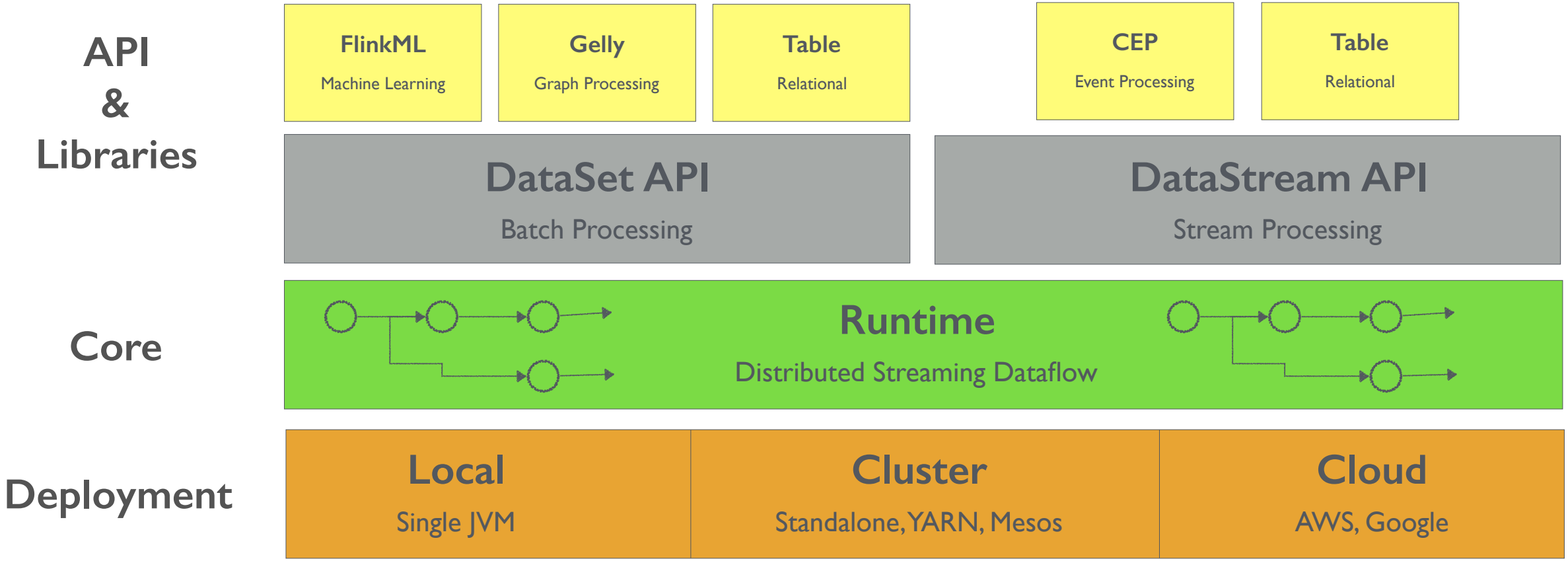
Flink Architecture



Flink Architecture



Flink Architecture



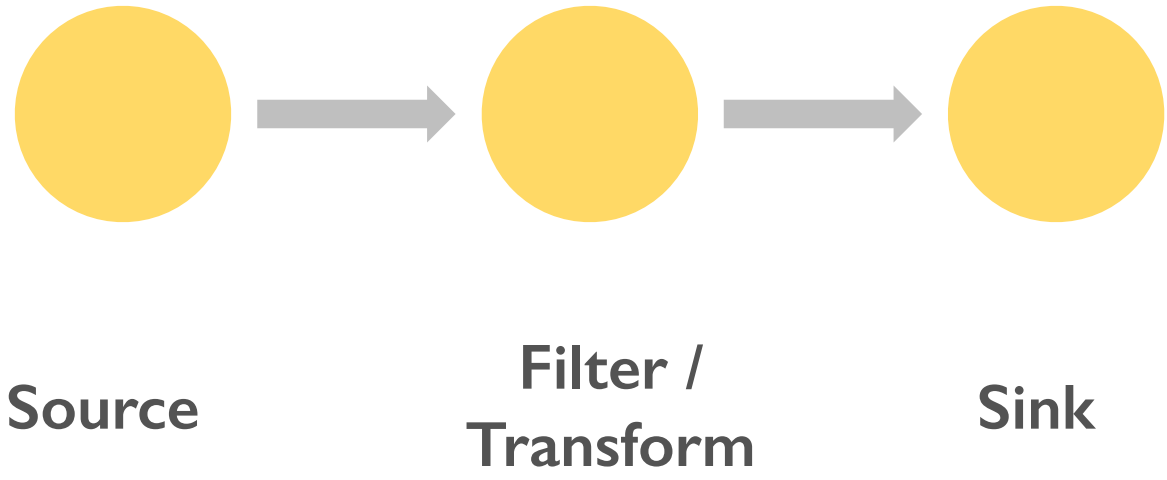
Demonstration

Flink Basics

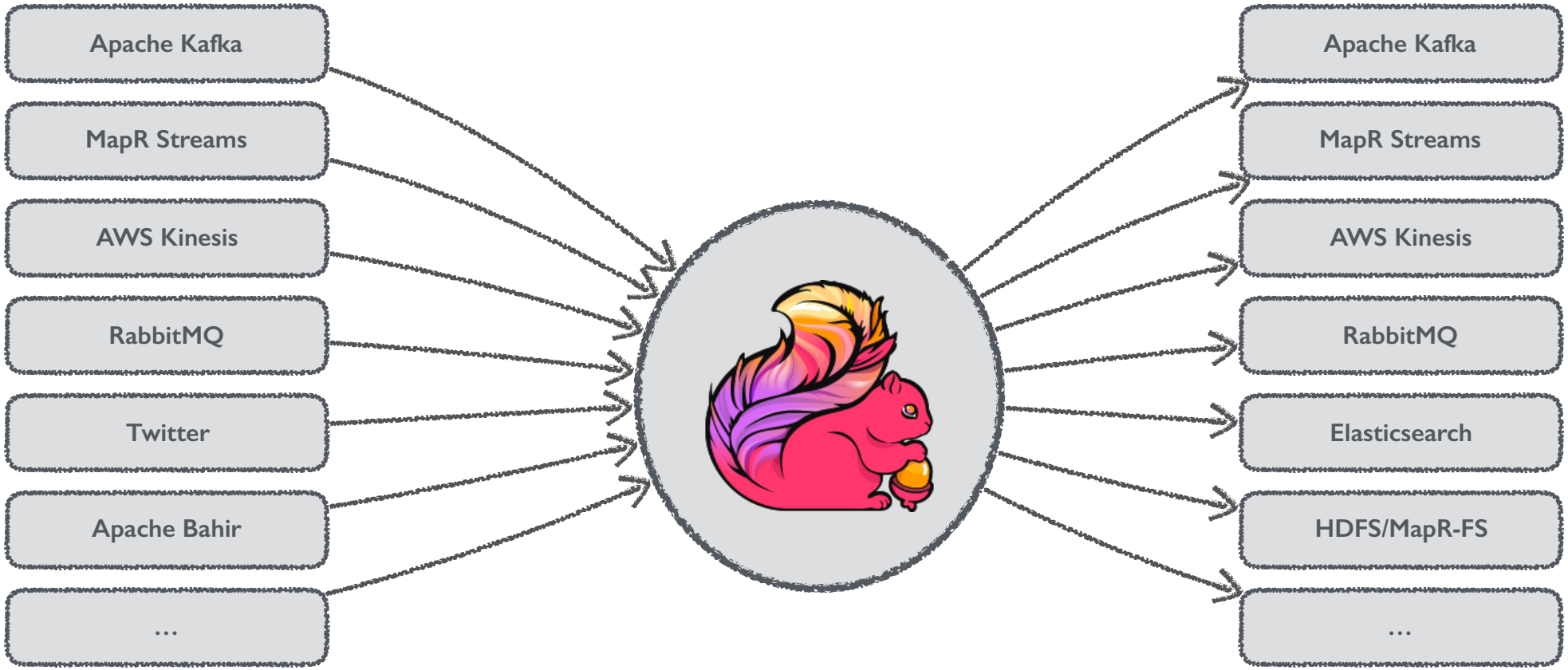
Batch & Stream

```
case class Word (word: String, frequency: Int)

// DataSet API - Batch
val lines: DataSet[String] = env.readTextFile(…)

lines.flatMap {line => line.split(" ").map(word => Word(word,1))}
         .groupBy("word").sum("frequency")
         .print()



// DataStream API - Streaming
val lines: DataSream[String] = env.fromSocketStream(...)

lines.flatMap {line => line.split(" ").map(word => Word(word,1))}
     .keyBy("word").window(Time.of(5,SECONDS))
     .every(Time.of(1,SECONDS)).sum("frequency")
     .print()
```

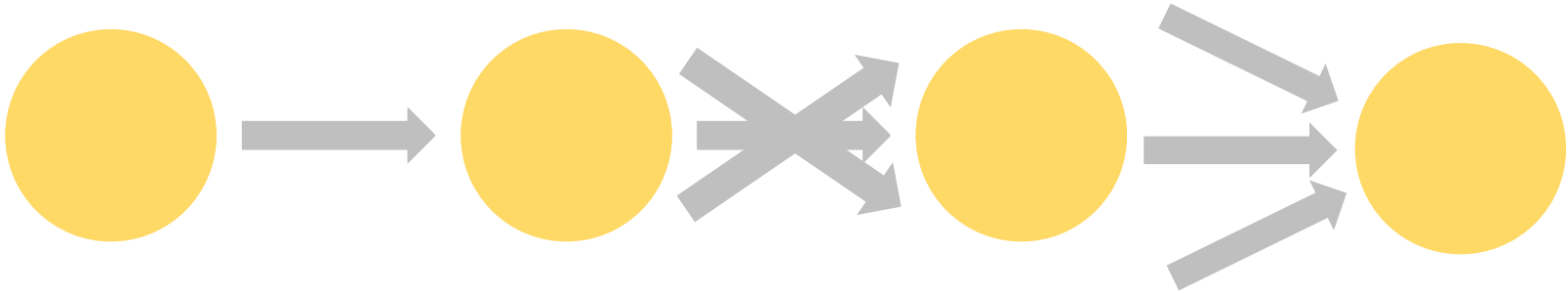
Steam Processing



Flink Ecosystem



Stateful Steam Processing



Source

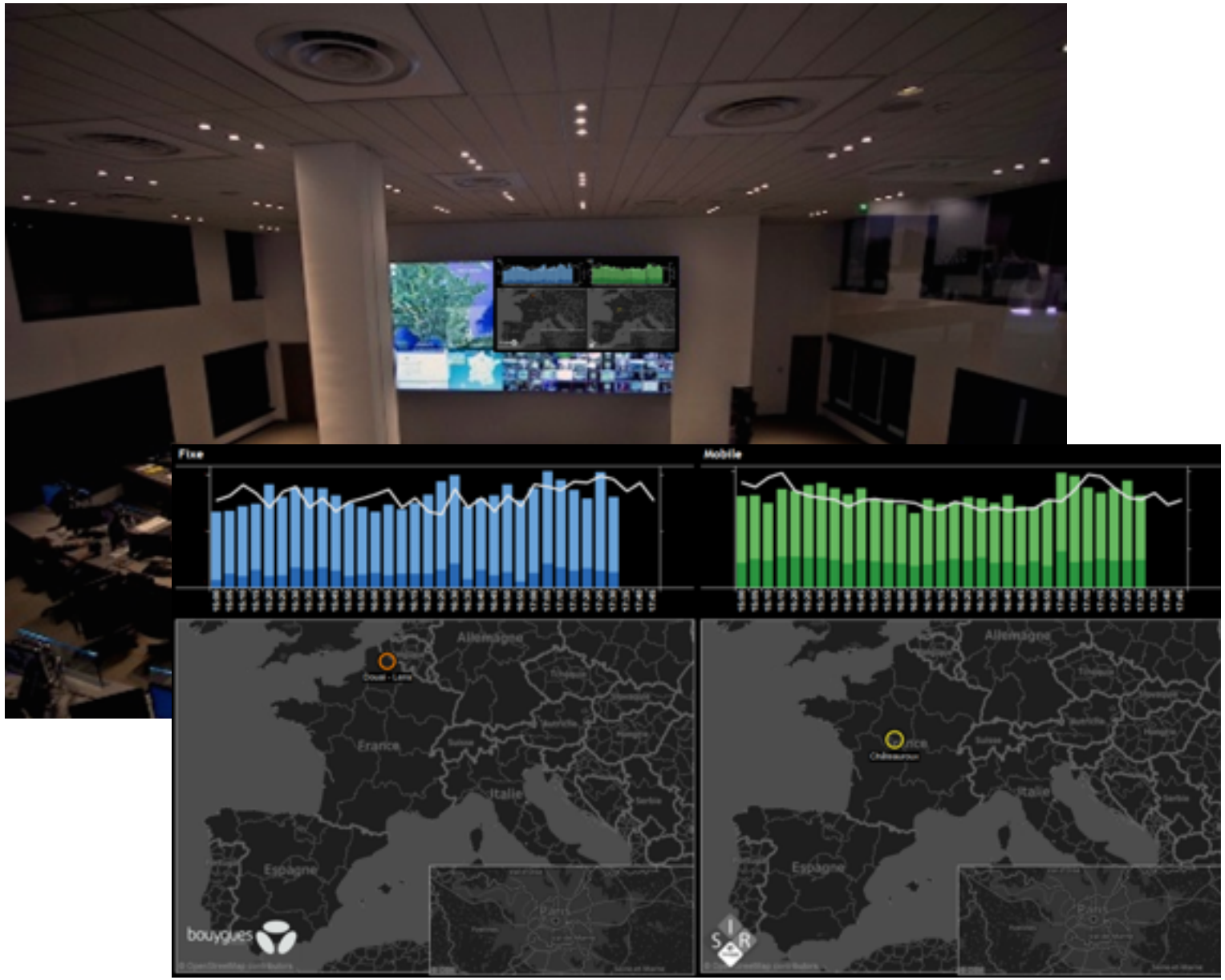
Filter / Transform

State read/write

Sink

Is Flink used?

Powered by Flink



bouygues TELECOM

10 Billion events/day
2Tb of data/day

30 Applications
2Pb of storage and growing

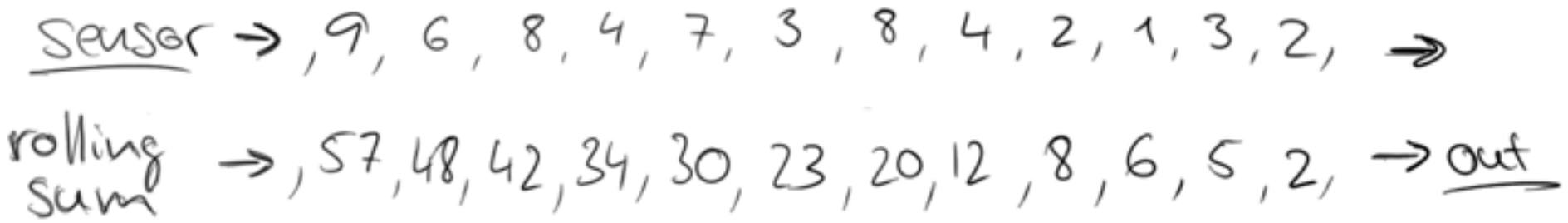
Source Bouyges Telecom http://berlin.flink-forward.org/wp-content/uploads/2016/07/Thomas-Lamirault_Mohamed-Amine-Abdessemed-A-brief-history-of-time-with-Apache-Flink.pdf

Stream Processing

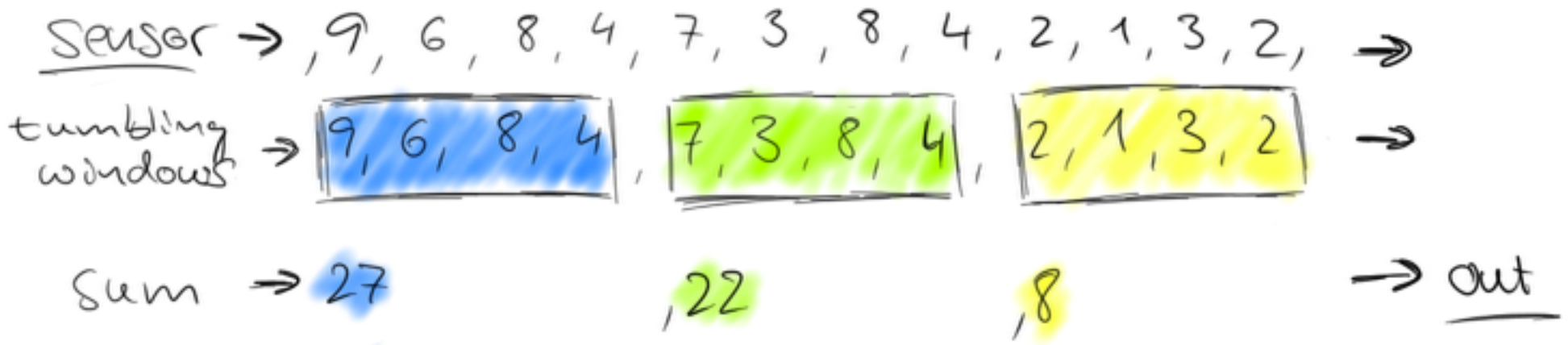
Windowing

Stream Windows

Stream Windows

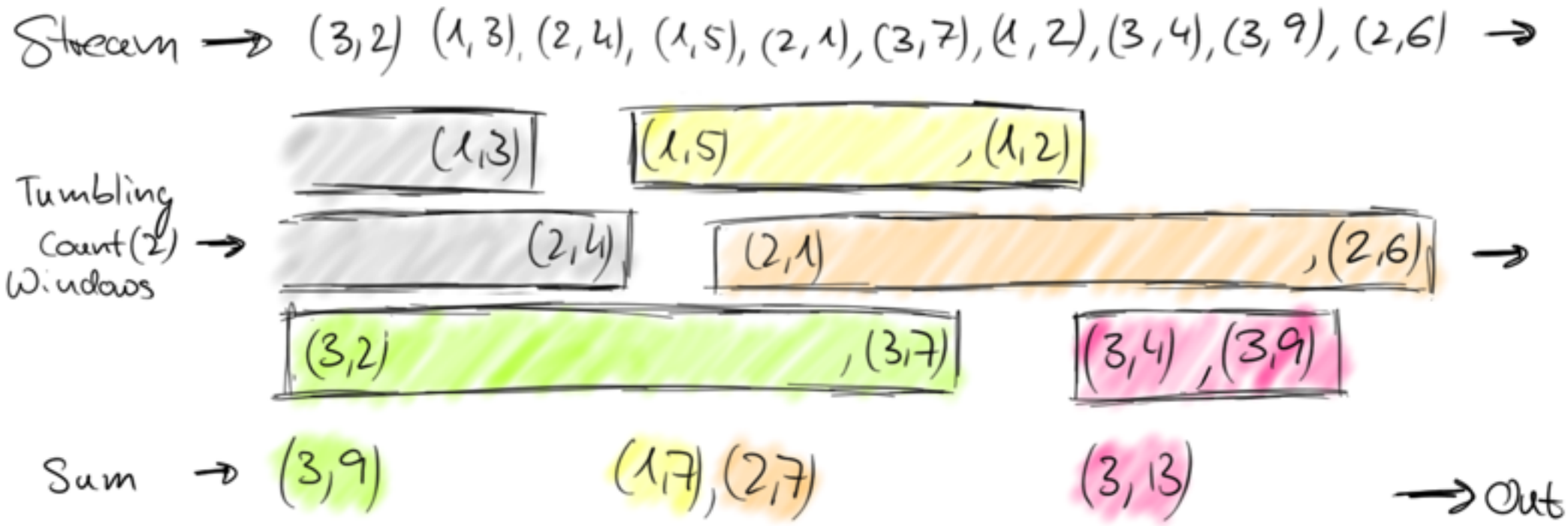


Stream Windows



Stream Windows

Stream Windows



Demonstration

Flink Windowing

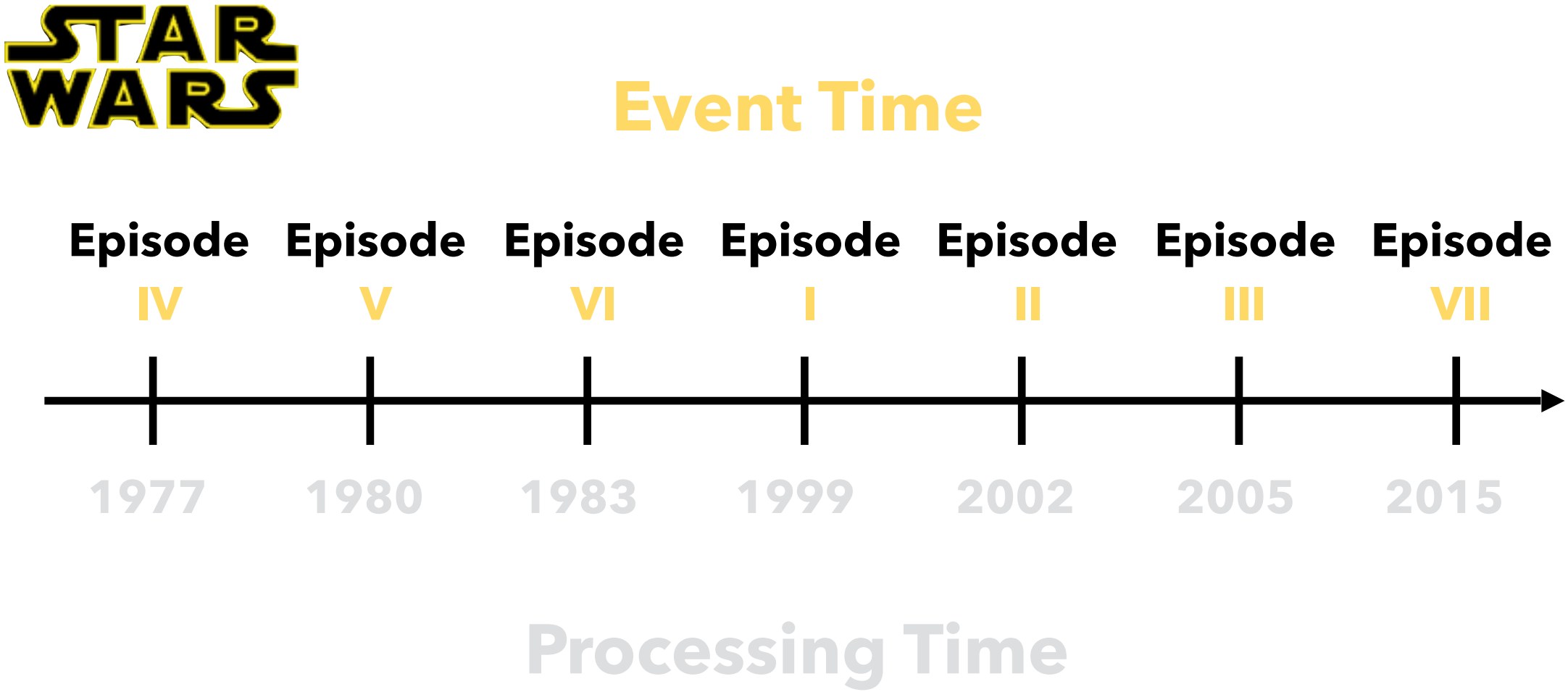
Time

What about it ?

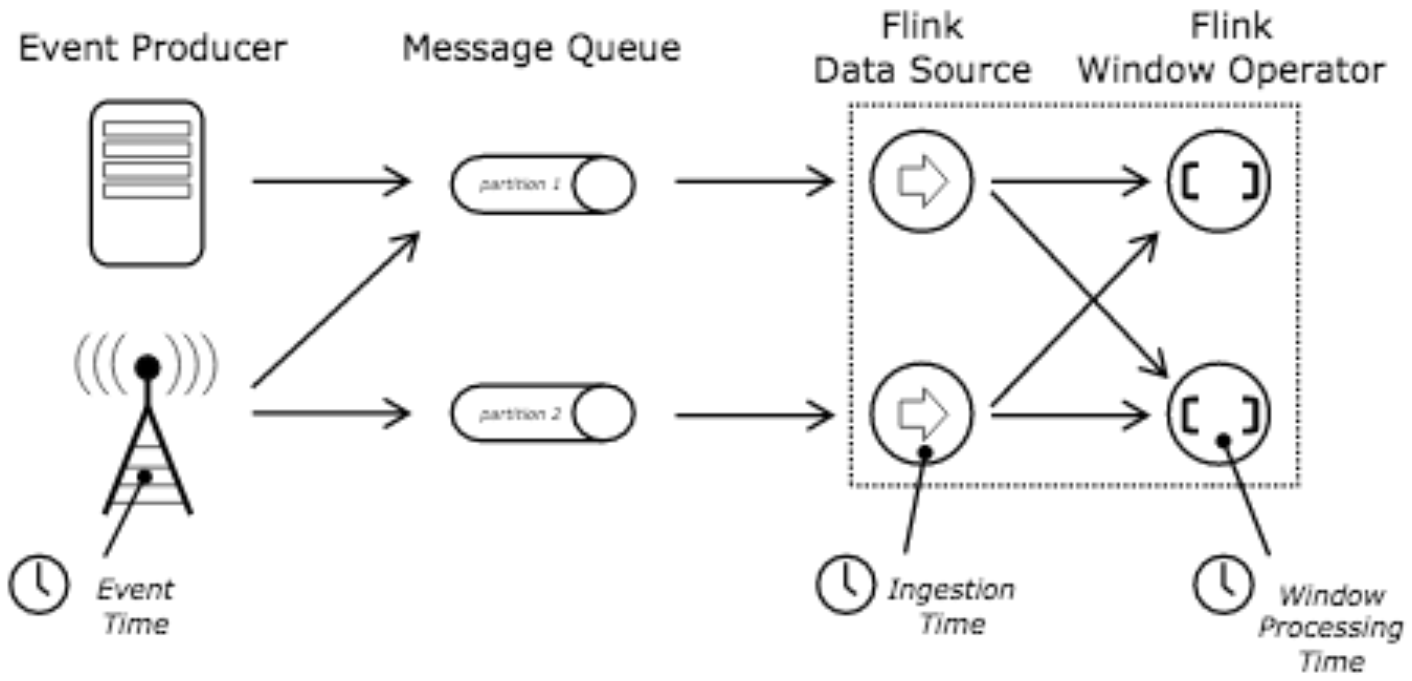
Time in Flink

- Multiple notion of “Time” in Flink
    - Event Time
    - Ingestion Time
    - Processing Time

What Is Event-Time Processing



Time in Flink

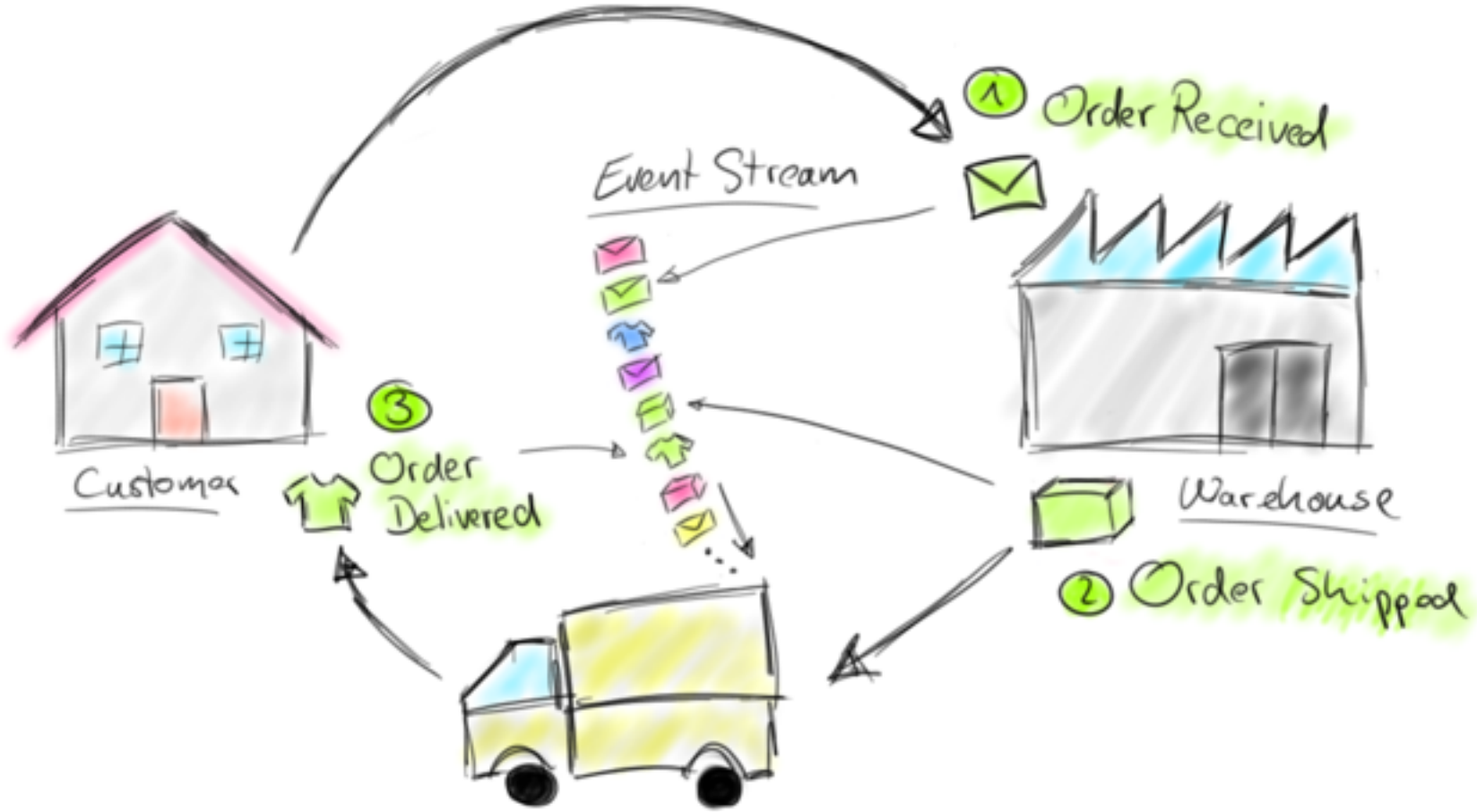


Complex Event Processing

Complex Event Processing

- Analyzing a stream of events and drawing conclusions
  - `“if A and then B → infer event C”`
- Demanding requirements on stream processor
  - Low latency!
  - Exactly-once semantics & event-time support

Use Case



Order Events

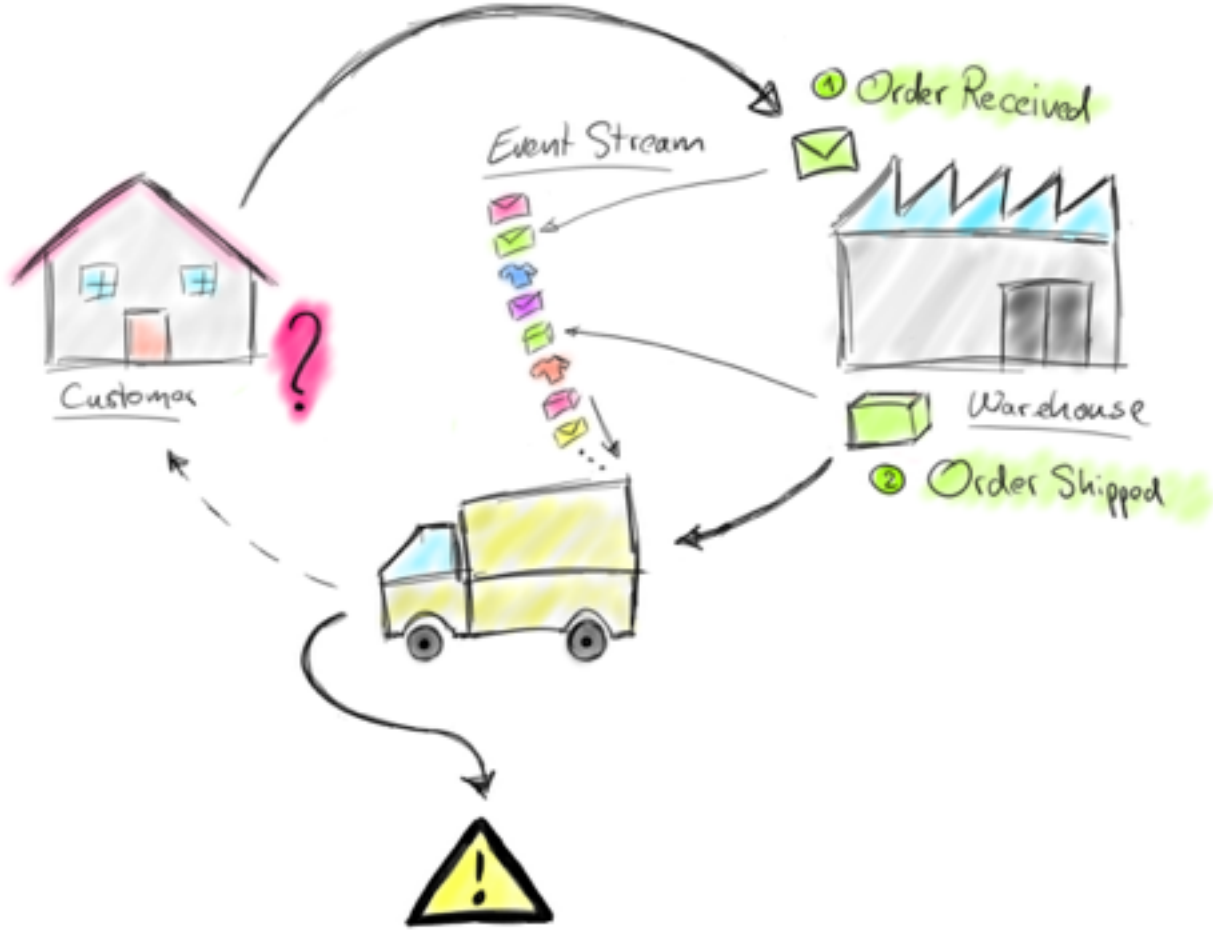
Process is reflected in a stream of order events

```
Order(orderId, tStamp, “received”)
Shipment(orderId, tStamp, “shipped”)
Delivery(orderId, tStamp, “delivered”)
```

`orderId`: Identifies the order

`tStamp`: Time at which the event happened

Real-time Warnings



CEP to the Rescue

Define processing and delivery intervals (SLAs)

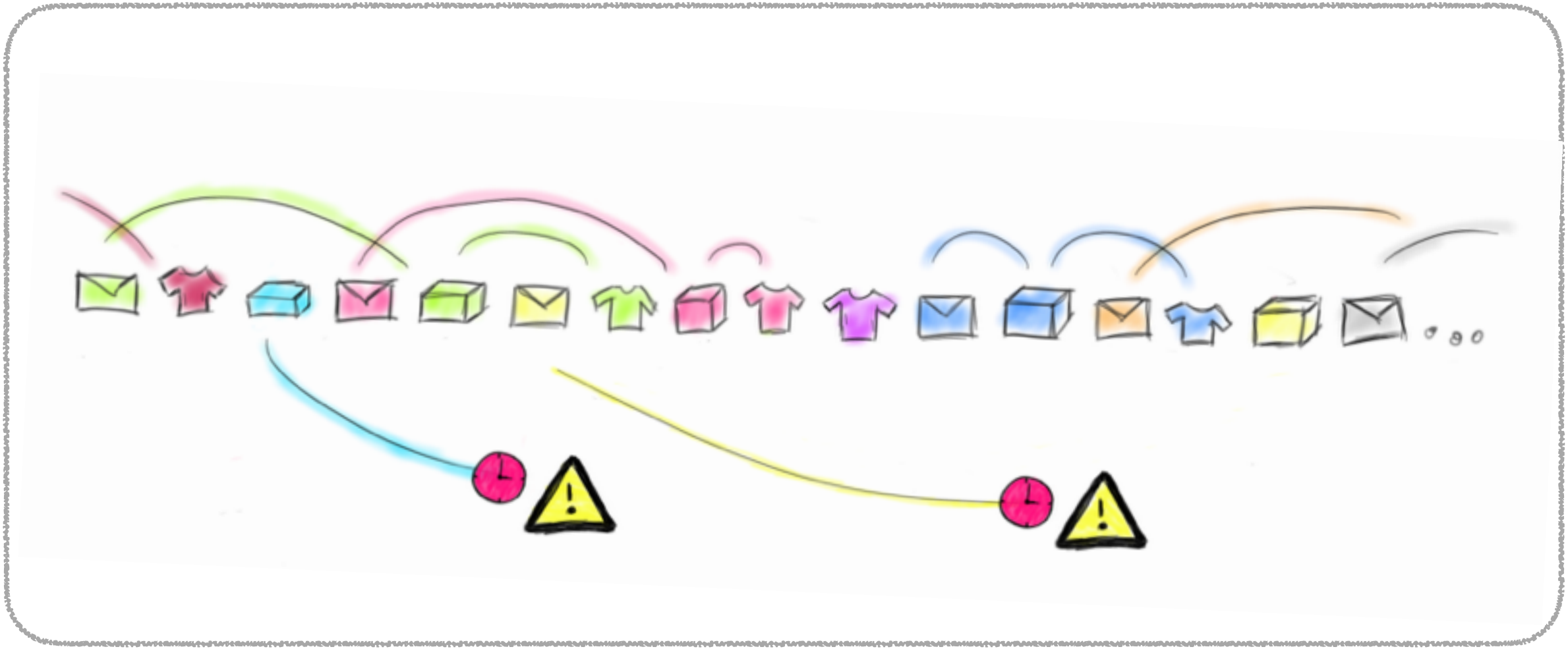
```
ProcessSucc(orderId, tStamp, duration)
ProcessWarn(orderId, tStamp)
DeliverySucc(orderId, tStamp, duration)
DeliveryWarn(orderId, tStamp)
```

`orderId:` Identifies the order

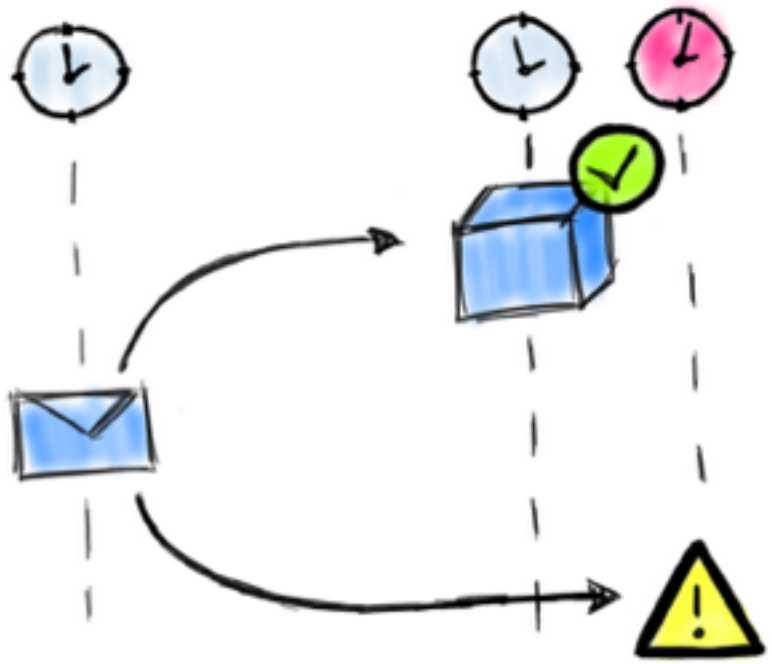
`tStamp:` Time when the event happened

`duration:` Duration of the processing/delivery

CEP Example

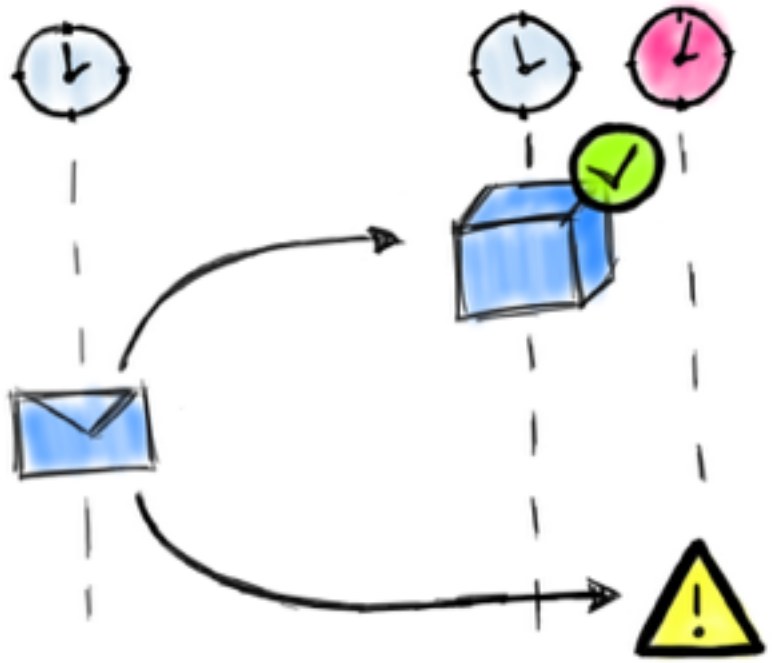


Processing: Order → Shipment



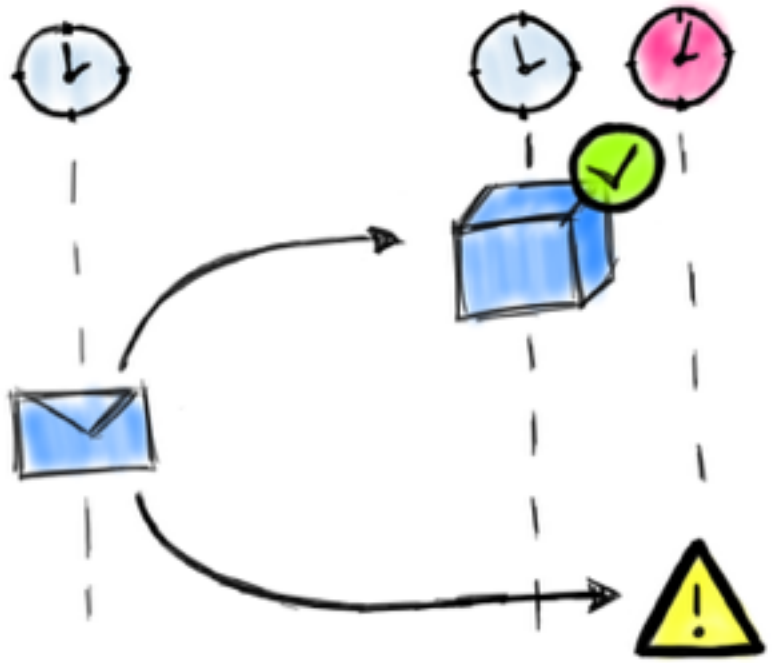
Processing: Order → Shipment

```
val processingPattern = Pattern
  .begin[Event]("received").subtype(classOf[Order])
  .followedBy("shipped").where(_.status == "shipped")
  .within(Time.hours(1))
```



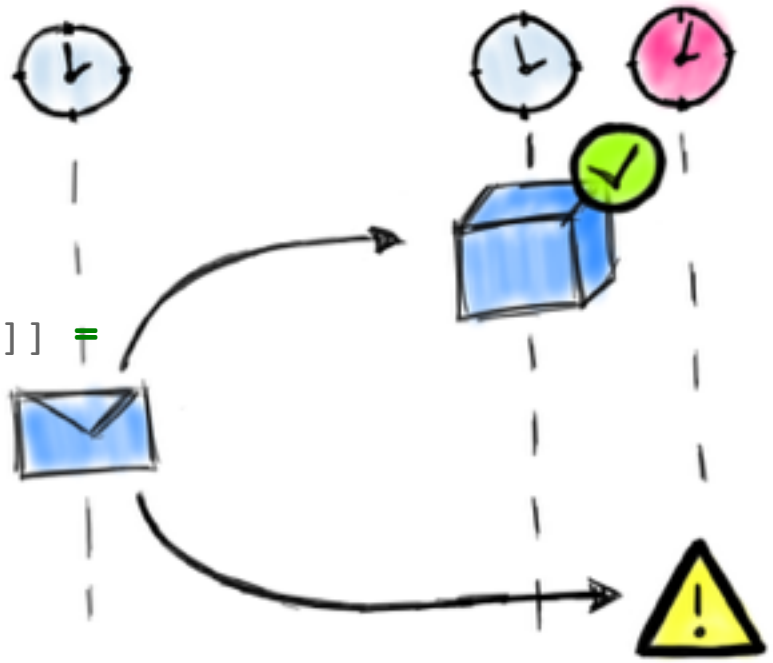
Processing: Order → Shipment

```
val processingPattern = Pattern
  .begin[Event]("received").subtype(classOf[Order])
  .followedBy("shipped").where(_.status == "shipped")
  .within(Time.hours(1))

val processingPatternStream = CEP.pattern(
  input.keyBy("orderId"),
  processingPattern)
```

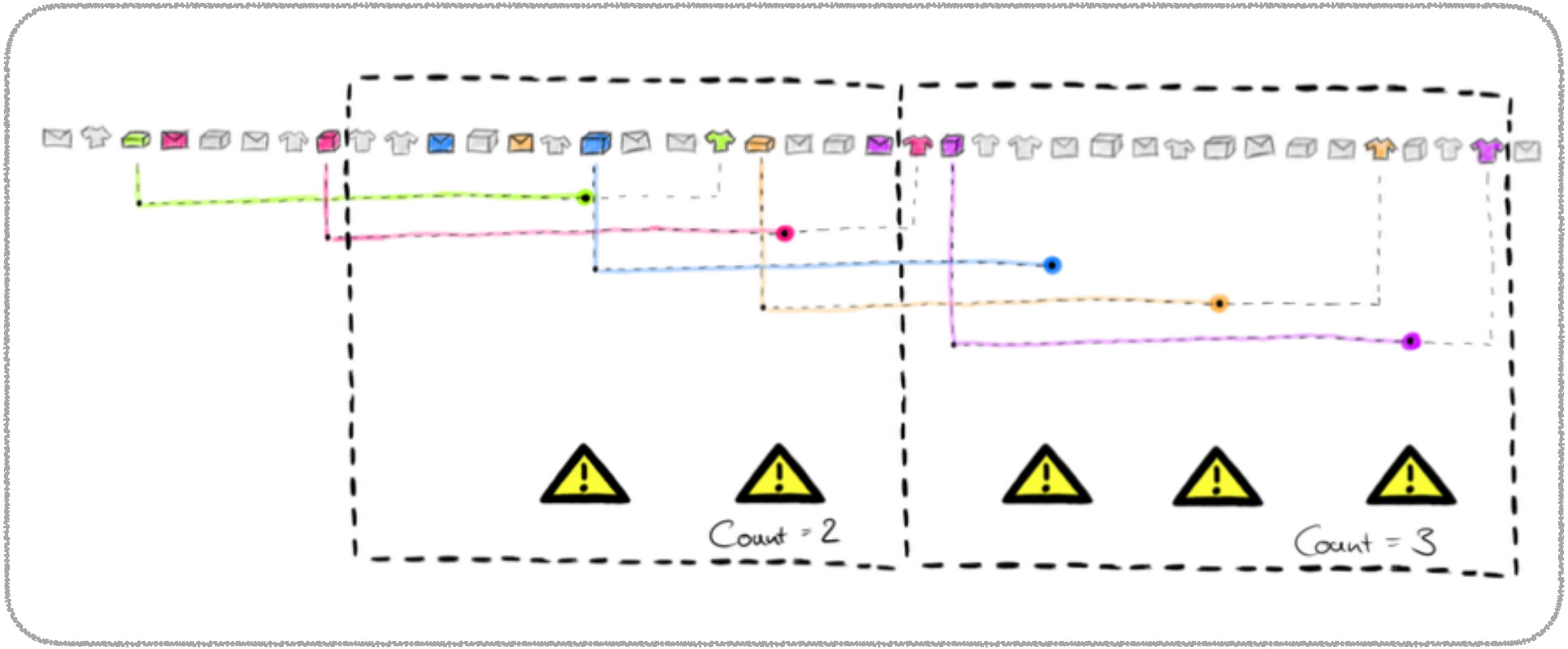


Processing: Order → Shipment

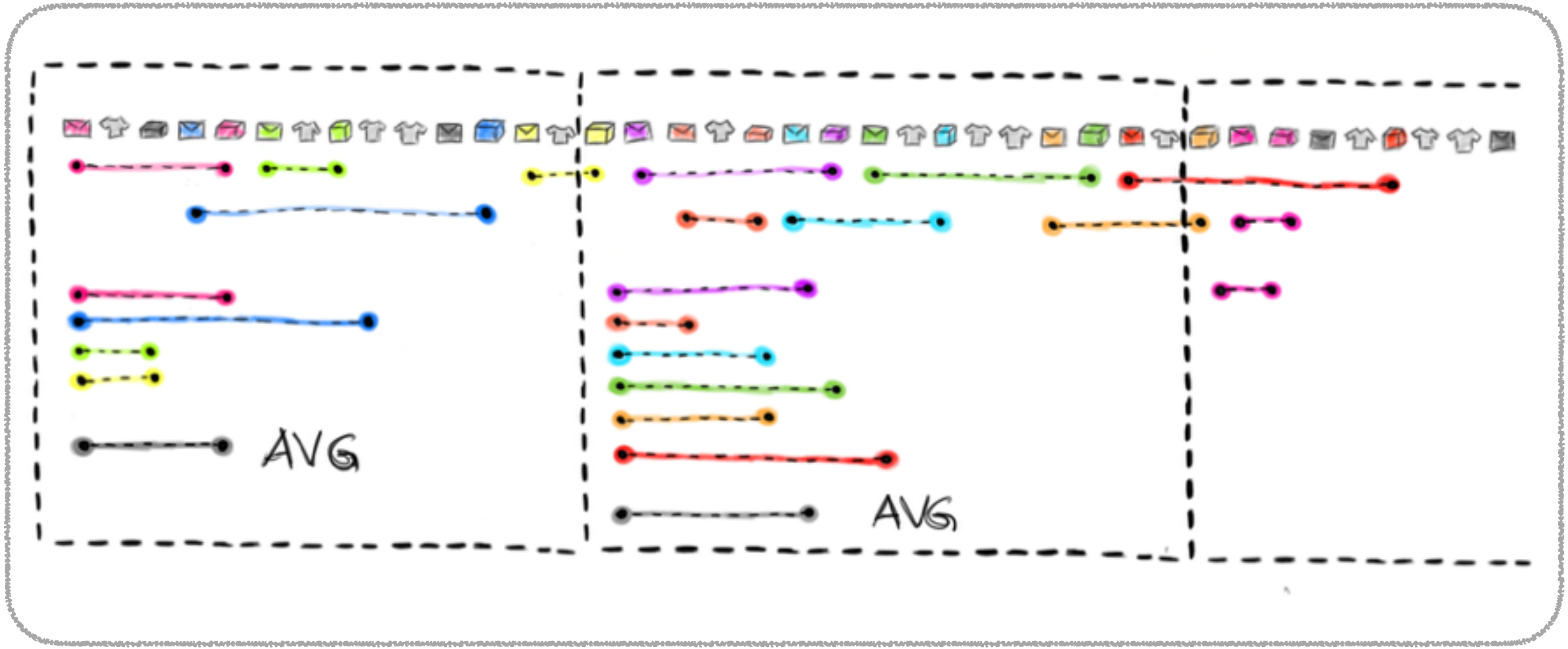
```
val processingPattern = Pattern
  .begin[Event]("received").subtype(classOf[Order])
  .followedBy("shipped").where(_.status == "shipped")
  .within(Time.hours(1))

val processingPatternStream = CEP.pattern(
  input.keyBy("orderId"),
  processingPattern)

val procResult: DataStream[Either[ProcessWarn, ProcessSucc]] =
  processingPatternStream.select {
    (pP, timestamp) => // Timeout handler
      ProcessWarn(pP("received").orderId, timestamp)
  } {
    fP => // Select function
      ProcessSucc(
        fP("received").orderId, fP("shipped").tStamp,
        fP("shipped").tStamp – fP("received").tStamp)
  }
```



Count Delayed Shipments



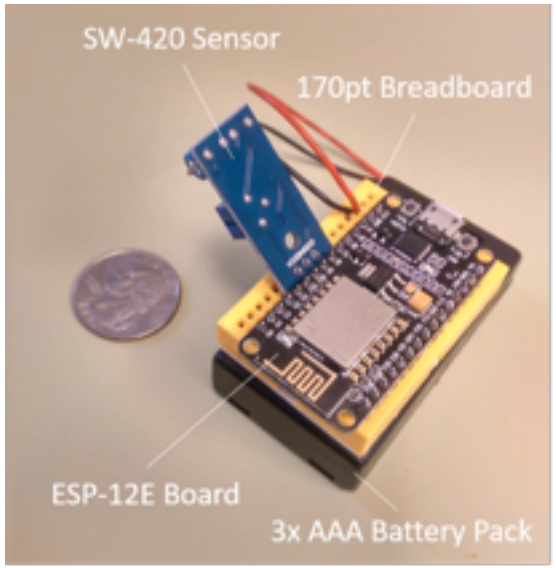
Compute Avg Processing Time



Demonstration

Streaming Analytics

Demonstration

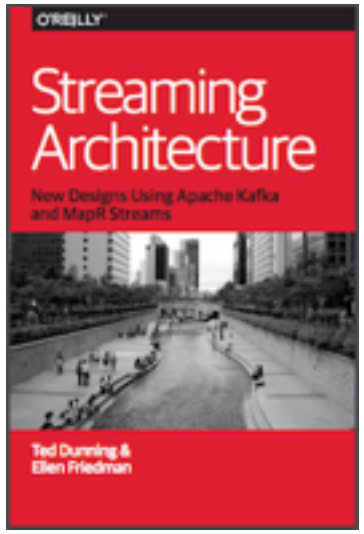


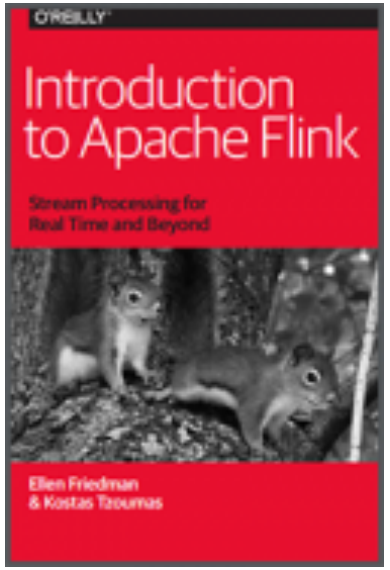
- https://github.com/mapr-demos/mapr-streams-flink-demo
- https://github.com/mapr-demos/wifi-sensor-demo
- http://tgrall.github.io/blog/2016/10/12/getting-started-with-apache-flink-and-kafka/
- http://tgrall.github.io/blog/2016/10/17/getting-started-with-apache-flink-and-mapr-streams/
- more soon….

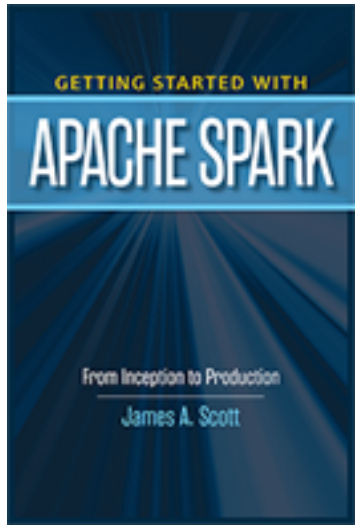
Thanks to

Kostas Tzoumas
Stephan Ewen
Fabian Hueske
Till Rohrmann
Jamie Grier

Streaming Architecture







Free ebooks & Online training

http://mapr.com/ebooks/

http://mapr.com/training/

Stream Processing with Apache Flink

Tugdual Grall
@tgrall