

Distributed systems for stream processing

Apache Kafka and Spark Structured Streaming

Alena Hall - @lenadroid

Alena Hall - **lenadroid**



- ✓ High Scale & Big Data
- ✓ Distributed Systems
- ✓ Functional Programming
- ✓ Data Science & Machine Learning

Data

Ever-increasing

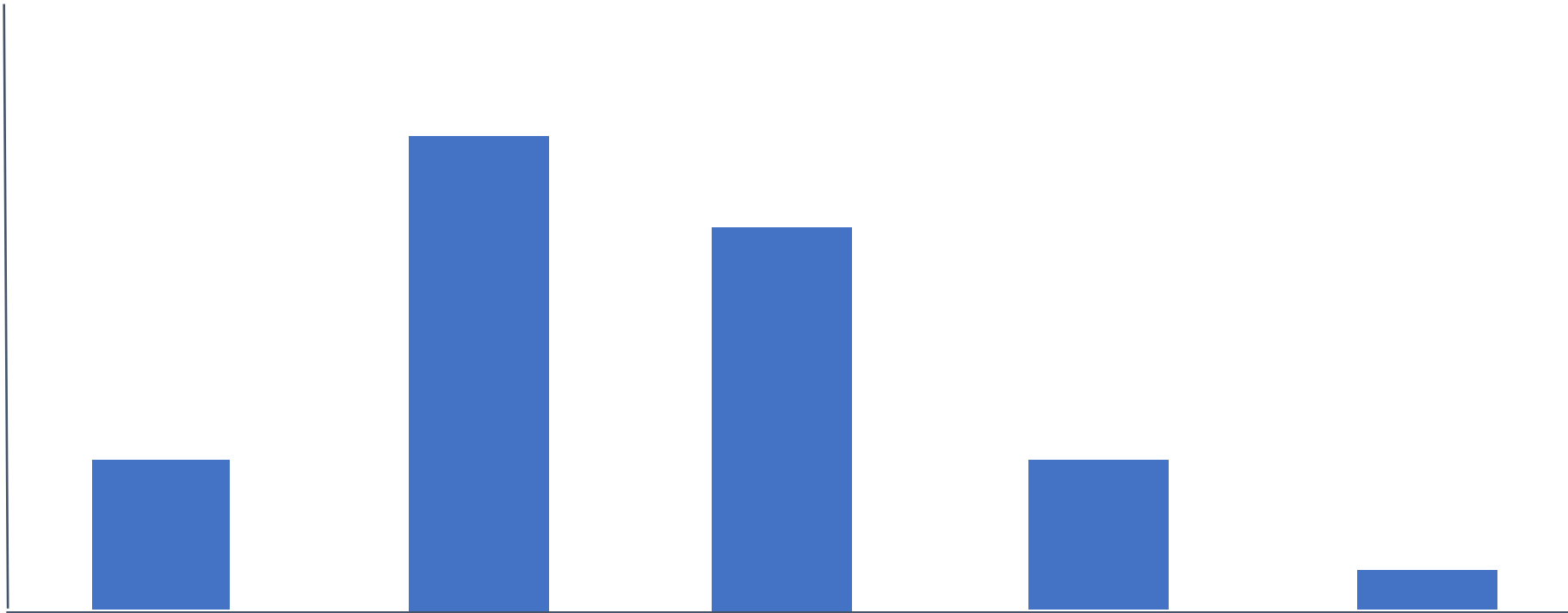
Direct result of some action



Produced as a **side effect**



Continuous **metrics**



Reaction

urgent

not-so-urgent

flexible

Reaction

urgent

not-so-urgent

flexible



real-time

~ sub milliseconds

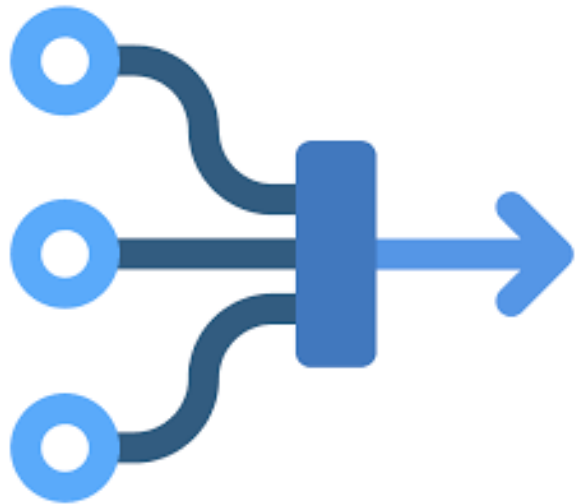
near-real-time

~ seconds

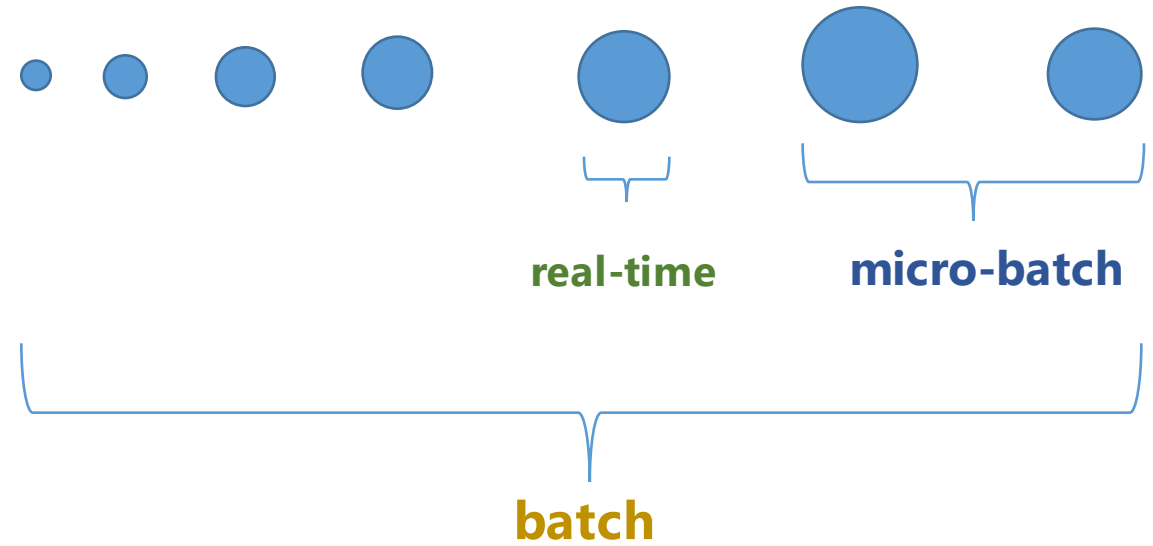
batch

~ minutes, hours, days, weeks

Event Ingestion

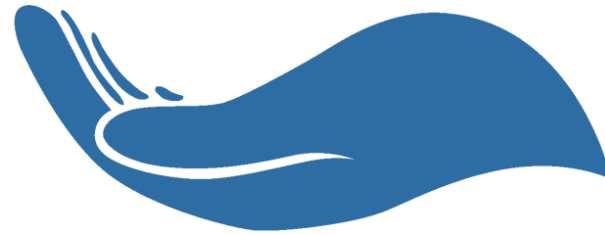


Processing & Reaction



Data Producers and Consumers

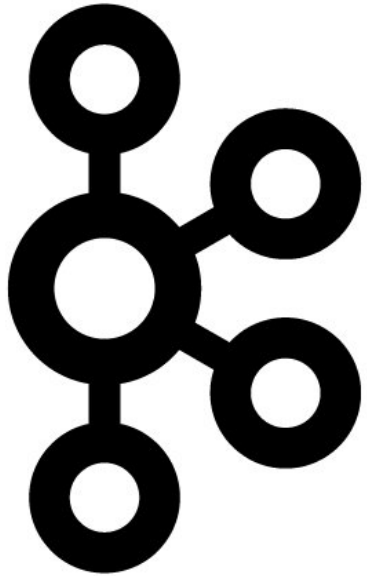
Are data workflows flexible enough?



Challenges

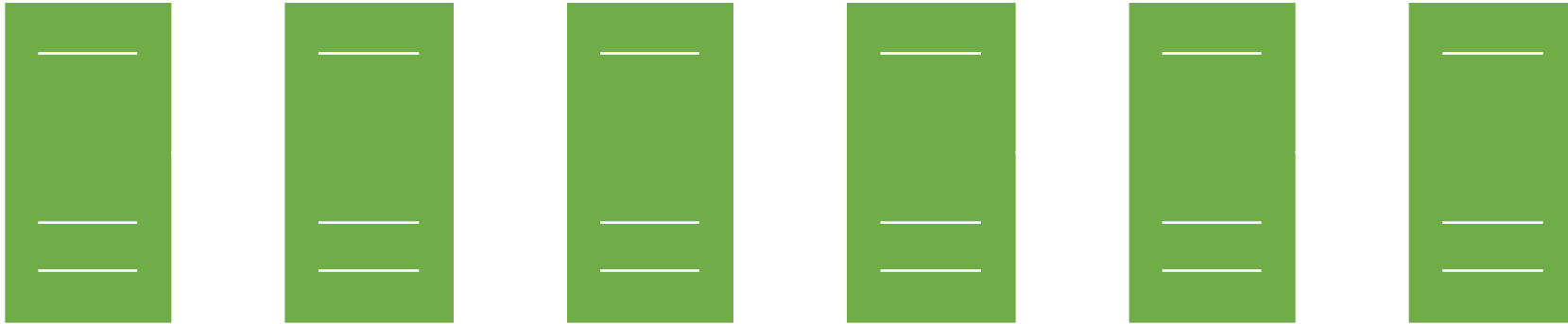
Simplicity. Scalability. Reliability

Meet Apache Kafka

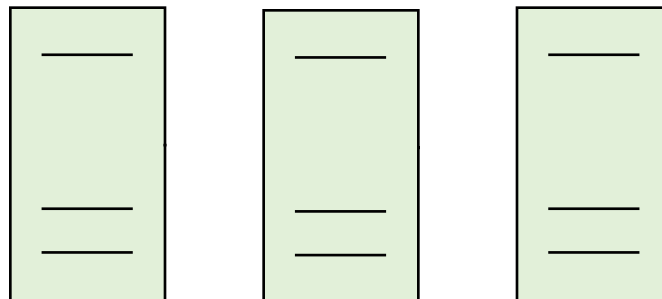


Apache Kafka is an open-source stream-processing software platform developed by the Apache Software Foundation written in Scala and Java.

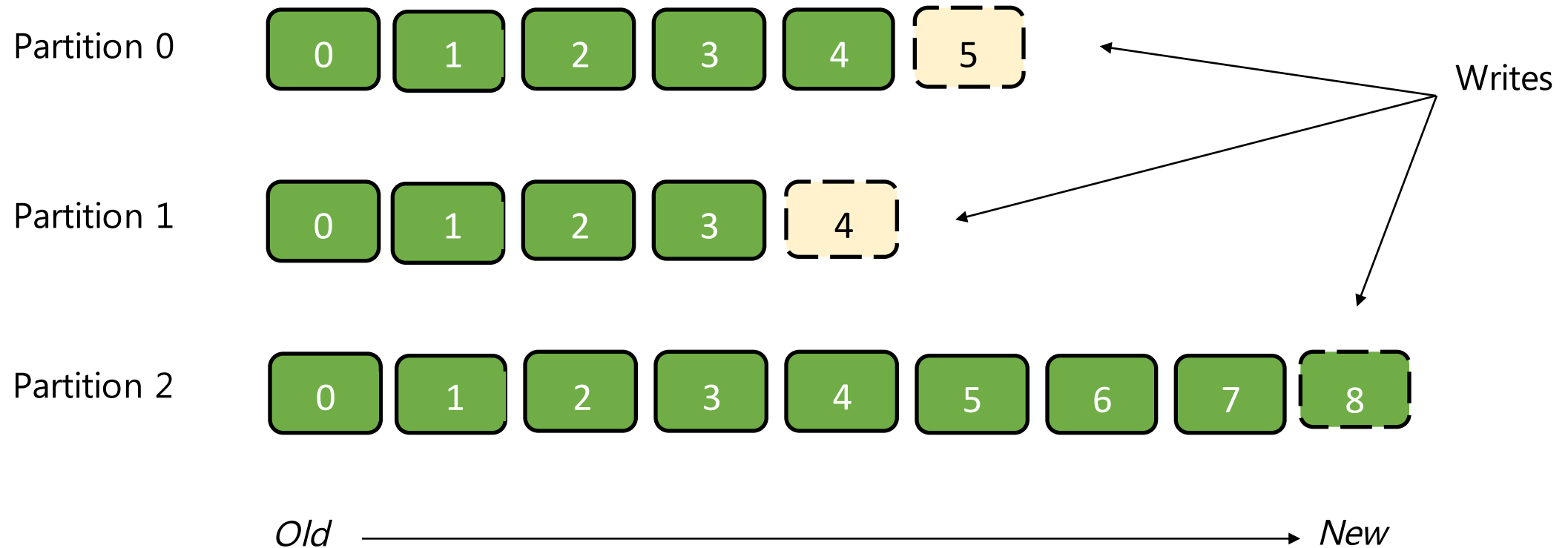
Kafka Brokers



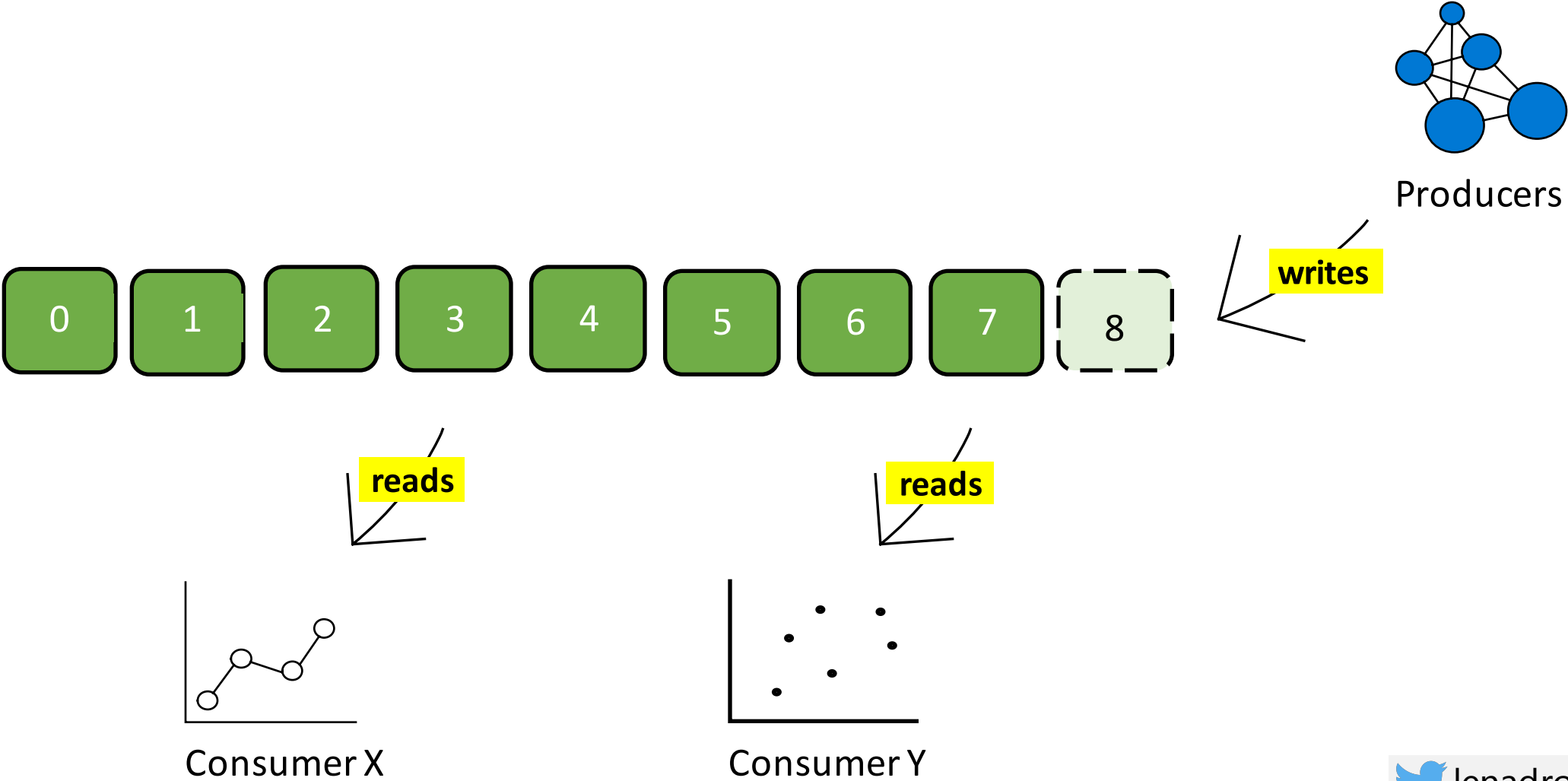
Zookeeper Servers



Inside of a Kafka Topic



Kafka Topic Partition



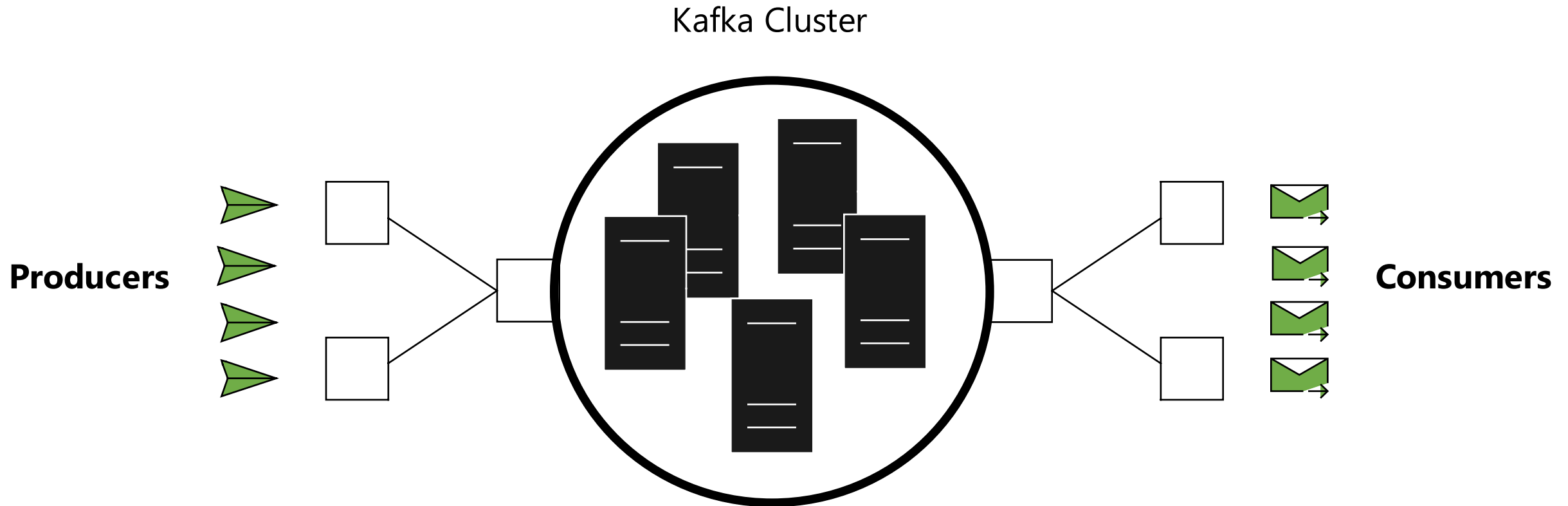
Create a Kafka topic



sshuser@hn0-lena-k: ~\$



Kafka Producers and Consumers



Systems for stream processing

Kafka

Spark

Storm

Flink

Meet Apache Spark



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.

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

* Experimental

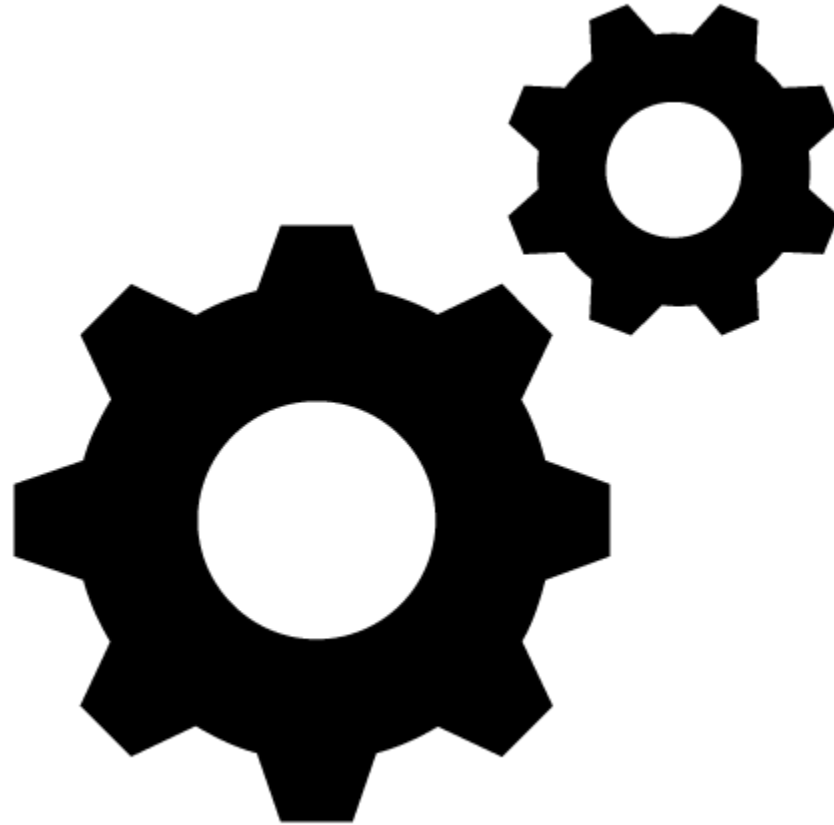
Spark program

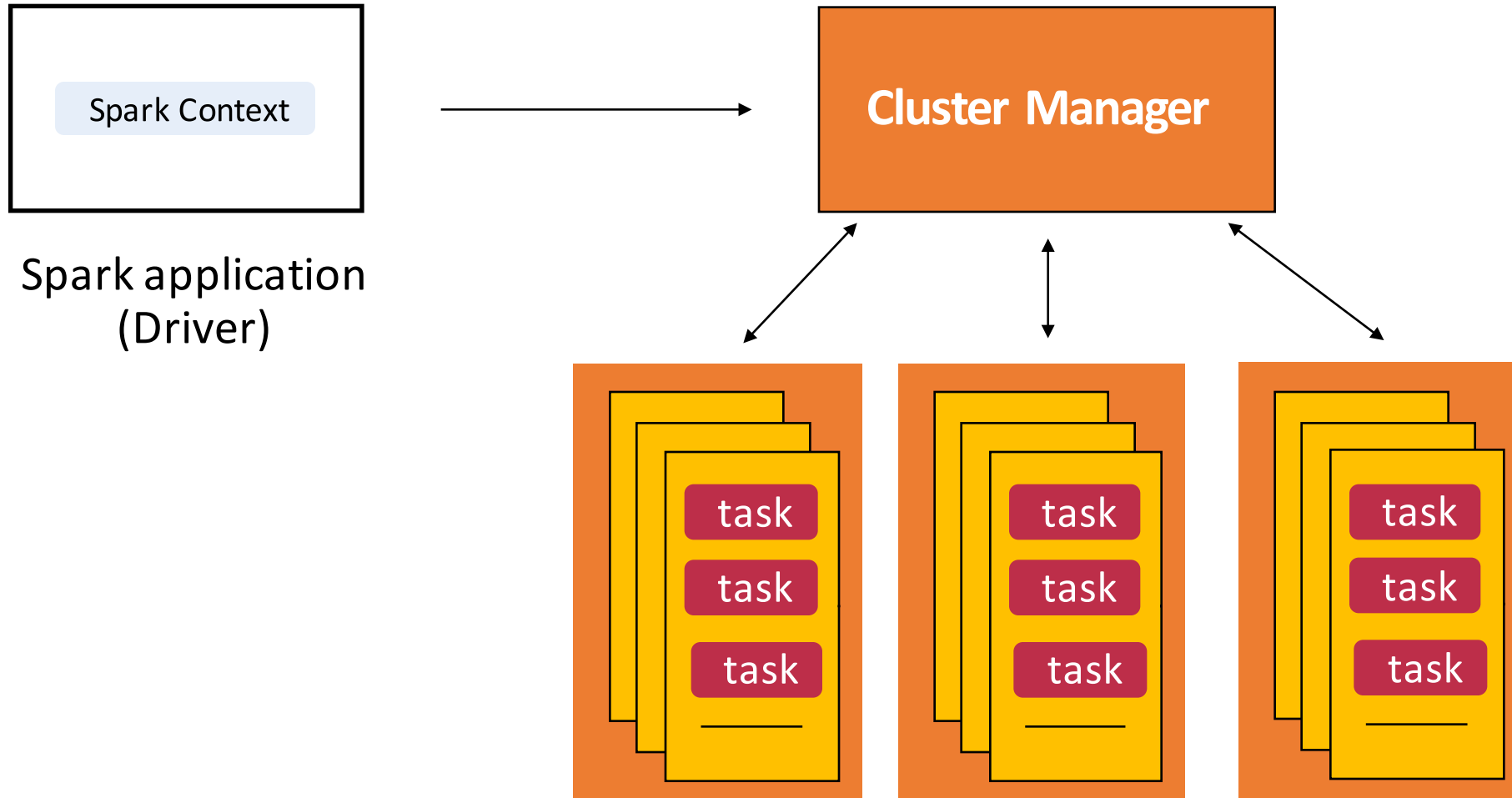
```
val textFile = sc.textFile("hdfs://...")
```

```
val counts =  
  textFile  
    .flatMap(line => line.split(" "))  
    .map(word => (word, 1))  
    .reduceByKey(_ + _)
```

```
counts.saveAsTextFile("hdfs://...")
```

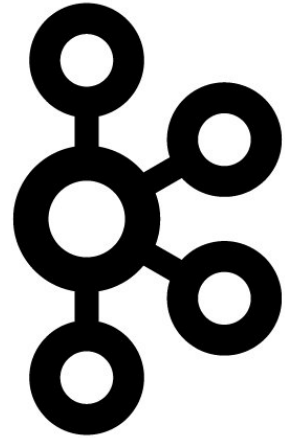

How does Spark work?





Spark workers have executors of tasks

Apache Kafka + Apache Spark



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

Databricks: Interactive Environment



Azure

Databricks



Home



Workspace



Recent



Data



Clusters



Jobs



Search



Azure Databricks

Featured Notebooks



[Introduction to Apache Spark on Databricks](#)



[Databricks for Data Scientists](#)



[Introduction to Structured Streaming](#)

New

- Notebook
- Job
- Cluster
- Table
- Library

Documentation

- [Databricks Guide](#)
- [Python, R, Scala, SQL](#)
- [Importing Data](#)

Open Recent

- 3. Spark From Kafka Message Receiver
- 4. Tweets To Kafka
- 2. Slack Listener For New Messages
- 1. Slack Message Generator
- Kafka Consume - Lena Hall
- EventHubs Data - Lena Hall

Example

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

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

Continuous Processing

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

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

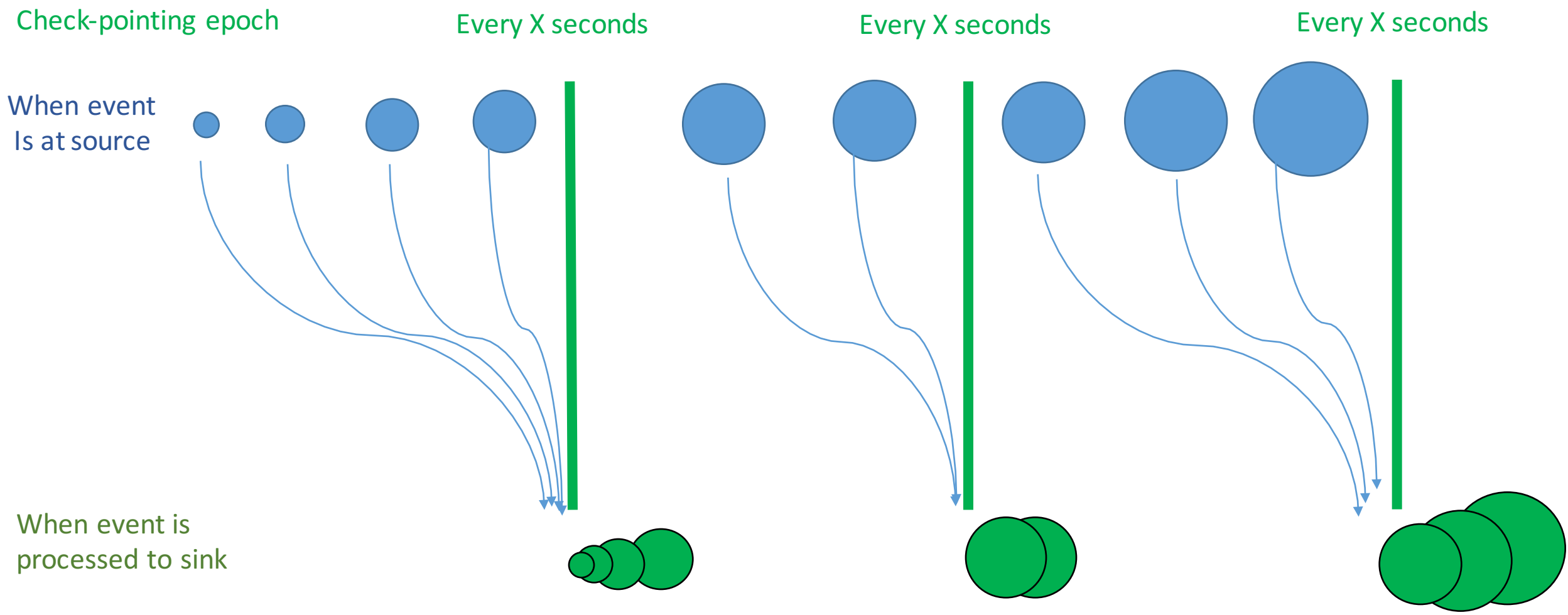
At-least-once fault-tolerance guarantees

Not nearly all operations are supported yet

No automatic retries of failed tasks

Needs enough cluster power to operate

Micro-batch



Continuous

Check-pointing epoch

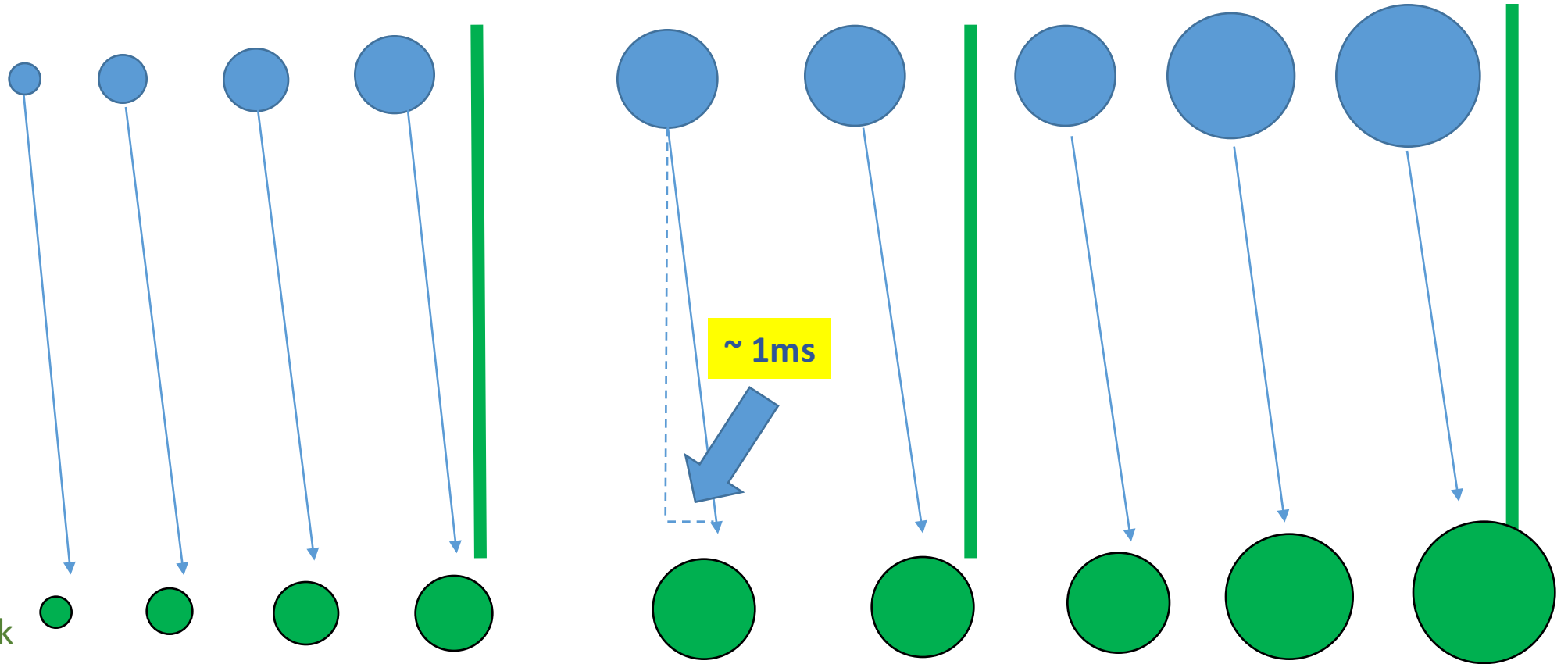
Every X seconds

Every X seconds

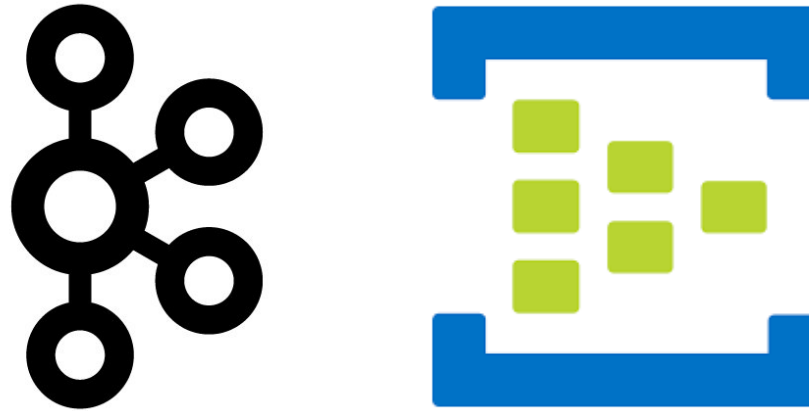
Every X seconds

When event
is at source

When event is
processed to sink



Kafka API for Event Hubs



aka.ms/eventhubs-kafka

Confluent/Kafka **Operator**

and other **Operators** ...

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

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

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

Alena Hall - lenadroid



- ✓ Works on Azure at  Microsoft
- ✓ Lives in  Seattle
- ✓ F# Software Foundation Board of Trustees
- ✓ Organizes [@ML4ALL](https://twitter.com/ML4ALL) 
- ✓ Program Committee for Lambda World
- ✓ Has a channel: [You**Tube**/c/AlenaHall](https://www.youtube.com/c/AlenaHall)