The Art of Possible

CAKESOLUTIONS
ENTERPRISE SOFTWARE SERVICES

Staging Reactive data pipelines using Kafka as the backbone

Jaakko Pallari (@lepovirta)

Simon Souter (@simonsouter)

/cakesolutions /scala-kafka-client

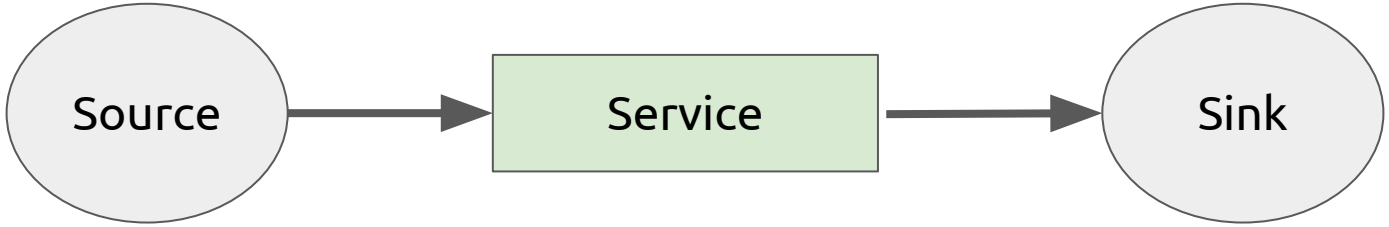
CAKESOLUTIONS
ENTERPRISE SOFTWARE SERVICES

Reactive Solutions at Cake

Contents

Stream Processing

- Big Data
- Processing in Real-time
- Event Throughput vs Number of Queries
- IoT



Distributed Streaming Engines

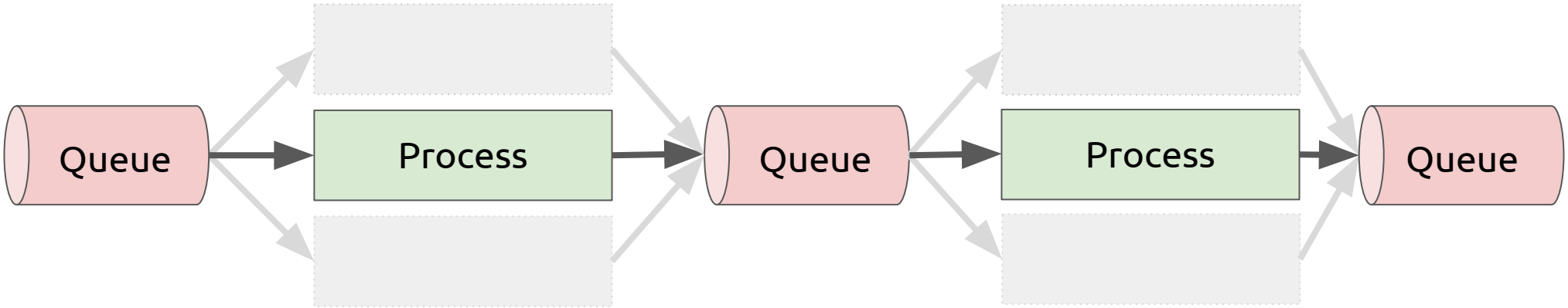
- Server Applications
- Stream topologies deployed to cluster
- Framework design

Streaming from ground-up

- Custom Streaming Applications
- Leverage existing tool stack

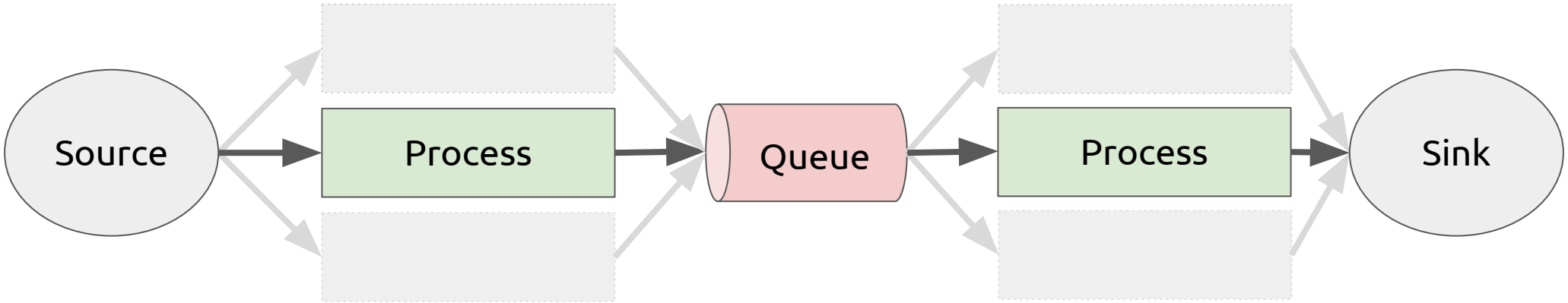
Staged data pipelines

- Staged Event Driven Architecture
- Processes separated by a queue
- Processing in stages



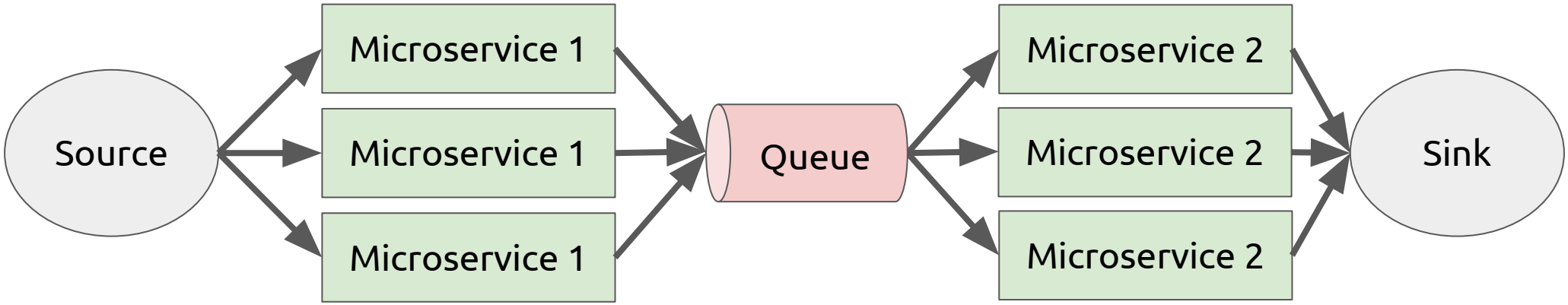
Reactive data pipelines

- Responsive
- Resilient
- Elastic
- Message Driven



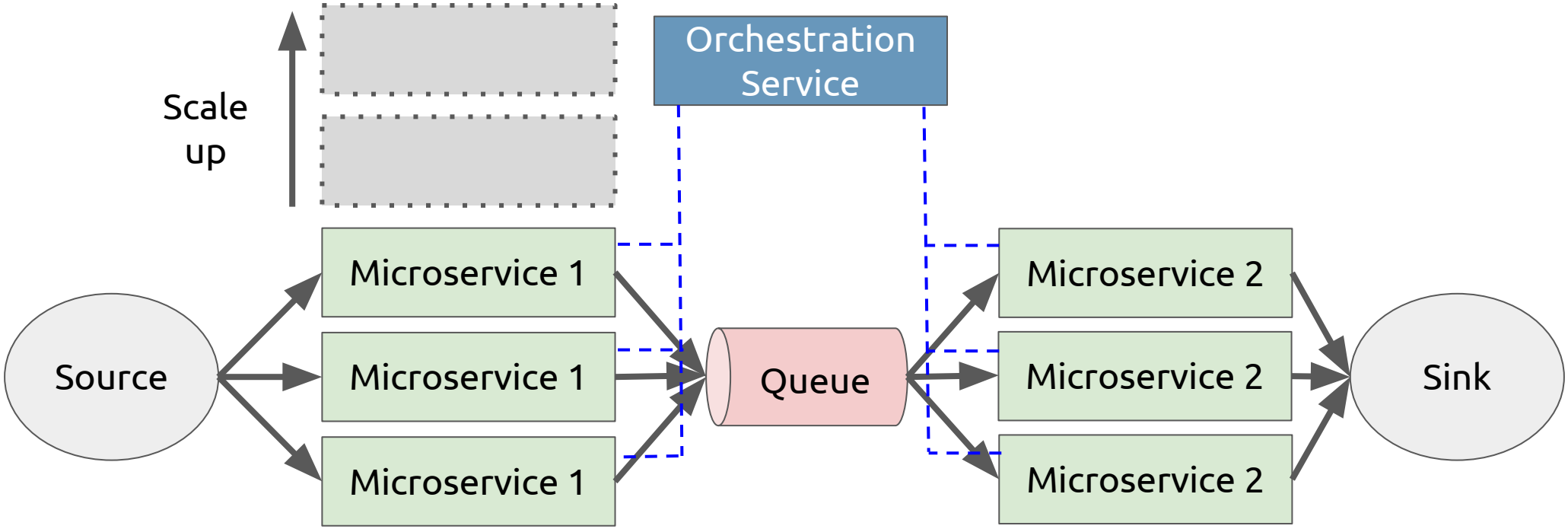
Streaming from ground-up

- **Microservices** as processing components



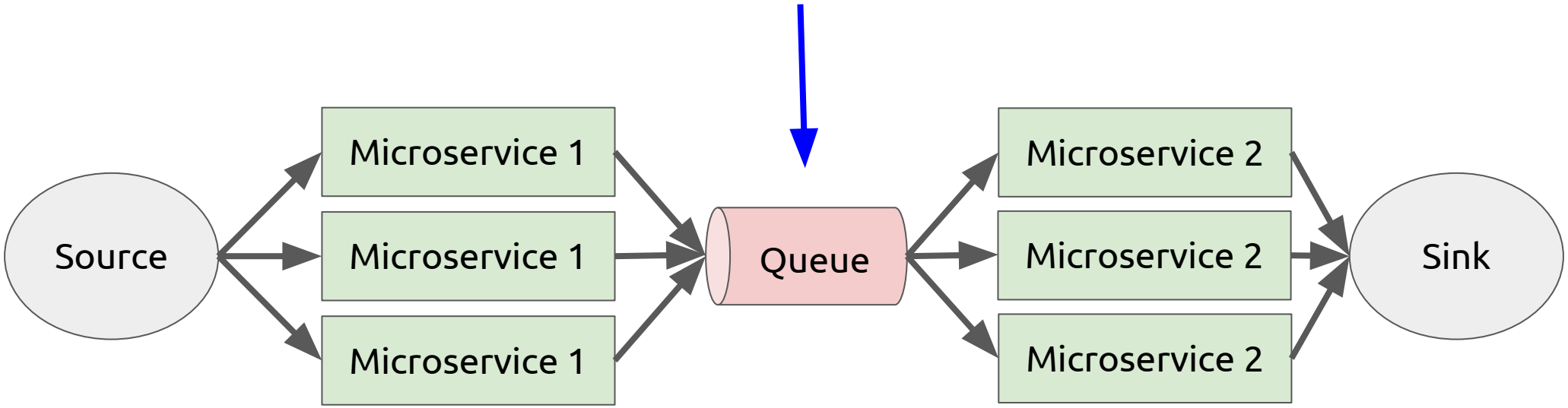
Streaming from ground-up

- Deployment via **cluster orchestration services**



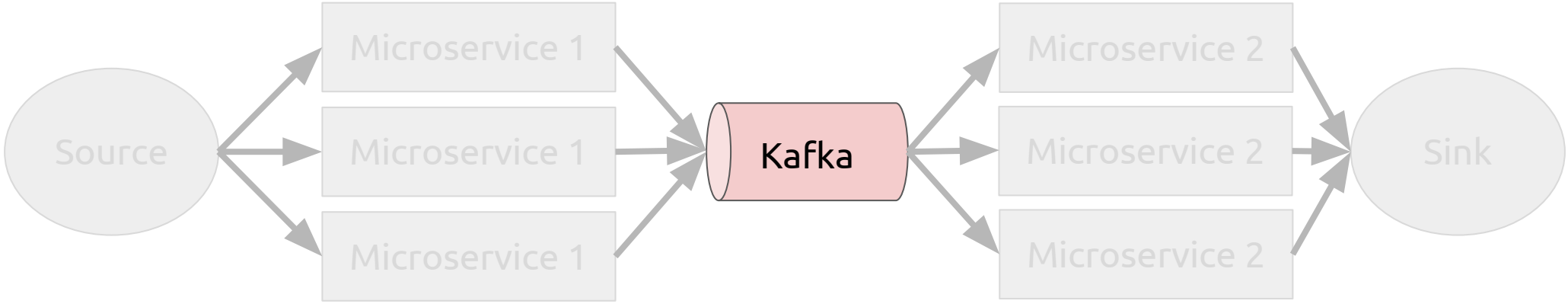
Streaming from ground-up

- **Messaging middleware** for resilient data distribution between microservices

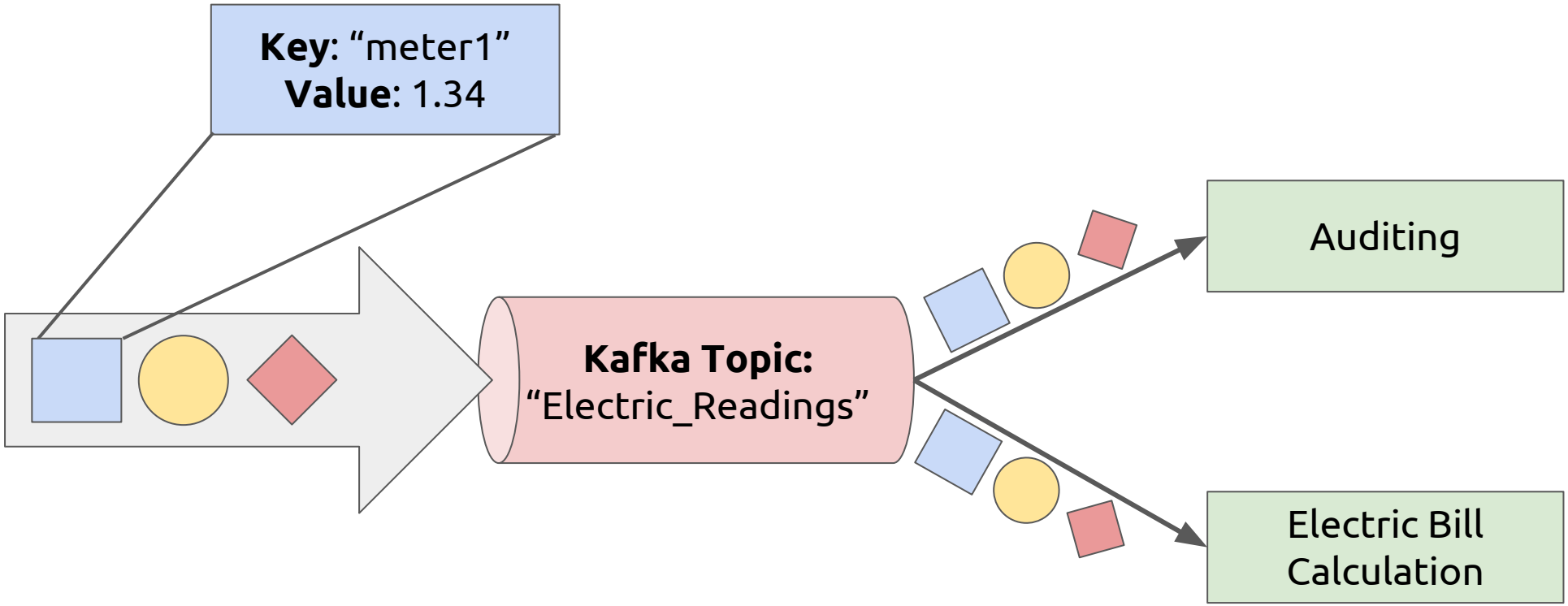


What is Kafka?

- Distributed Message Broker
- Supports Parallel Streaming
- Kafka as a Reactive MQ

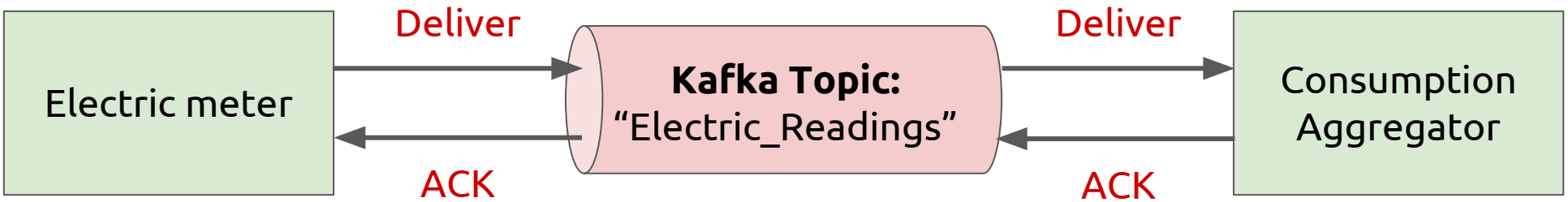


Kafka: topic and message anatomy



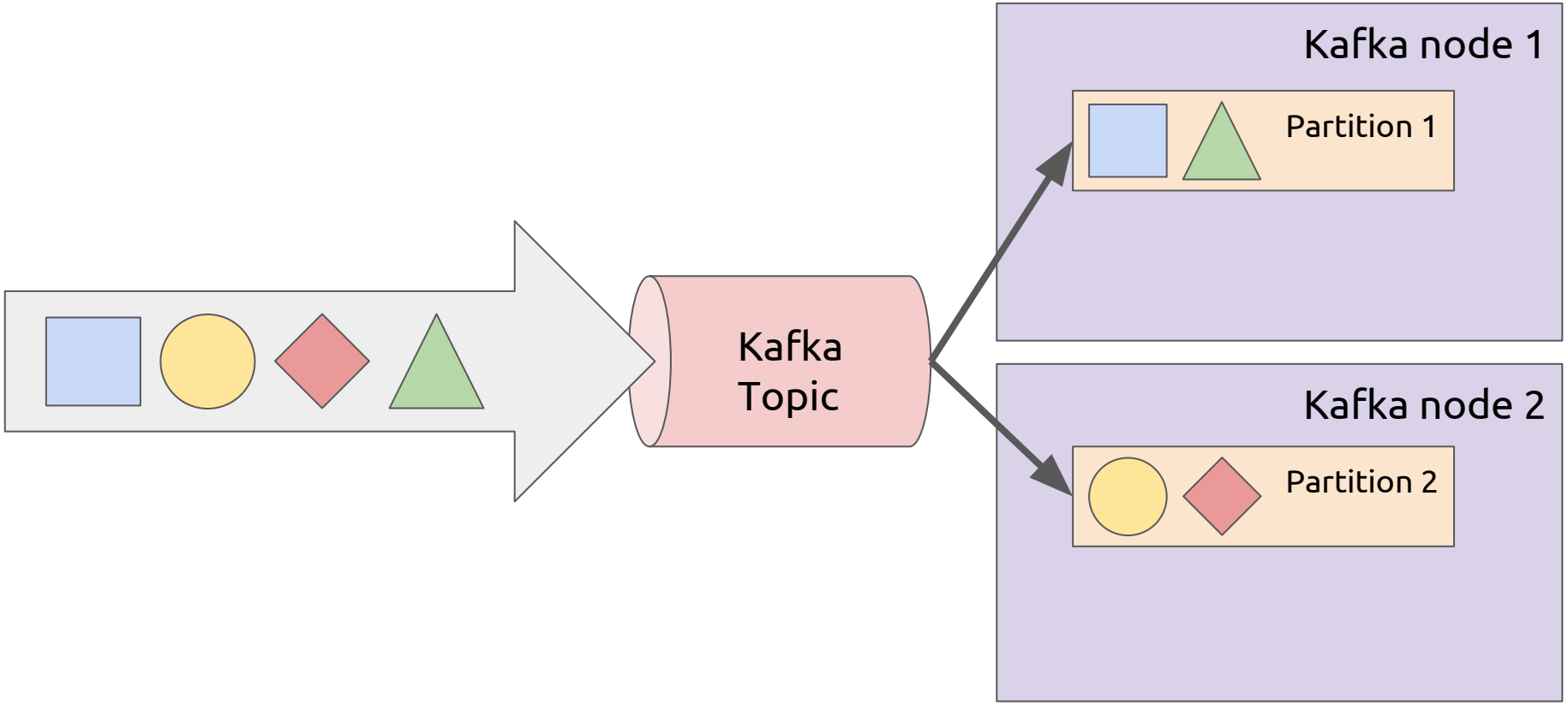
Message Driven

Kafka: at-least-once delivery



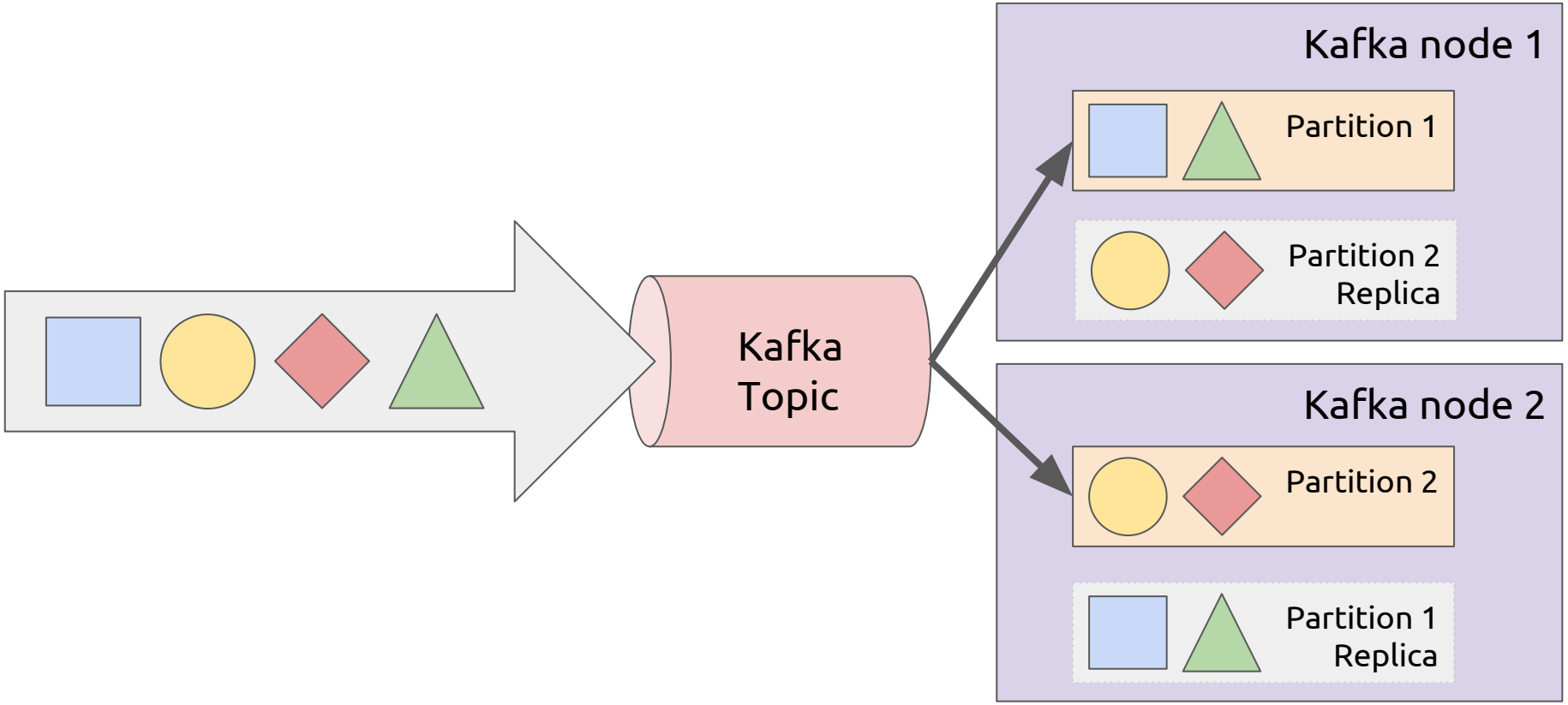
Resilient

Kafka: clustering - arrangement



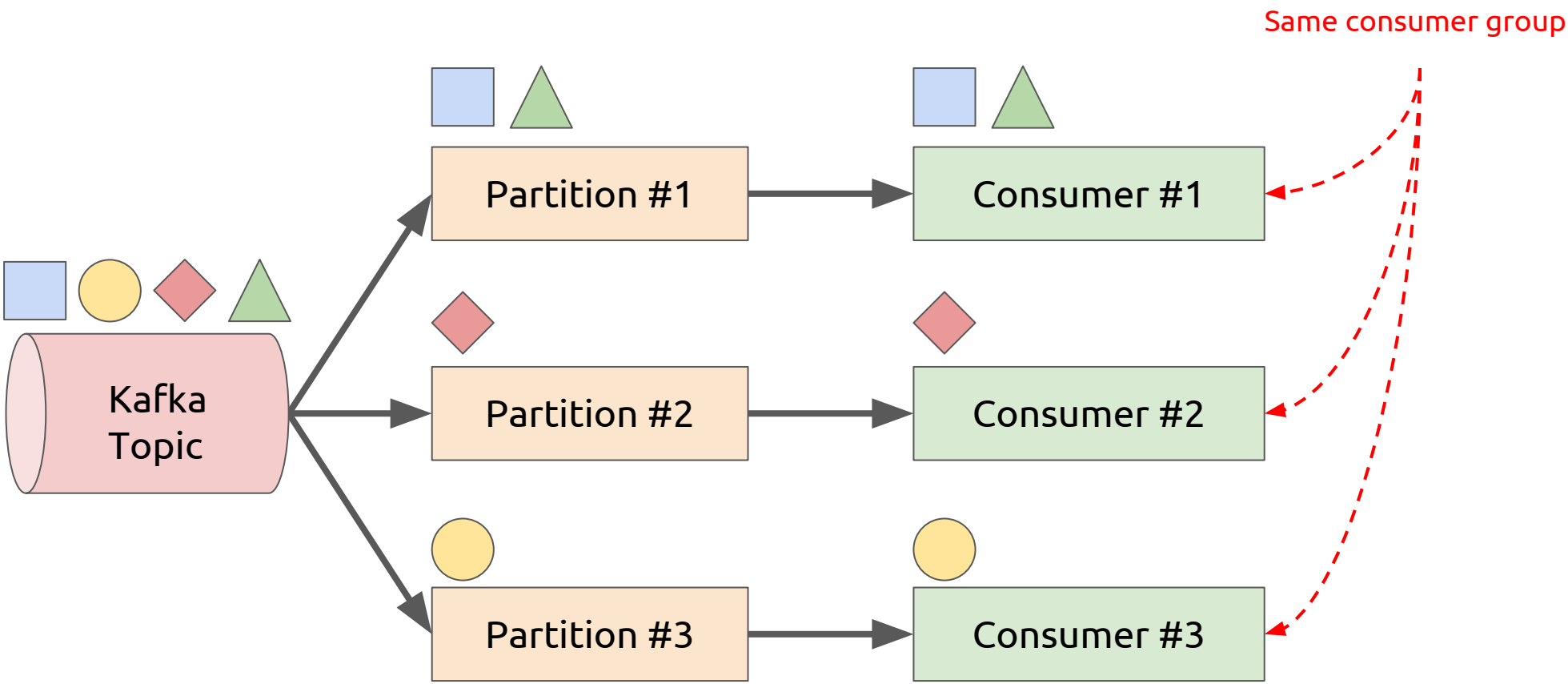
Elastic

Kafka: clustering - replication



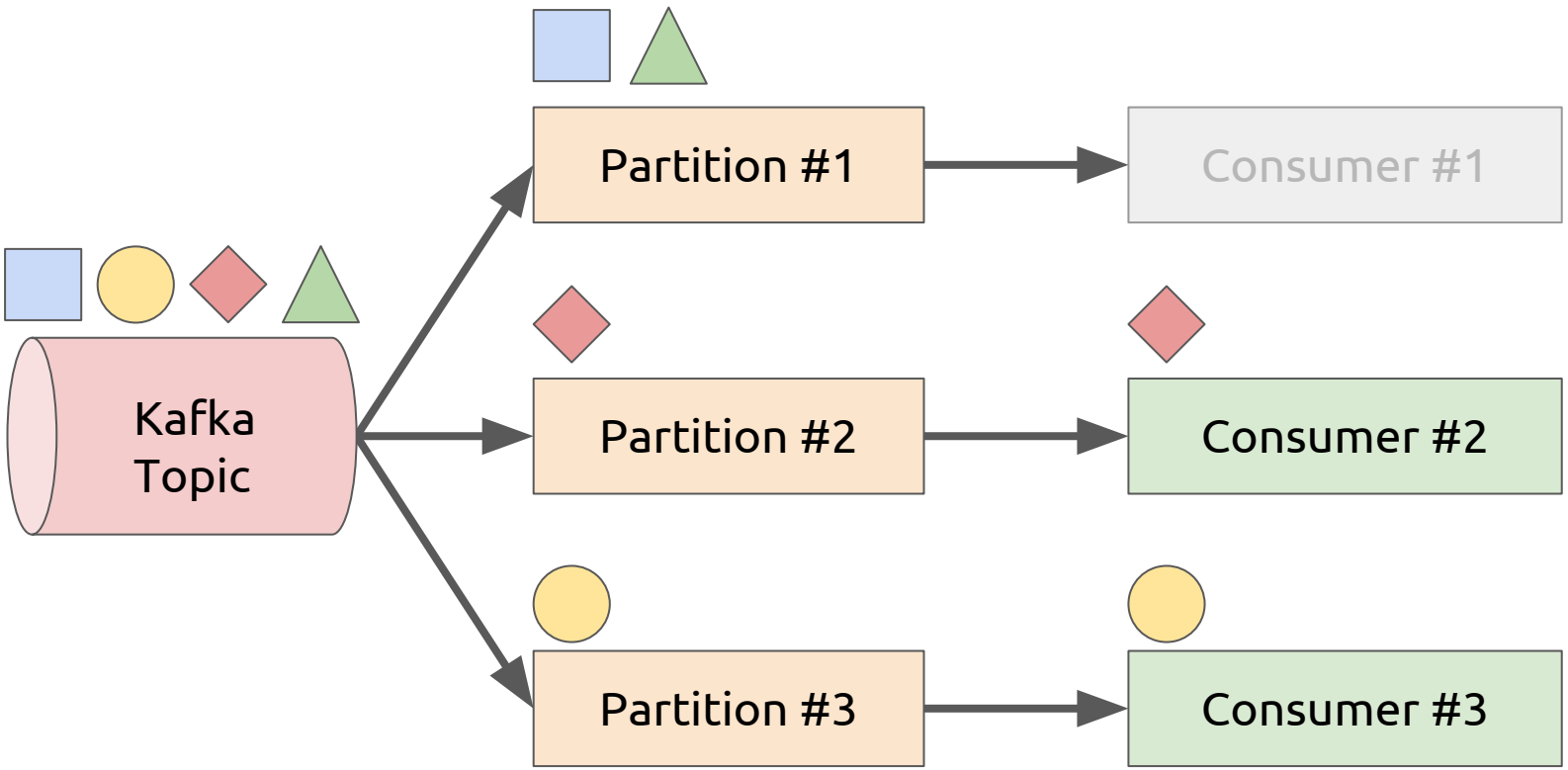
Resilient

Kafka: clustering - consumer



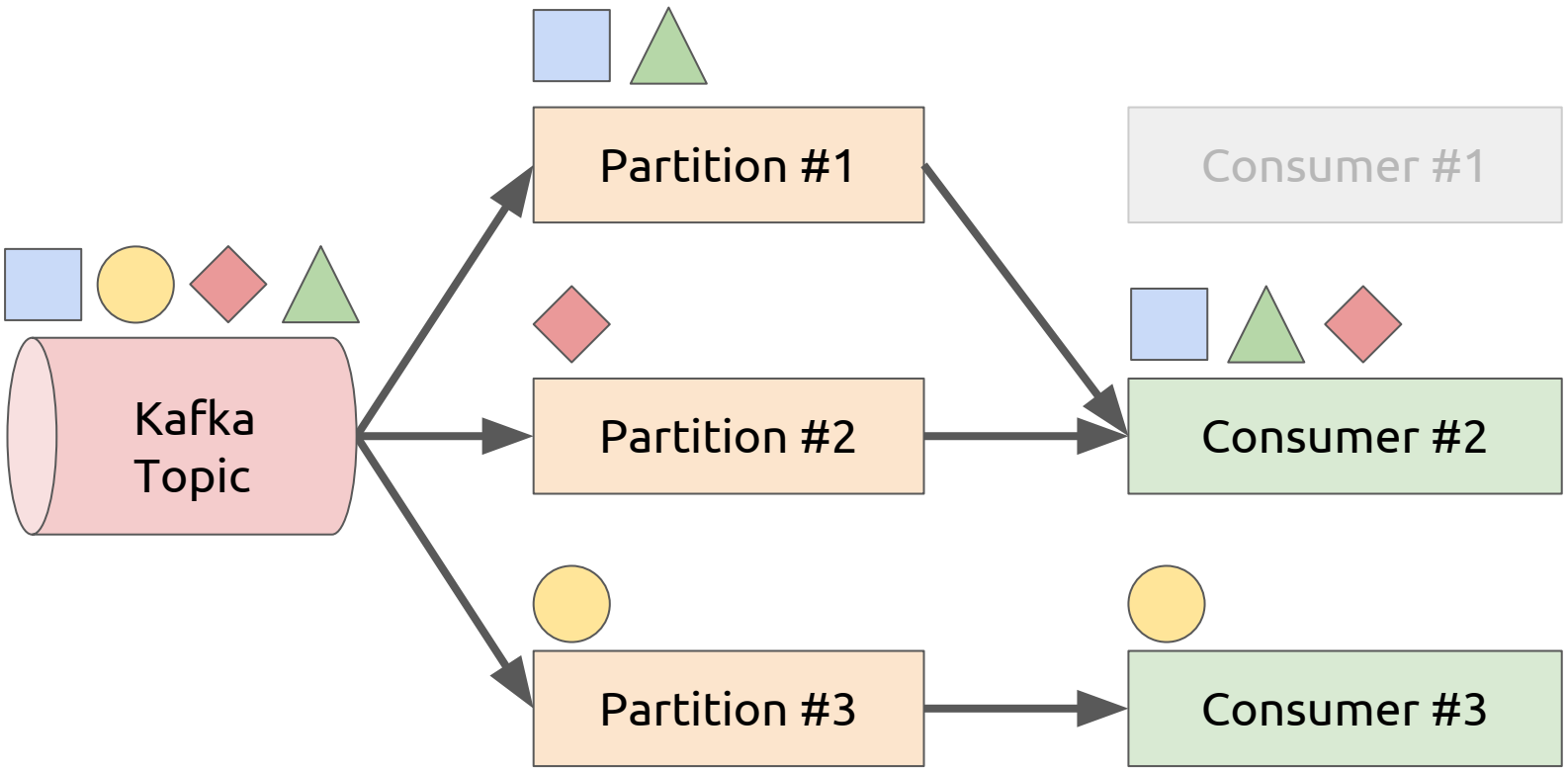
Responsive

Kafka: clustering - consumer



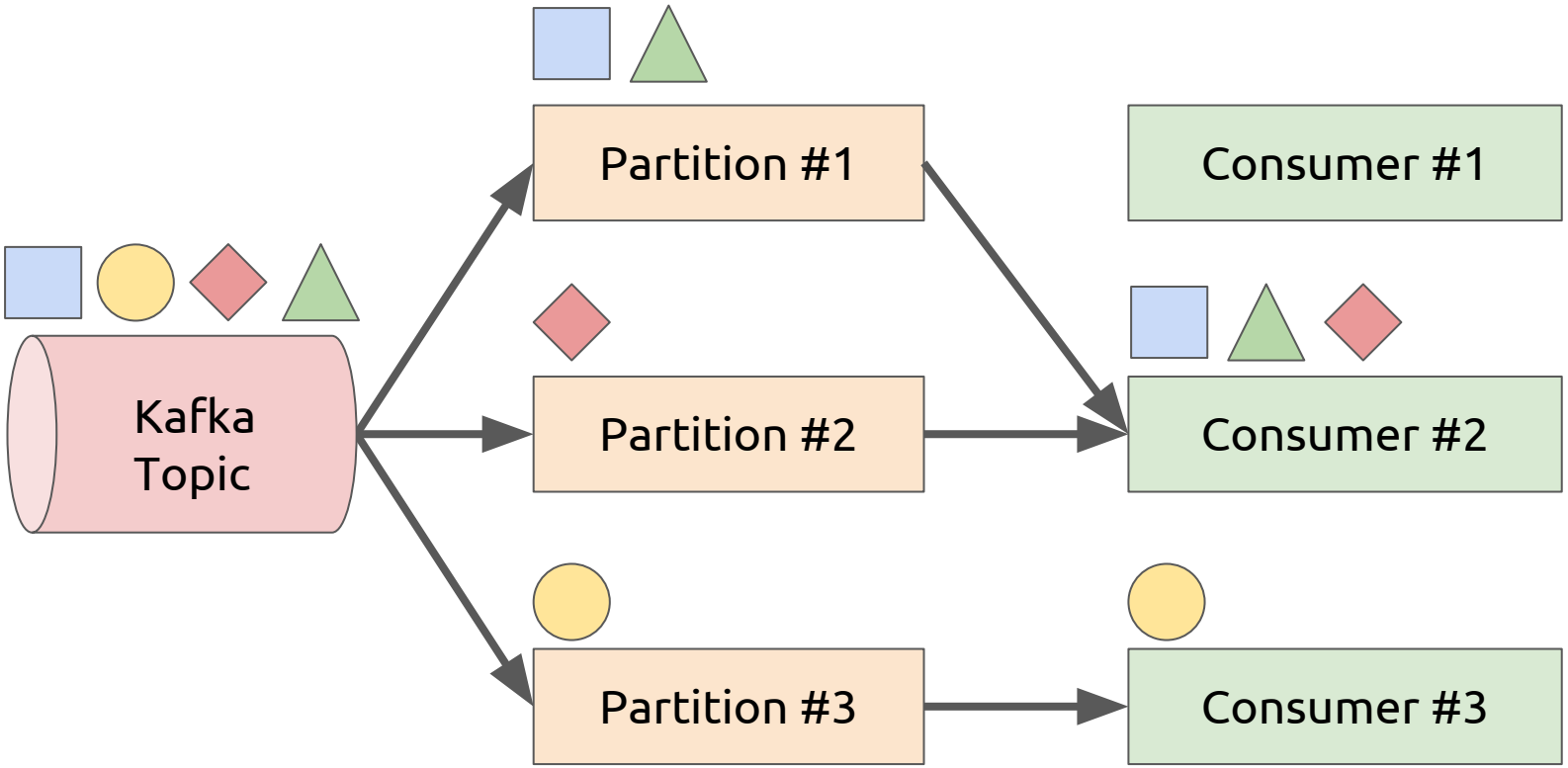
Responsive

Kafka: clustering - consumer



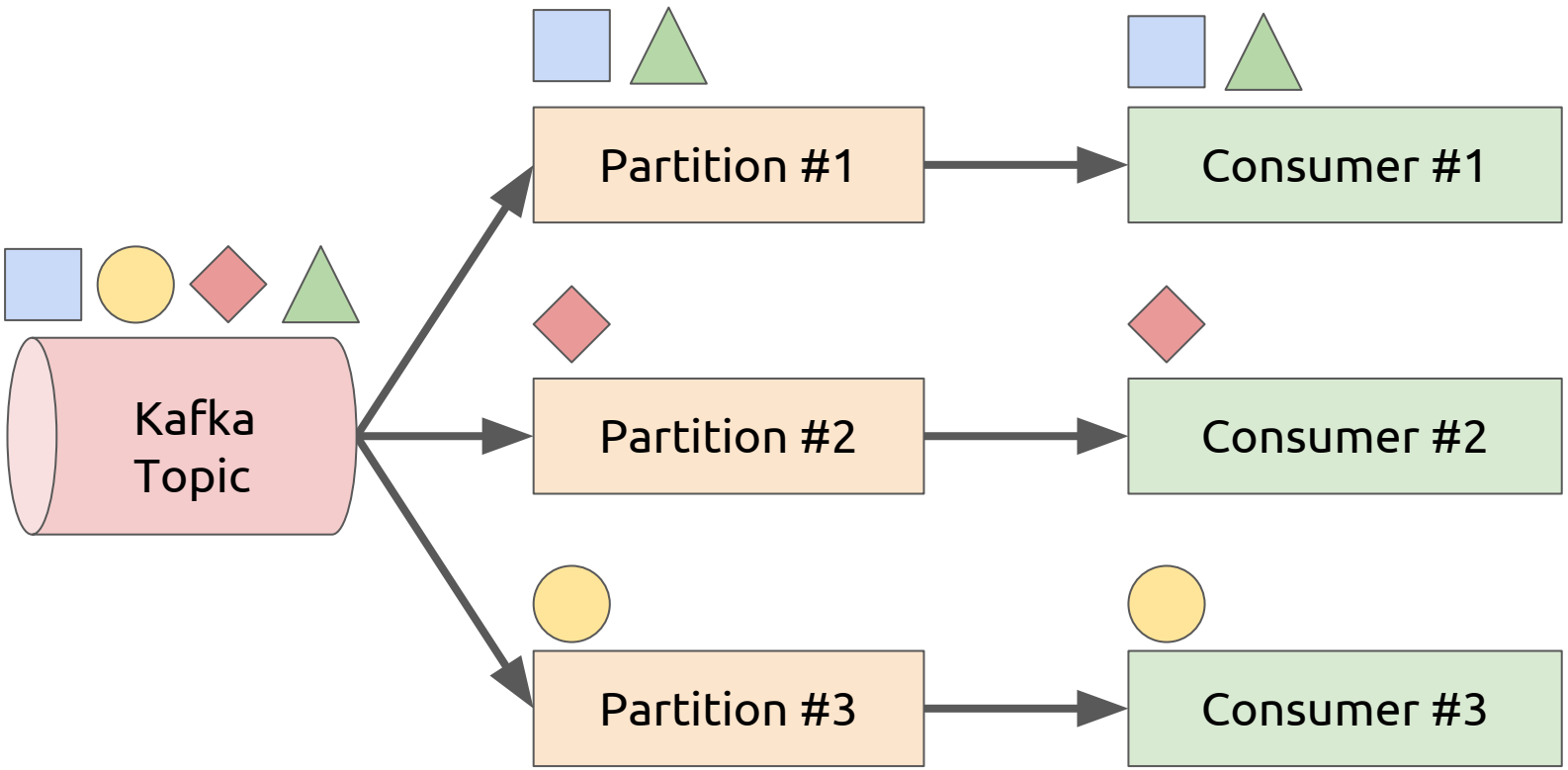
Responsive

Kafka: clustering - consumer



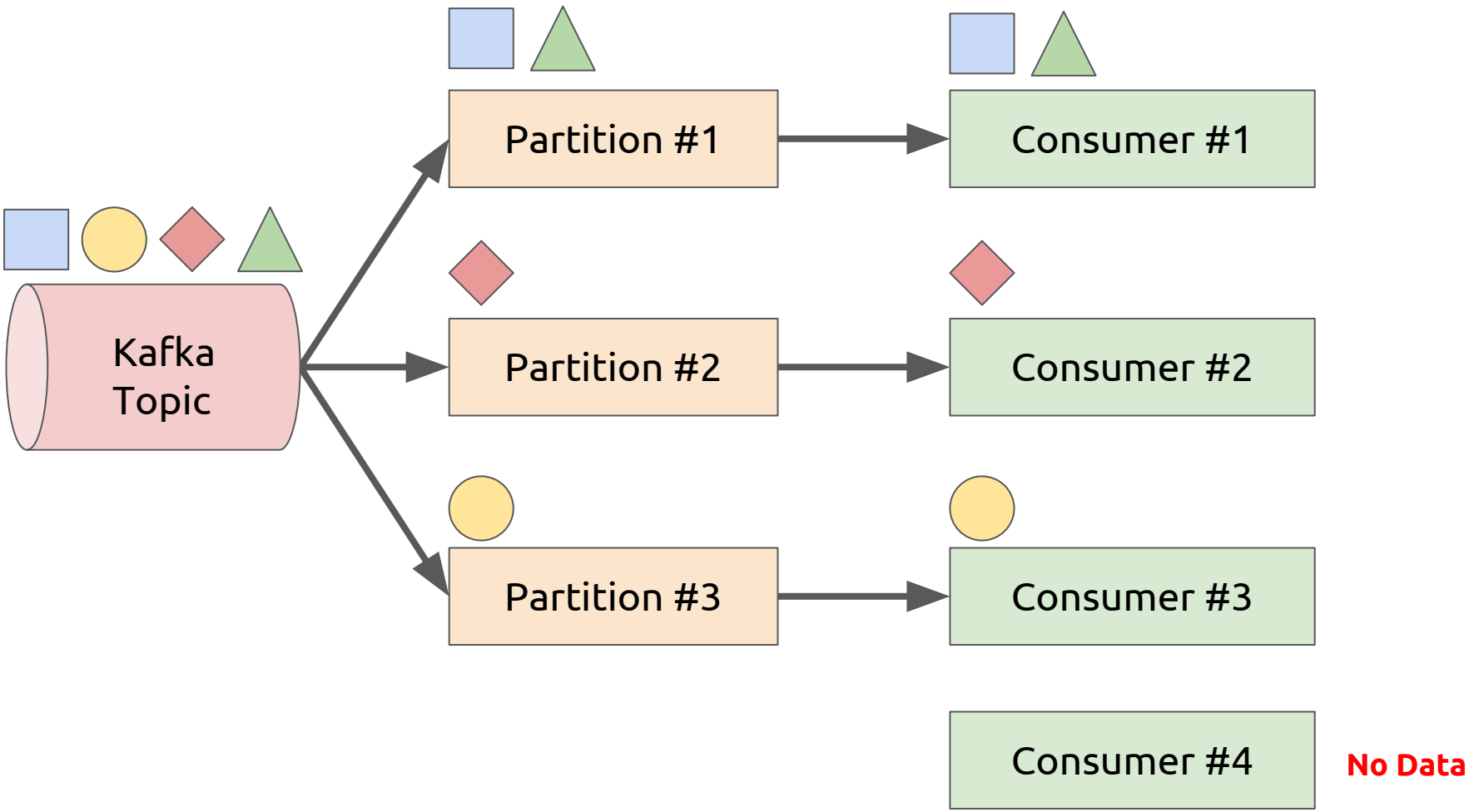
Responsive

Kafka: clustering - consumer



Responsive

Kafka: clustering - consumer



Responsive

Kafka: high throughput

- Single partition consumer: **20-90 Mb/sec**

Responsive

Kafka the Reactive MQ

Responsive

- Consumer clustering
- High throughput

Resilient

- At-least-once delivery
- Replication

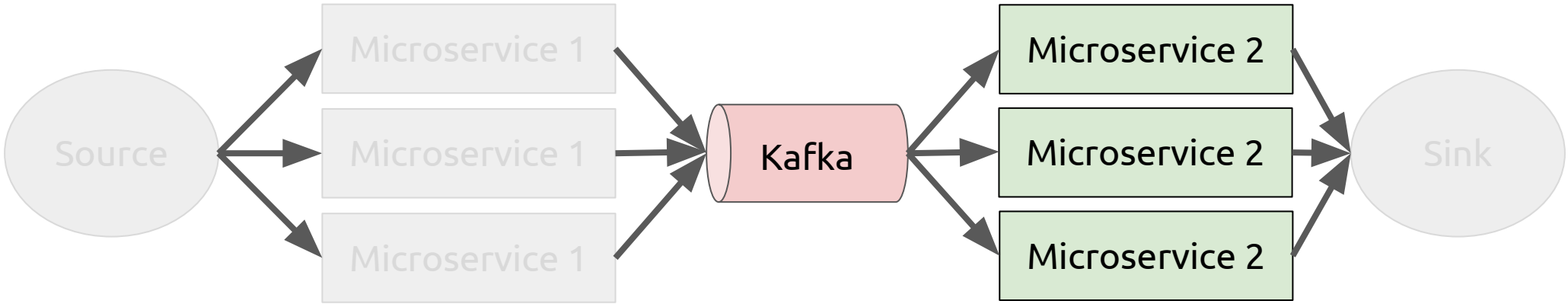
Elastic

- Linear scalability

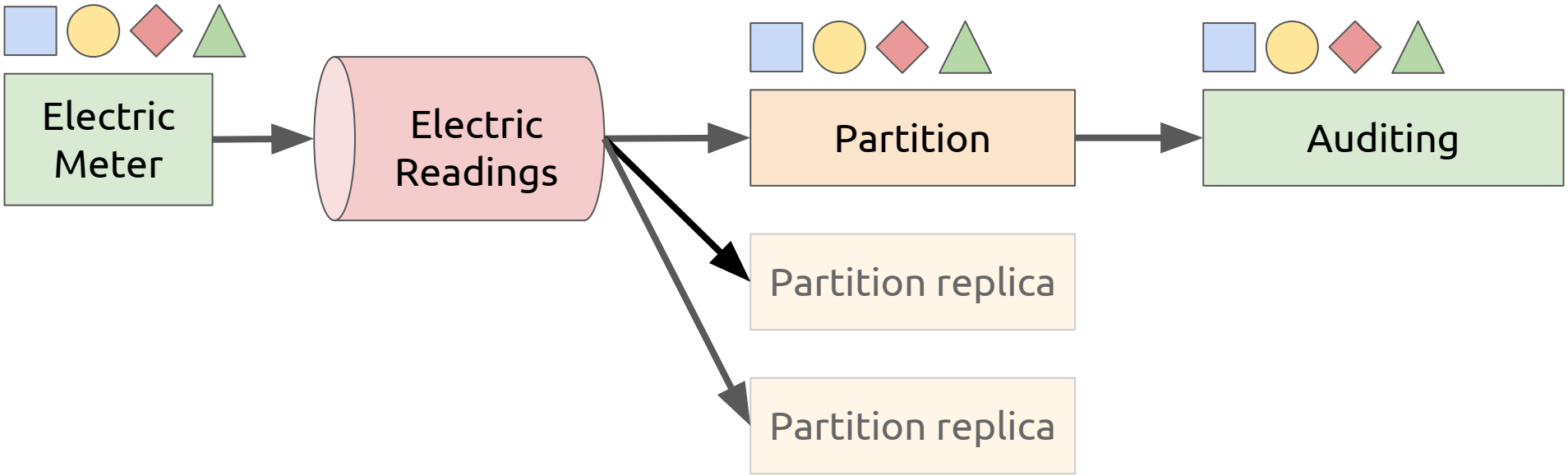
Message Driven

- Key-value messages

Kafka consumer patterns



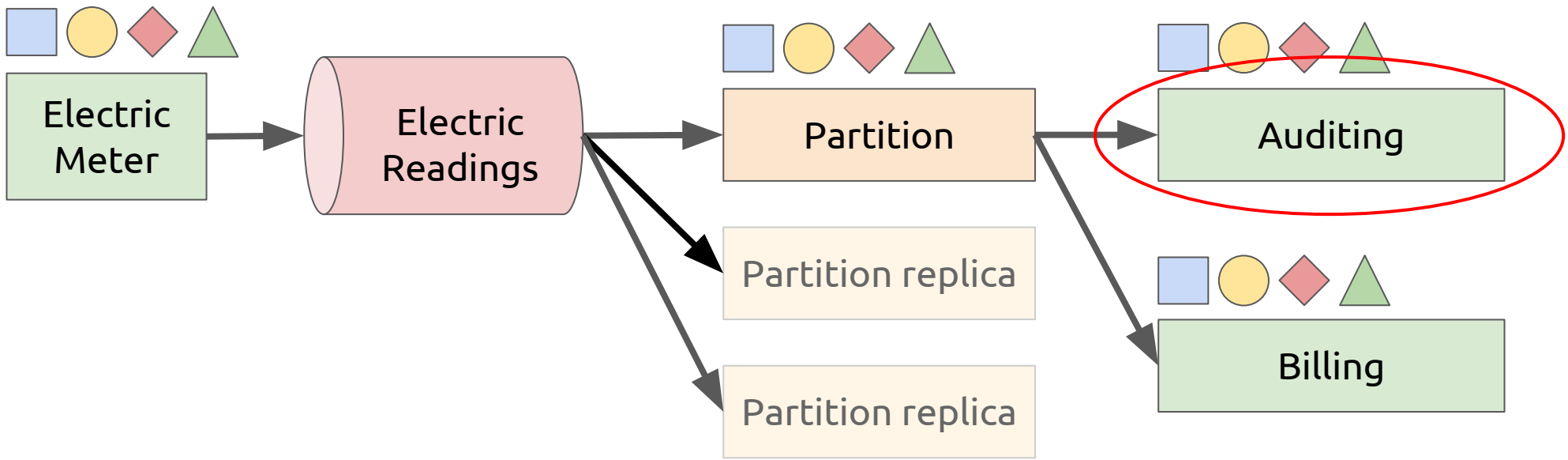
Simple message queue



Kafka Terminology:

- Partition Count: **1**

Simple message queue - fanout

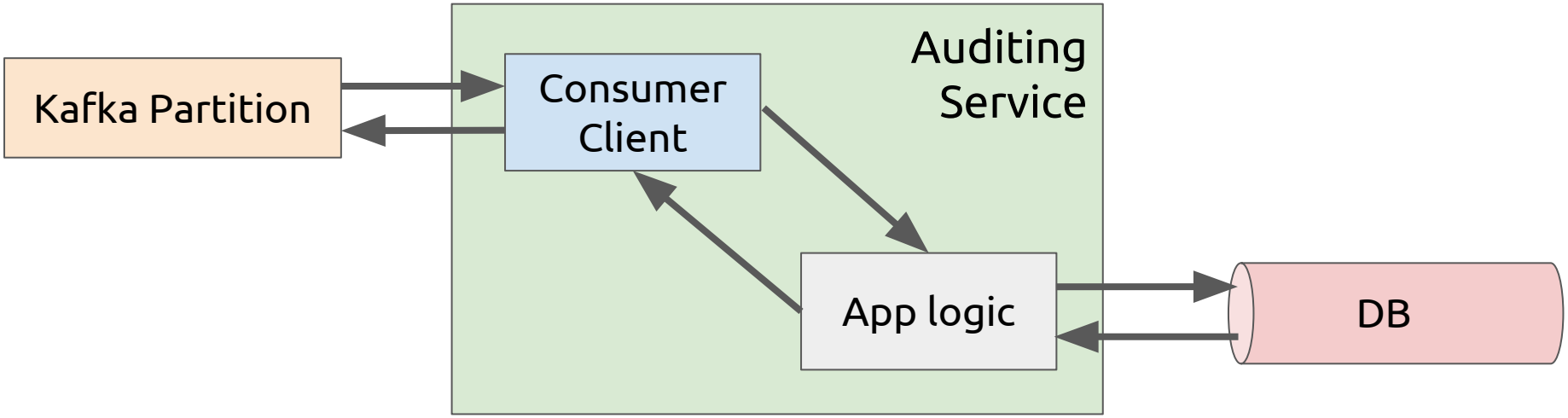


Kafka Terminology:

- Partition Count: **1**
- **Multiple Consumer Groups**

Simple message queue - consumer

1. Consume a batch of messages from Kafka
2. Process messages and send results to wherever necessary (e.g. another Kafka topic)
3. Confirm delivery to Kafka

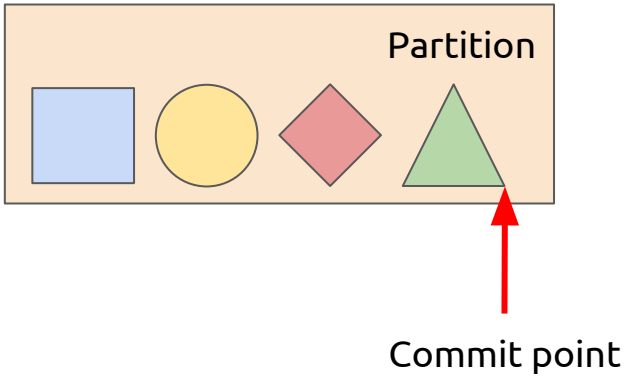


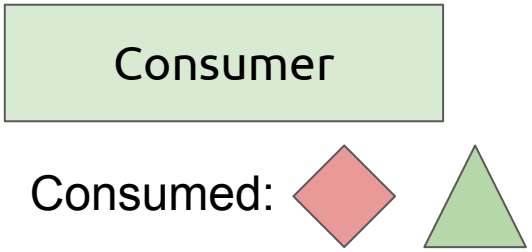
Kafka Terminology:

- Commit Mode: **Manual**

Kafka: message confirmation

- Messages confirmed by offset (not individually)



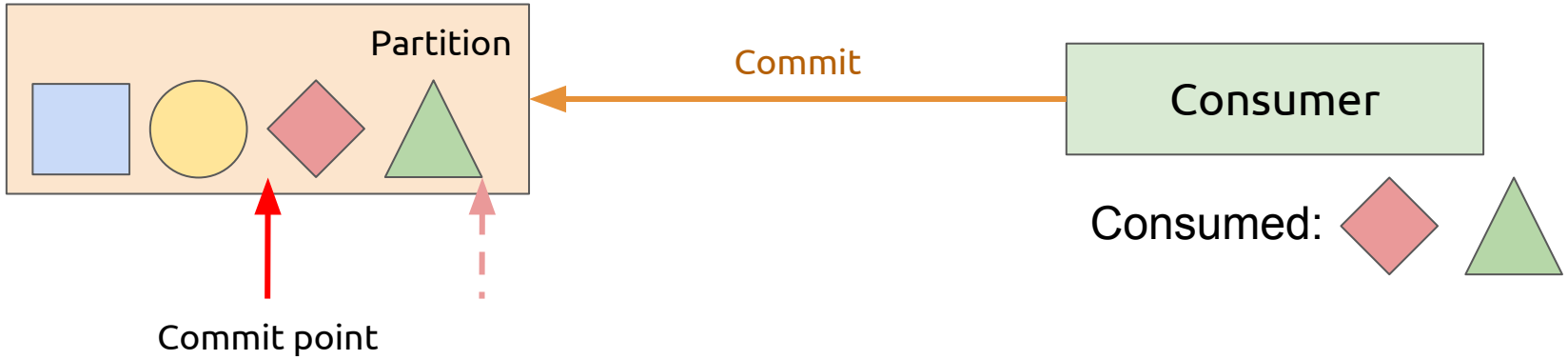


Kafka Terminology:

- Commit Mode: **Manual**

Kafka: message confirmation

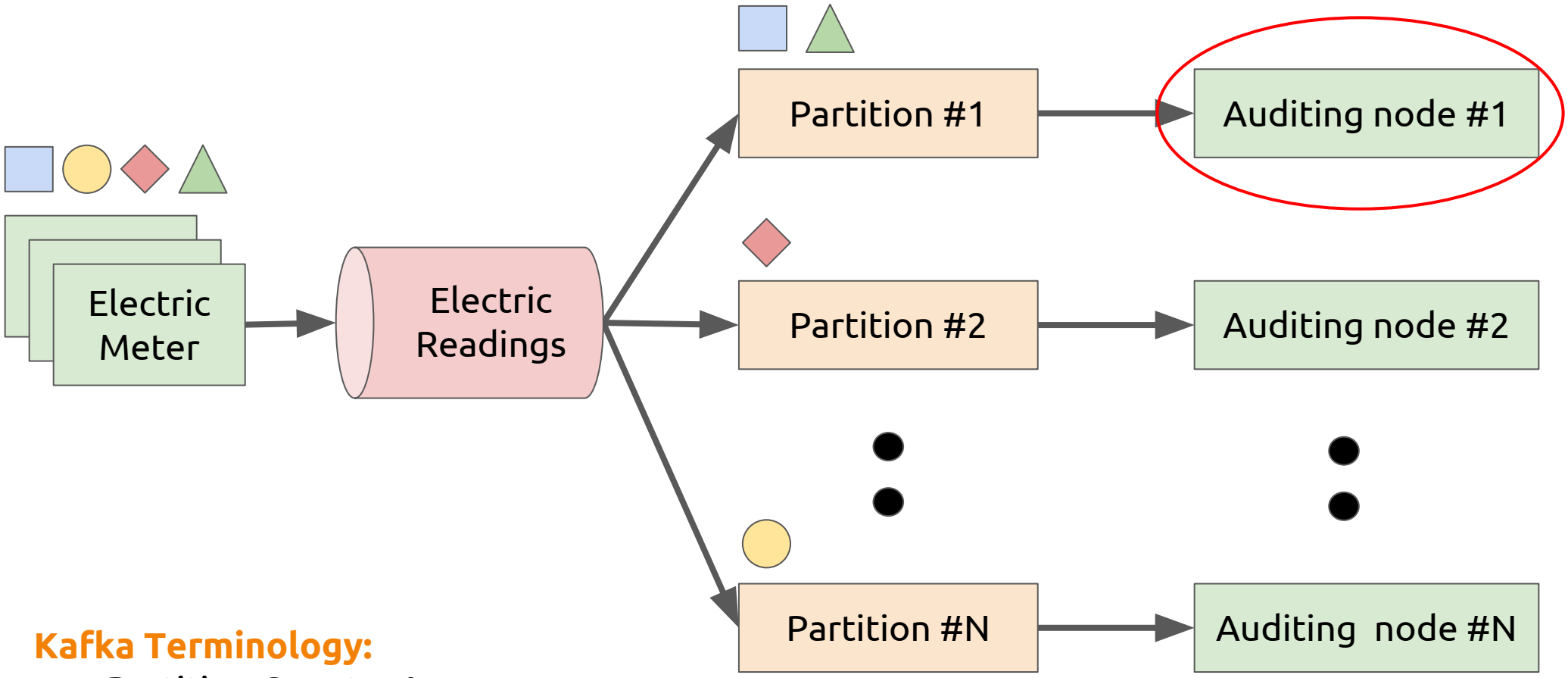
- Messages confirmed by offset (not individually)



Kafka Terminology:

- Commit Mode: **Manual**

Parallel workers

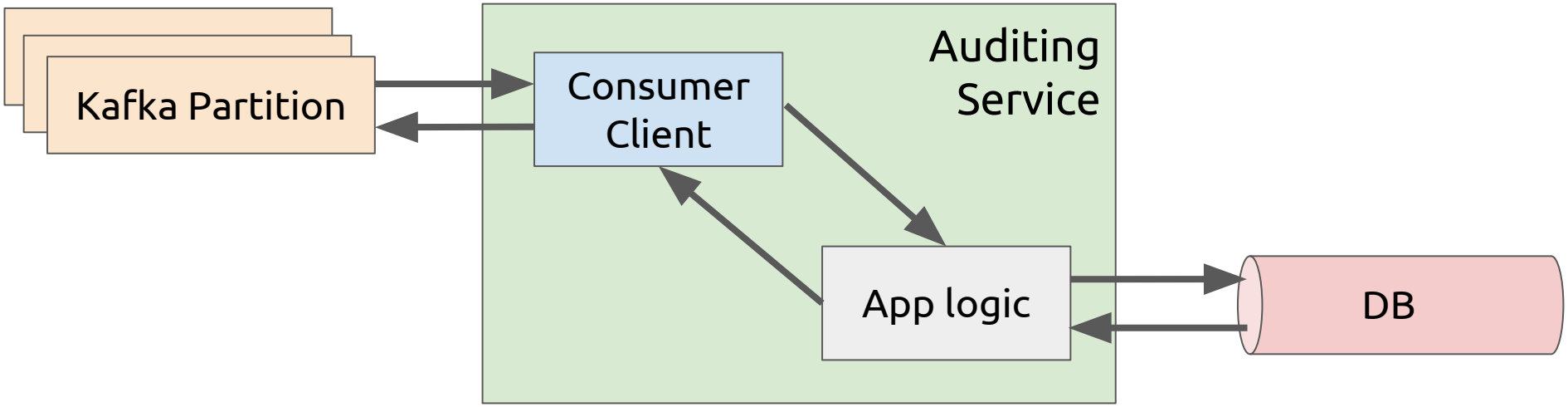


Kafka Terminology:

- Partition Count: **>1**
- **Single Consumer Group**

Consumer for parallel processing

- Same arrangement from consumer perspective

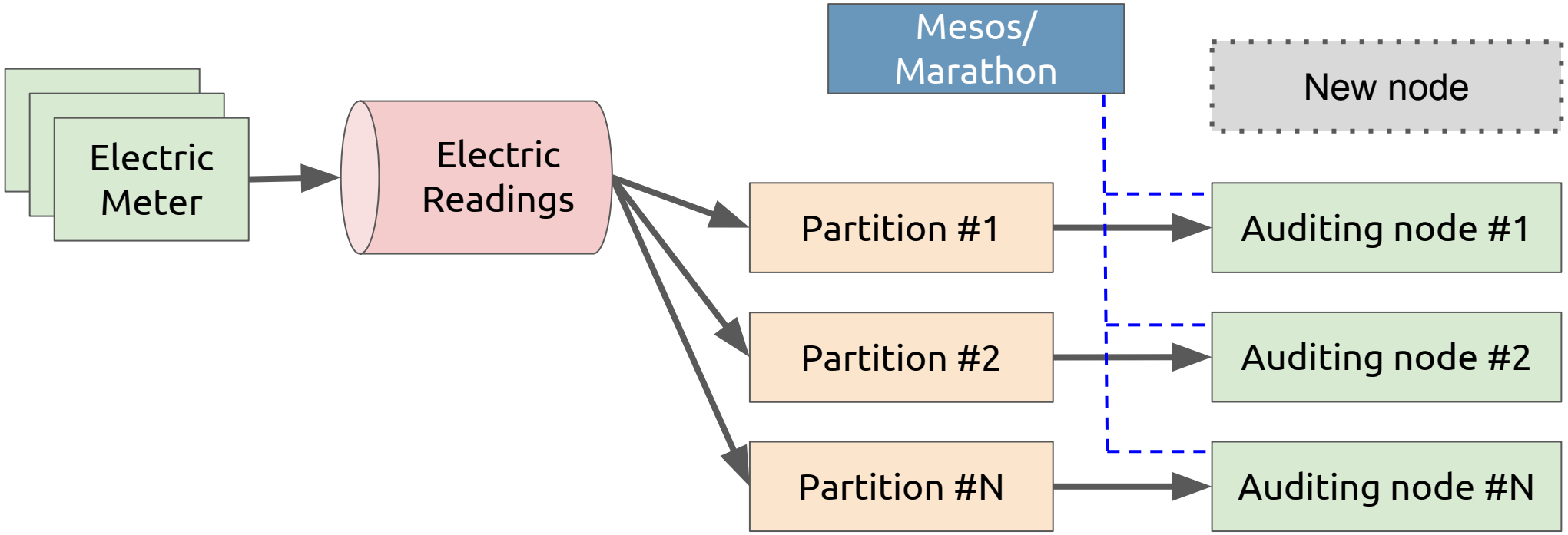


Kafka Terminology:

- Partition Count: **>1**
- Commit Mode: **Manual**

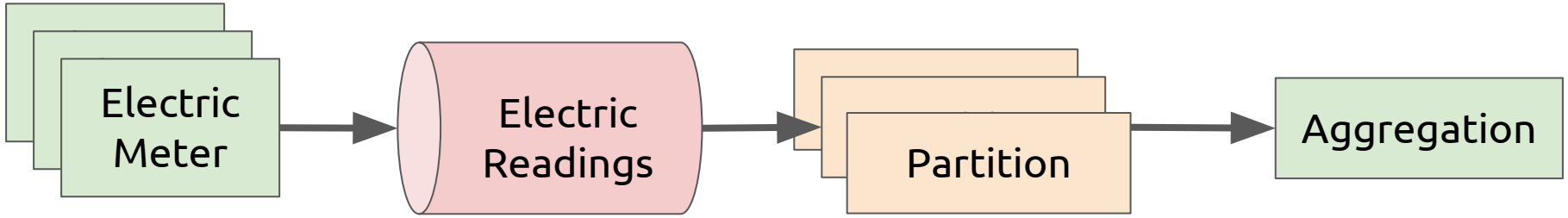
Orchestration

- Provide Scaling Capability
- Restart or replace failed nodes



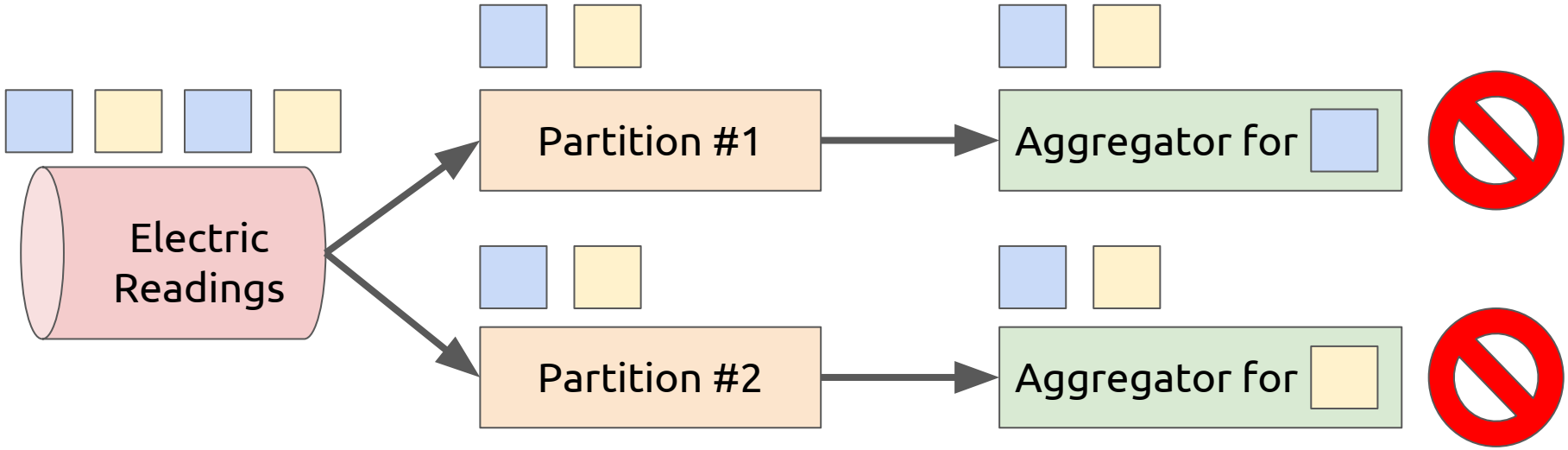
Stateful Processing

- Example:
  Average electricity consumption per meter for the last hour



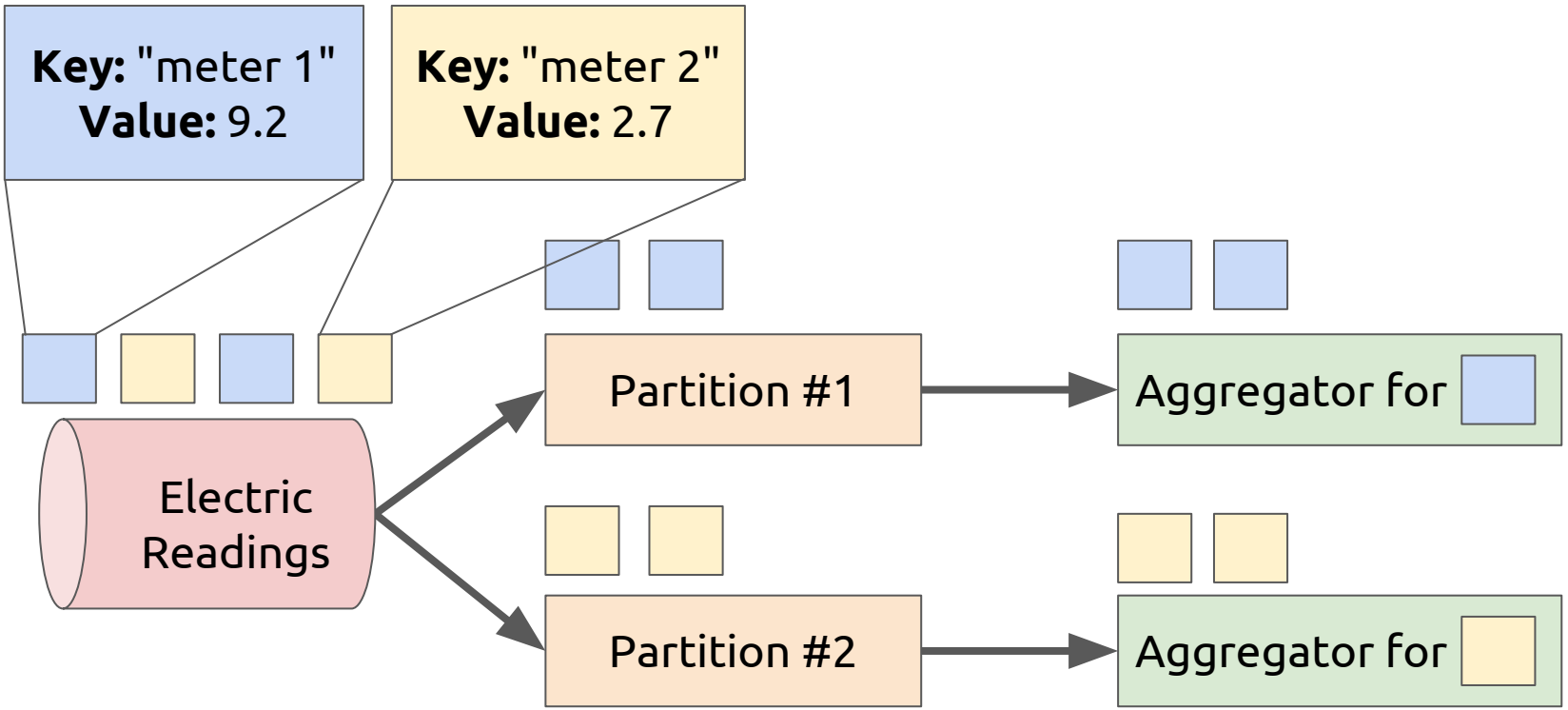
Stream and state

- Data locality



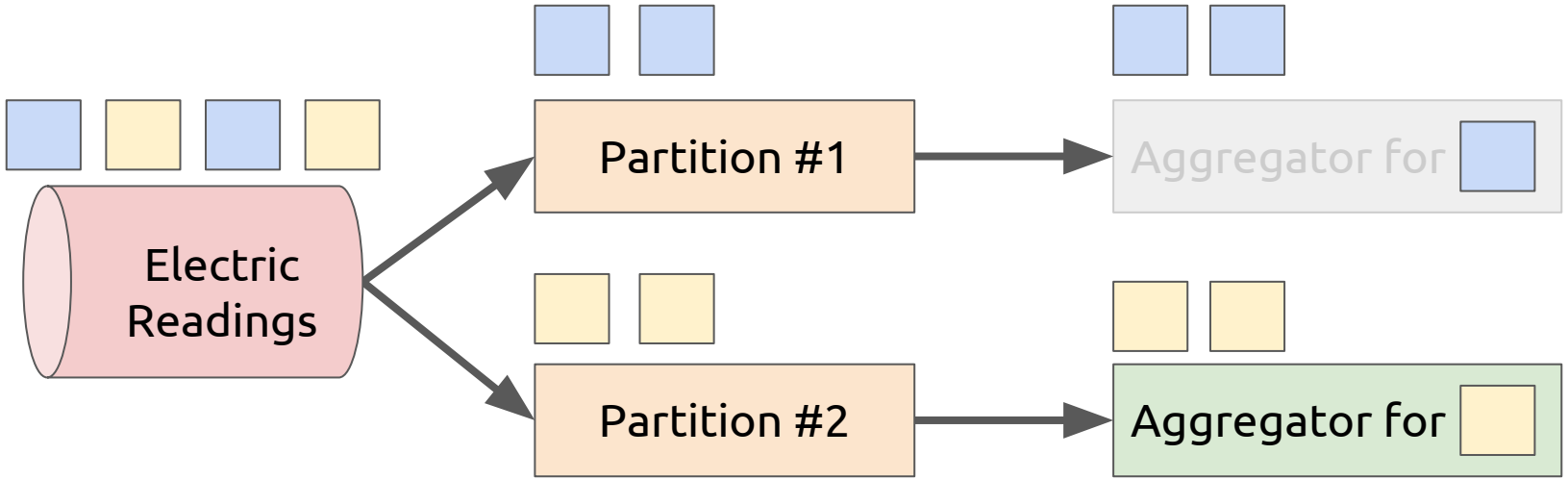
Stream and state

- Data locality



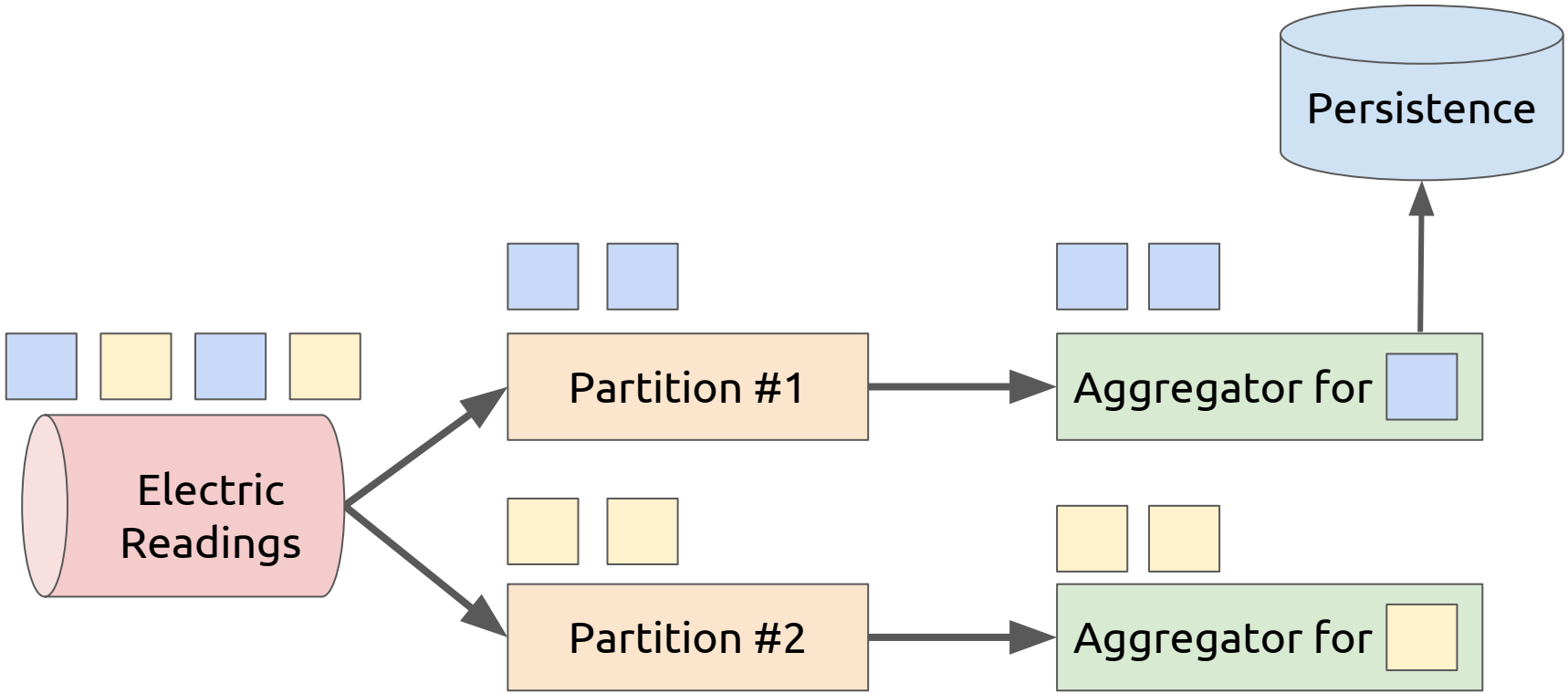
Fault tolerance

- State persistence and recovery

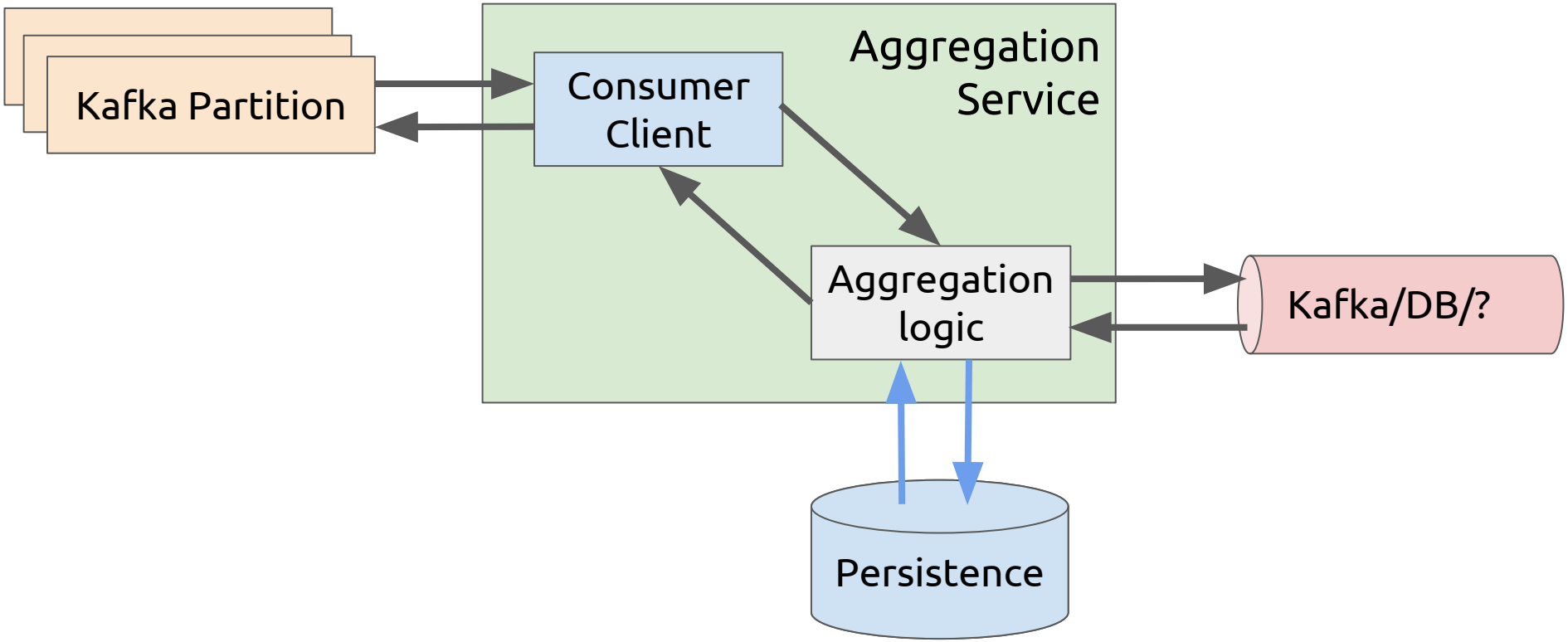


Fault tolerance

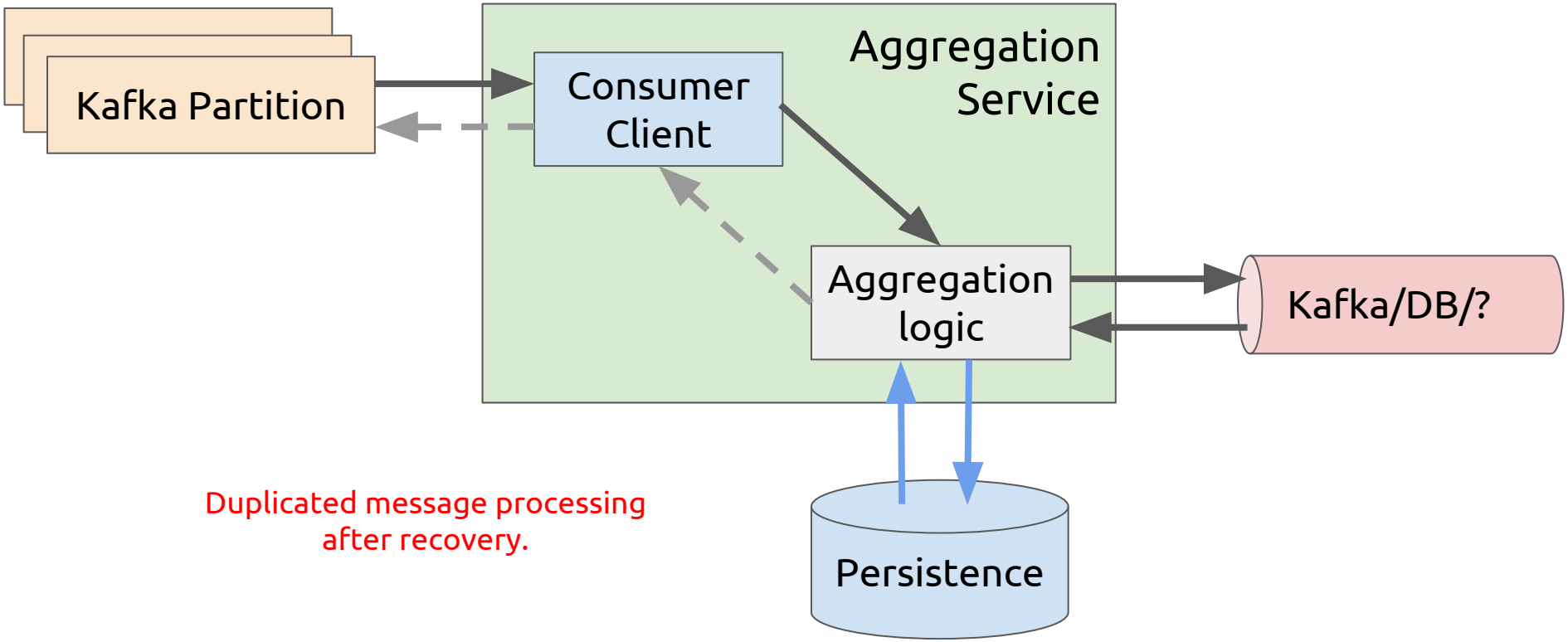
- State persistence and recovery



Stateful Processing app

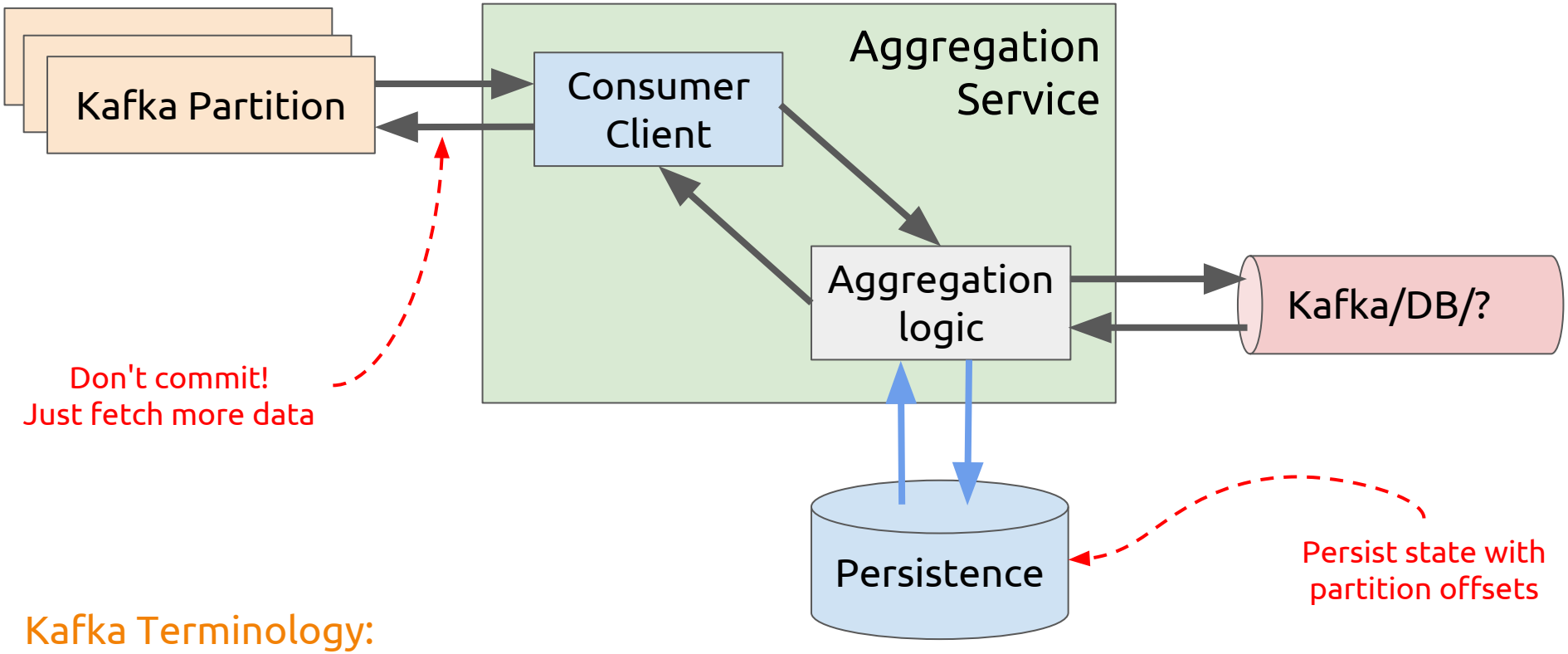


Stateful Processing app



Duplicated message processing after recovery.

Stateful Processing app

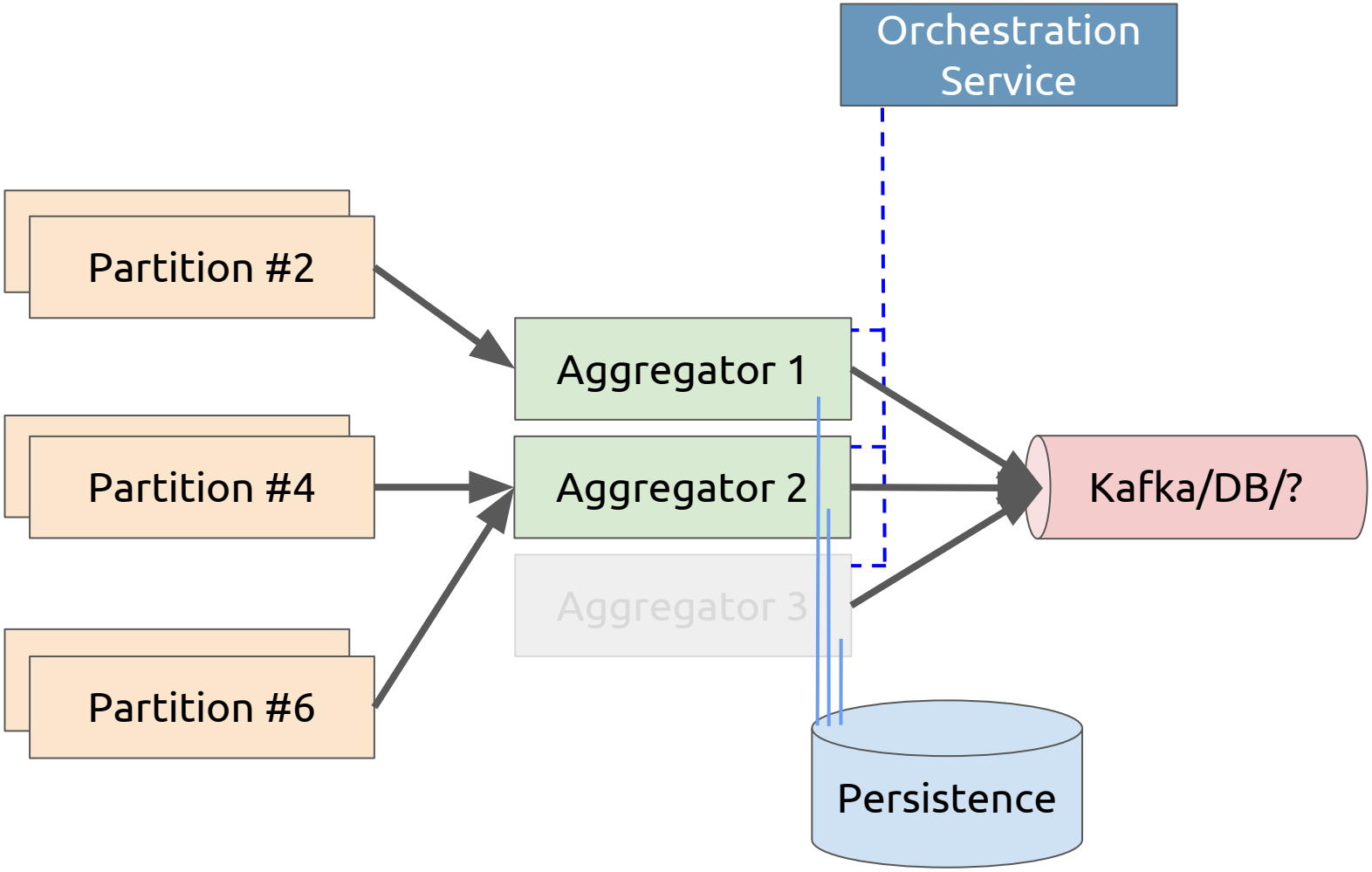


Kafka Terminology:

- Commit Mode: **Self Managed Offsets**

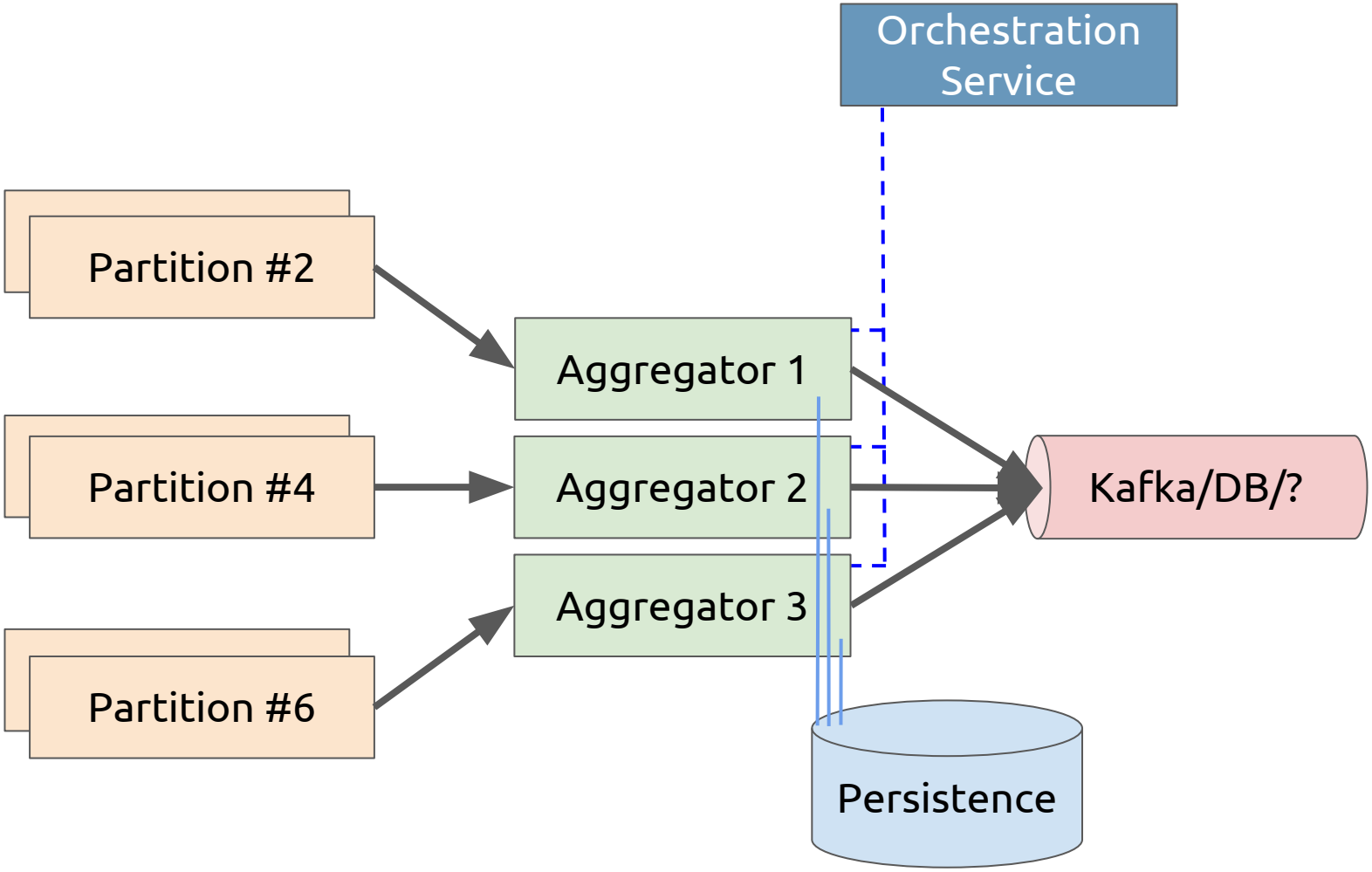
Stateful Processing architecture

- Dynamic partition assignment
- Shared Persistence for State



Stateful Processing architecture

- Dynamic partition assignment
- Shared Persistence for State



Streaming Patterns

Single Partition Topic

- Strong ordering guarantees
- Limited failure recovery
- Scalability is limited

Multi Partition Topic

- Parallel processing
- Limited ordering guarantees
- Kafka managed processing state

Fanout

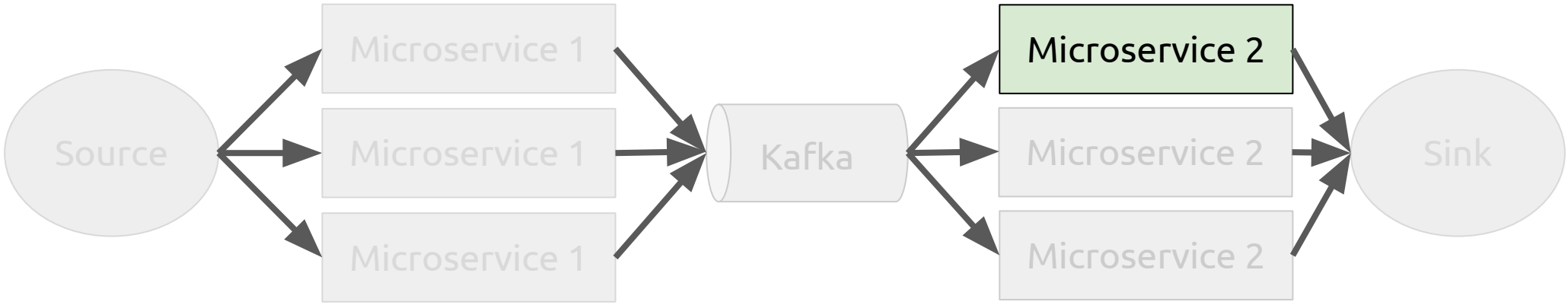
- Independent consumer groups

Stateful Processing

- Self-managed processing state

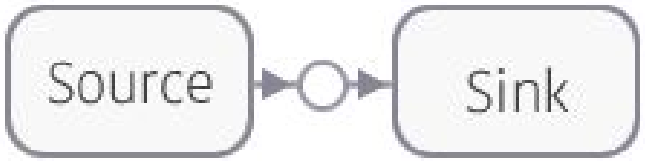
Kafka libraries

- Kafka client support in many languages
- Scala, Java, C
- C bindings -> Haskell, OCaml, Python etc.



Reactive Streaming APIs

- Similar paradigm as in real-time streaming platforms
- Reactive Kafka
  - Based on Akka Reactive Streams API
  - Scala + Java
  - Developed by Akka team
- Kafka Streams
  - Official streaming API for Kafka
  - Java
  - Developed by Confluent



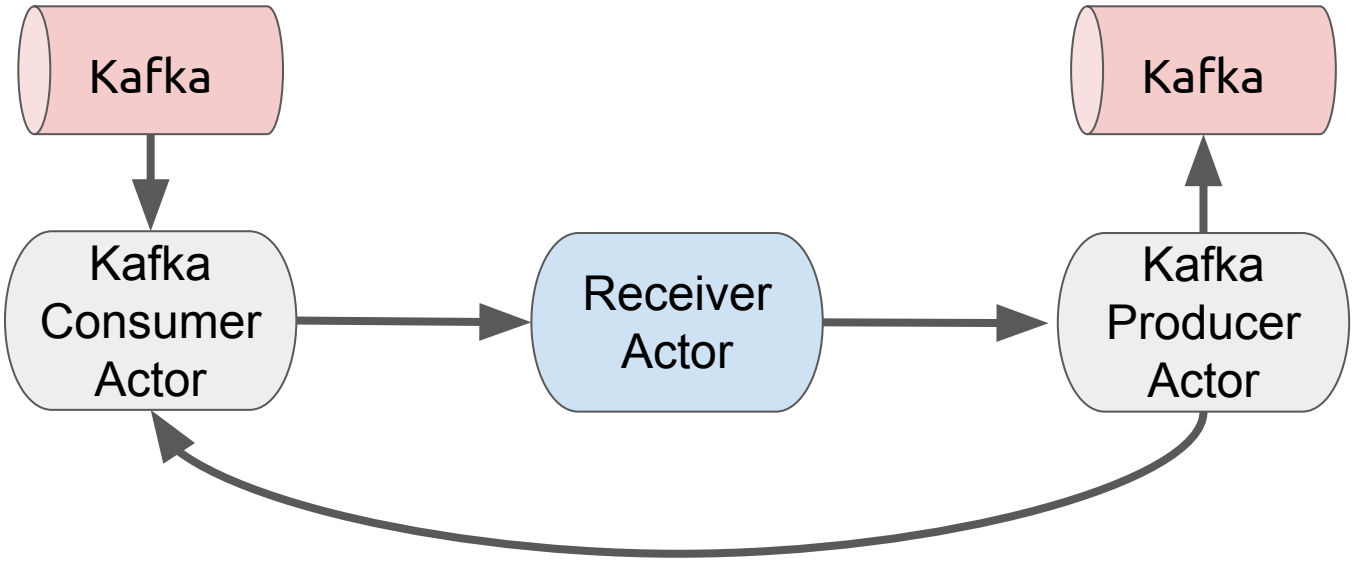
scala-kafka-client

- Kafka client developed for Scala
- Async and non-blocking
- Built on top off the official Java driver
- Easy API with high performance

/cakesolutions /scala-kafka-client

scala-kafka-client

- Leverage extensive Akka feature set
- Processing logic implemented using Actor Model



/cakesolutions /scala-kafka-client

Summary

- Leverage Microservice based techniques.
- Streaming topologies can be varied and complex
  - Many use-cases fall under a small set of consumer patterns.
- Challenges around scalable and reactive data pipelines
- Kafka provides first-class support for reactive streaming to your applications.
- Stateful processing remains a challenging area.

We didn’t discuss...

- Data serialisation
- Application rolling updates
- Complex streaming topologies

Questions?

CAKESOLUTIONS
ENTERPRISE SOFTWARE SERVICES

/cakesolutions /scala-kafka-client

@cakesolutions

+44 845 617 1200

enquiries@cakesolutions.net