KAFKA

introduction of kafka,

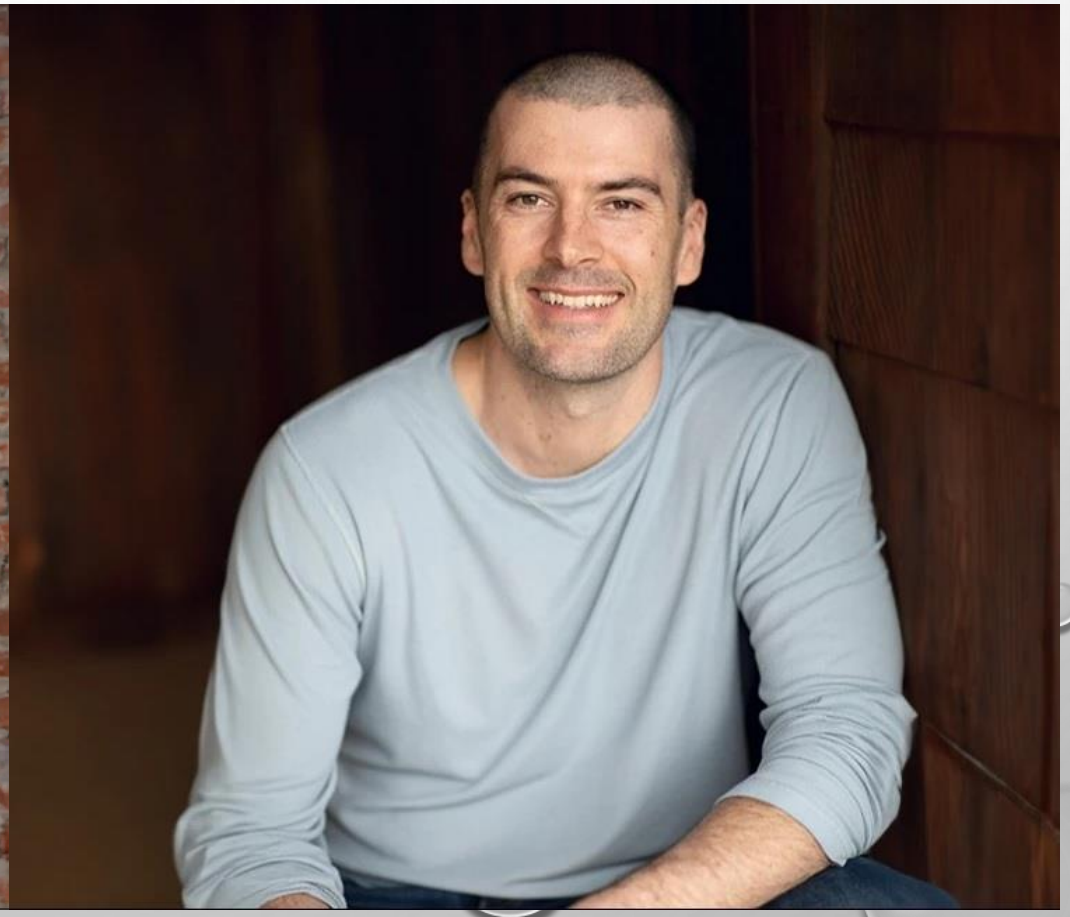
kafka stream and

kafka stateful & stateless stream

KAFKA HISTORY

- OPEN-SOURCE
- DEVELOPED BY THE ASF
- WRITE BY JAVA AND SCALA
- HIGH-THROUGHPUT
- SCALABLE
- MAINTAINABLE
- LOW-LATENCY PLATFORM FOR HANDLING REAL-TIME DATA FEEDS
- KAFKA CAN CONNECT TO EXTERNAL SYSTEMS(KAFKA CONNECTION)
- USES A BINARY TCP-BASED PROTOCOL
- KAFKA STREAMS LIBRARIES FOR STREAM PROCESSING APPLICATIONS

- KAFKA WAS ORIGINALLY DEVELOPED AT LINKEDIN FROM 2011(13 YAERS)
- APACHE KAFKA IS BASED ON THE COMMIT LOG
- JAY KREPS & NEHA NARKHEDE & JUN RAO IS COCREATER KAFKA



- APACHE LICENSE
- IT ALLOWS USERS TO USE THE SOFTWARE FOR ANY PURPOSE, TO DISTRIBUTE IT, TO MODIFY IT, AND TO DISTRIBUTE MODIFIED VERSIONS OF THE SOFTWARE UNDER THE TERMS OF THE LICENSE, WITHOUT CONCERN FOR ROYALTIES

THE APACHE™ SOFTWARE FOUNDATION

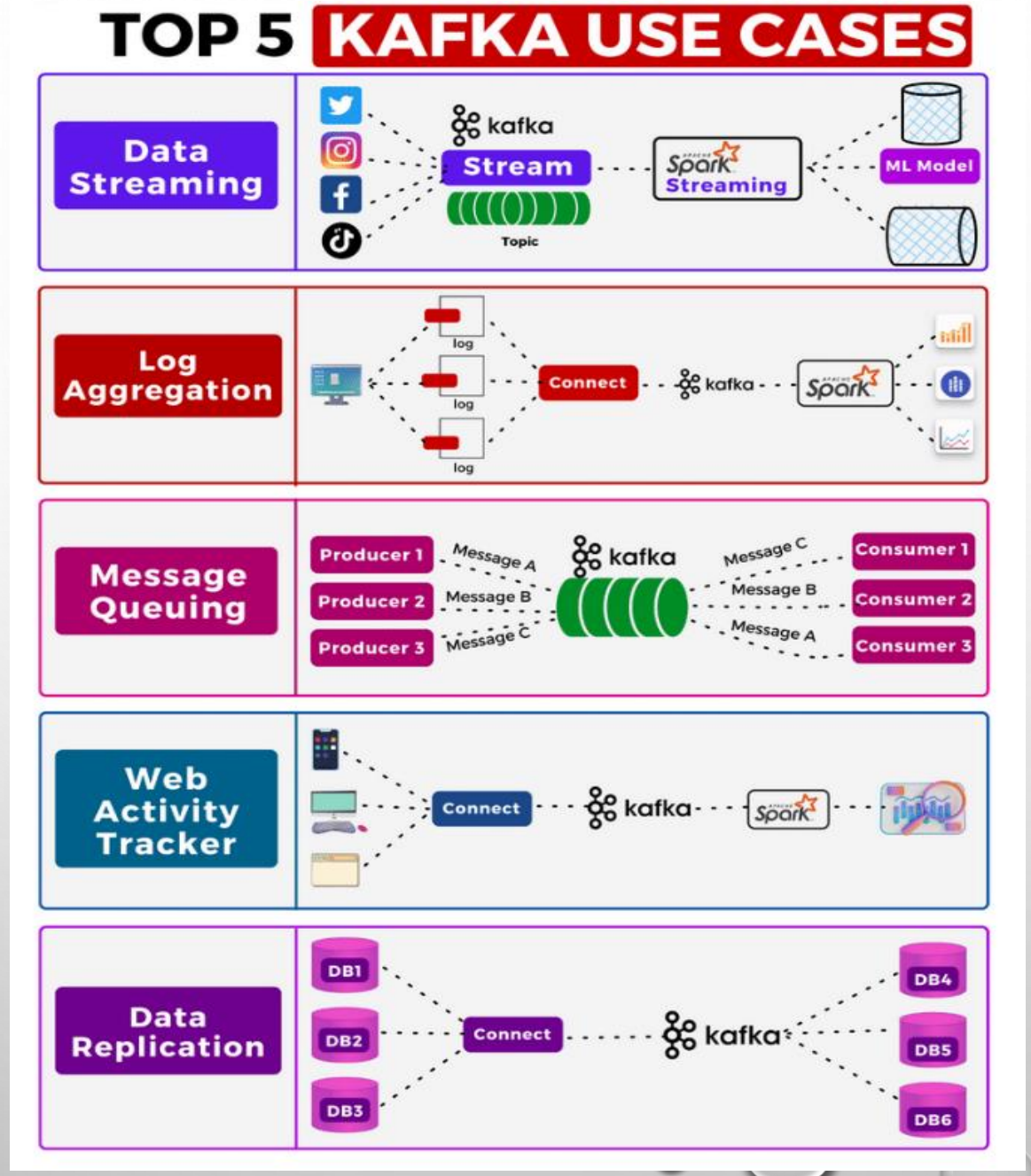
- APACHE KAFKA REPOSITORIES

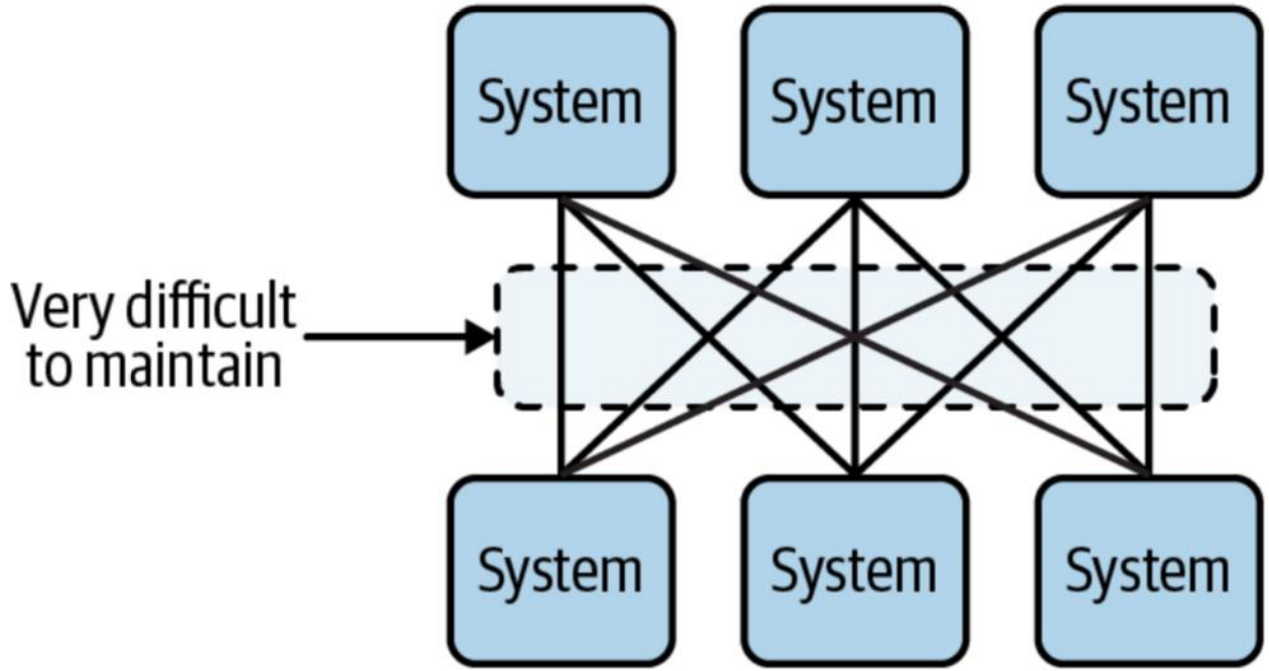
https://github.com/orgs/apache/repositories

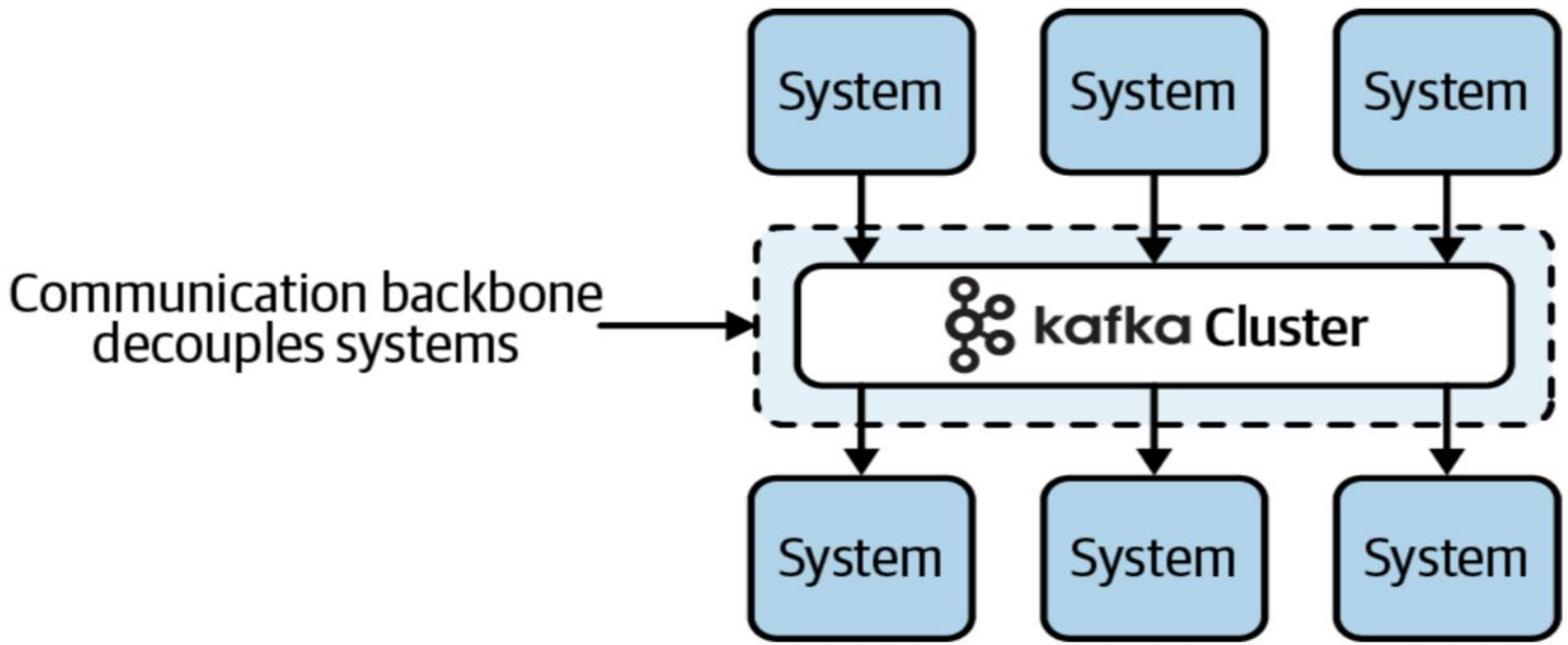
- APACHE KAFKA PMC MEMBER

https://kafka.apache.org/committers

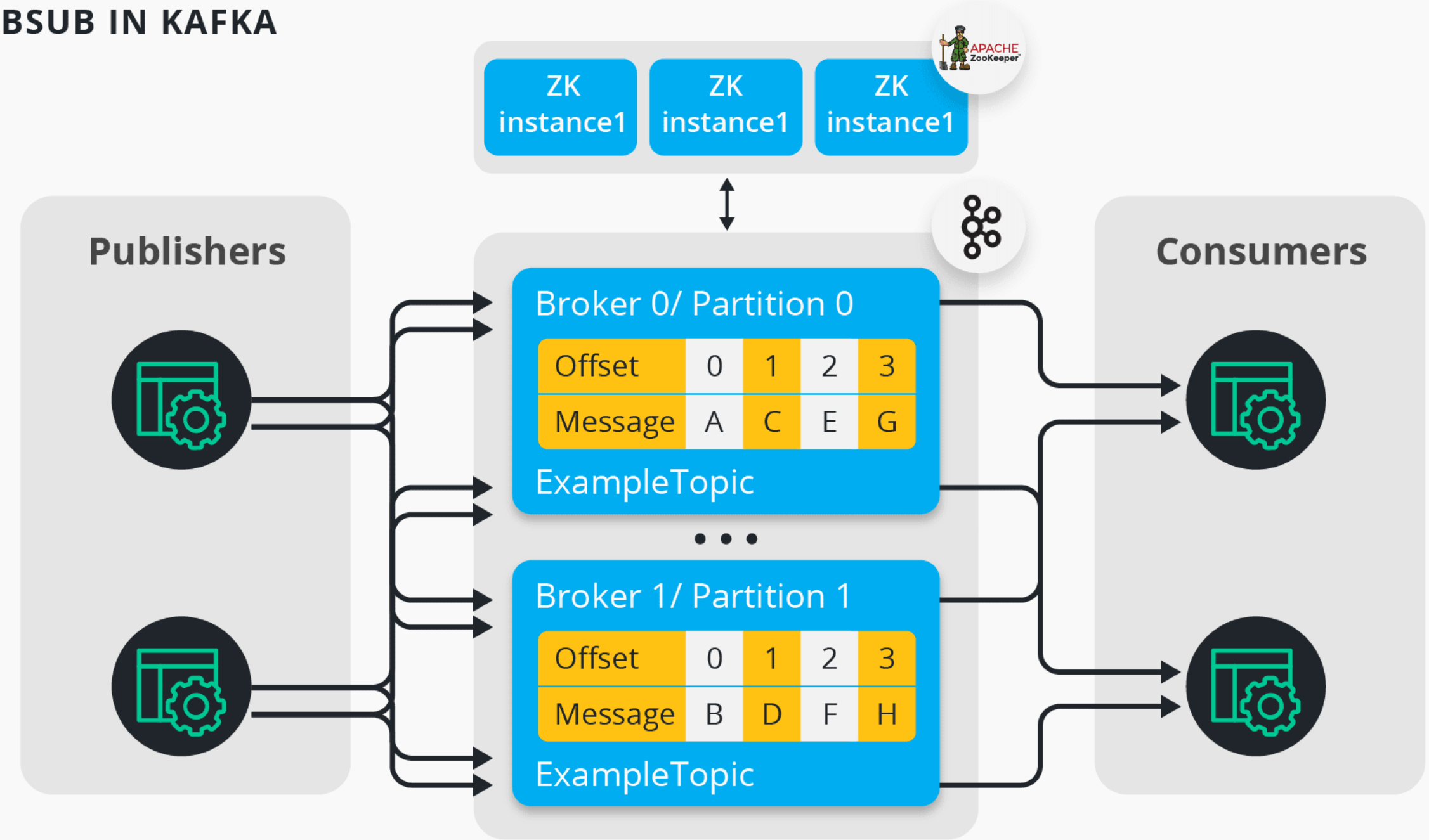
- KAFKA USE CASE

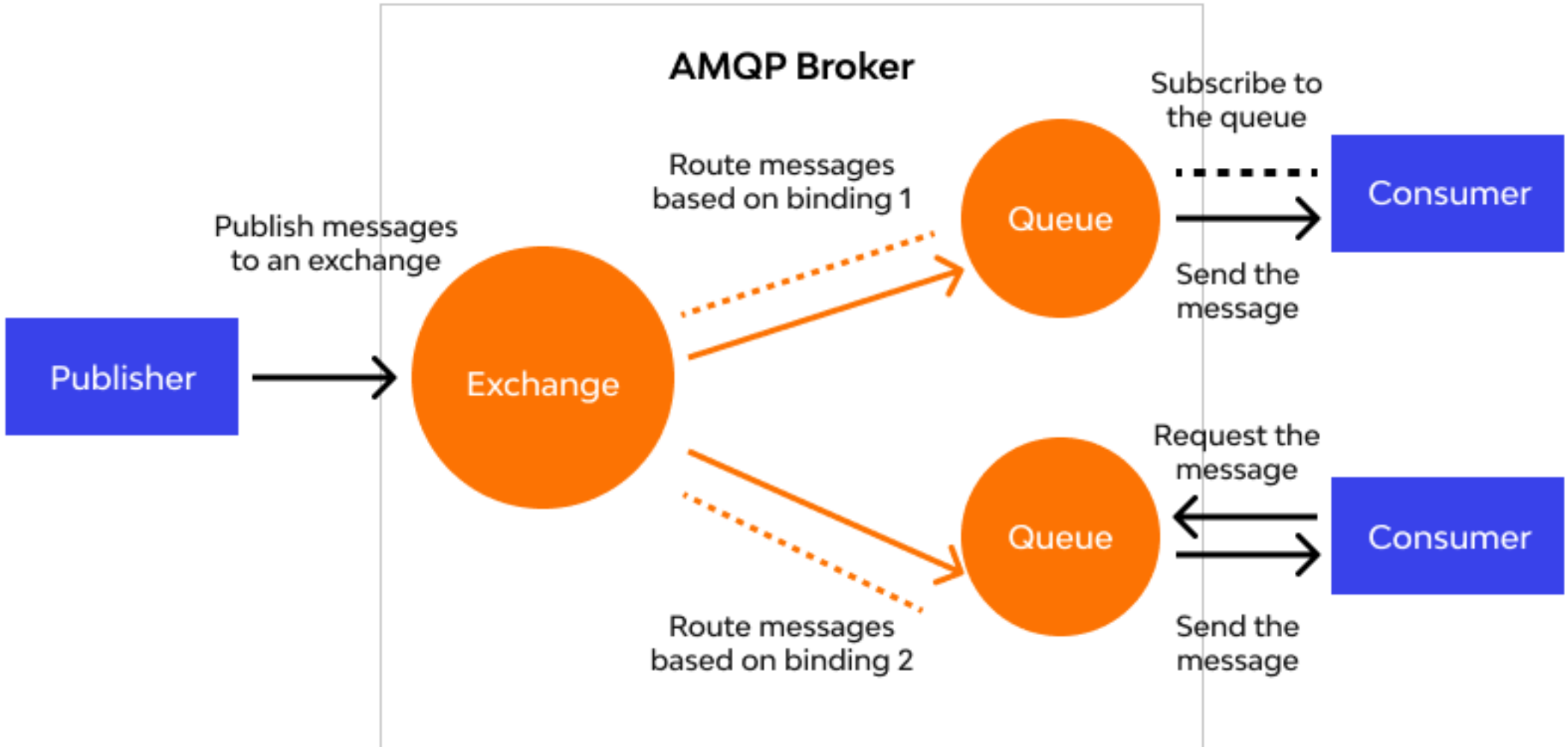



System
System
System
Very difficult
to maintain
System
System
System

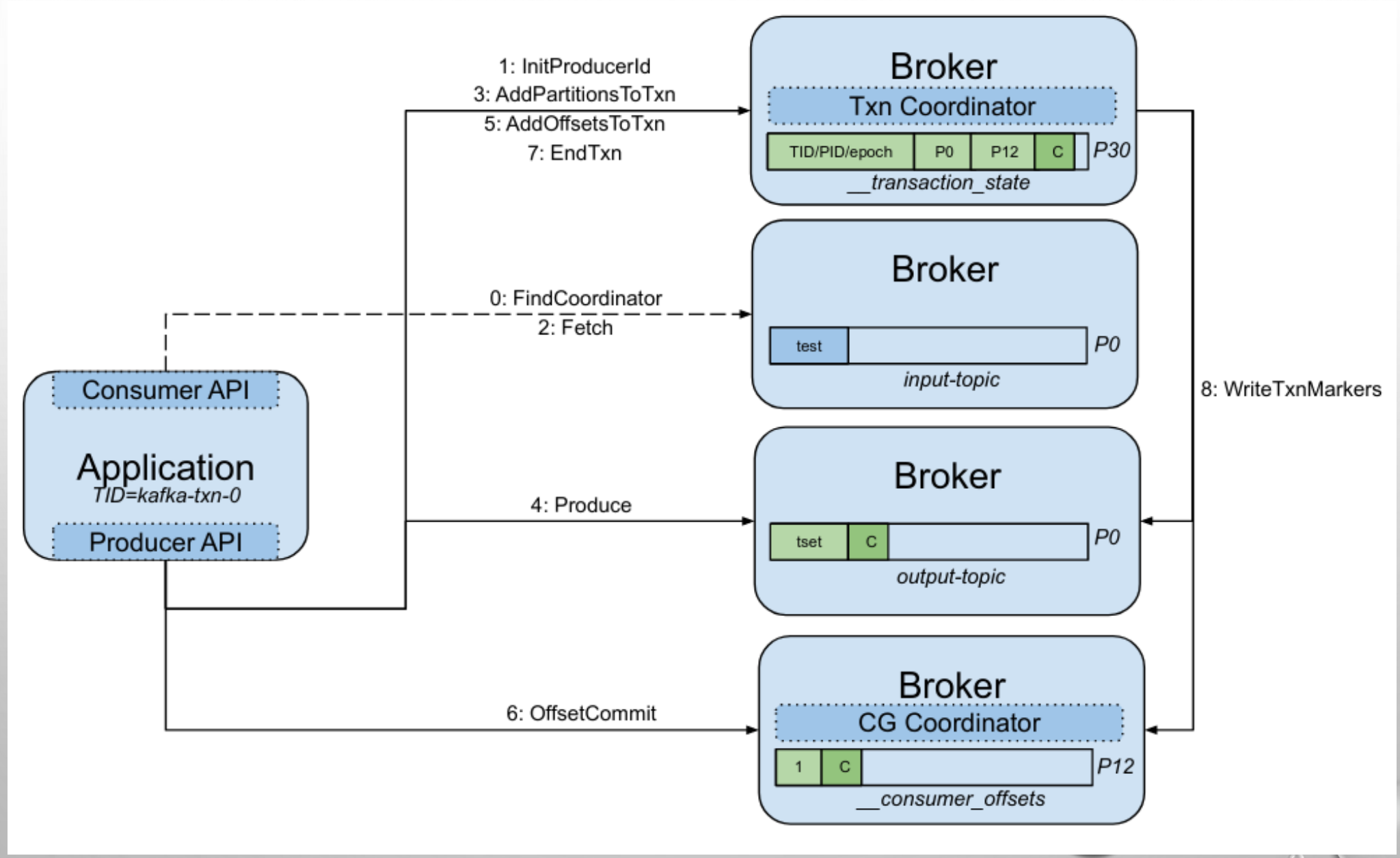

System
System
System
Communication backbone
decouples systems
kafka Cluster
System
System
System

PUBSUB IN KAFKA

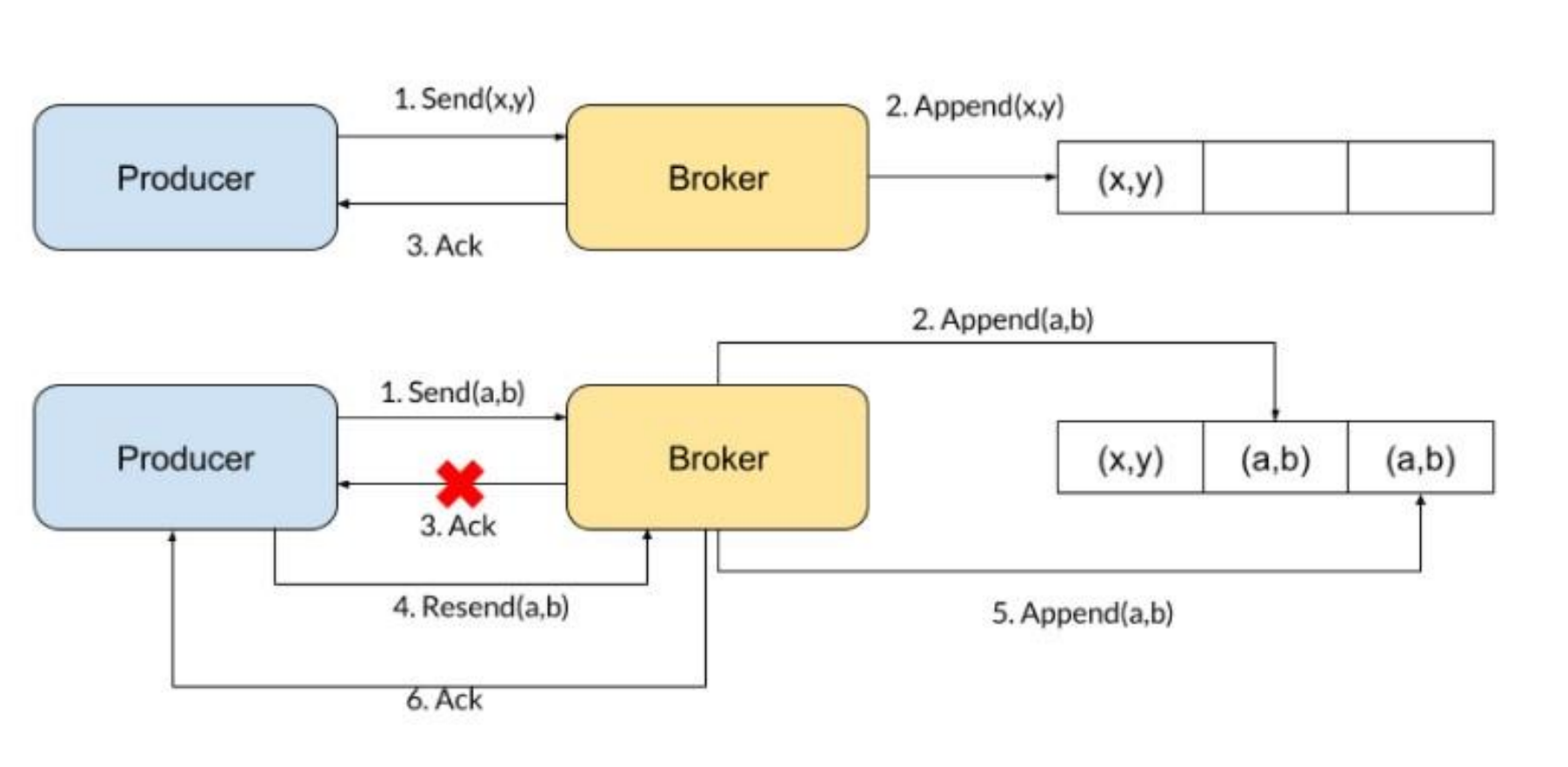



AMQP Broker
Publisher
Publish messages
to an exchange
Exchange
Route messages
based on binding 1
Route messages
based on binding 2
Queue
Queue
Subscribe to
the queue
Send the
message
Request the
message
Send the
message
Consumer
Consumer

EOS

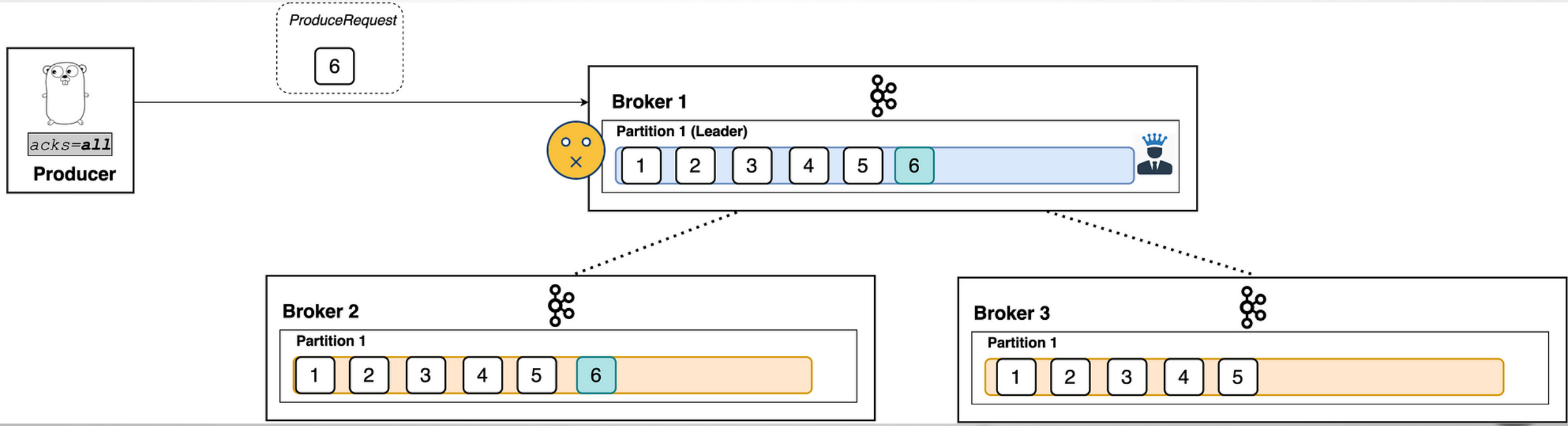

1: InitProducerId
3: AddPartitionsToTxn
5: AddOffsetsToTxn
7: EndTxn
Broker
Txn Coordinator
TID/PID/epoch
P0
P12
C
P30
__transaction_state
0: FindCoordinator
2: Fetch
Broker
test
P0
input-topic
8: WriteTxnMarkers
Consumer API
Application
TID=kafka-txn-0
Producer API
4: Produce
Broker
tset
C
P0
output-topic
6: OffsetCommit
Broker
CG Coordinator
1
C
P12
__consumer_offsets

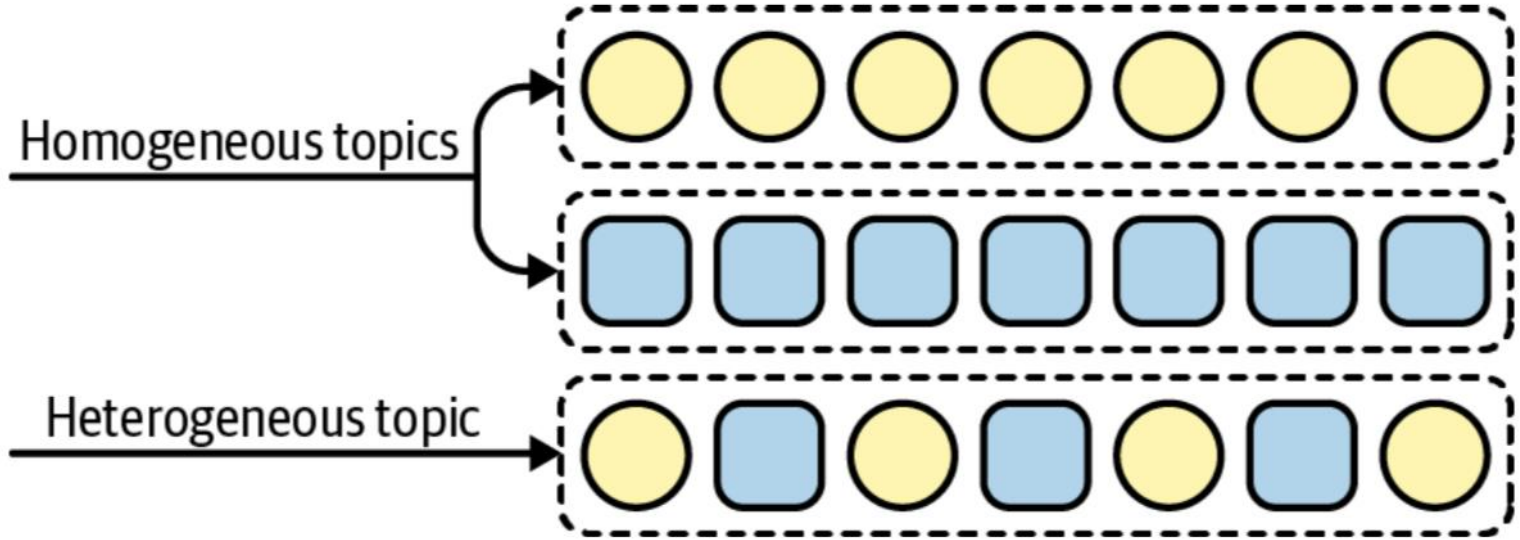
MESSAGE DELIVERY GUARANTEE IN KAFKA

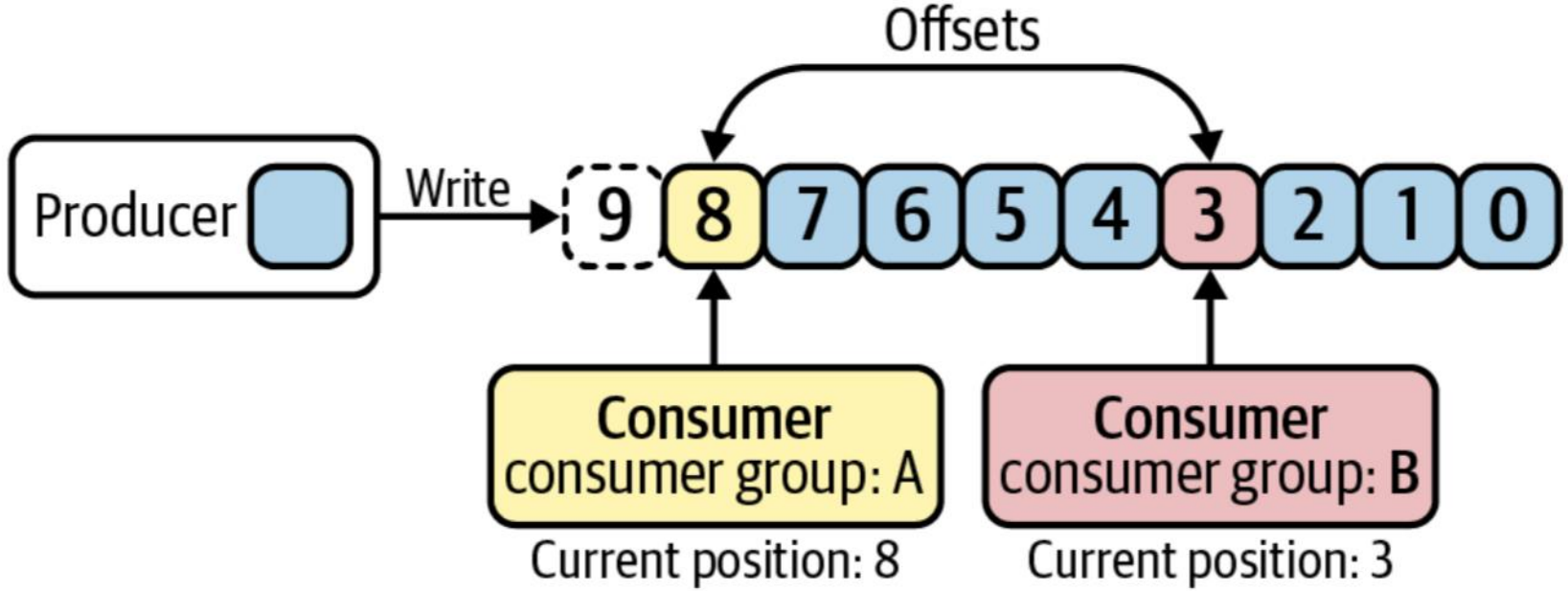


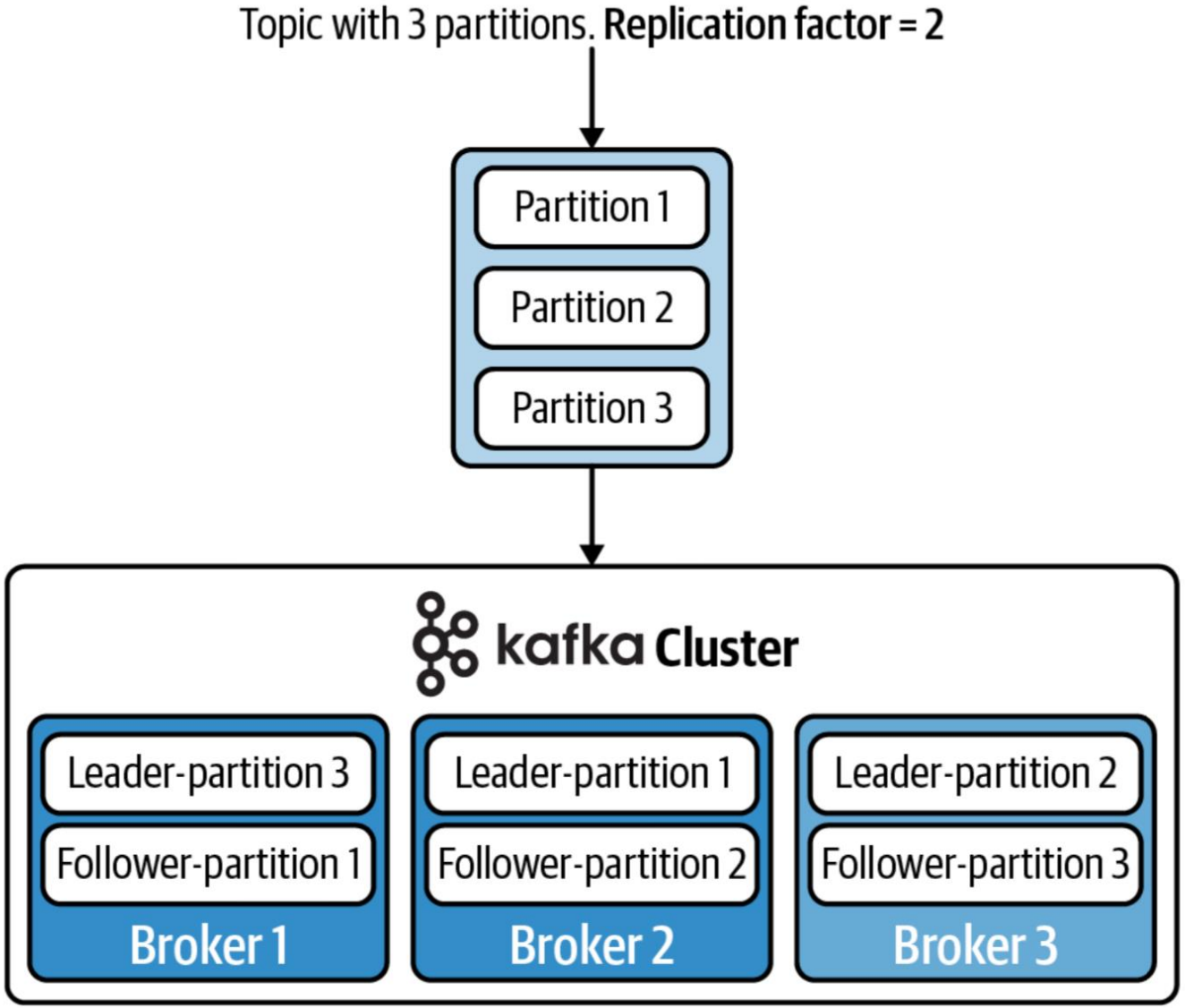
ACKS IN KAFAK

Acks Mean Acknowledgments. Kafka Producer Can Choose To Receive Acknowledgment Of Data Writes. Acknowledgment Is Also Known As Confirmation. And So There Are Three Confirmation Or Acknowledgment Modes Available In Apache Kafka Producer

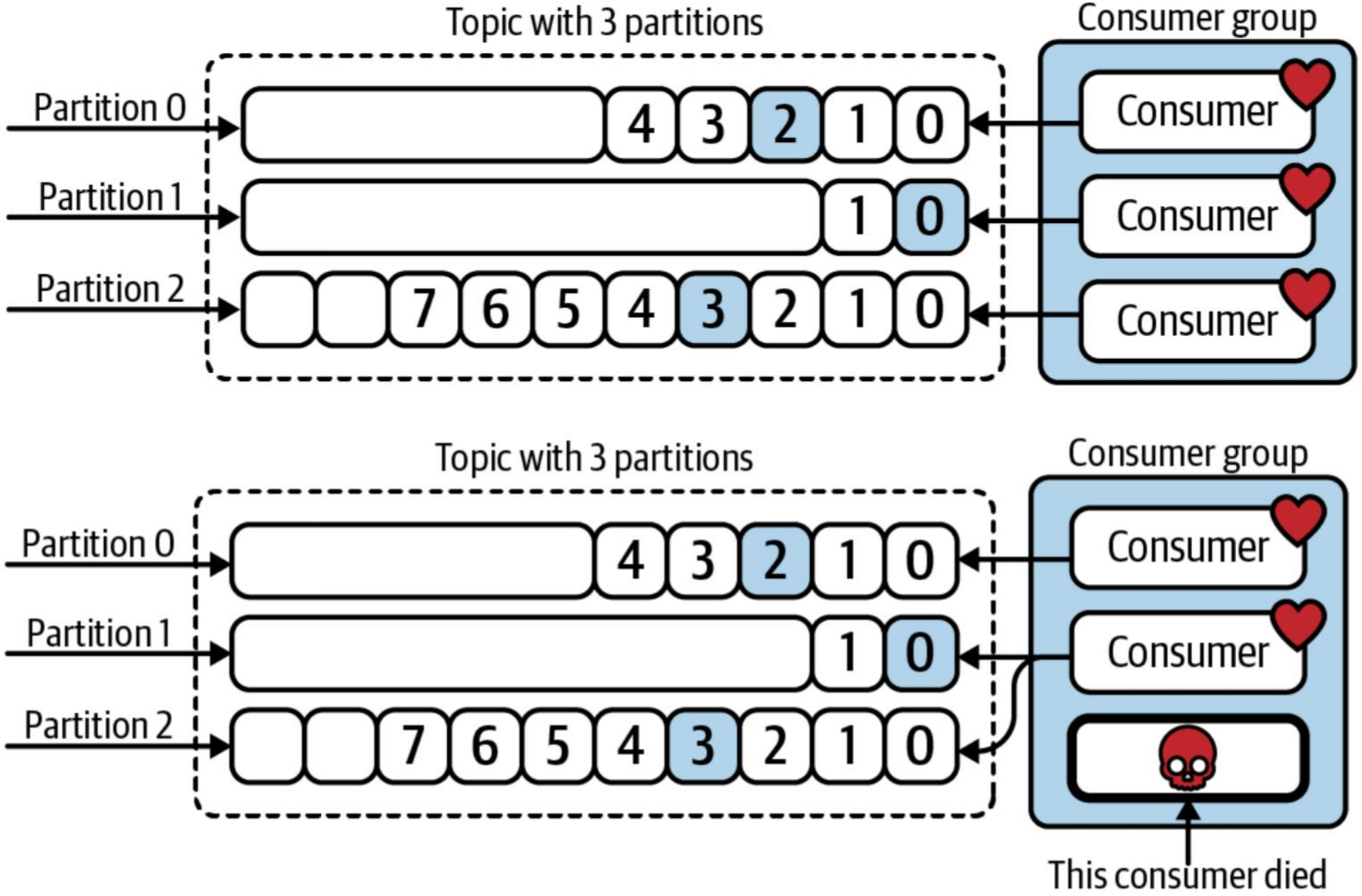



Homogeneous topics
Heterogeneous topic


Offsets
Producer
Write
9
8
7
6
5
4
3
2
1
0
Consumer
consumer group: A
Current position: 8
Consumer
consumer group: B
Current position: 3

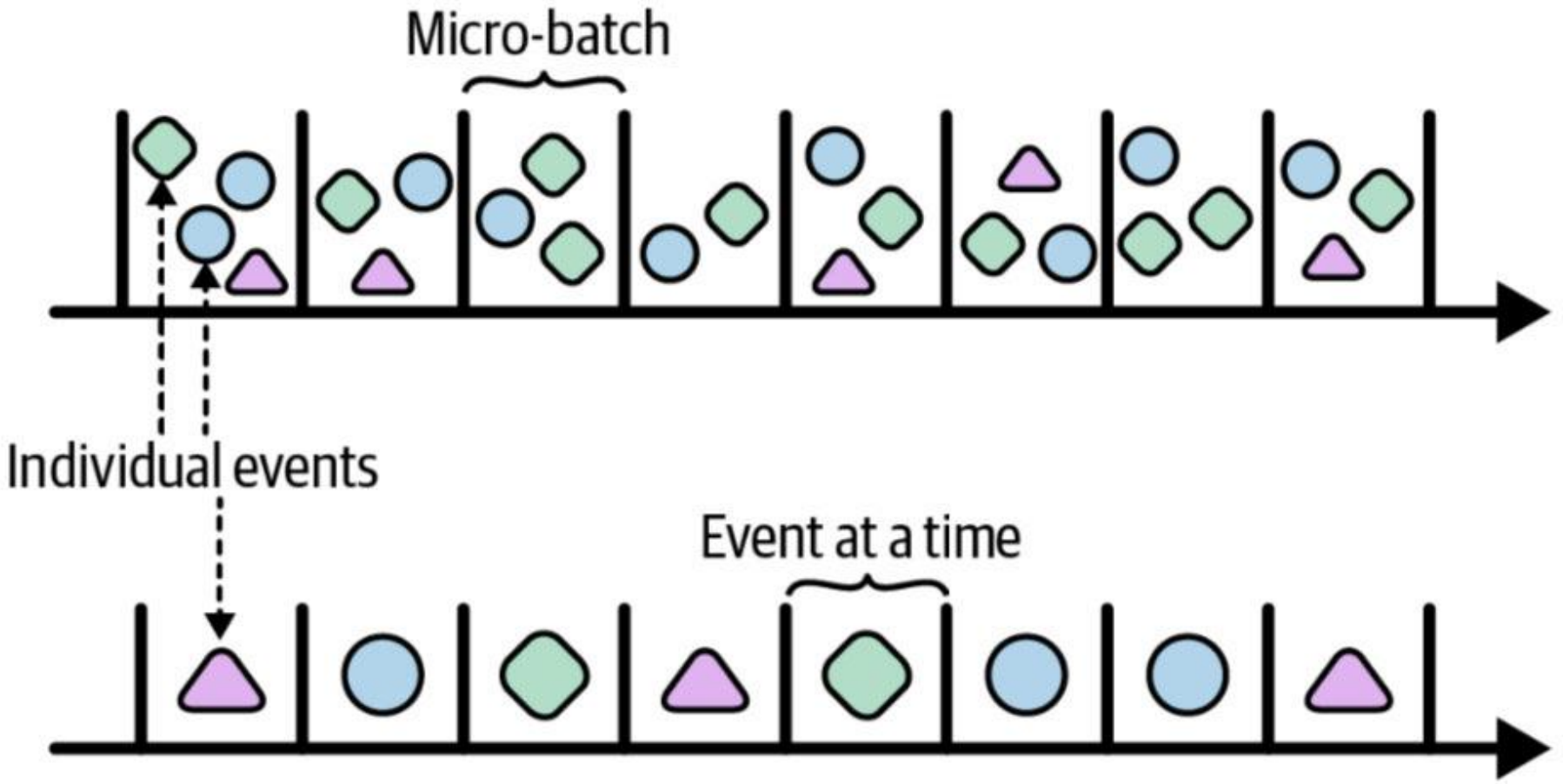

Topic with 3 partitions. Replication factor = 2
Partition 1
Partition 2
Partition 3
kafka Cluster
Leader-partition 3
Follower-partition 1
Broker 1
Leader-partition 1
Follower-partition 2
Broker 2
Leader-partition 2
Follower-partition 3
Broker 3

CONSUMER GROUP

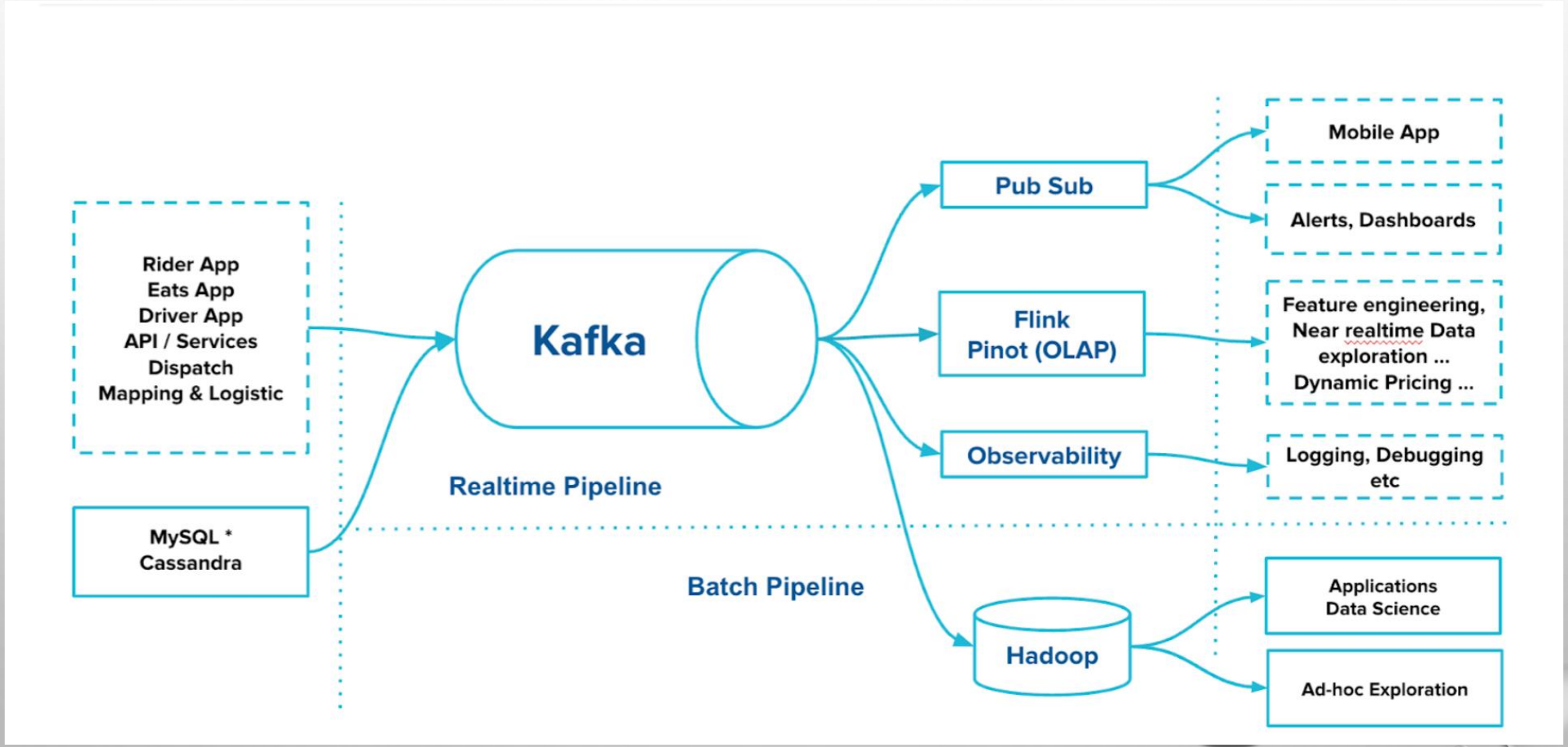


PROCESSING MODEL

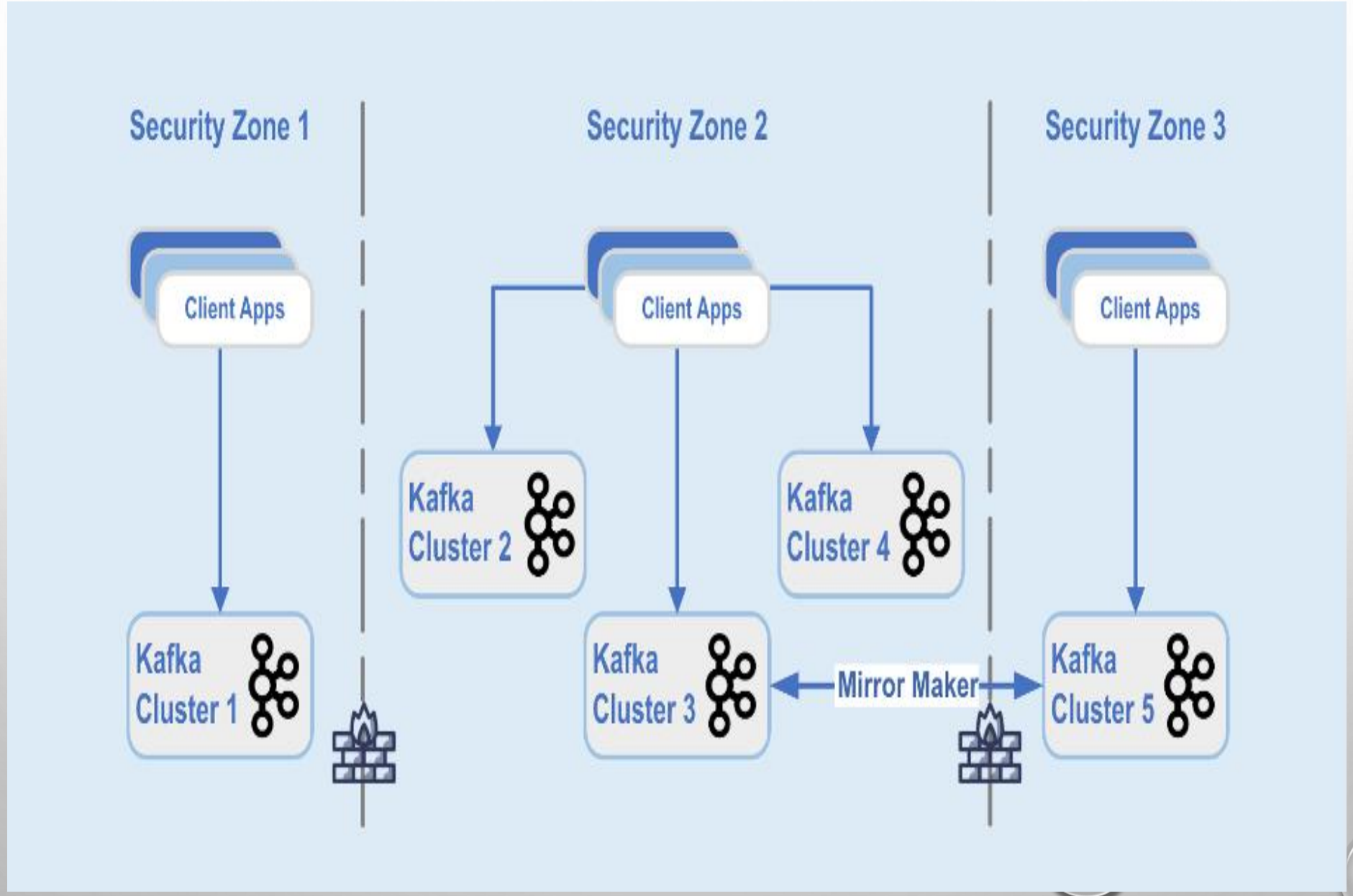
- micro batching invoives grouping records into small batches and emitting them to downstream processing at a fixed interval , event-at-time processing allows each event to be processed at soon as it comes in instead of waiting for a batch to materialize



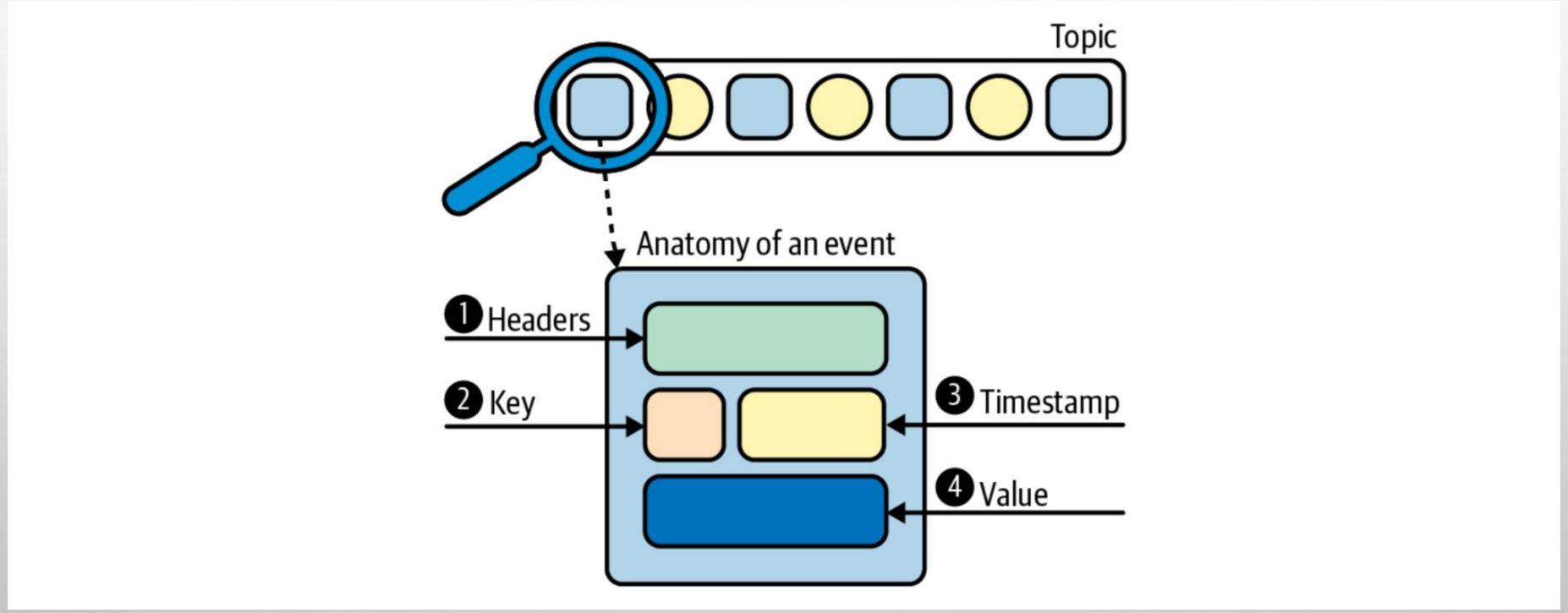
KAFKA IN UBER

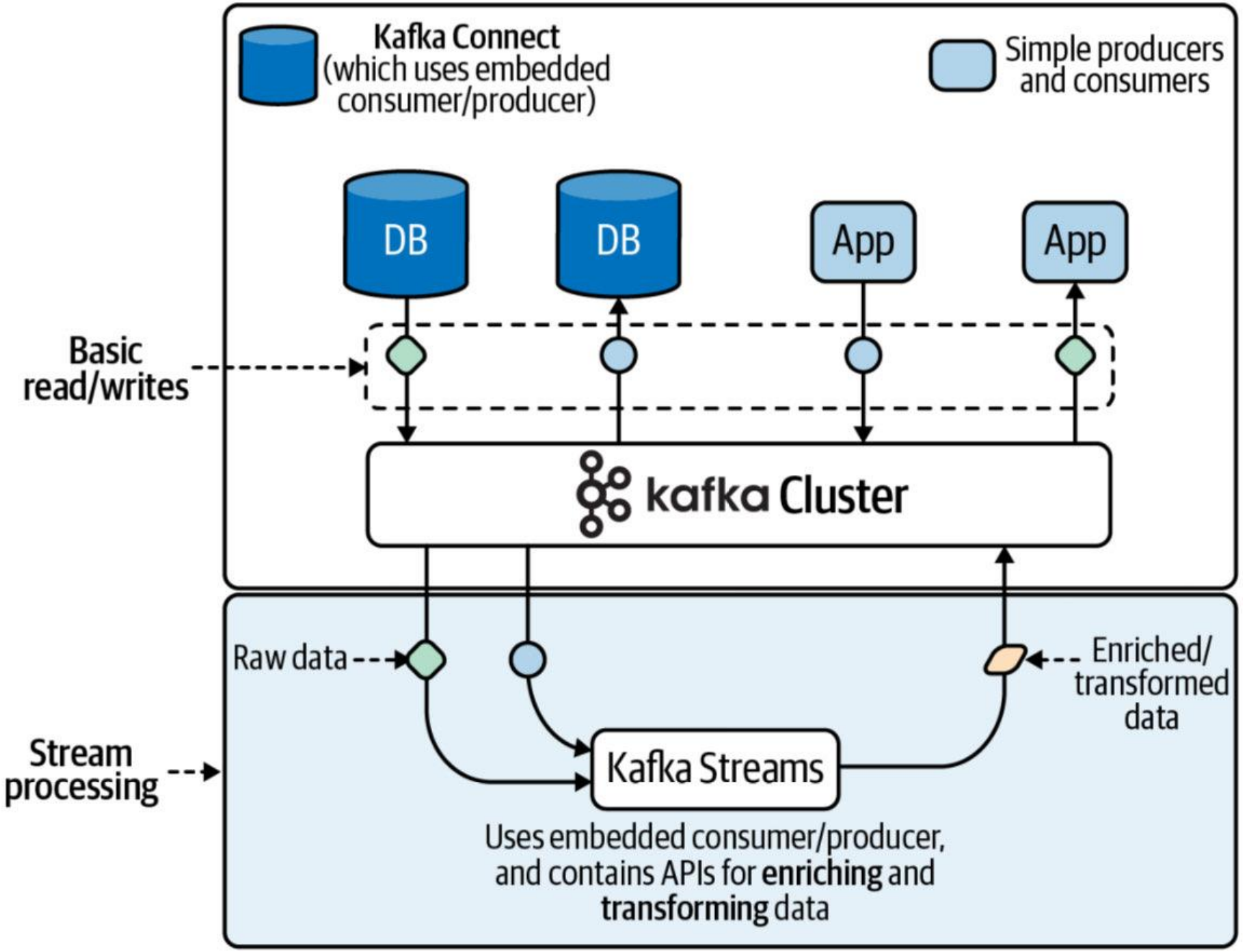


KAFKA IN PAYPAL



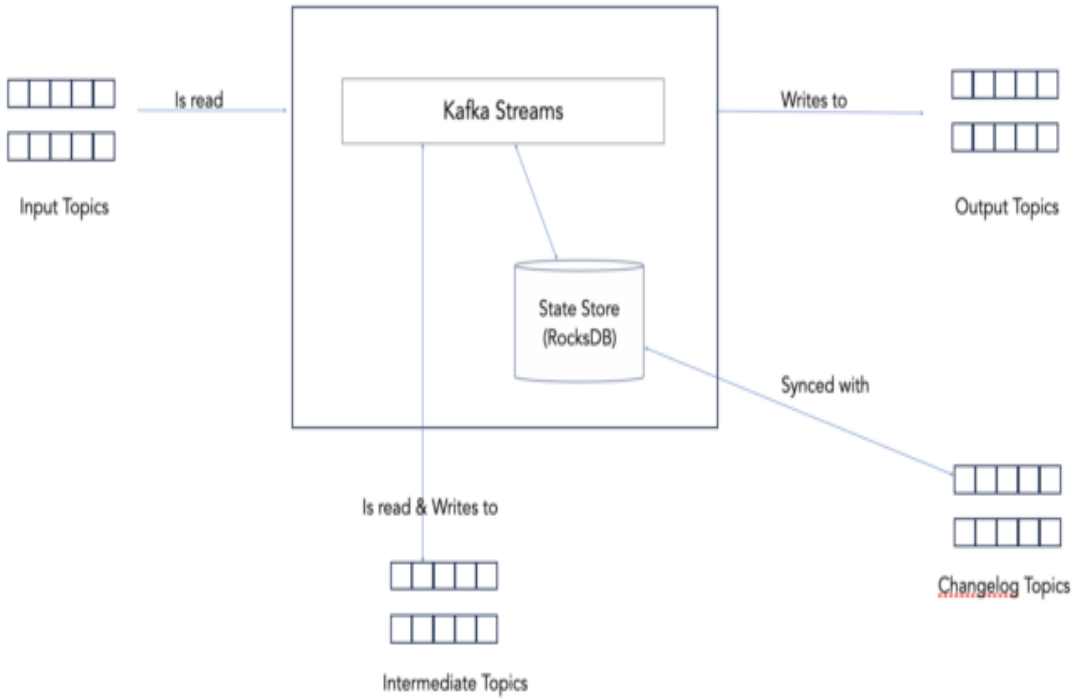
STREAM MESSAGE SCHEMA

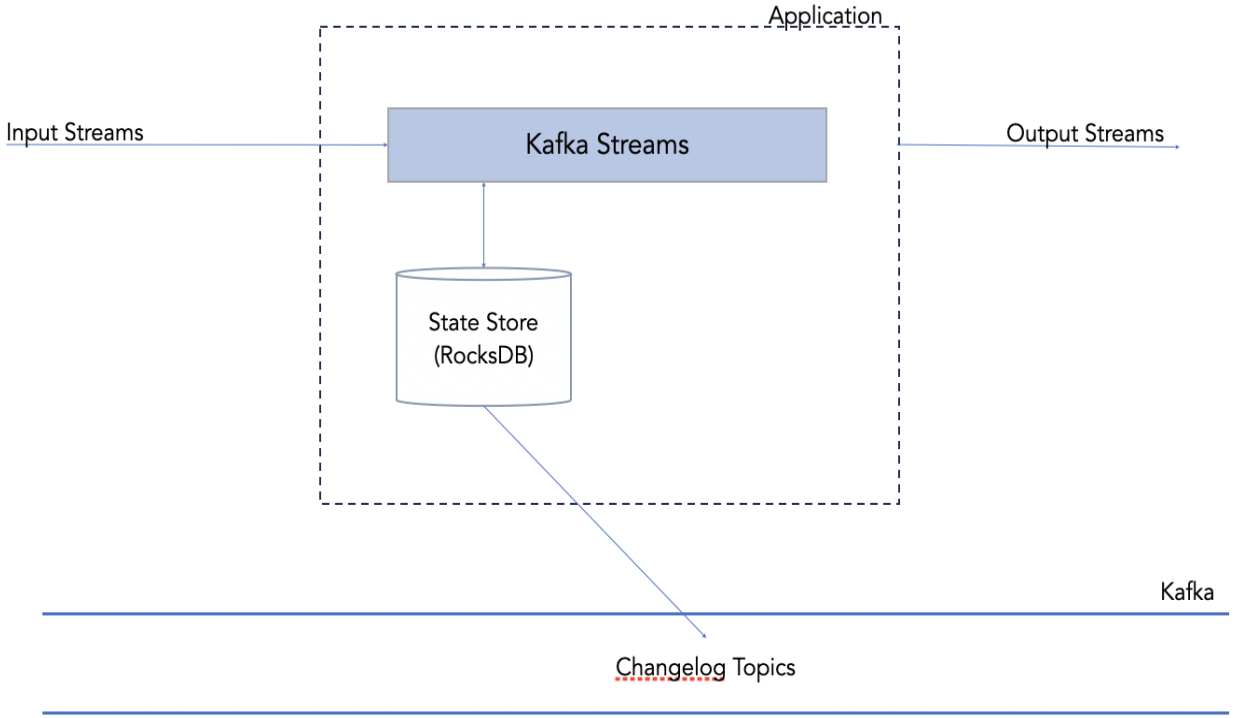



Kafka Connect
(which uses embedded
consumer/producer)
Simple producers
and consumers
DB
DB
App
App
Basic
read/writes
kafka Cluster
Raw data
Enriched/
transformed
data
Stream
processing
Kafka Streams
Uses embedded consumer/producer,
and contains APIs for enriching and
transforming data

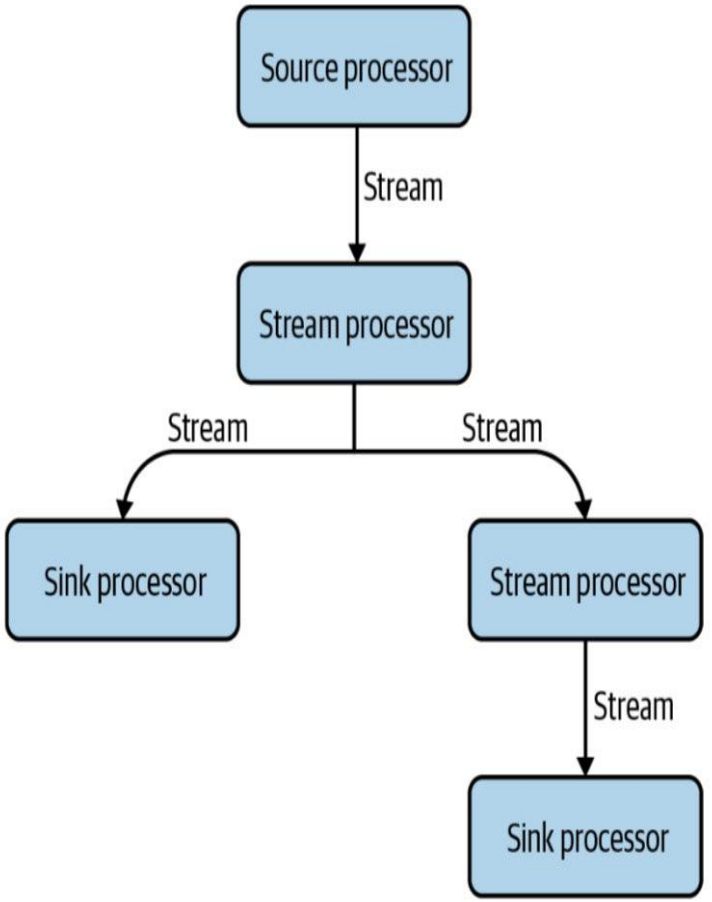
- KAFKA STREAMS CHALLENGES

1. **NOT BUILT FOR FAILURE RECOVERY**
2. **LACK OF INTERNAL CONSISTENCY**
3. **FAILURE TO SCALE WITH COMPLEXITY**





PROCESSOR TOPOLOGIES IN KAFKA STREAM



architectural view of where the deserialization and serialization processes occur in kafka stream application