# Other Systems

# Flink & Spark

https://flink.apache.org/

https://spark.apache.org/

# Flink: 1 Slide is Enough

- Unified batch and streaming;
- Event-driven applications (timers, CEP);
- Production-ready SQL support;
- First class state management;
- Fault-tolerance for correctness;
- Connectors for I/O (Kafka, Cassandra);
- Support for multiple deploys (K8s, Yarn, Mesos) and storage systems (HDFS, S3, RocksDB) for snapshots.

# Spark

- Fully fledged batch/stream processor as Flink;
- (Now) it also adopts a unified approach;
- Spark had a different vision (it was born with the idea of batch processing), but it is now converging towards Flink ideas (e.g. structured streaming);
- Great support for ML libraries as Flink.

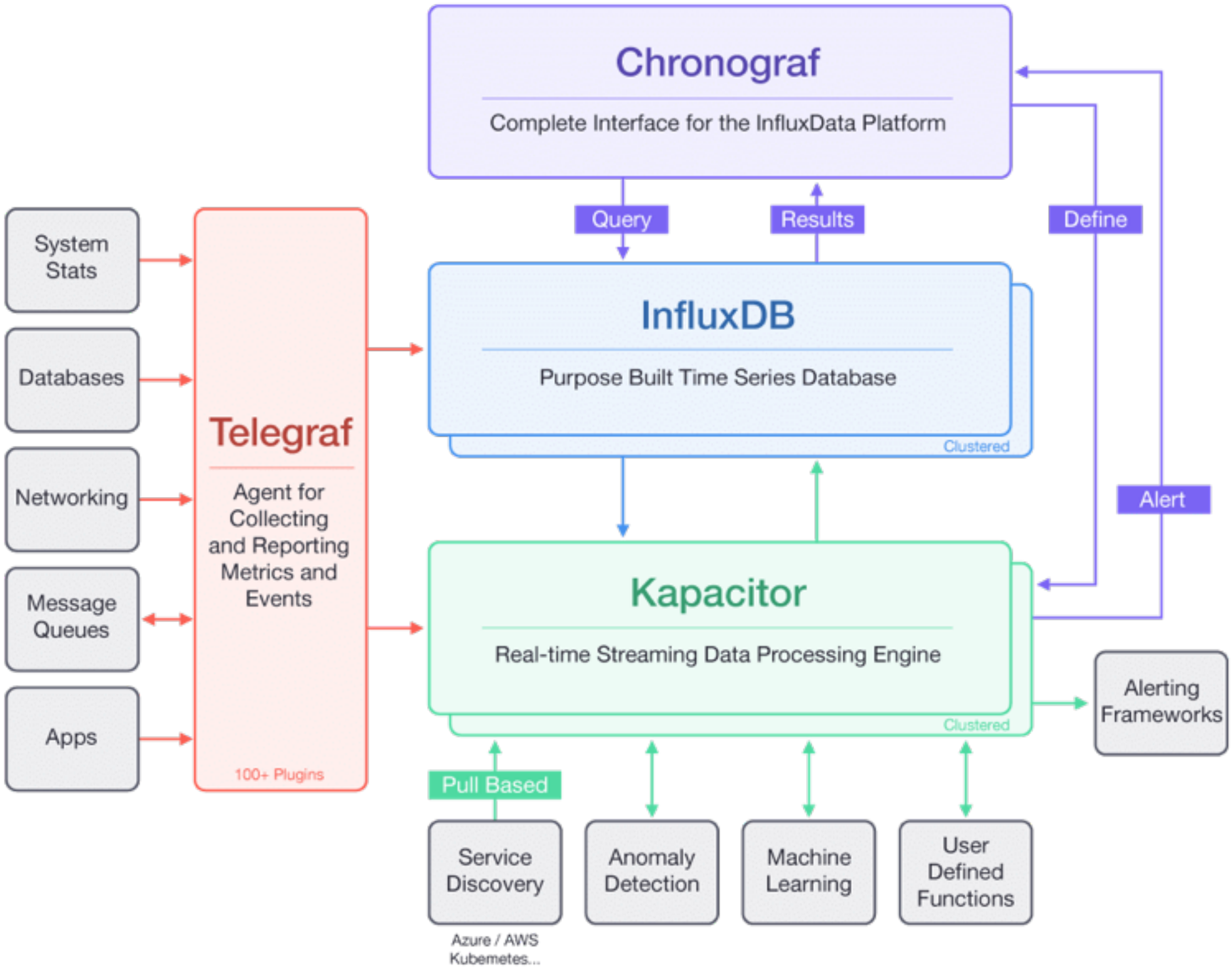
# Influx

https://www.influxdata.com/products/

# The TICK stack

- **T**elegraph: gather data from sources;
- **I**nfluxDB: the core, the Time Series DataBase;
- **C**hronograph: visualize data;
- **K**apacitor: act basing on continuous queries on data;

# The TICK stack



# TSDB

A Time Series DataBase is optimised for managing timestamped data:

- Timestamp Data compression;
- Optimised for scans over huge *periods*;
- Out-of-the-box data summarization (downsampling);
- Time aware queries.

# How to query the Database

- Influx is realizing that static queries, continuous queries and event-processing are very similar;
- They are moving to a SQL-like declarative query language to a unified scripting language: **flux**[1];
- The parser reads the script and generates a plan (as in Flink).



[1] Github repository

# A Flux Query

```
from(db:"foo")
    |> filter(
        fn: (r) => r[ "_measurement"] == "cpu"
        AND r["_field" ]== "usage_system")
|> range(start:-12h)
|> window(every:10m, period: 5m)
|> min()
```

# Kapacitor VS CQ

- Influx has a system to perform Continuous Queries
  - It provides a SQL-like language for CQs
  - Transitioning to TICKscript[1]



[1] Kapacitor as a CQ engine

# Kapacitor Script (1/2)

```
dbrp "telegraf"."autogen"

stream
    |from()
        .measurement(‘cpu')
    |window()
        .period(5m)
        .every(5m)
        .align()
    |mean('usage_idle')
        .as('usage_idle')
```

# Kapacitor Script (2/2)

```
|alert()
  .crit(lambda: int("usage_idle") <  70)

  // Write it to a file.
  .log('/tmp/alerts.log')

  // … or send to REST endpoint
  .post('https://alerthandler.example.com')

  // Execute script
  .exec('/bin/custom_alert_handler.sh')
```

# Distributed Features (1/2)

- Data is sharded and replicated over nodes;
- *Shard groups* are based on *retention policy* (the time that data looses importance);
- On write, data is sent to the right shard within the shard group by using the *tag set* of the record (like indexing in normal DBs, or keyBy in Flink);
- If your retention policy is 1 day, then, for any given tag set, all data on any given day lies in a single shard;
- Queries that span multiple days must be *reduced* by the coordinator.

# Distributed Features (2/2)

- Different levels of consistency for replication:
  - any: the write has been logged (*hinted handoff*)
  - one: the write is on at least one replica
  - quorum: the write is on the majority of replicas
  - all: synchronous replication
- Fault-tolerance: hinted handoff is a persisted log (for every node to every node) that is replayed in case of failure.

# A Final Note

- Single machine is open-source;
- Distributed cluster requires payment;
- The database is distributed, the computation is not;
- Every query is executed by a single node, if it fails, then it fails as a whole.

# Kafka

https://kafka.apache.org/

# About Kafka

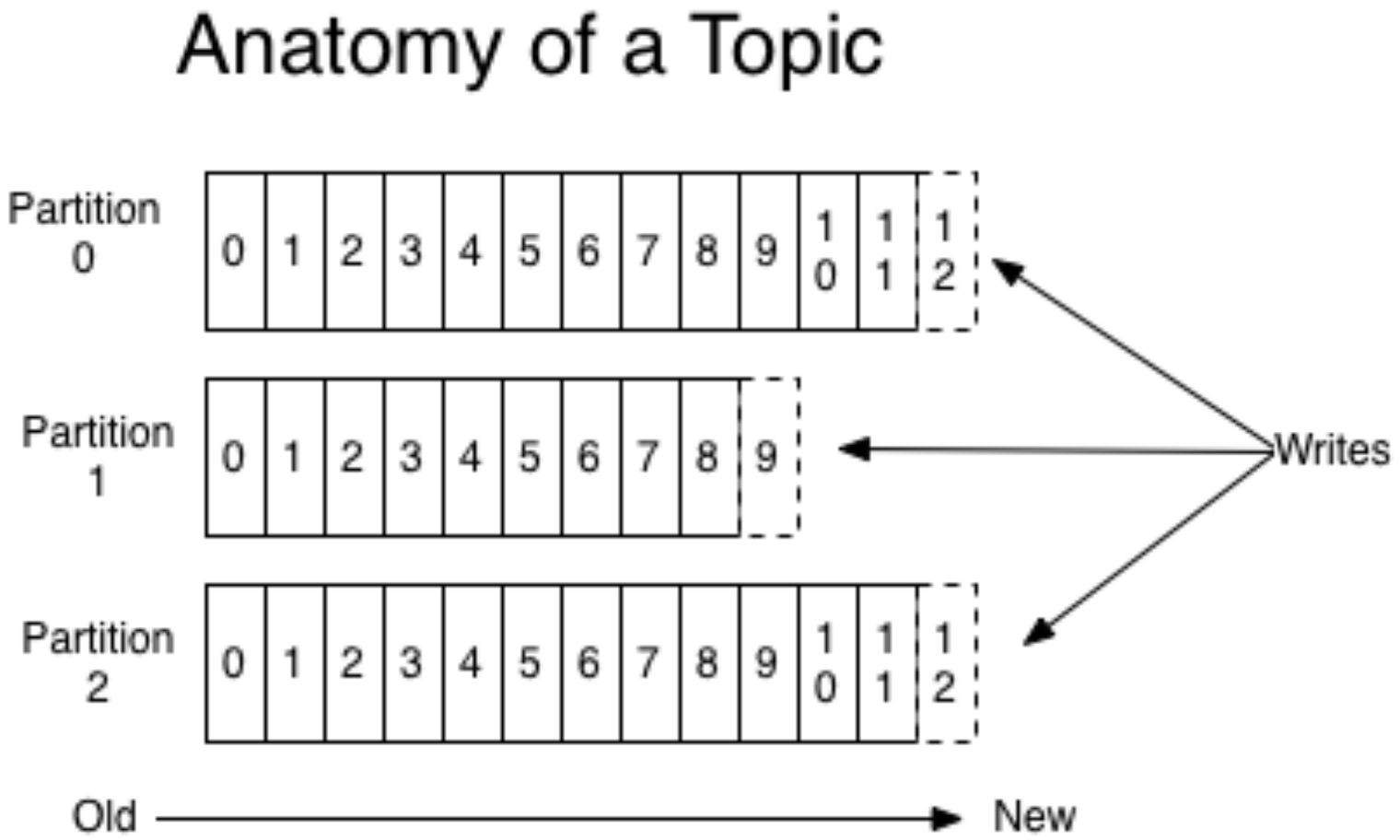
- **Kafka** *is a*[1] distributed streaming platform;
- Its core is a highly scalable and fault tolerant log store;
- **Kafka Streams** is a library to write streaming pipelines consume kafka logs and produce kafka logs.



[1] or, better, it has become

# Kafka Concepts (1/3)

**The topic**: a partitioned stream



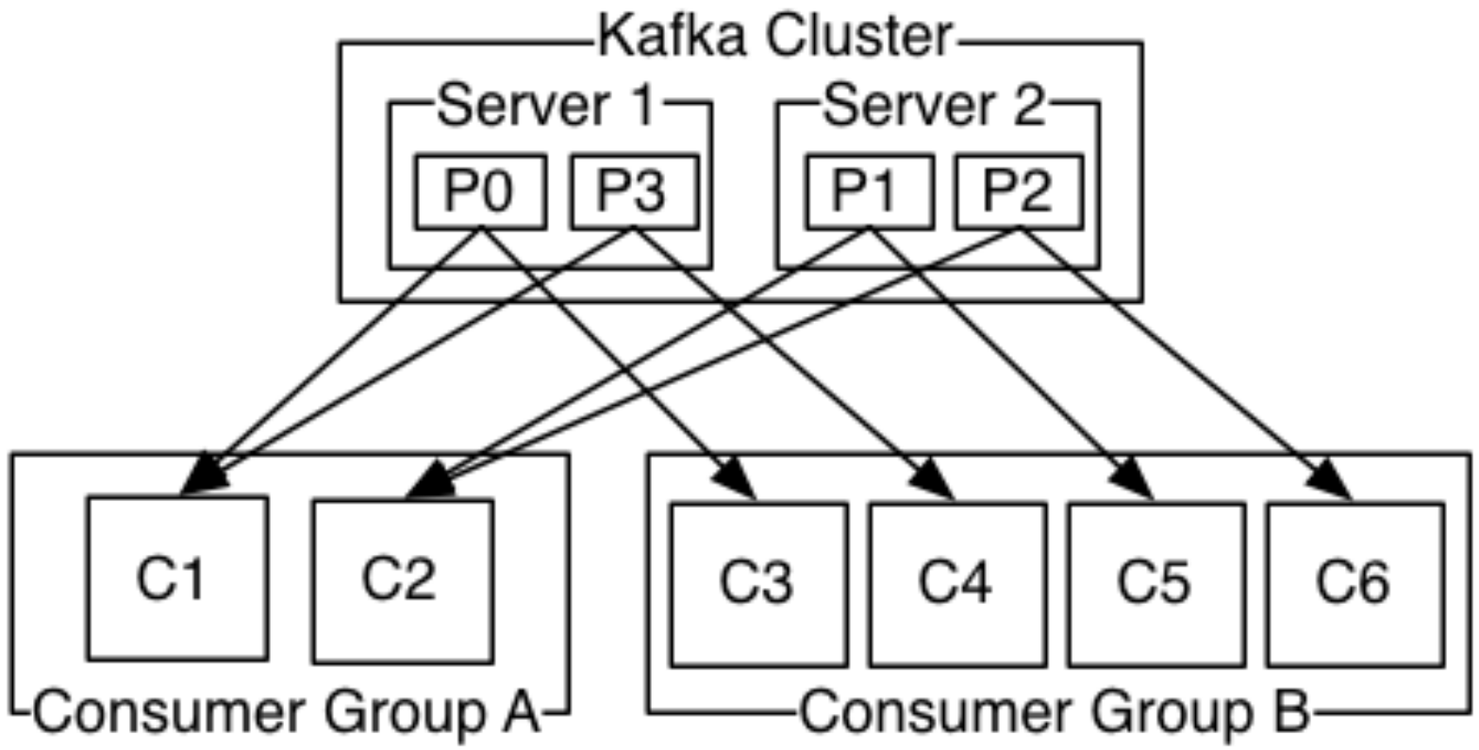

- Every partition is an append-only log;
- Every record is assigned an *offset* in the partition.

# Kafka Concepts (2/3)

- Partitions allow for scale;
- Every partition can be replicated for fault-tolerance (master-slave replication);
- The producers of data pushes data to some partition basing on some policy (e.g. it could use a key).

# Kafka Concepts (3/3)

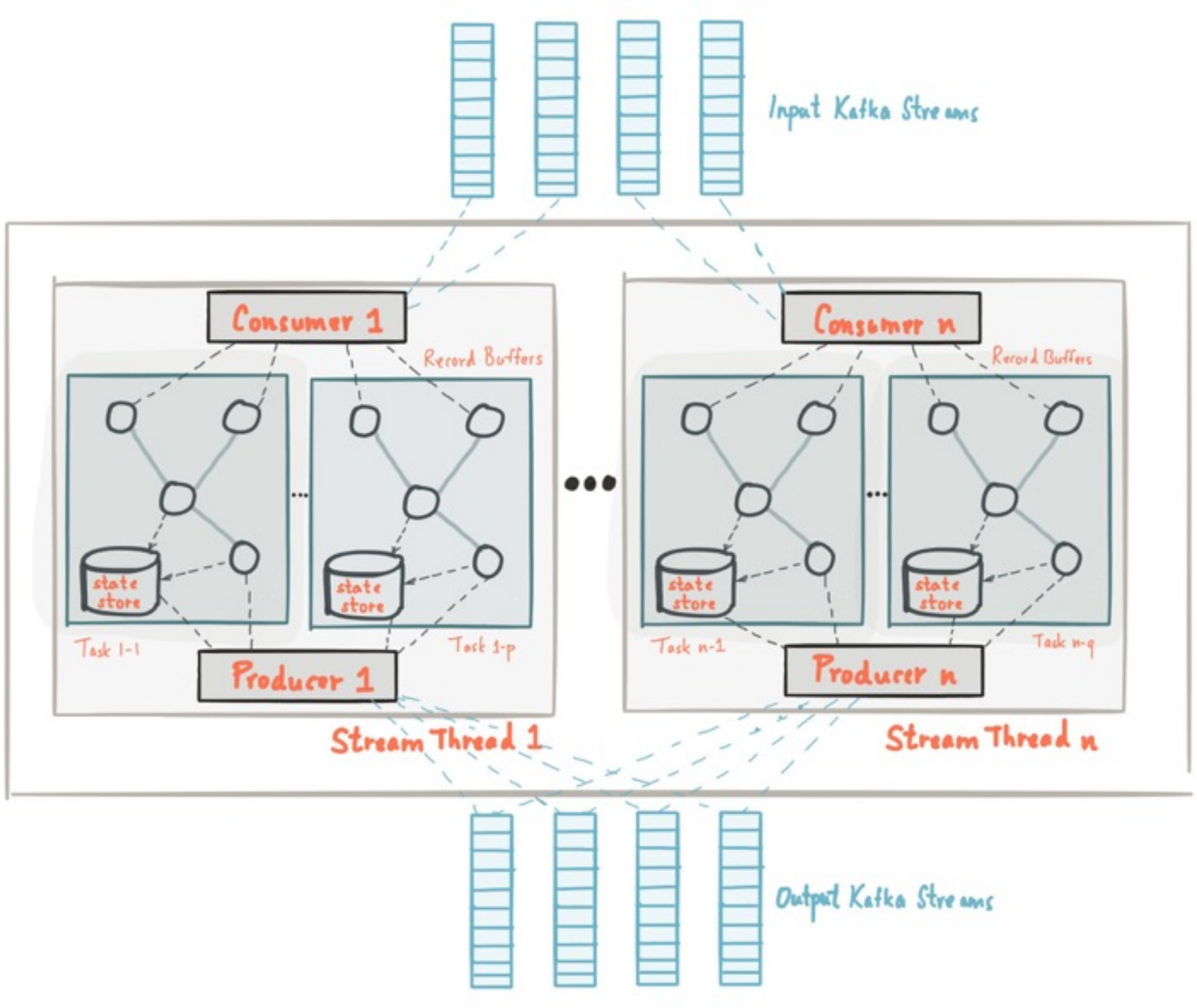
- Consumers are organized in groups;
- Every consumer in a consumer group is assigned to a partition;
- Consumers can join/leave a group —> partition assignment is handled transparently.



# Kafka Streams

- A simple library to write stream processing jobs;
- There is NO runtime —> no schedulers, no taskmanagers, no jobmanagers;
- Kafka Streams is built on top of the consumer group concept;
- There is a stream processing *task* for each partition in the input topic(s);
- Each consumer in the consumer group runs some of the tasks and produces to other topics;

# Kafka Streams



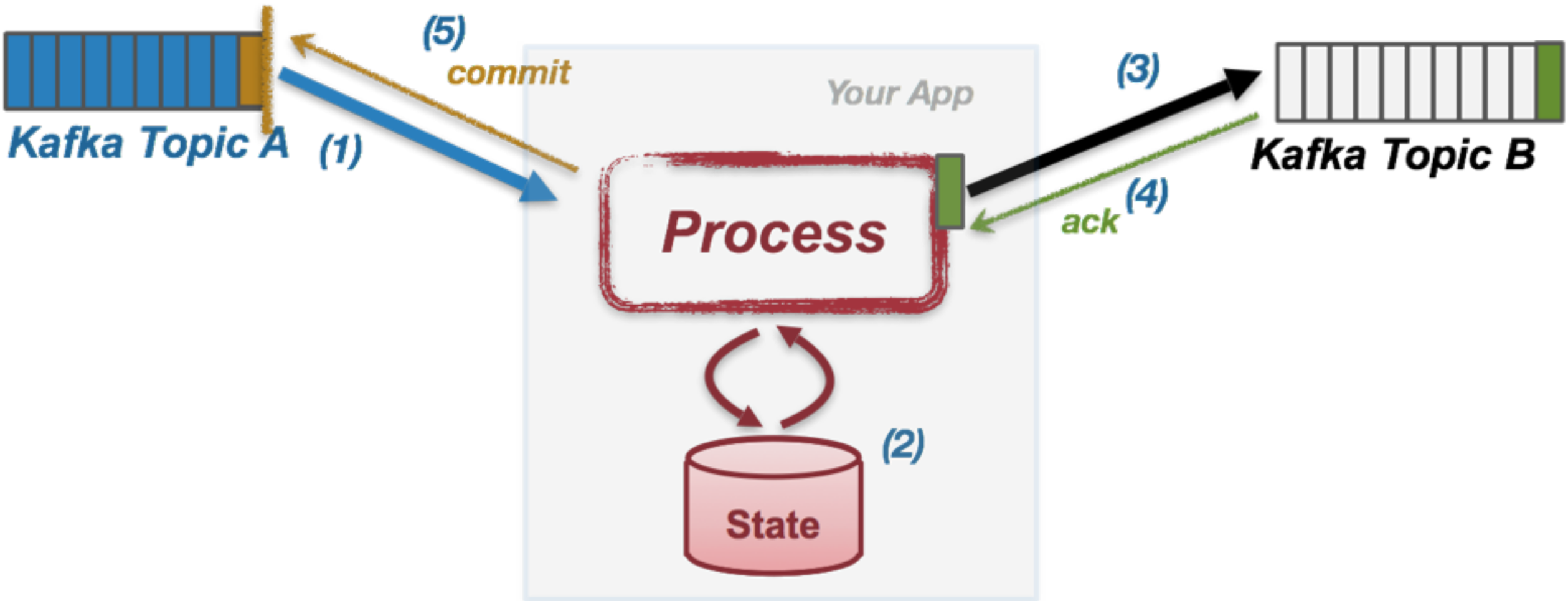
# Kafka Streams - Failure (1/2)

- Local state changes are persisted to dedicated kafka topics too!
- If some consumer fails, its tasks are assigned to another one that can replay the state changes;
- Kafka guarantees exactly-once processing guarantees among multiple kafka topics by storing offsets and committing records once they are produced towards the next kafka topic;
- NOTE that re-partitioning causes to write to a kafka topic (again).



more here

# Kafka Streams - Failure (2/2)




Differences wrt Flink? Opinions?

# KSQL (1/2)

SQL-like language for streaming

```
CREATE STREAM pageviews_enriched
AS \
  SELECT pv.viewtime, \
         pv.userid AS userid, \
         ...
         u.contact_info \
  FROM page_views pv \
  LEFT JOIN users u ON pv.userid =
users.userid;
```

# KSQL (2/2)

SQL-like language for streaming

```
CREATE TABLE
windowed AS \
SELECT regionid, count(*) \
FROM pageviews_enriched \
WINDOW TUMBLING (SIZE 1 MINUTE) \
GROUP BY regionid;
```

# Connectors

- Kafka Connect is a separate project[1] that aims at integrating Kafka broker with other systems;
- Source and sink connectors;
- Some connectors are developed by Confluent (Kafka's company), some are certified, some are from the community;
- To name a few:
  - JDBC, Elasticsearch, JMS, HDFS;
  - VoltDB, MongoDB, Redis, RabbitMQ.



[1] Kafka Connect link