Distributed systems for stream processing

Apache Kafka and Spark Structured Streaming

Alena Hall

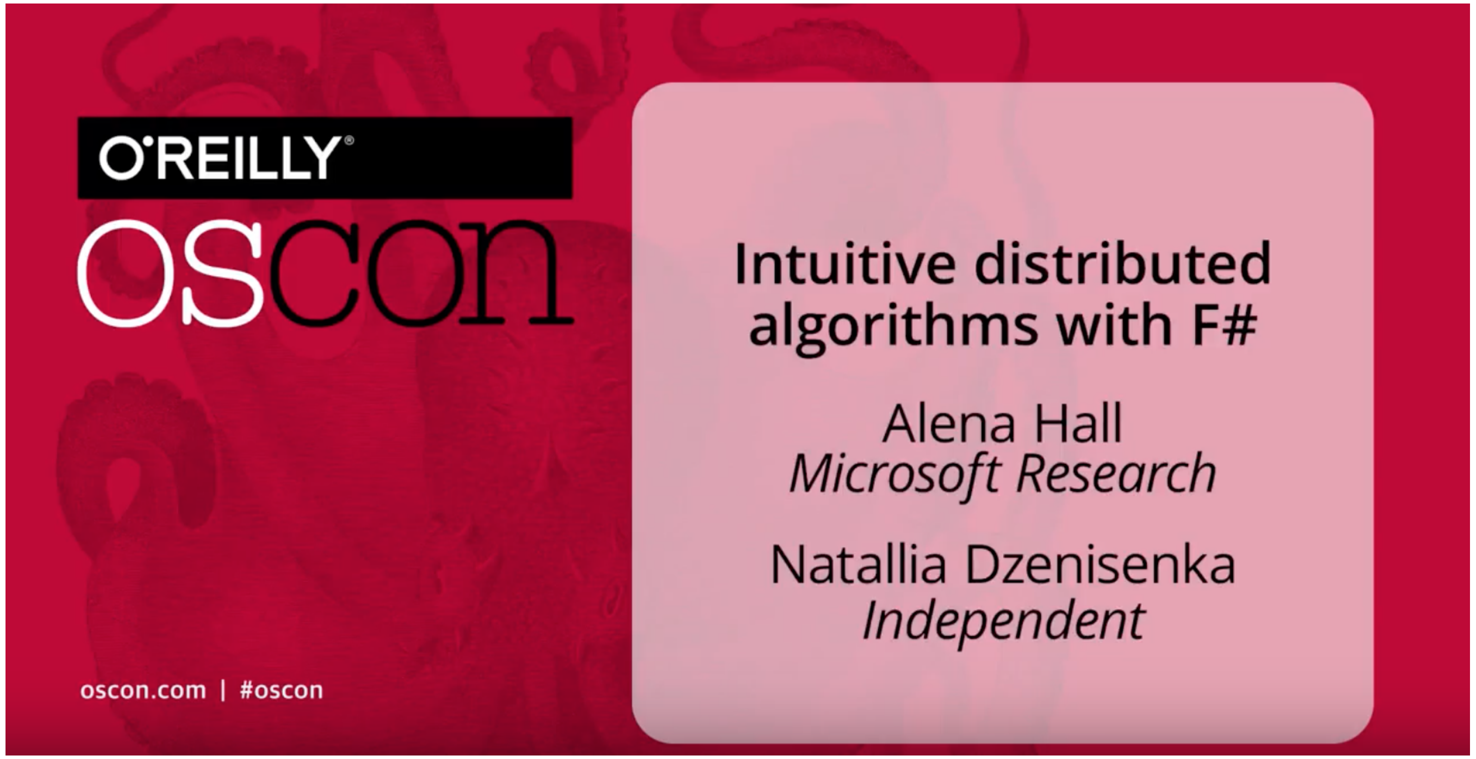
lenadroid

OSCON 20 YEARS 2018

Alena Hall - lenadroid

- ✓ Large-scale data processing
- ✓ Distributed Systems
- ✓ Functional Programming
- ✓ Data Science & Machine Learning

Natallia Dzenisenka



O'REILLY

oscon

Intuitive distributed algorithms with F#

Alena Hall
Microsoft Research

Natallia Dzenisenka
Independent

oscon.com | #oscon

nata_dzen

bit.ly/oscon-17

Data

Ever-increasing

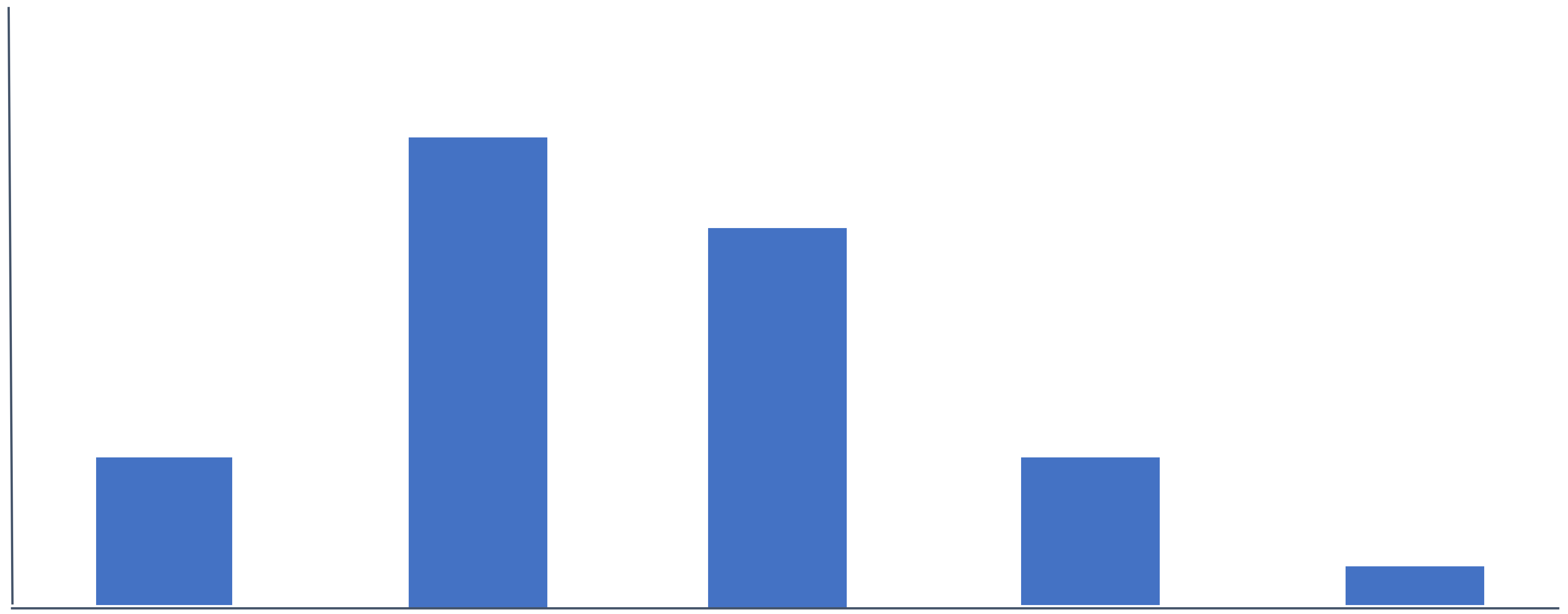
lenadroid

Direct result of some action

Produced as a **side effect**

lenadroid

Continuous **indicators**



lenadroid

Reaction

urgent **not-so-urgent** **flexible**

lenadroid

Reaction

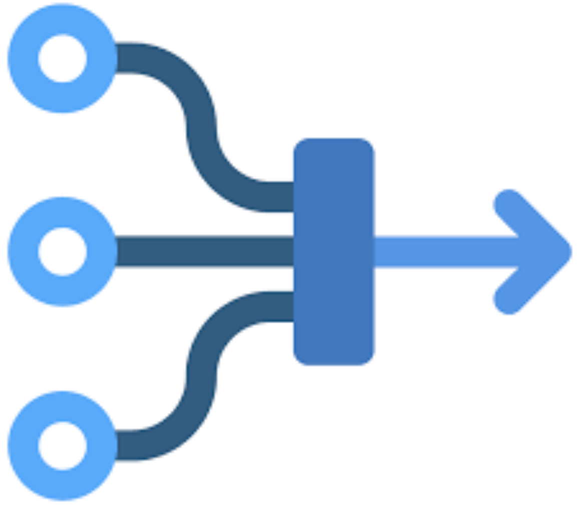
urgent **not-so-urgent** **flexible**

real-time
~ sub milliseconds

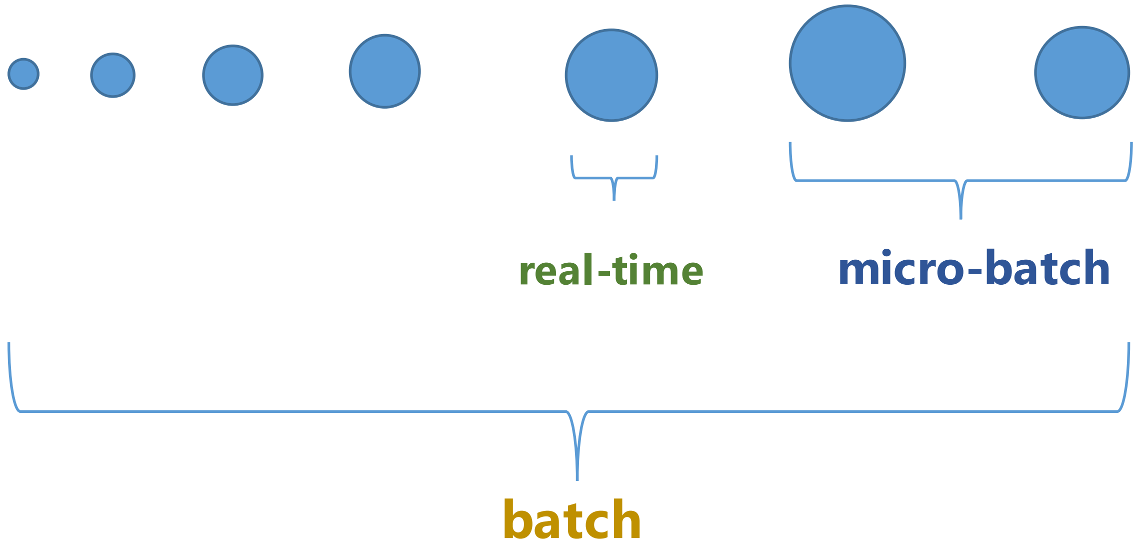
near-real-time
~ seconds

batch
~minutes, hours, days, weeks

Event Ingestion



Processing & Reaction



lenadroid

Data Producers and Consumers

Are data workflows flexible enough?

lenadroid

Challenges

Simplicity. Scalability. Reliability

Meet Apache Kafka

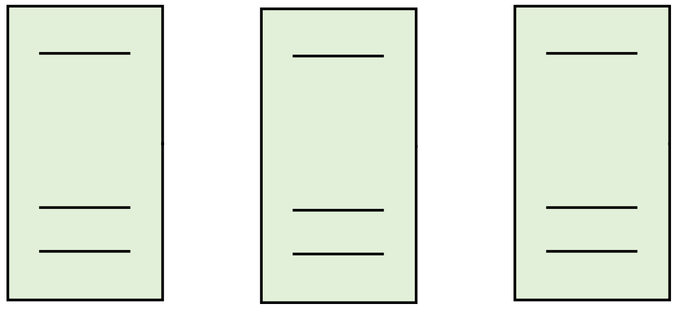
lenadroid

Apache Kafka is an open-source stream-processing software platform developed by the Apache Software Foundation written in Scala and Java.

lenadroid

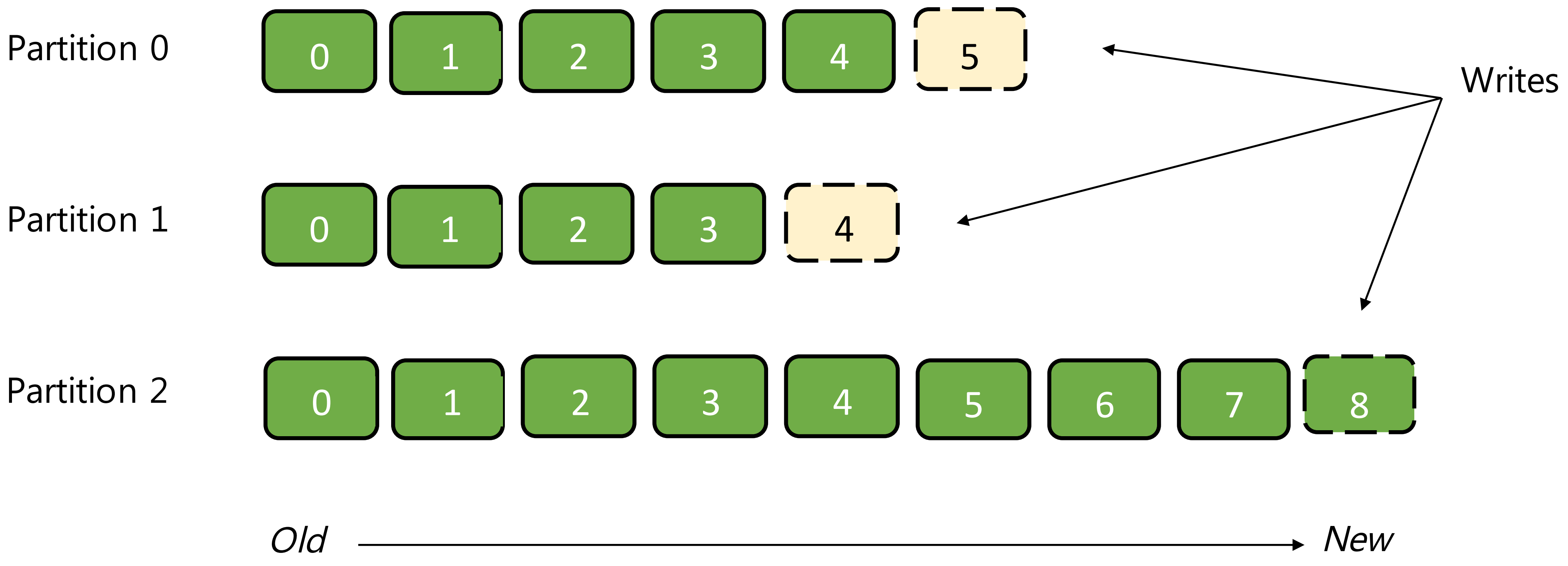
Kafka Brokers

Zookeeper Servers



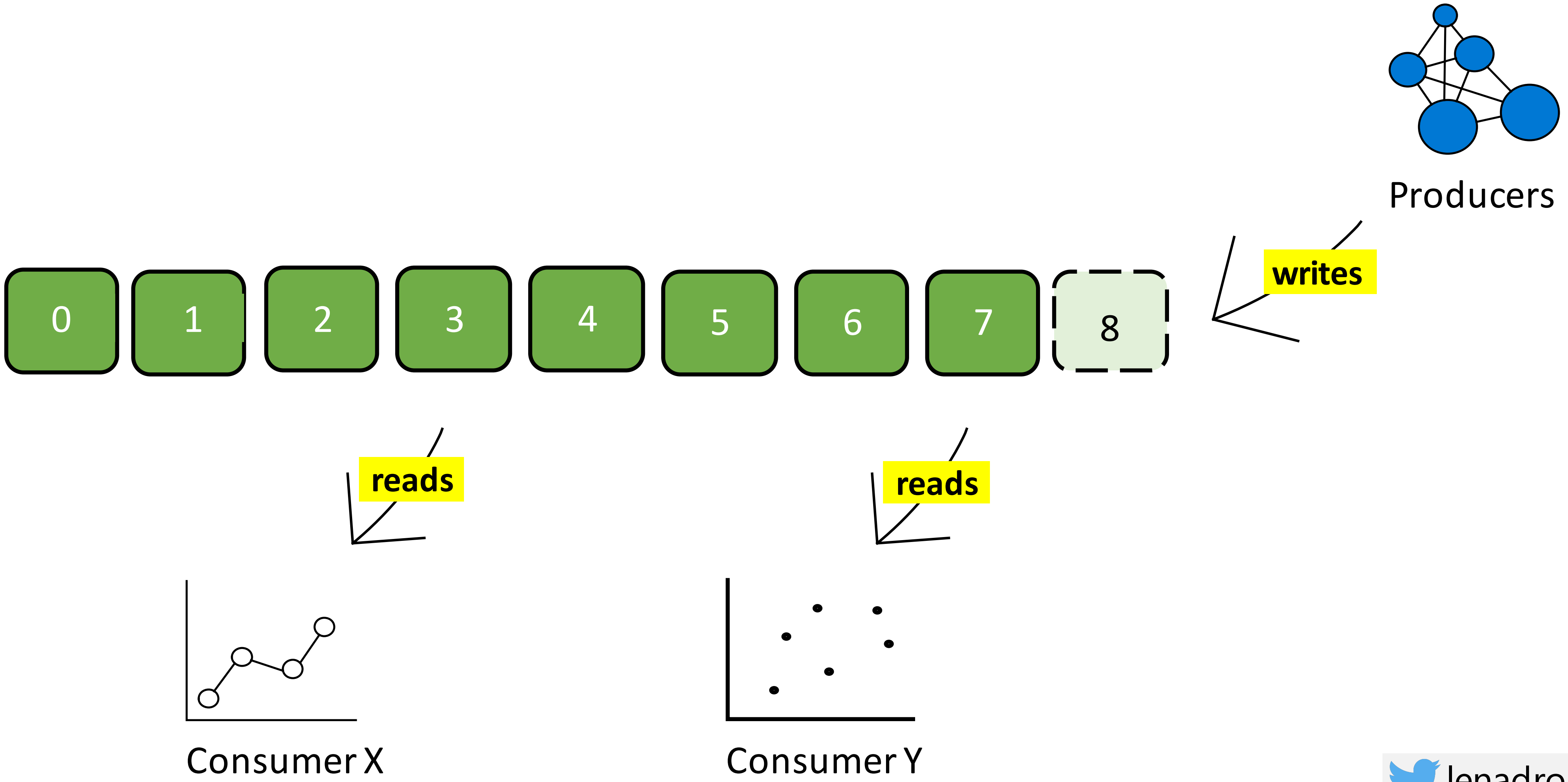
lenadroid

Inside of a Kafka Topic



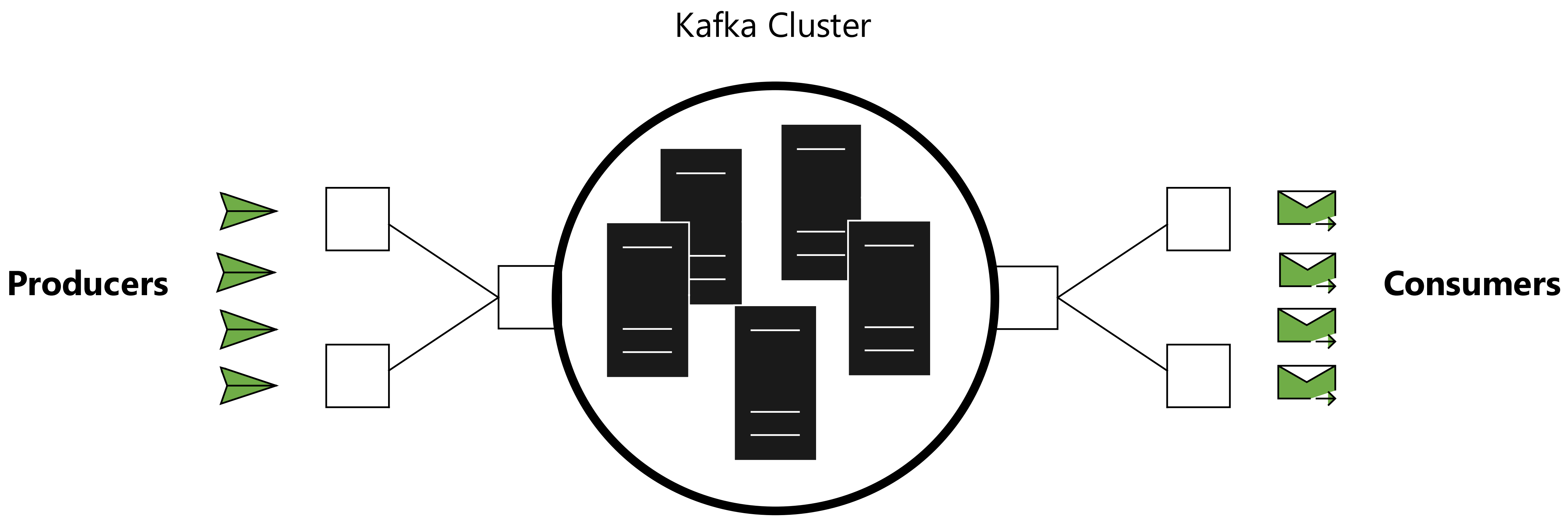
lenadroid

Kafka Topic Partition



lenadroid

Kafka Producers and Consumers



lenadroid

Systems for stream processing

Kafka Streams

Spark

Storm

Flink

lenadroid

Meet Apache Spark

lenadroid

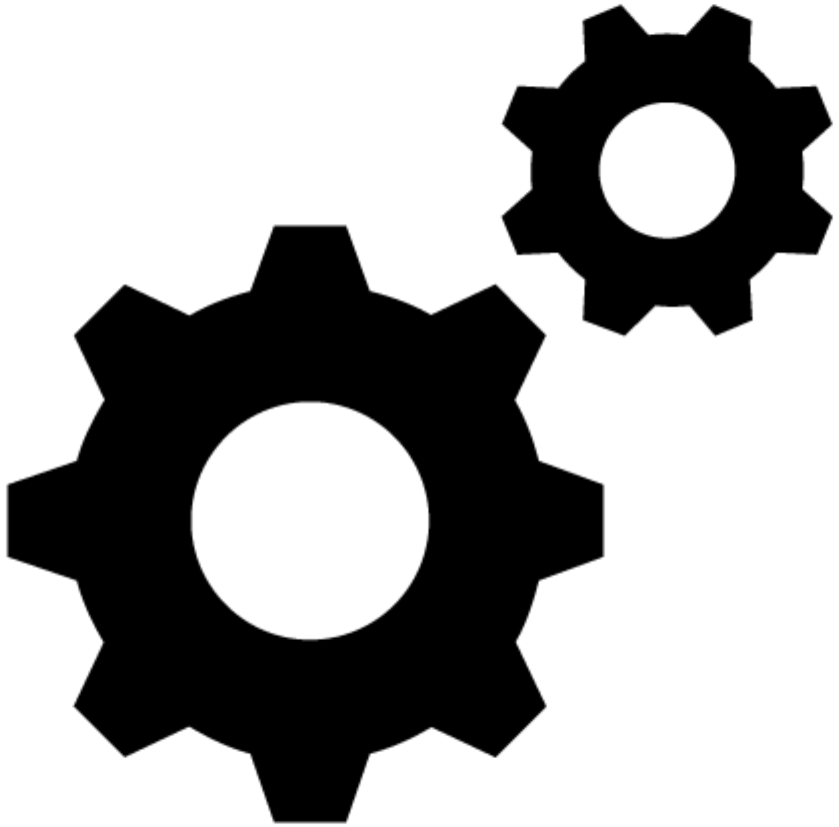
Apache Spark is a unified analytics engine for large-scale data processing: batch, streaming, machine learning, graph computation with access to data in hundreds of sources.

lenadroid

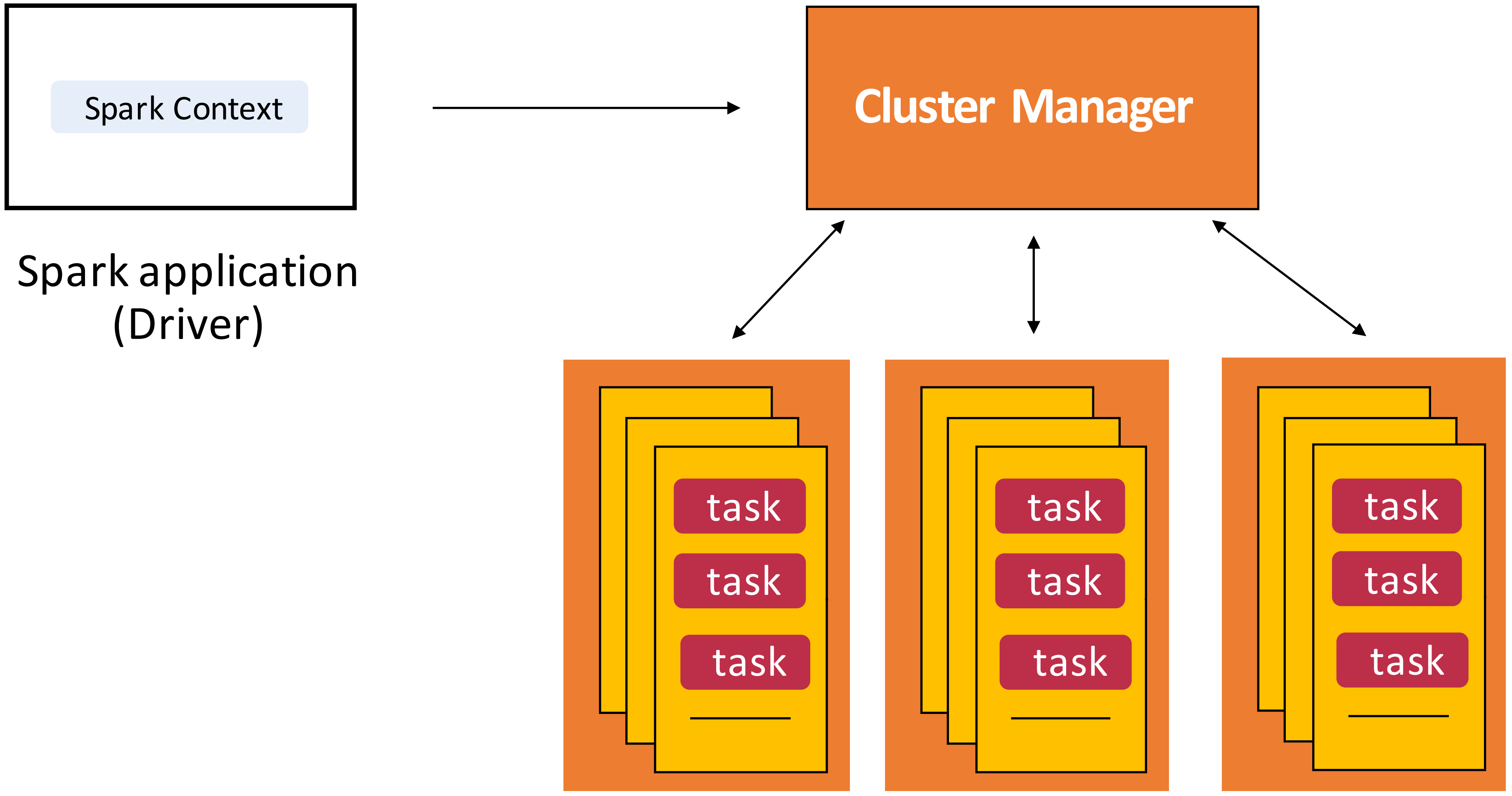
- ✓ Spark SQL and batch processing
- ✓ Stream processing with Spark Streaming and Structured Streaming
- ✓ * Continuous processing
- ✓ Machine Learning with Mllib
- ✓ Graph computations with GraphX

* Experimental

How does Spark work?



lenadroid



Spark workers have executors of tasks

lenadroid

Apache Kafka + Apache Spark

lenadroid

Existing infrastructure and resources

- ✓ Kafka cluster (HDInsight or other)
- ✓ Spark cluster (Azure Databricks workspace, or other)
- ✓ Peered Kafka and Spark Virtual Networks
- ✓ Sources of data: Twitter & Slack & Nomics APIs

Databricks: Interactive Environment

Processing crypto currency trading data

Example

lenadroid

markets

ETH / BTC

BTC / USDT

...

base

quote

exchanges

Bitfinex

Binance

...

trades

{ ... }

lenadroid

markets **exchanges** **trades**

```
{
    "volume":"5",
    "price":"3.0871",
    "id":"123456",
    "timestamp":"2018-07-17T17:00:00.00Z"
}
```

Indicators to watch and act on

- ✓ Price spikes (all-time high, all-time low)
- ✓ Significant changes in price or volume of trades
- ✓ Profitability of potential trade at current moment
- ✓ Price or volume of trades crossing given threshold during the past X minutes
- ✓ More

lenadroid

Getting trades data from API

- ✓ Market and exchange data
- ✓ Trades data for given market and base/quote currencies
- ✓ Sending data to Kafka

Processing trades

- ✓ Consuming data coming from Kafka topics
- ✓ Watching relevant indicators

More examples?

Processing streams of events from multiple sources with Apache Kafka and Spark

lenadroid

Data sources: external, internal, ...

- Big number of data sources
- Most of the data sources are independent
- Sources of data used for many processing tasks & end-goals

Feedback from Slack

- ✓ Sending messages to Slack

Listener for new Slack messages

- ✓ Messages under specific channels
- ✓ Focused on a particular topic
- ✓ Sent to a specific Kafka topic

Receiving events in Kafka topic

- ✓ Spark consumer for Kafka topics
- ✓ Sending only topic related messages to Kafka

Sending Twitter feedback to Kafka

- ✓ Getting latest tweets about specific topic to Kafka
- ✓ Receiving those events from Kafka in Spark

Analyzing feedback in real-time

- ✓ Kafka is receiving events from many sources
- ✓ Sentiment analysis on incoming Kafka events
- ✓ Sentiment <= 0.3 → **#negative-feedback** for review
- ✓ Sentiment >= 0.9 → **#positive-feedback** channel

Kafka + Spark =

Reliable, scalable, durable event ingestion and efficient stream processing

lenadroid

Bonus Topics

Continuous Processing

```
trigger(Trigger.Continuous("1 second"))
```

Low (~1 ms) end-to-end latency

At-least-once fault-tolerance guarantees

Not nearly all operations are supported yet

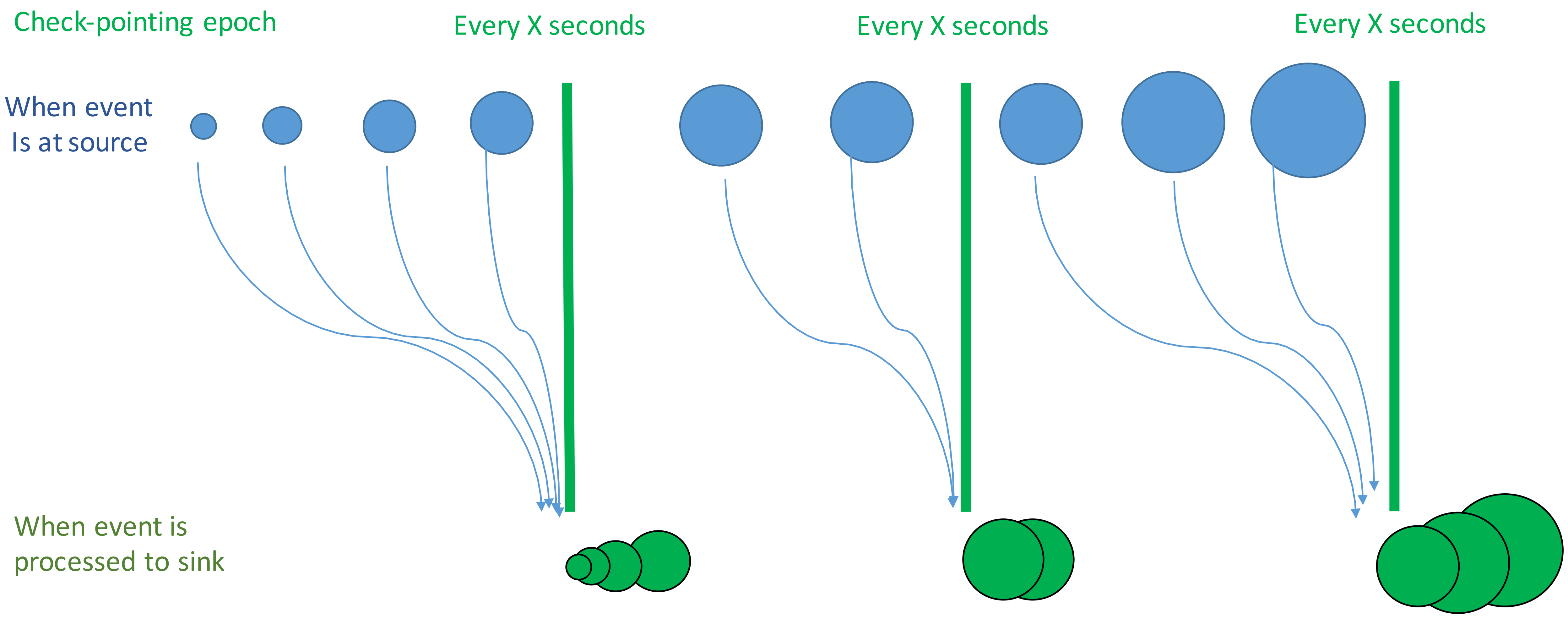
No automatic retries of failed tasks

Needs enough cluster power to operate

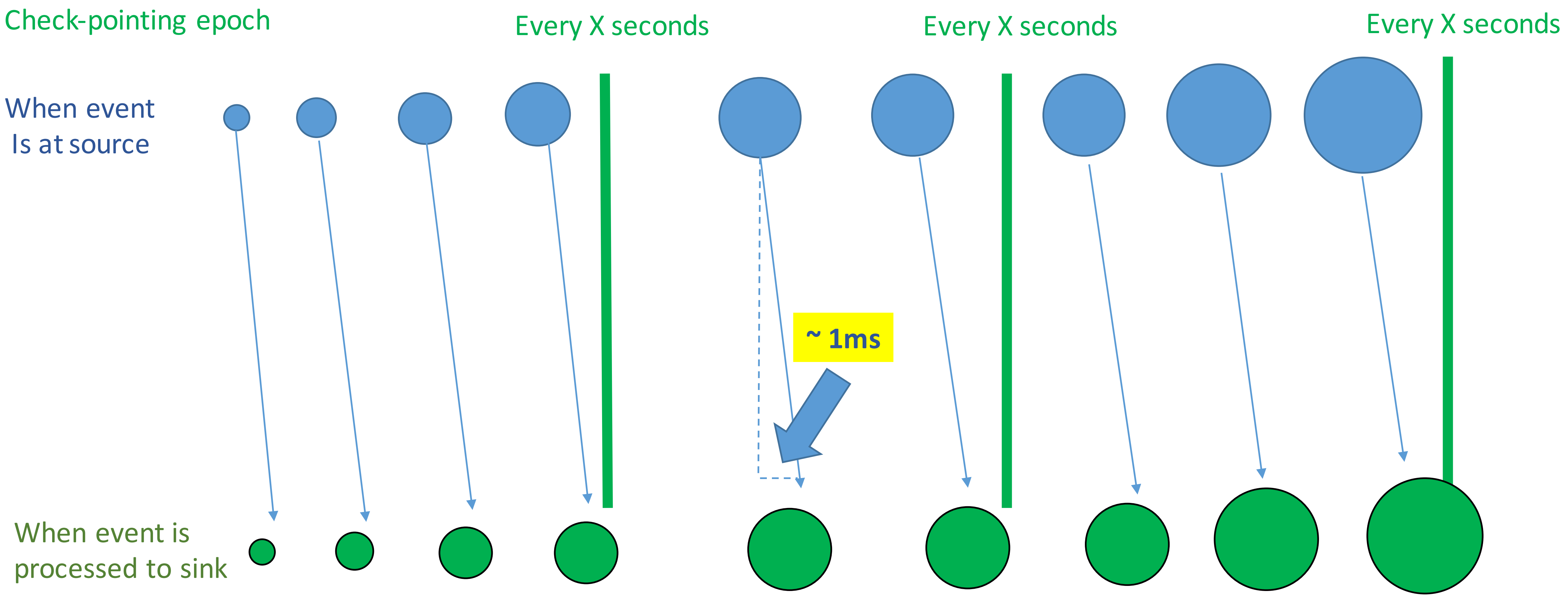
lenadroid

lenadroid

Micro-batch



Continuous


Check-pointing epoch
Every X seconds
Every X seconds
Every X seconds
When event Is at source
~ 1ms
When event is processed to sink

Kafka API for Event Hubs

aka.ms/eventhubs-kafka

lenadroid

Confluent/Kafka **Operator**

and other **Operators** …

lenadroid

Thank you!

Apache Kafka: aka.ms/apache-kafka

Apache Spark: aka.ms/apache-spark

Event stream processing architecture on Azure with Apache Kafka and Spark: aka.ms/kafka-spark-azure and aka.ms/oscon-18

Create HDInsight Kafka cluster using ARM: aka.ms/hdi-kafka-arm

Create Kafka topics in HDInsight: aka.ms/hdi-kafka-topic

lenadroid

Alena Hall - lenadroid

- ✓ Works on Azure at Microsoft
- ✓ Lives in Seattle
- ✓ F# Software Foundation Board of Trustees
- ✓ Organizes @ML4ALL
- ✓ Program Committee for Lambda World
- ✓ Has a channel: YouTube /c/AlenaHall