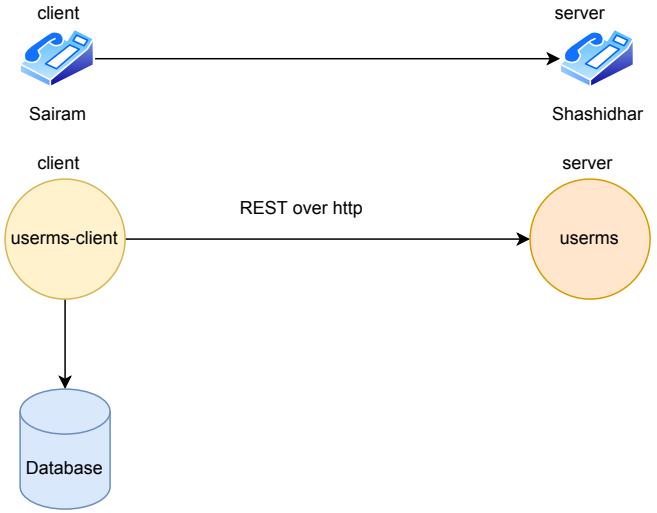
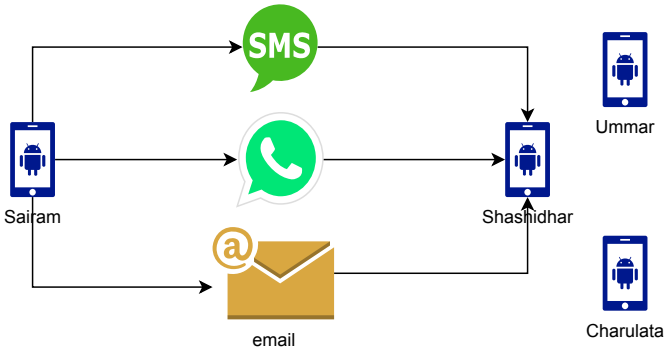
REST vs Messaging

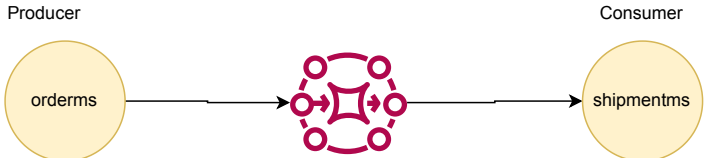
RESTful

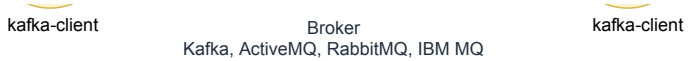
Synchronous Communication



Messaging

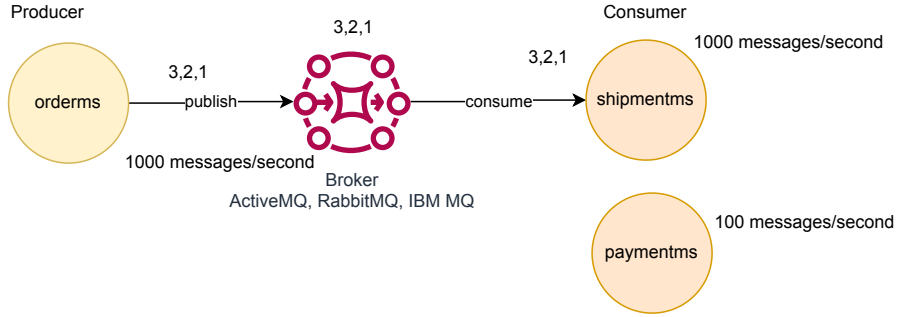




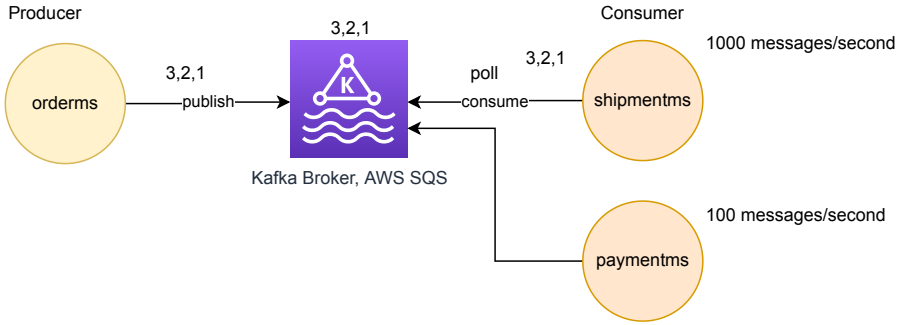

kafka-client
Broker
Kafka, ActiveMQ, RabbitMQ, IBM MQ
kafka-client

Messaging (Contd.)

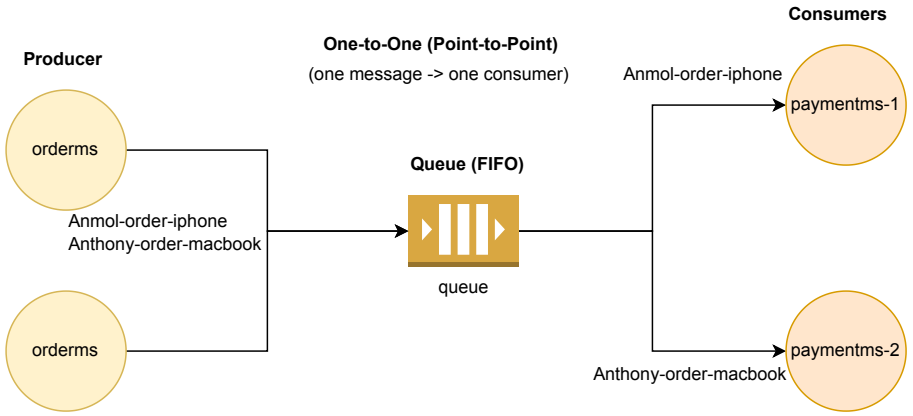
Push-based
(Broker's responsibility to guarantee delivery of message)



Pull-based
(Consumer's responsibility to consume the message)

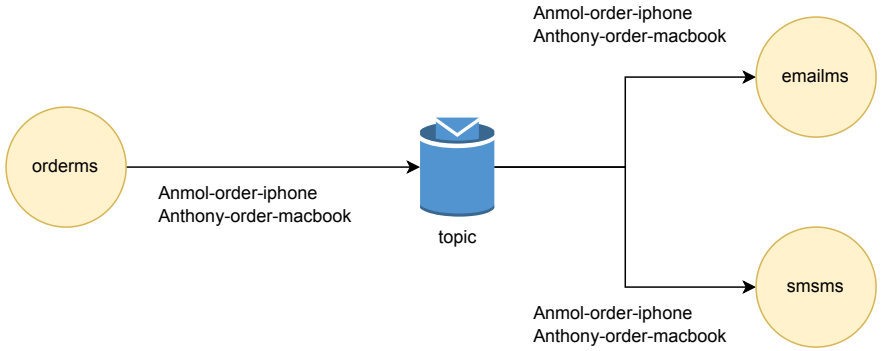


Messaging (Contd.)



One-To-Many (Publisher/Subscriber - Pub/Sub)

(one message -> many consumers)



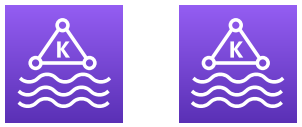
Kafka

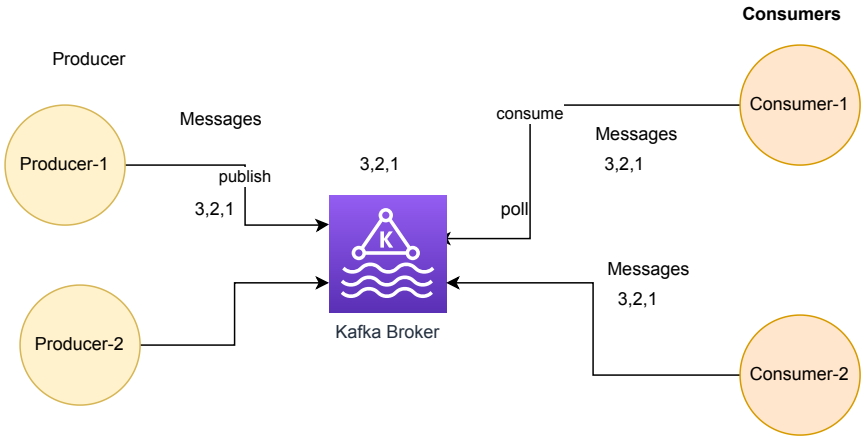
1. Apache Kafka
2. Confluent Kafka

Initially developed at Linkedin.
Later came under Apache Open Source Licence.

Characteristics of Kafka:

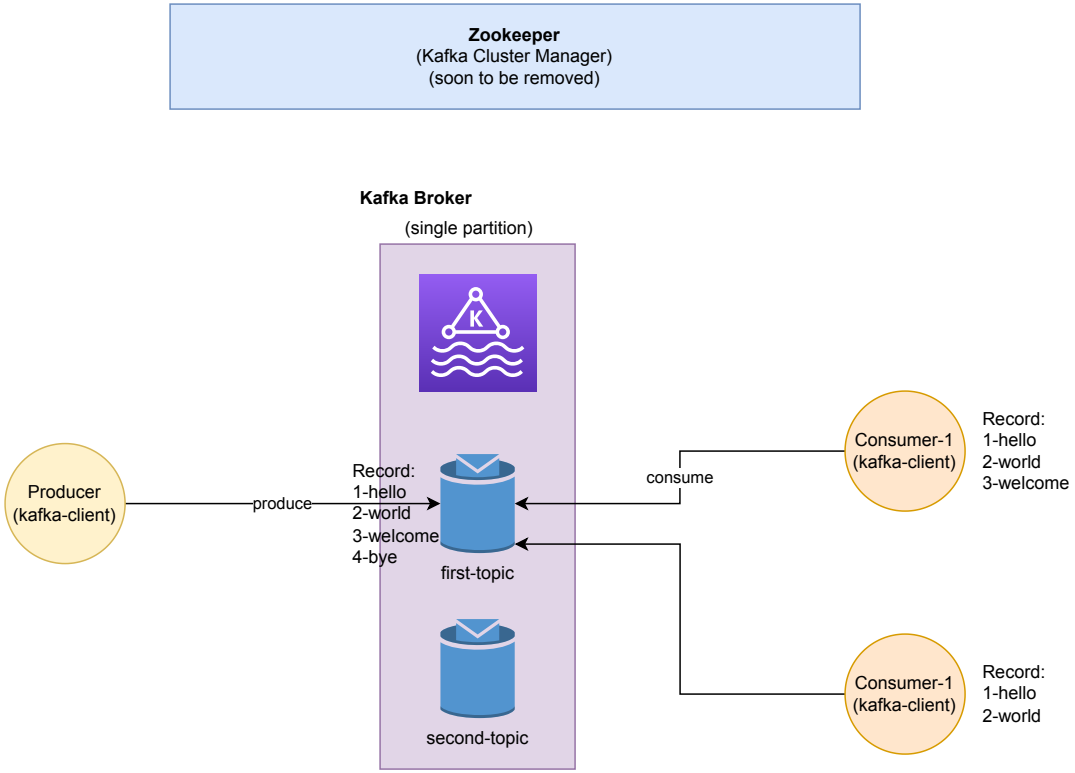
1. Highly Available
2. Scalable
3. Resilient
4. Fault Tolerant
5. Distributed
6. Low Latency
7. High Throughput



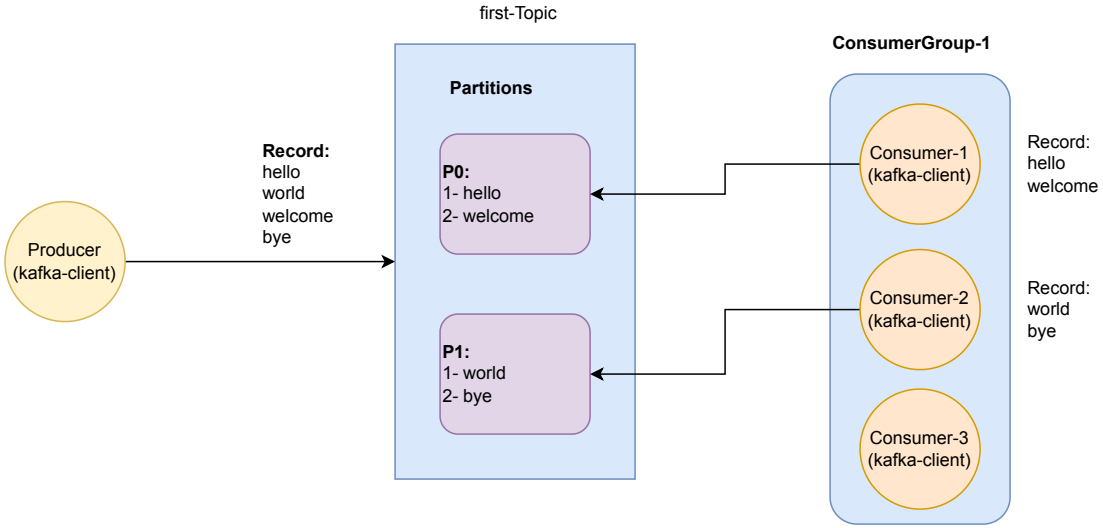


Kafka Terminologies

1. Broker
2. Topic
3. Record(message)
4. Producer
5. Consumer
6. Consumer Group
7. Kafka Client
8. Partition (Scalable)
9. Replication Factor (Highly Available)
10. Offset
11. Zookeeper



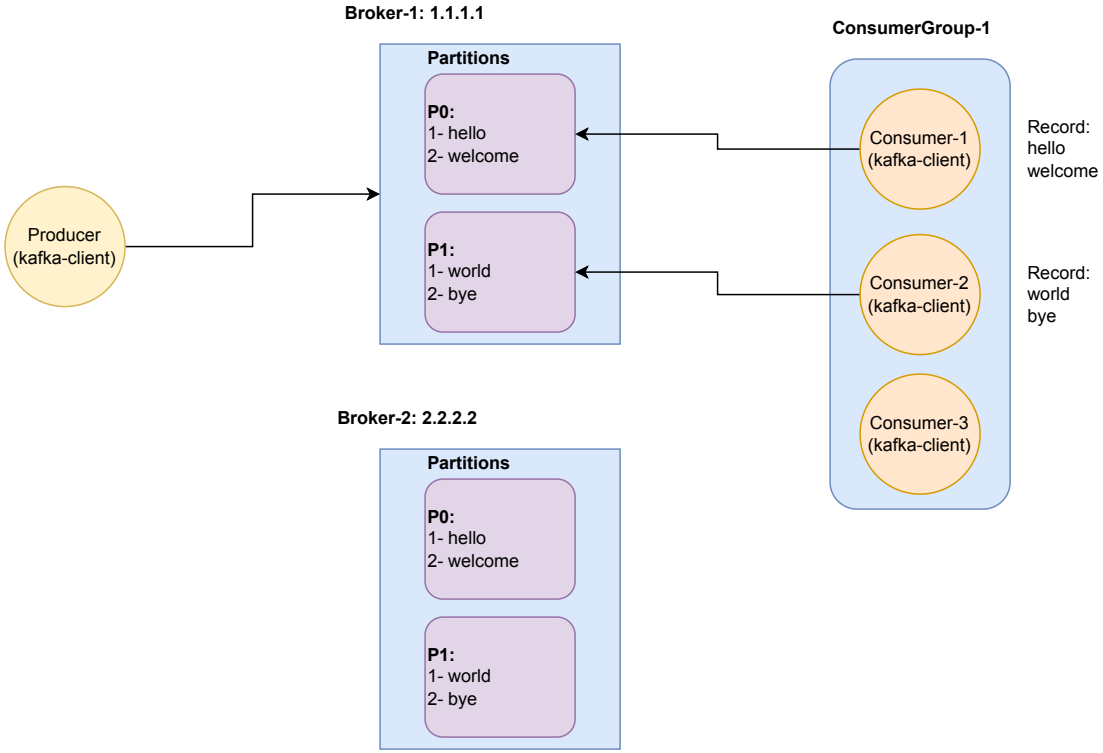
Kafka Partitions + Consumer Group
(Scalable)



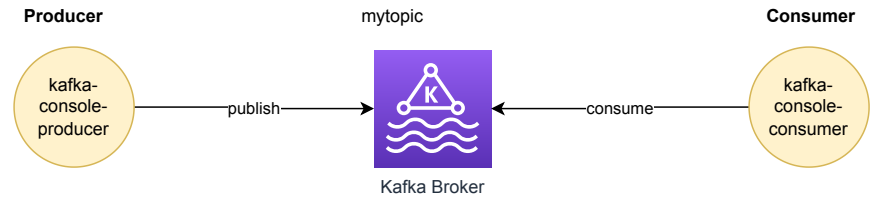
Replication Factor
(Highly Available)

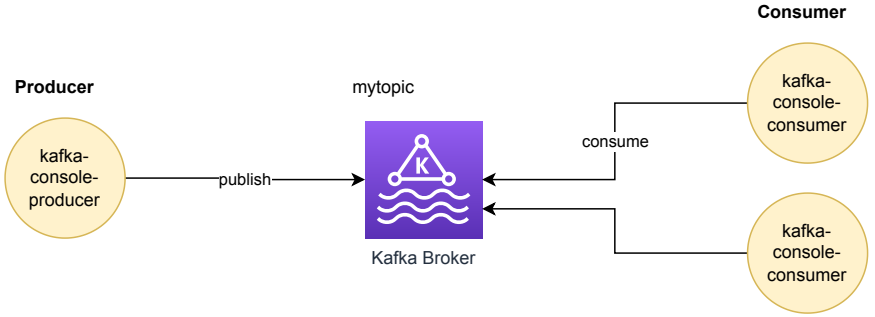
first-Topic

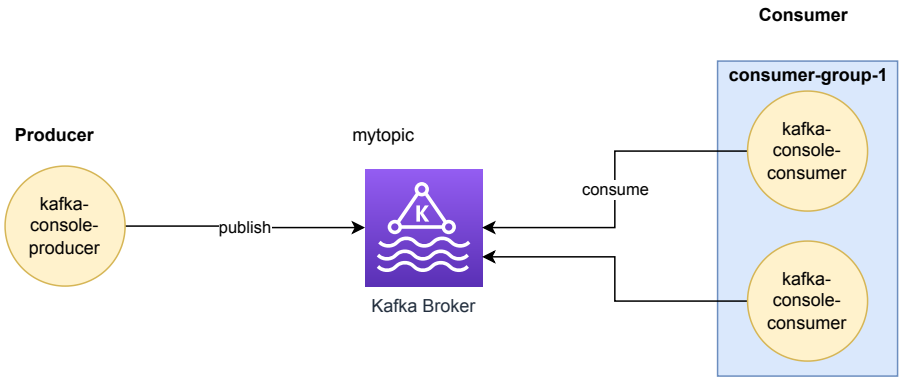
(Replication Factor = 2)



Kafka Console Producer & Kafka Console Consumer







Day-2:

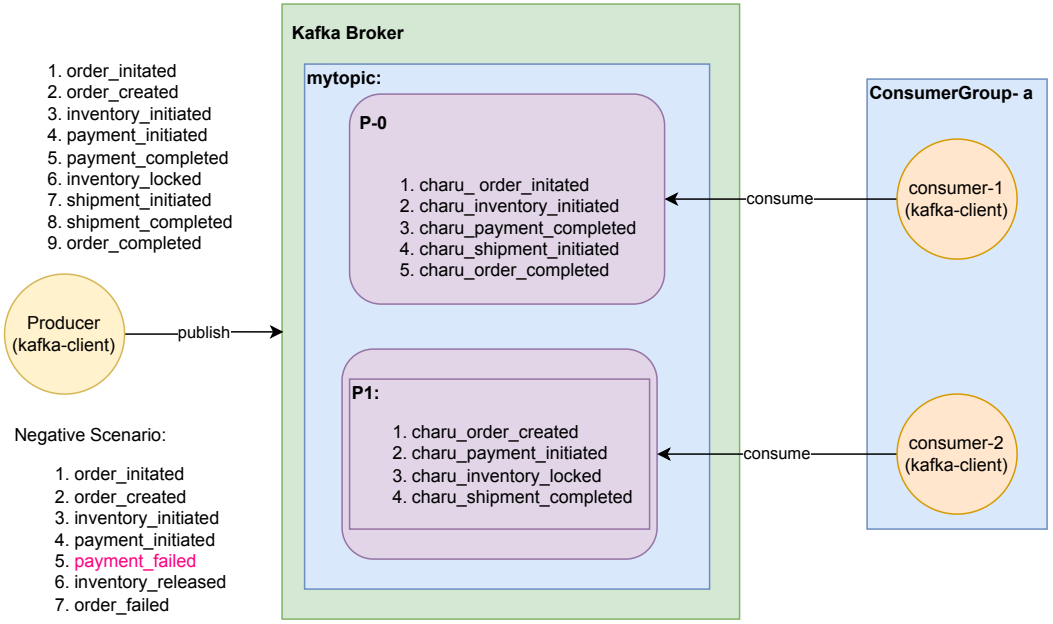
Use Case
(Placing of Order should be sequential)

Positive Scenario:

1. order_initated
2. order_created
3. inventory_initiated
4. payment_initiated
5. payment_completed
6. inventory_locked
7. shipment_initiated
8. shipment_completed
9. order_completed

Negative Scenario:

1. order_initated
2. order_created
3. inventory_initiated
4. payment_initiated
5. payment_failed
6. inventory_released
7. order_failed



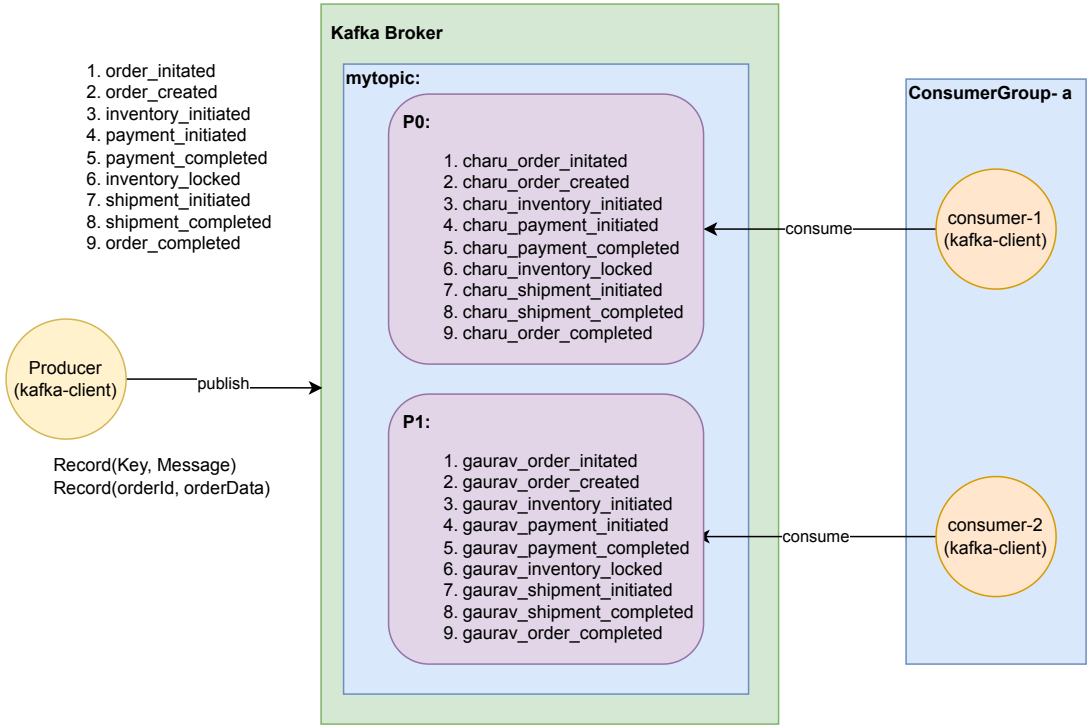
Use Case
(Placing of Order should be sequential)

Positive Scenario:

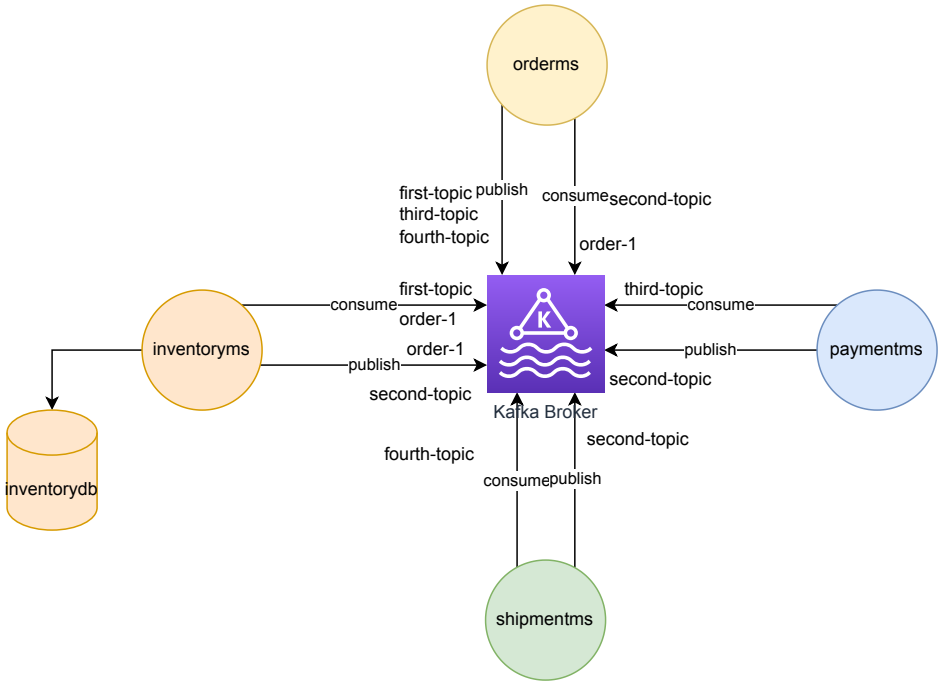
1. order_initated
2. order_created
3. inventory_initiated
4. payment_initiated
5. payment_completed
6. inventory_locked
7. shipment_initiated
8. shipment_completed
9. order_completed

Negative Scenario:

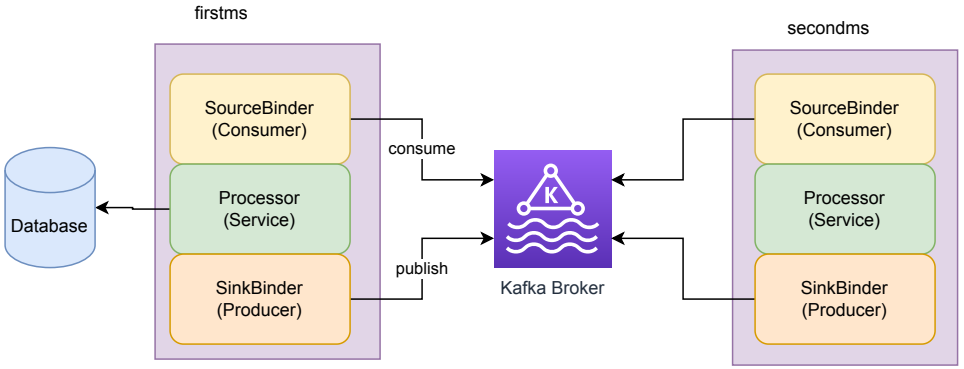
1. order_initated
2. order_created
3. inventory_initiated
4. payment_initiated
5. payment_failed
6. inventory_released
7. order_failed

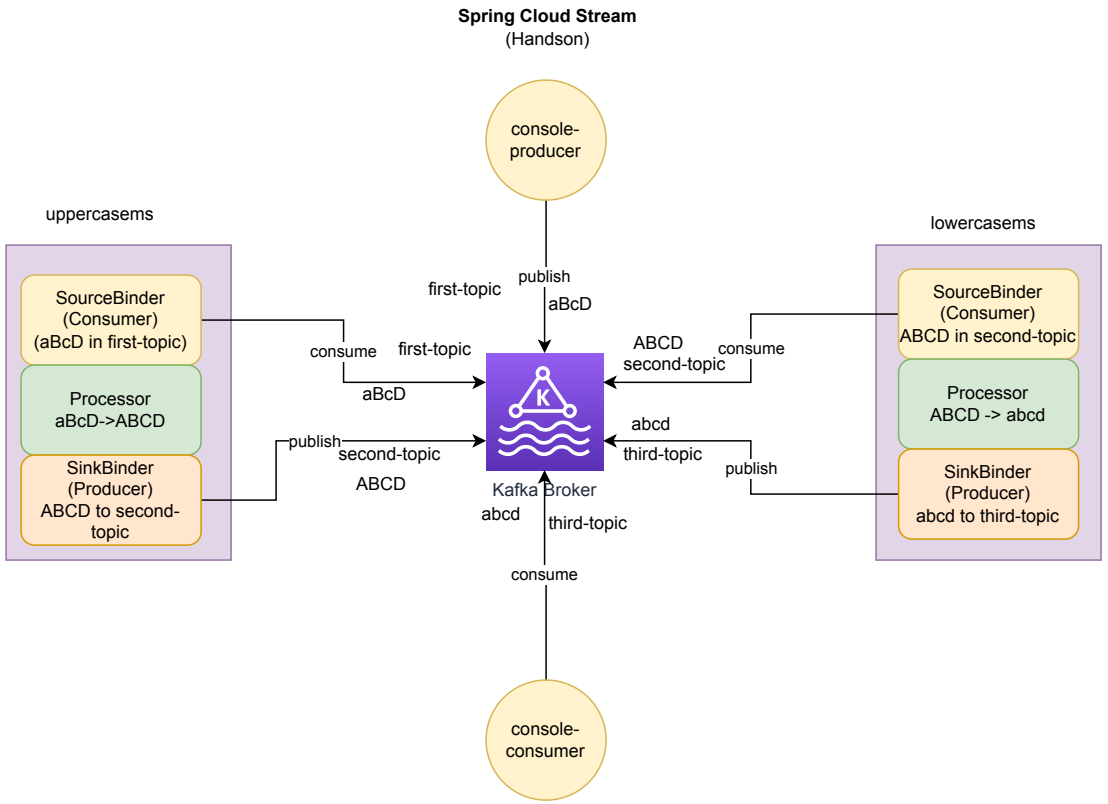


EDA/MDA - Spring Cloud Stream
Event Driven Architecture



Spring Cloud Stream




Spring Cloud Stream
(Handson)
console-
producer
uppercasems
lowercasems
SourceBinder
(Consumer)
(aBcD in first-topic)
Processor
aBcD->ABCD
SinkBinder
(Producer)
ABCD to second-
topic
SourceBinder
(Consumer)
ABCD in second-topic
Processor
ABCD -> abcd
SinkBinder
(Producer)
abcd to third-topic
first-topic
publish
aBcD
consume
first-topic
aBcD
ABCD
second-topic
consume
publish
second-topic
ABCD
abcd
third-topic
publish
Kafka Broker
abcd
third-topic
consume
console-
consumer

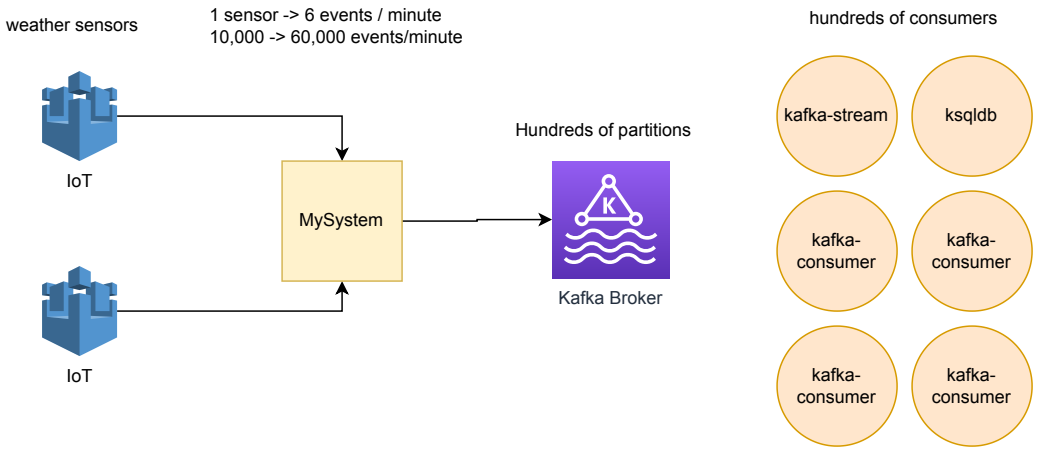
Confluent Kafka Products

1. Kafka Connect
2. Kafka Stream
3. KSQL
4. Kafka GUI
5. Confluent Kafka Cloud (Managed Service)
6. AWS MSK (Managed Streaming for Apache Kafka)
7. Confluent CLI

Batch vs Stream Apache Hadoop, Apache Spark, Apache Storm, Apache Flink, Kafka Stream, KSQL

IMPS vs NEFT

Kafka Stream, KSQLDB = Streaming Data

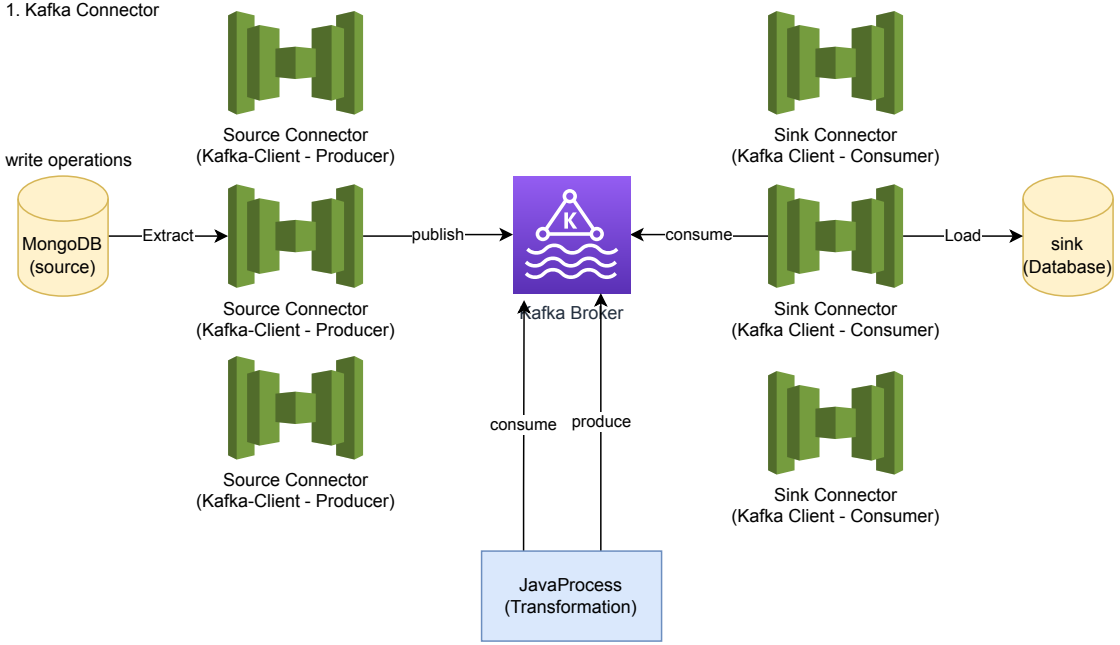


Day-3

Kafka Connect

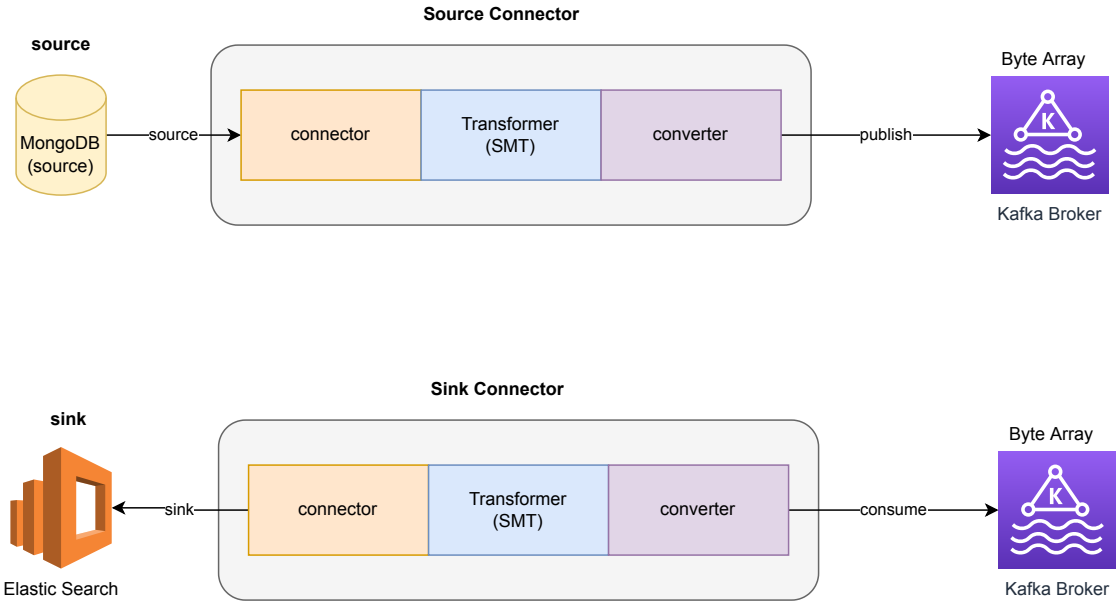
ETL
(Extract Transform Load)

1. Kafka Connector



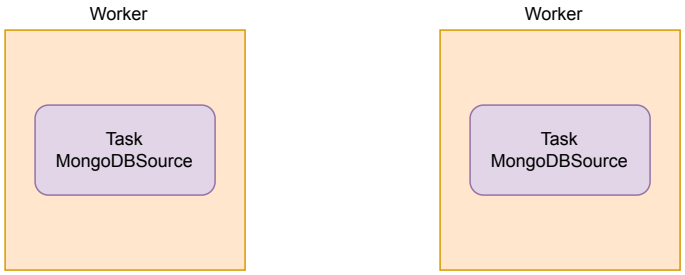
Kafka Connect

SMT: Single Message Transform

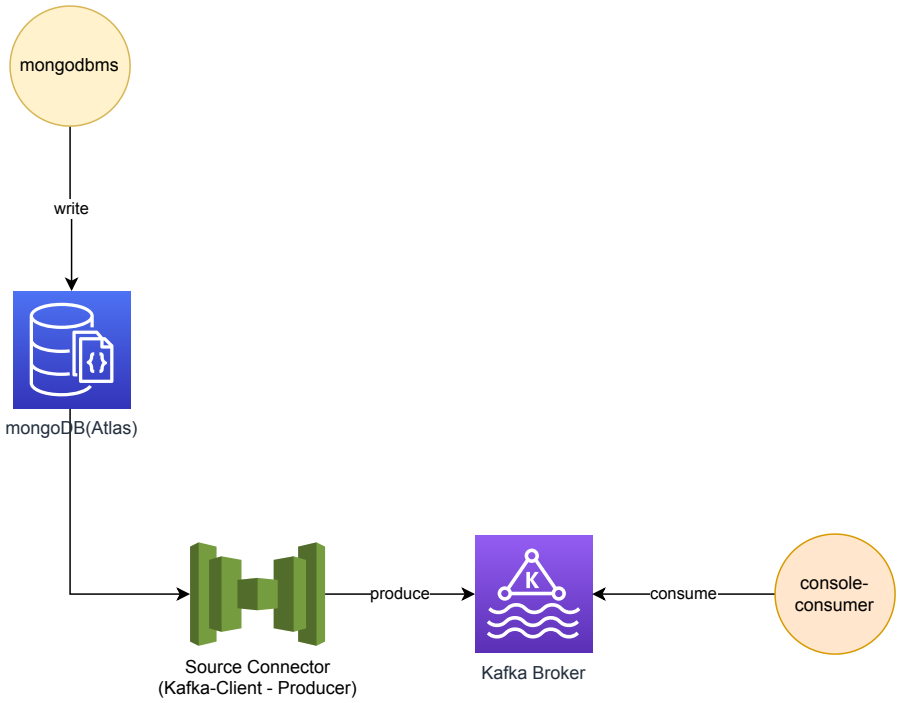


Kafka Connect

Worker and Tasks



Handson



Relational

1. Database
2. Table
3. Rows
4. Columns

No-SQL

1. Database
2. Collection
3. Document
4. Field

BSON: Binary Json