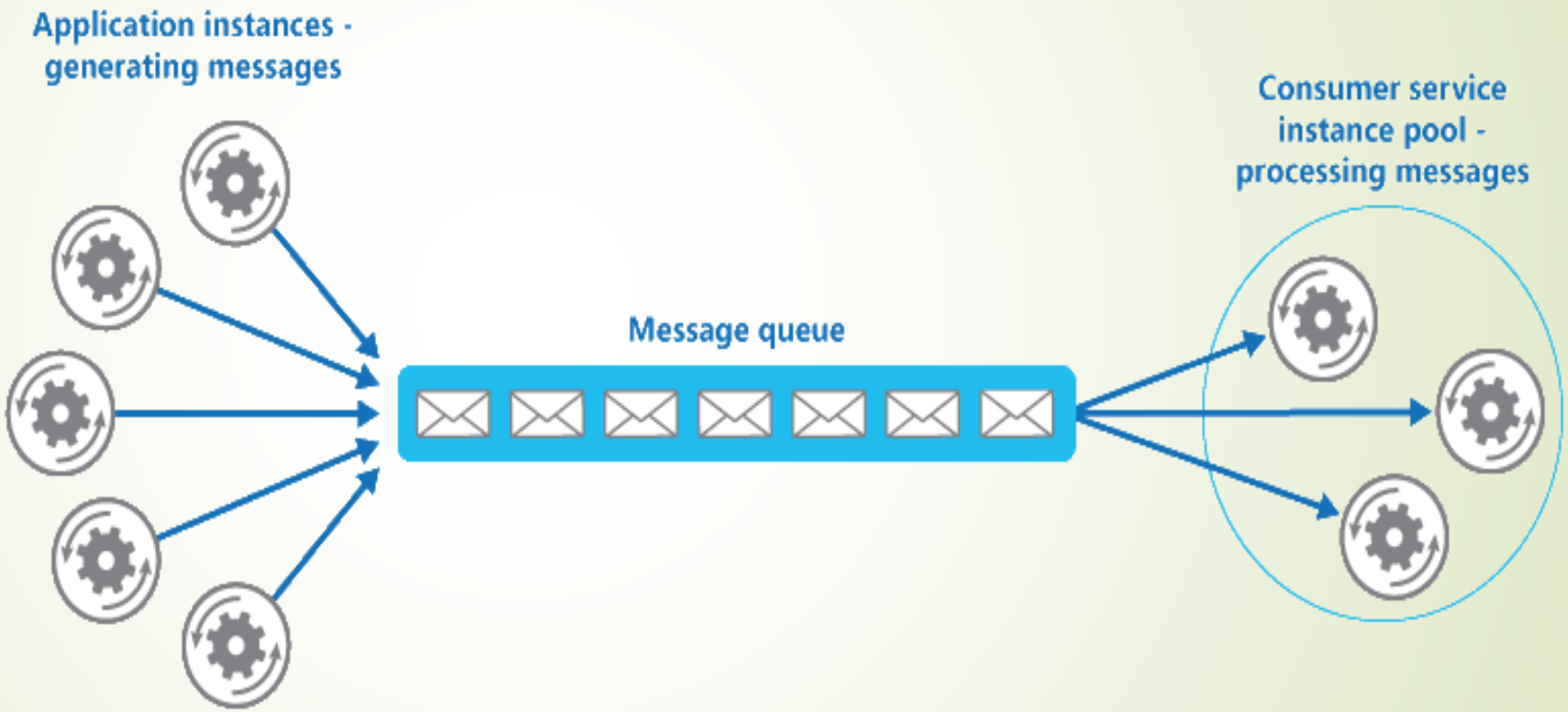
Message Queues

Sharjeel Tahir
Cole Vikupitz

What is a message queue?

- As the name implies, a queue of messages. Allows for inter-process communication.
- There are client applications (producers) that send messages to one another, are sent to a “mailbox”.
- There are other client applications that receives/processes these messages from the “mailbox” (consumers).
- This “mailbox” is called the message queue

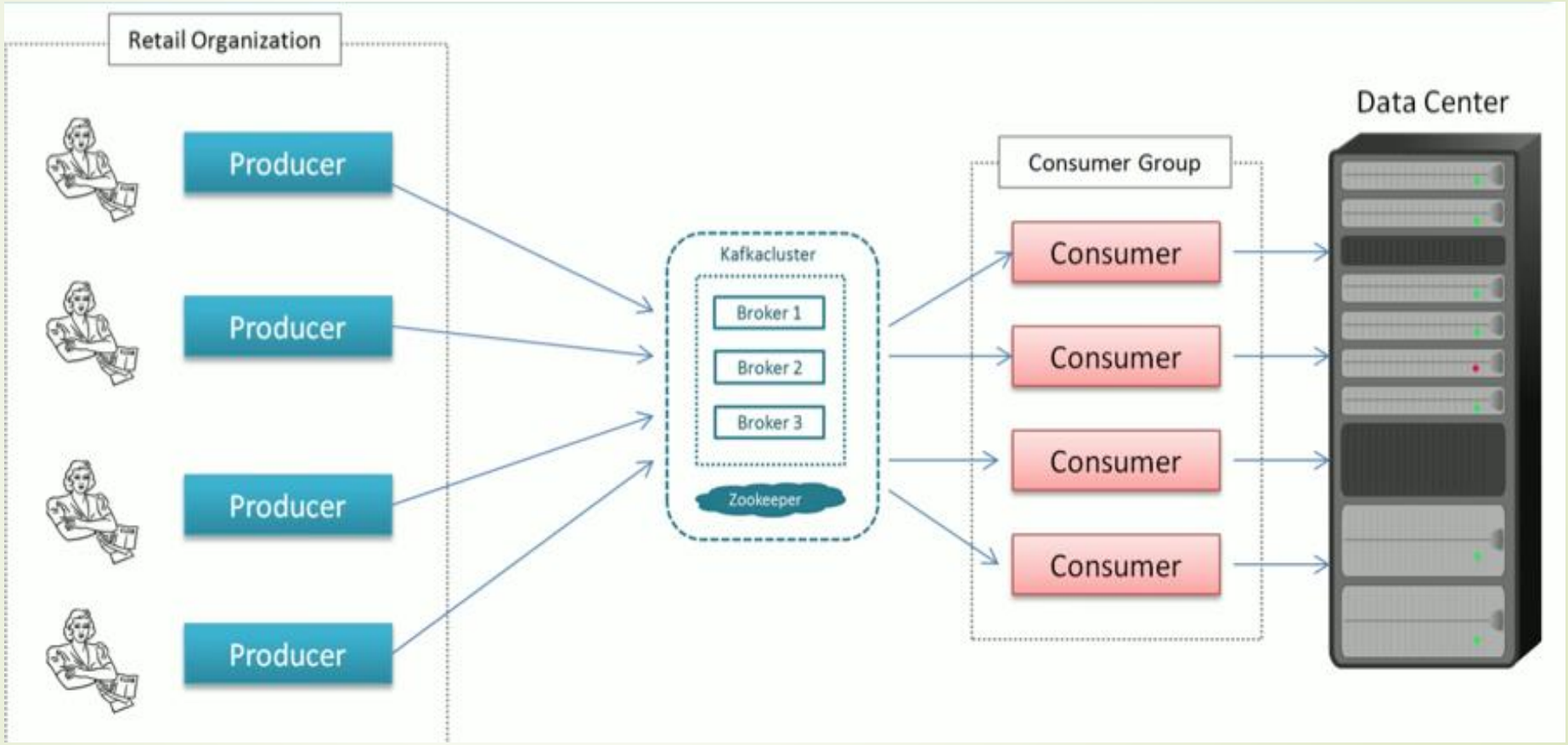
Basic Architecture



Why use message queues?

- Provides an asynchronous communications protocol.
- Decouples the consumers from the producers.
- Helps with traffic spikes; ensures all messages are delivered.
- Makes your system easily scalable.

Apache Kafka Architecture



Apache Kafka vs. Rabbit MQ

- RabbitMQ is broker-centric, focused on delivery guarantees between producers and consumers.
- Kafka is producer-centeric, based around partitioning a fire-hose of event data into durable message brokers, supporting batch consumers that may be offline, or online consumers that want messages at low latency.
- The whole job of Kafka is to provide the "shock absorber" between the flood of events and those who want to consume them in their own way -- some online, others offline - only batch consuming on an hourly or even daily basis.
- RabbitMQ ensures queued messages are stored in published order even in the face of requeues or channel closure.

Apache Kafka vs. Rabbit MQ (cont.)

- Difference in Open Source:
  - RabbitMQ: Mozilla Public License
  - Kafka: Apache License 2.0
- Difference in Language:
  - RabbitMQ: Erlang
  - Kafka: Scala (JVM)
- Federated Queues:
  - Yes for RabbitMQ and No for KafkaScaling
- Methods:
  - RabbitMQ: Vertical
  - Kafka: Horizontal type

Similarities: Rabbit MQ Apache Kafka

- Neither offers filter/processing capabilities.
- Both solutions run as distributed clusters.