MESSAGING

Teaching Faculty: Umur INAN

Prepared by Umur INAN

Remoting Technologies

- JAX-WS
  - Spring provides remoting support for web services via JAX-WS [SOAP]
- JMS
  - Remoting using JMS is supported
- AMQP
  - Remoting using AMQP as underlying protocol

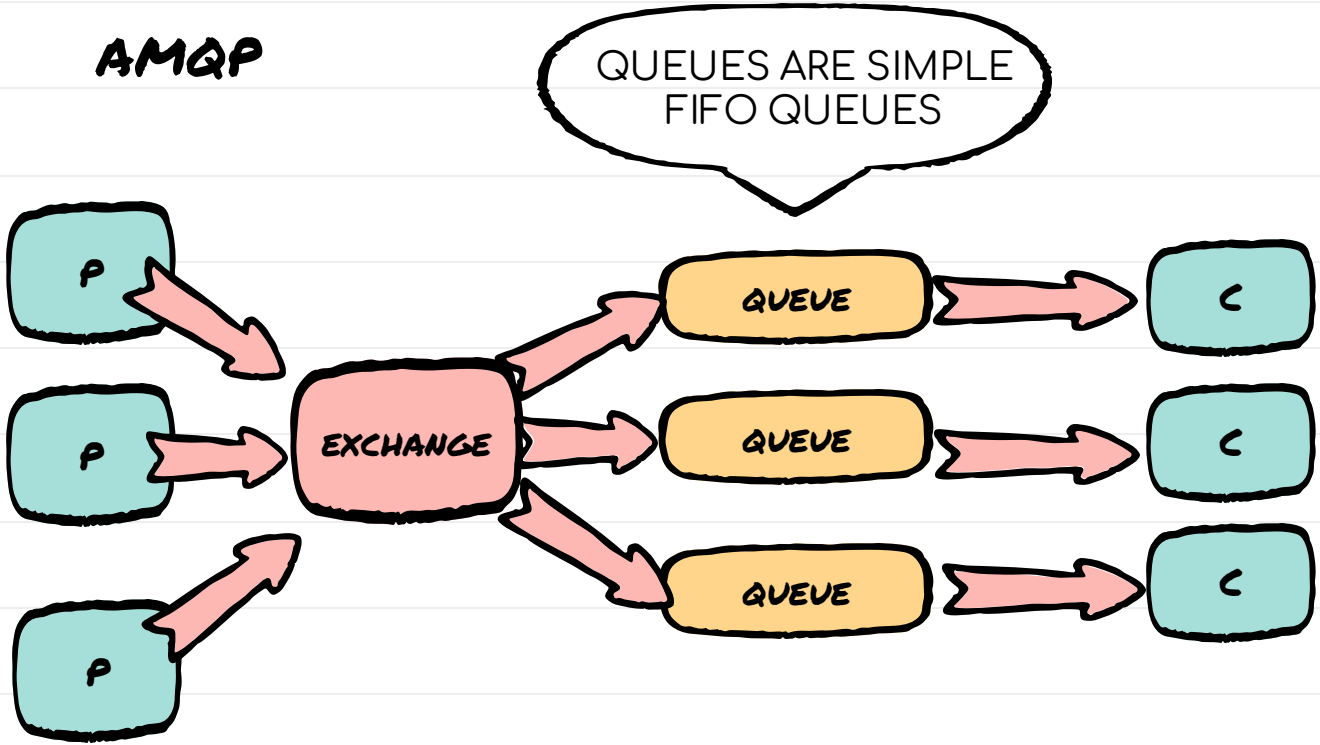
JMS (Java Messaging Service)

- JMS has queues and topics.
  - A message sent on a queue is consumed by no more than one client.
  - A message sent on a topic may be consumed by multiple consumers.

AMQP (Advanced Messaging Queue Protocol)

- AMQP only has queues and exchanges.
  - Queues are only consumed by a single receiver.
  - AMQP doesn't publish directly to queues.
    - A message is published to an exchange.
    - routed to one queue or multiple queues.

AMQP


QUEUES ARE SIMPLE
FIFO QUEUES
P
P
P
EXCHANGE
QUEUE
QUEUE
QUEUE
C
C
C

PRODUCER VS CONSUMER

- Queues are only consumed by a single receiver.
  - If more than one consumer subscribes to the queue, the messages are dispensed in a round-robin fashion.
- Consumer defined queues.
  - Multiple queues, bound to the same exchange/routing key.
  - Emulate a broadcast message.
  - A queue per consumer.

AMQP CONCEPTS

- Exchanges
  - Message routing agents
  - Accept messages from producers routes to queues
- Bindings
  - Binds/maps a queue & exchange
- Routing Key
  - Optional attribute to customize binding/routing
- Queues
  - Message placeholders

AMQP Exchanges

- Direct
  - Queue binding requires a direct match based on a “simple” Routing Key.
  - Corresponds to JMS PTP.
  - Can have multiple Queues/Consumers.

AMQP Exchanges

- FanOut
  - Queue is bound directly to exchange no Routing Key.
  - Publisher / Subscriber

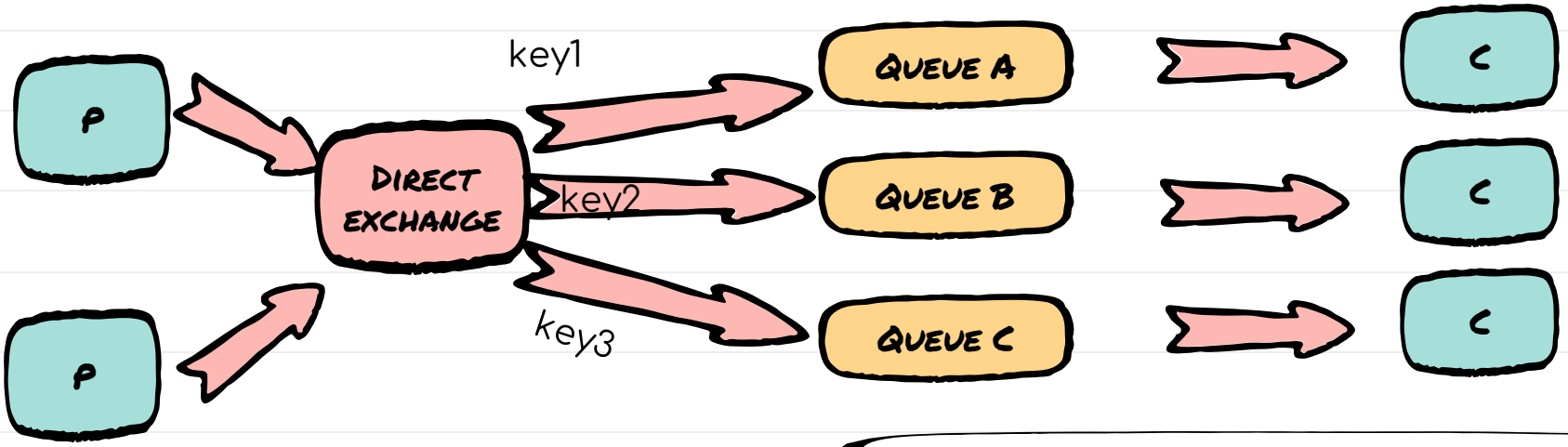
AMQP Exchanges

- Topic
  - Queue binding requires a direct match based on a “complex” Routing Key
  - Beyond Pub / Sub

AMQP Exchanges

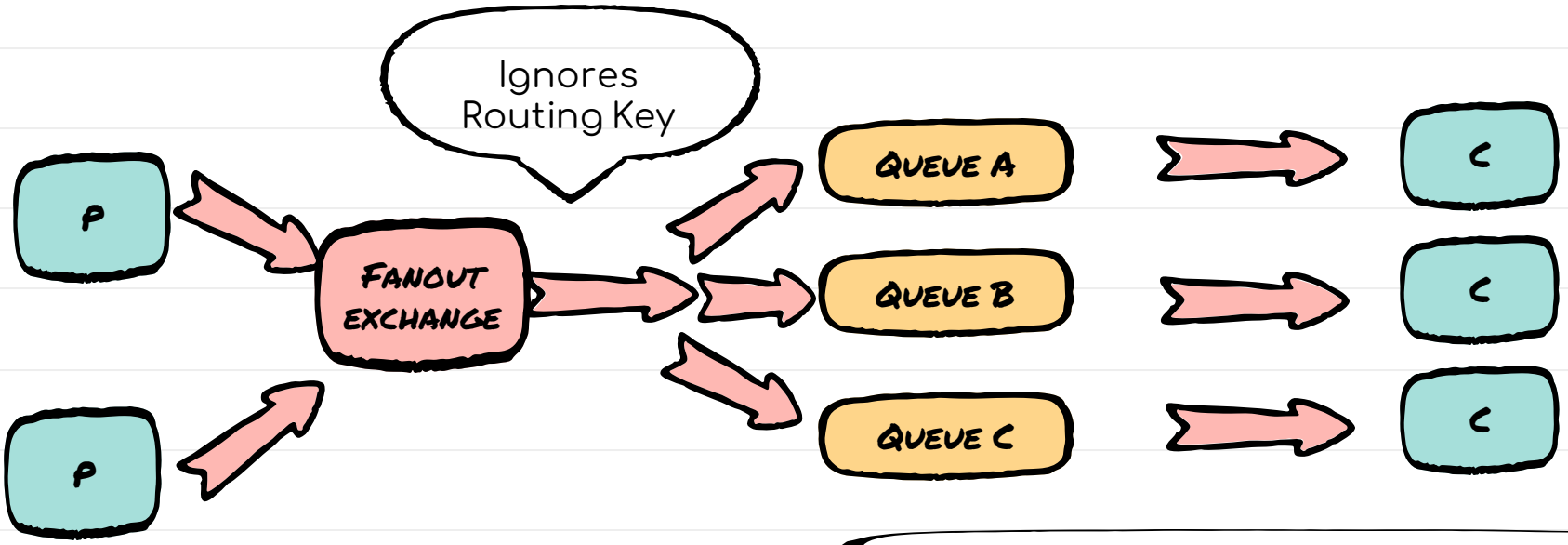
- Header
  - Similar to Topic only uses message headers instead of explicit Routing Key

Direct exchange



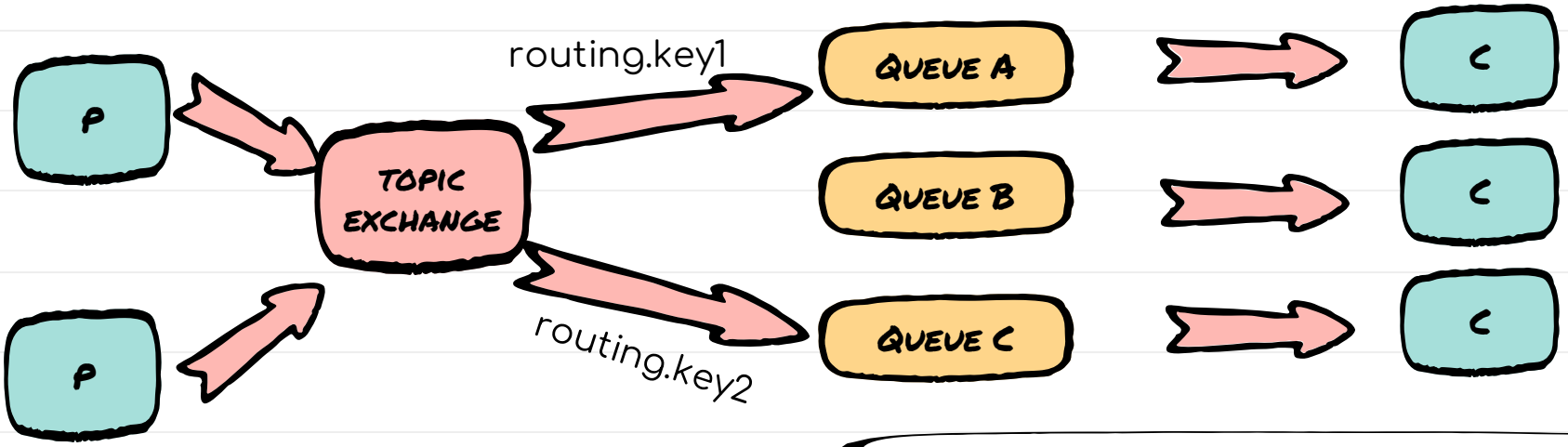
A Direct exchange will be used to route messages intended for a specific customer directly to that customer.

FanOut exchange



Fanout exchange for messages intended for all consumers

Topic exchange



A Topic exchange will be used to route messages that might apply to some consumers but not others.

RABBITMQ

- RabbitMQ is an open-source messaging server that makes use of the AMQP.
- Rabbit is compatible with many programming languages and allows you to handle message traffic simply and reliably.

SPRING AMQP

- AMQP entities
  - Are created entities with the Message, Queue, Binding, and Exchange classes.
- Connection Management
  - CachingConnectionFactory is used to connect to RabbitMQ broker.
- Message Publishing
  - RabbitTemplate is to send messages.
- Message Consumption
  - @RabbitListener is used to read messages from a queue.

Dependencies

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
```

KAFKA

- Apache Kafka is an open-source distributed event streaming platform.
- It is an open-source software platform developed by the Apache Software Foundation written in Scala and Java.
- Kafka uses a binary TCP-based protocol that is optimized for efficiency and relies on a "message set" abstraction that naturally groups messages together to reduce the overhead of the network roundtrip.

Event streaming

- Event streaming is the practice of capturing data in real-time from event sources like
  - Databases
  - Sensors
  - Mobile devices
  - Cloud services
  - Software applications

in the form of streams of events;

Event streaming

- storing these event streams durably for
  - Later retrieval
  - Manipulating
  - Processing
  - Reacting

  to the event streams in real-time as well as retrospectively; and
- routing the event streams to different destination technologies as needed.

Event streaming

- Event streaming thus ensures a continuous flow and interpretation of data so that the right information is at the right place, at the right time.

Event streaming use cases

- To process payments and financial transactions in real-time.
  - Stock exchanges, banks, and insurances.
- To track and monitor cars, trucks, fleets, and shipments in real-time.
  - Logistics and the automotive industry.
- To continuously capture and analyze sensor data from IoT devices or other equipment.
  - factories and wind parks.

Event streaming use cases

- To collect and immediately react to customer interactions and orders.
  - Retail, the hotel and travel industry, and mobile applications.
- To monitor patients in hospital care and predict changes in condition to ensure timely treatment in emergencies.
- To connect, store, and make available data produced by different divisions of a company.
- To serve as the foundation for data platforms, event-driven architectures, and microservices.

Event streaming platform

- Kafka combines three key capabilities
  - To publish (write) and subscribe to (read) streams of events, including continuous import/export of data from other systems.
  - To store streams of events durably and reliably for as long as wanted.
  - To process streams of events as they occur or retrospectively.

Event streaming platform

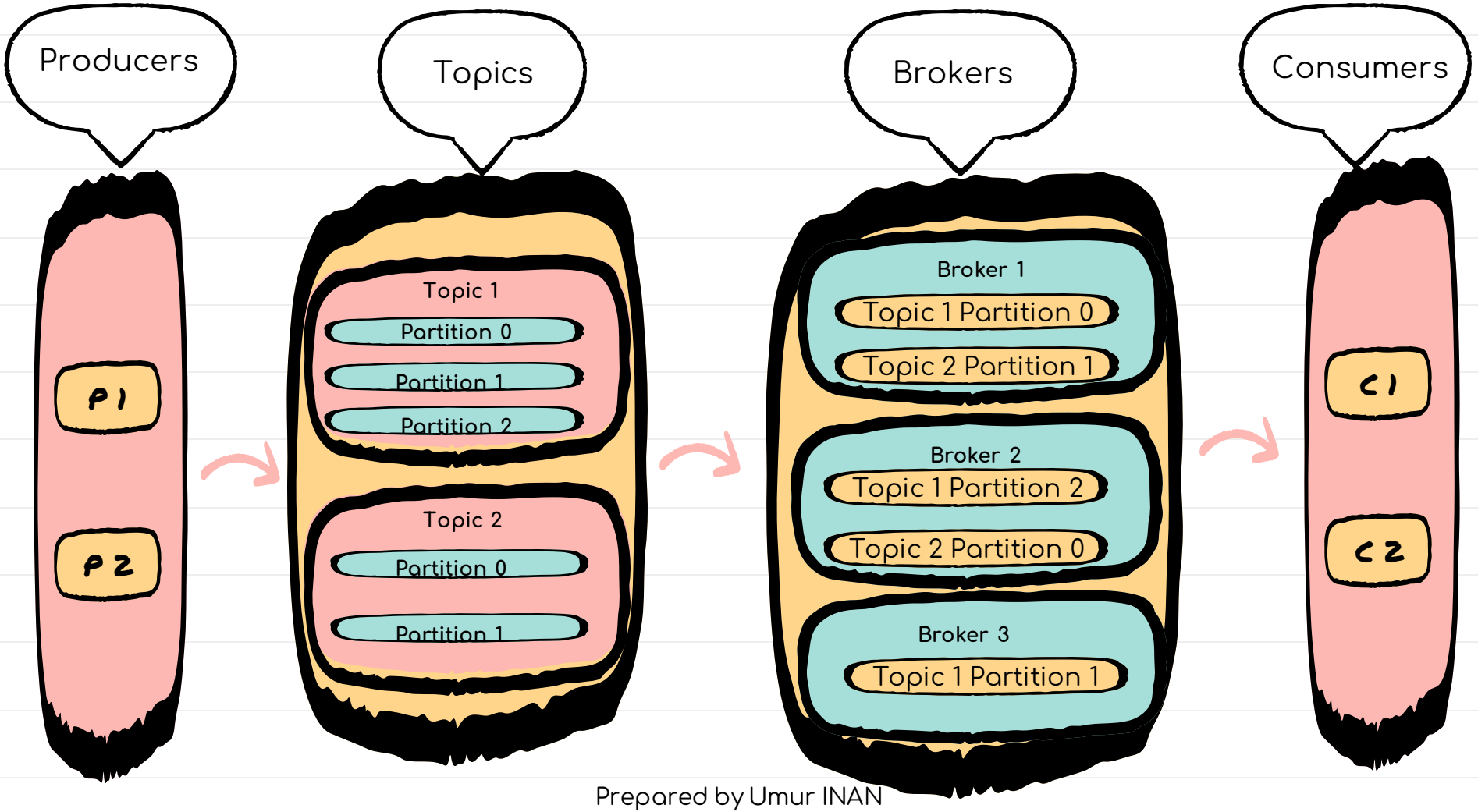
- All this functionality is provided in a
  - Distributed
  - Highly scalable
  - Elastic
  - Fault-tolerant
  - Secure manner

Kafka advantages

- Low Latency
- High Throughput
- Fault tolerance
- Durability
- Easily accessible
- Real-Time handling
- Scalability

Kafka disadvantages

- Message tweaking issues.
- Clumsy Behavior.
- Do not have complete set of monitoring tools.
- Lack some message paradigms.



Prepared by Umur INAN

TERMINOLOGY

- Topic
  - A stream of messages belonging to a particular category is called a topic.
  - Data is stored in topics.
  - Topics are split into partitions.
    - For each topic at least 1 partition.

TERMINOLOGY

- Partition
  - Partition contains messages in an immutable ordered sequence.
  - Topics may have many partitions.
    - it can handle an arbitrary amount of data.
- Partition Ofset
  - Each partitioned message has a unique sequence id called as offset.

TERMINOLOGY

- Brokers
  - Brokers are simple system responsible for maintaining the published data.
  - Each broker may have zero or more partitions per topic.
- Kafka Cluster
  - Kafka's having more than one broker are called as Kafka cluster.
  - clusters are used to manage the persistence and replication of message data.

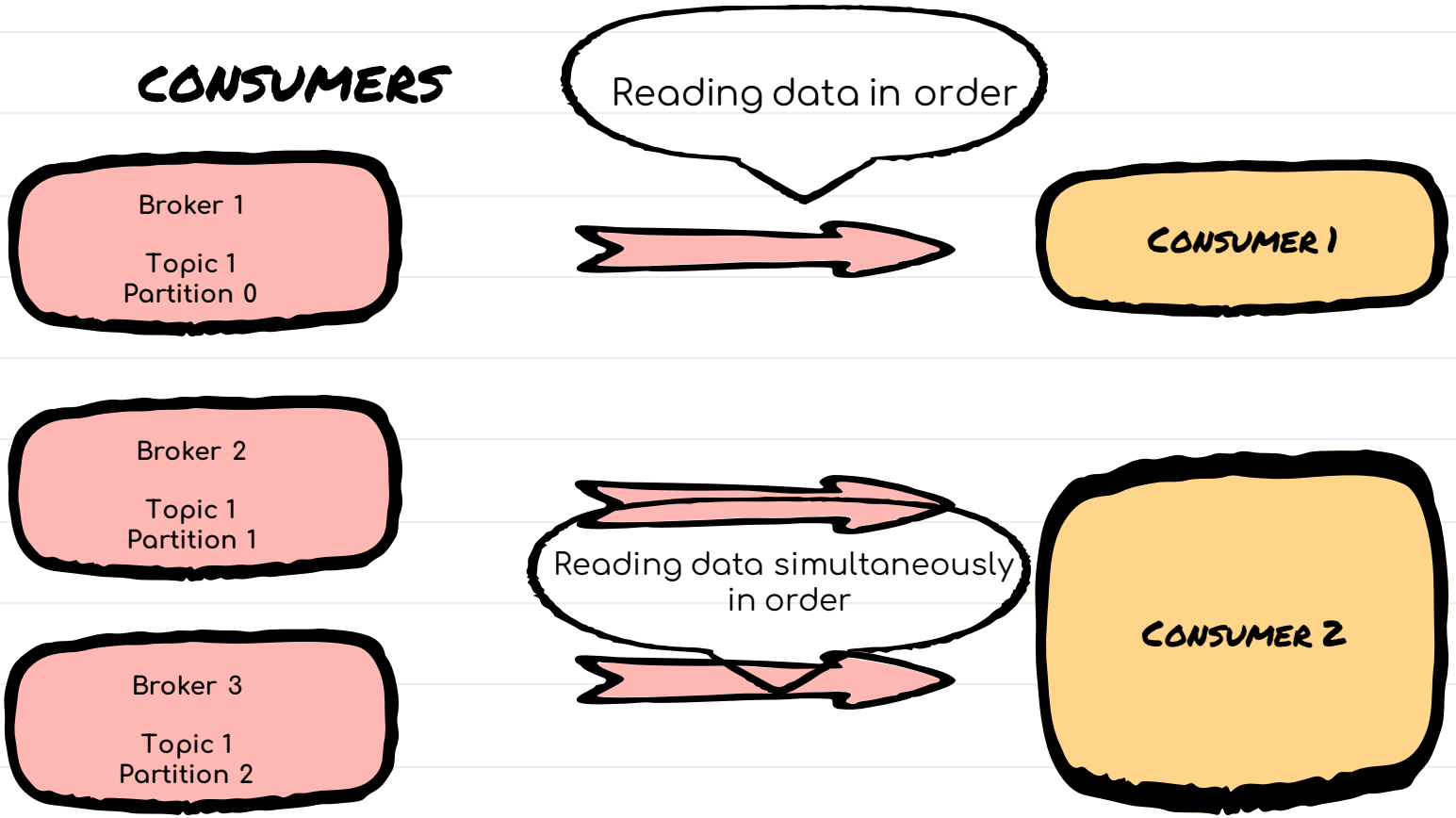
TERMINOLOGY

- Producers
  - Producers are the publisher of messages to one or more Kafka topics.
  - Producers send data to Kafka brokers.
  - Producer can also send messages to a partition of their choice.

TERMINOLOGY

- Consumers
  - Consumers read data from brokers.
  - Consumers subscribes to one or more topics and consume published messages by pulling data from the brokers.
  - A consumer also knows that from which broker, it should read the data.
  - The consumer reads the data within each partition in an orderly manner.

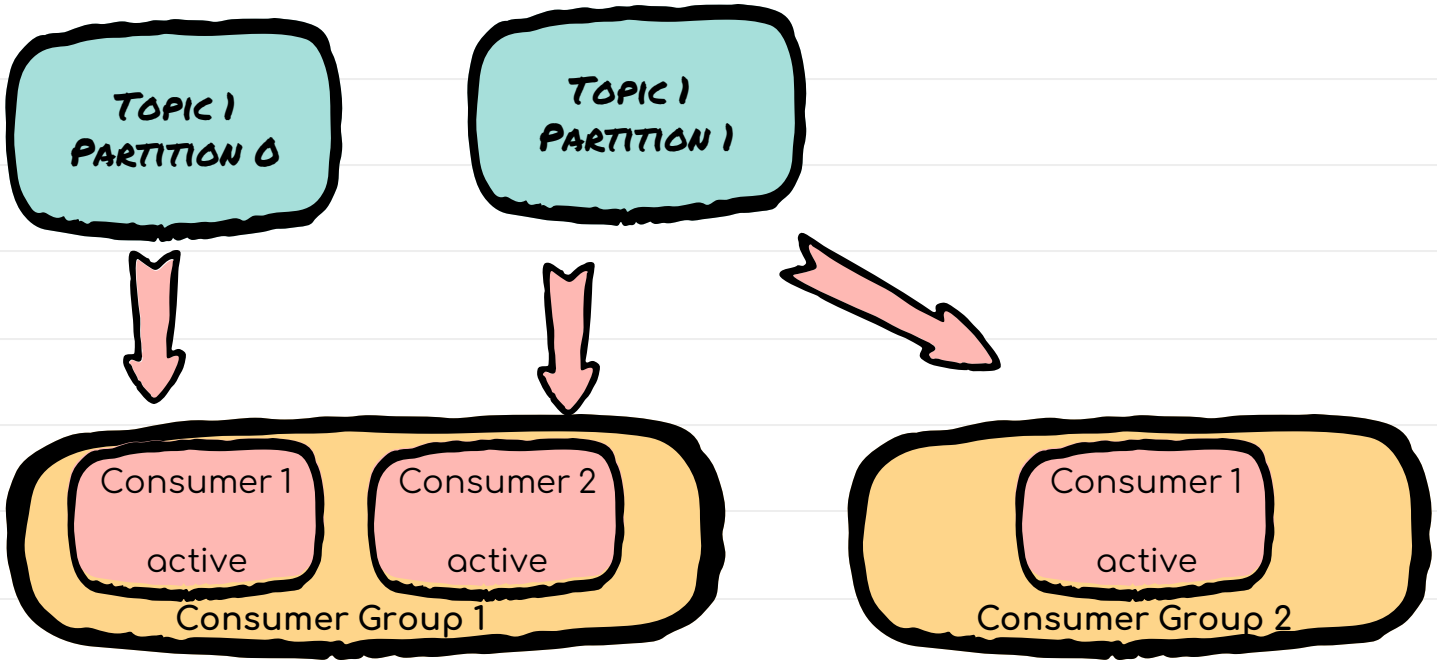
CONSUMERS


Reading data in order
Broker 1
Topic 1
Partition 0
CONSUMER 1
Broker 2
Topic 1
Partition 1
Reading data simultaneously
in order
CONSUMER 2
Broker 3
Topic 1
Partition 2

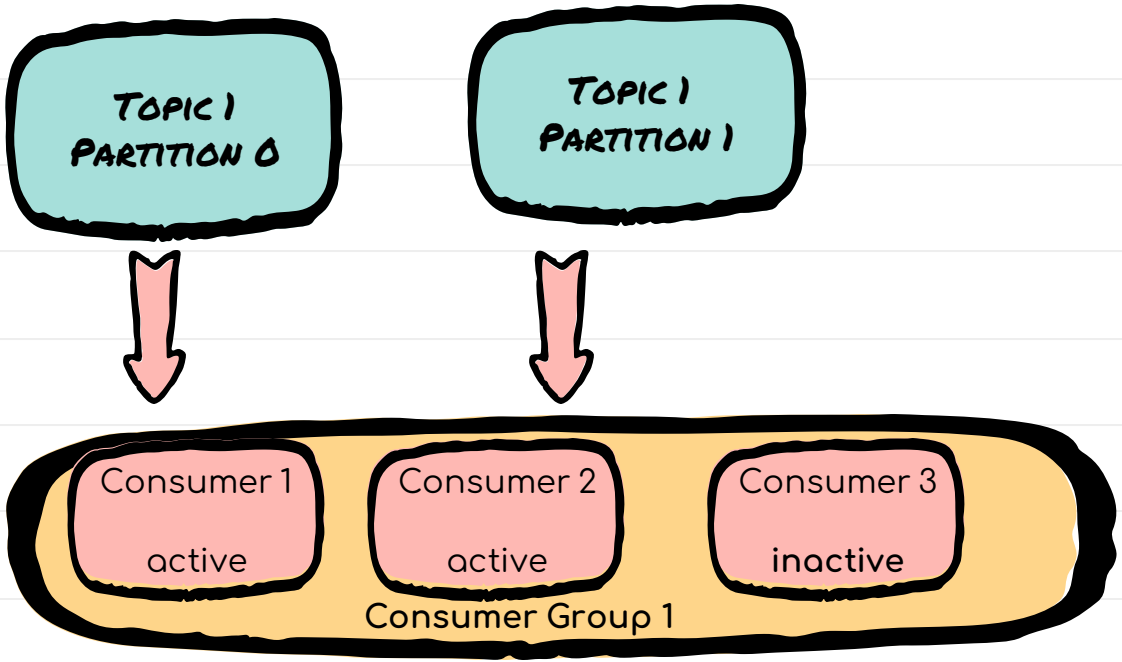
TERMINOLOGY

- Consumer Gorups
  - A consumer group is a group of multiple consumers which visions to an application.
  - Each consumer present in a group reads data directly from the exclusive partitions.
  - In case, the number of consumers are more than the number of partitions, some of the consumers will be in an inactive state.

CONSUMERS


Topic 1
Partition 0
Topic 1
Partition 1
Consumer 1
active
Consumer 2
active
Consumer Group 1
Consumer 1
active
Consumer Group 2

CONSUMERS

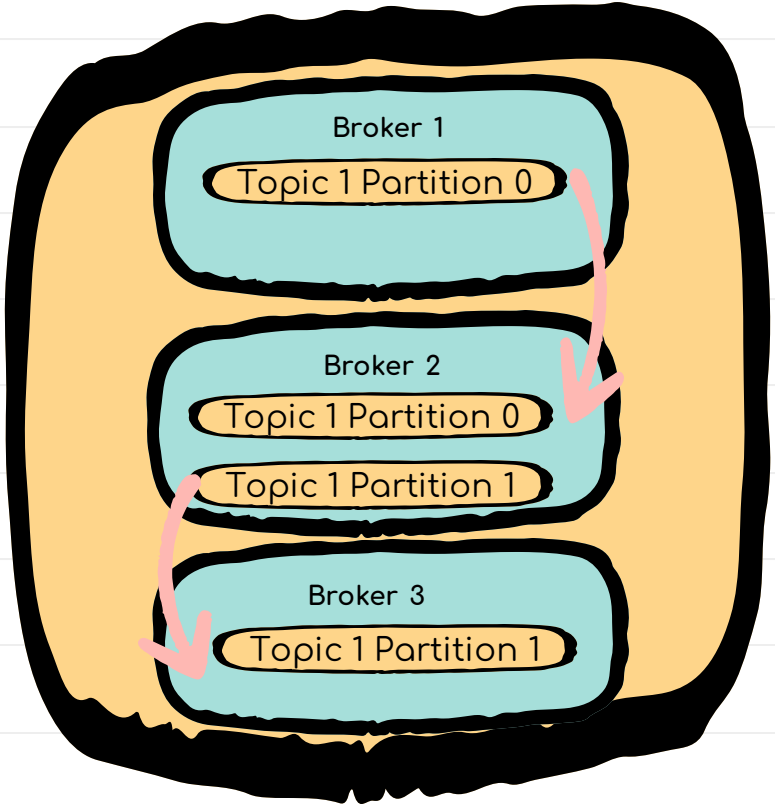

Topic 1
Partition 0
Topic 1
Partition 1
Consumer 1
active
Consumer 2
active
Consumer 3
inactive
Consumer Group 1

TERMINOLOGY

- Leader
  - Leader is the node responsible for all reads and writes for the given partition.
  - Every partition has one server acting as a leader.
- Follower
  - Node which follows leader instructions are called as follower.
  - If the leader fails, one of the follower will automatically become the new leader.
  - A follower acts as normal consumer, pulls messages and up-dates its own data store.

Replication factor

- A replication factor is the number of copies of data over multiple brokers.
- A replication factor is created for the topics contained in any particular broker.
- The replication factor value should be greater than 1 always (between 2 or 3).
- It supports data replication at the partition level.



Topic with 2 partitions and replication factor 2

Replication

- At any time only 1 broker can be a leader for a given partition.
- Only that leader can receive and serve data for a partition.
- The other brokers will synchronize the data.
- Each partition has 1 leader and multiple ISR (in-sync replica)

SPRING BOOT KAFKA

```
<dependency>
    <groupId>org.springframework.kafka</groupId>
    <artifactId>spring-kafka</artifactId>
</dependency>
```

Main points

- AMQP integrates heterogeneous systems through the introduction of a basic wire protocol.
- Acting from the level of Transcendental Consciousness, thoughts and actions are more integrated and harmonious.
- Messaging oriented services guarantee a reliable communication and simplify the complexity of the applications.
- Pure Consciousness is simple, reliable, efficient and precise.