ACP
Event- / Message-Brokers

Kafka / RabbitMQ / SQS

Michael Glienecke, PhD

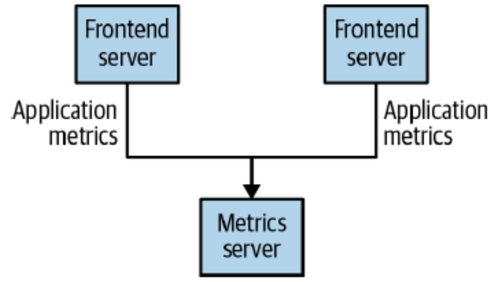
Welcome again

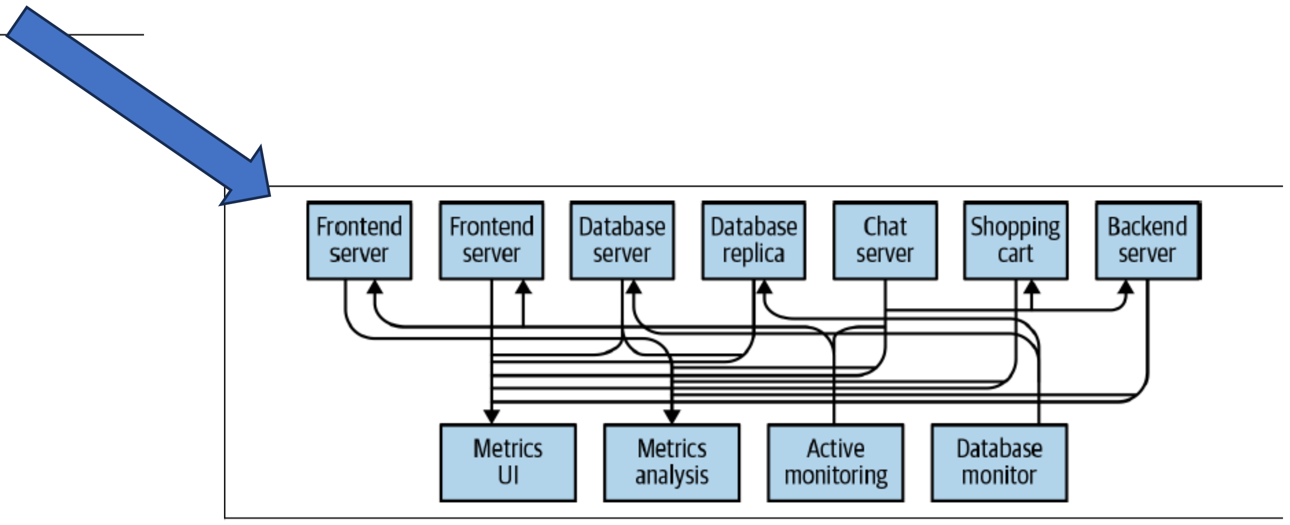
- The problem areas
- Kafka, RabbitMQ, SQS
- Topics / Queues
- Message formats (JSON, AVRO, etc.), Canonical formats
- Producer / Consumer
- chaining, flows
- event-driven applications, synchronization

What is the problem?

- Types of publishing data
  - Direct, Publish-Subscribe, Multi Publish Subscribe
- Events and messages
- Difference between Event- and Message-Broker

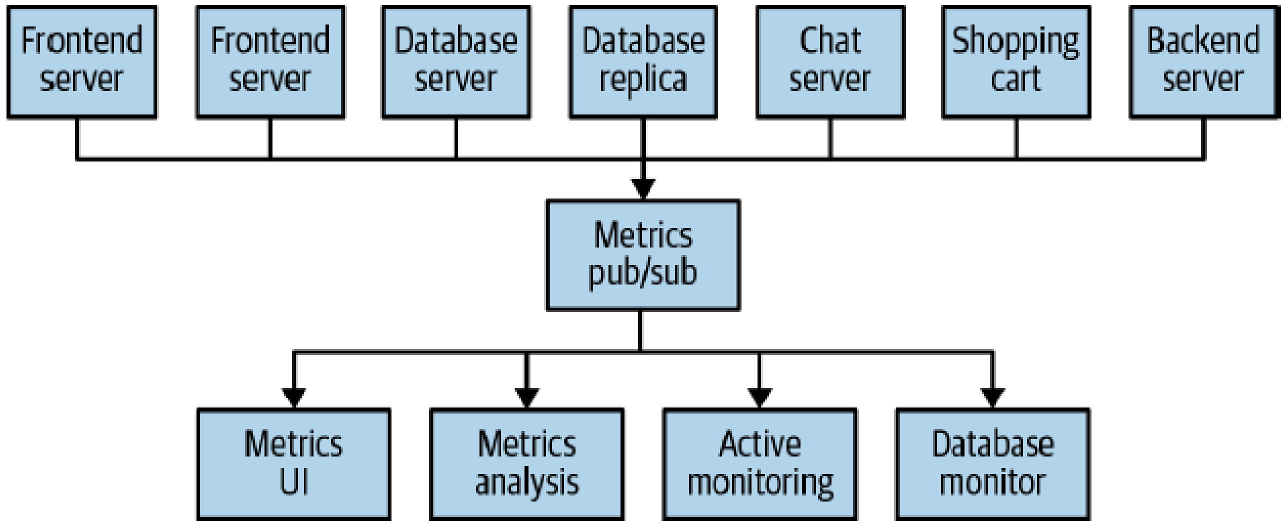
Direct publishing



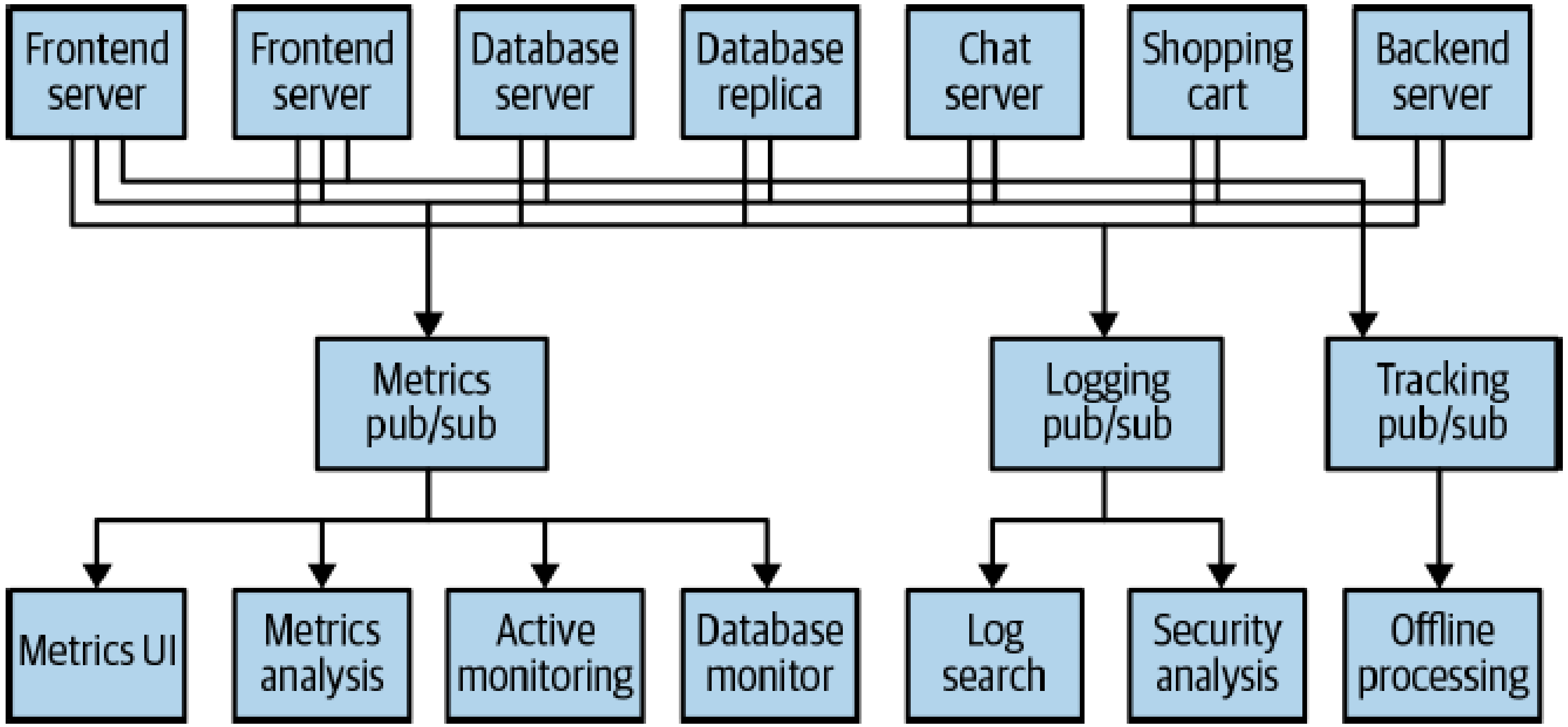


https://ebookcentral.proquest.com/lib/ed/detail.action?docID=6800597

Publish Subscribe



Multi publish-subscribe



Events and Messages

- Event and Message are highly overloaded terms
- Everybody has a different interpretation - often you find:
  - an event is an indication (like a signal) that something happened with some data as payload
  - A message is data exchange between communication partners (POST-request, gRPC-Request -> all messages)

- Kafka is unifying in this context: events, messages or records are the same
  - A message (or event) is a collection of bytes

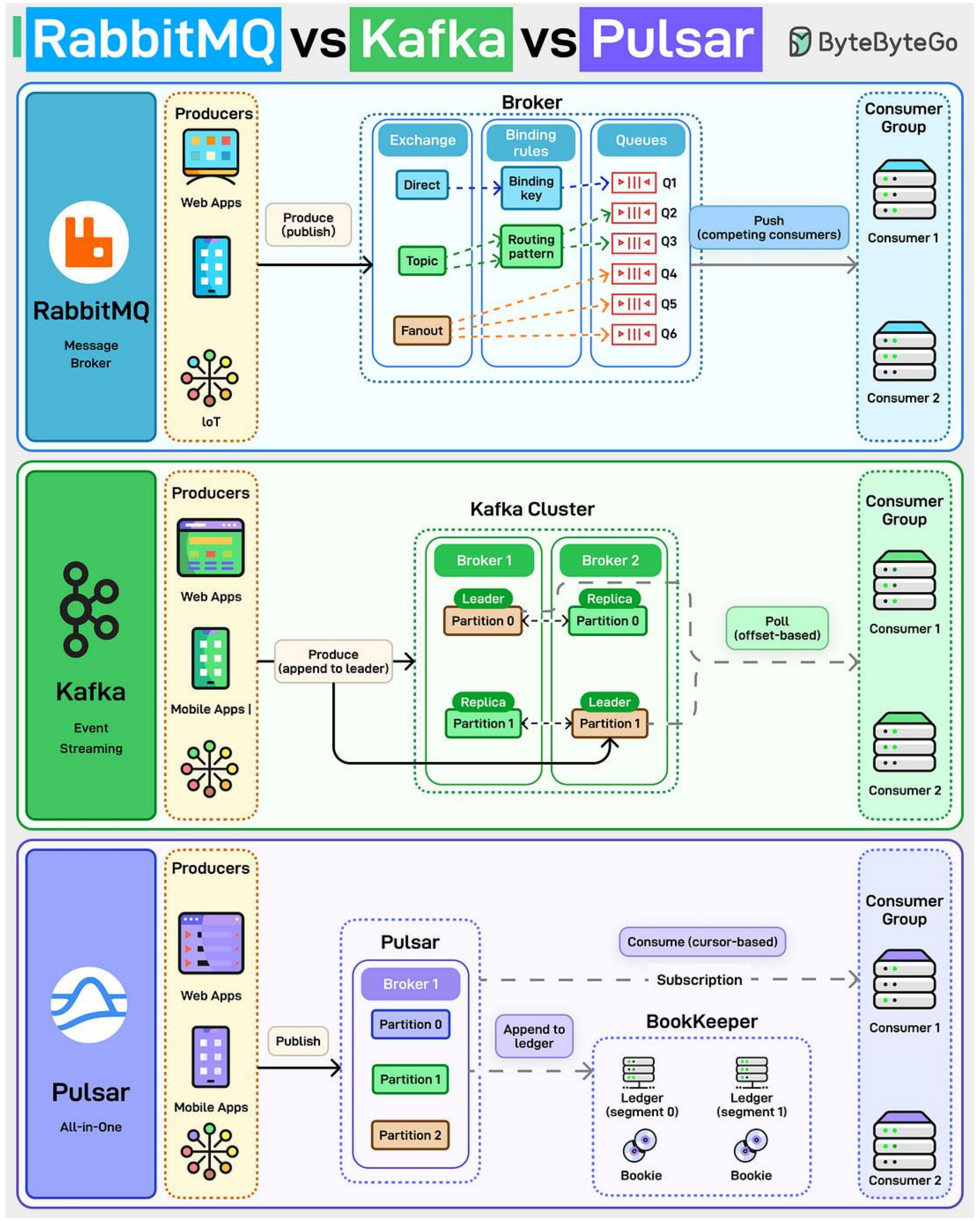
Difference between Event- and Message-Brokers

- Event-Broker
  - Stores a sequence of events usually appended to a log in order of arrival
  - Events are published to a queue or topic
  - Made available to multiple subscribers
  - Event sequence cannot be altered (Log)
  - Events can be potentially stored for a long time
- Message-Broker
  - Used for async exchanging data (messages) between components or services
  - Usually using queues where messages are stored for short periods of time
  - Messages are usually not stored
  - Sequence of message can be altered

https://medium.com/riskified-technology/message-broker-vs-event-broker-when-to-use-each-one-of-them-15597320a8ba (for more details)

Feature comparison

Feature	Kafka	RabbitMQ	SQS
Event Replay	✅ Native	❌ No	❌ No
High Throughput Streaming	⭐⭐⭐⭐	⭐⭐	⭐⭐⭐
Complex Routing	⭐	⭐⭐⭐⭐	⭐
Managed Service	Self / Managed	Self / Managed	Fully Managed
Best Fit	Data pipelines	App messaging	Cloud decoupling



THE UNIVERSITY of EDINBURGH informatics

https://blog.bytebytego.com/p/ep203-rabbitmq-vs-kafka-vs-pulsar?utm_source=post-email-title&publication_id=817132&post_id=188638970&utm_campaign=email-post-title&isFreemail=true&r=2tp8hj&triedRedirect=true&utm_medium=email

RabbitMQ as message broker

- RabbitMQ is one of the most popular message brokers (IBM MQ Series another) https://www.rabbitmq.com/
- Now (> 3.9) can handle streams as well (closing the gap)
- Very fast, very reliable
- Easy to use
- Built-in routing (if needed)
- If used properly, better than Kafka. Good analysis: https://www.cloudamqp.com/blog/when-to-use-rabbitmq-or-apache-kafka.html
- Interesting tutorials (logic applies to Kafka as well): https://www.rabbitmq.com/tutorials

SQS as message broker

- SQS is very simple to run (aws – no servers, no infra)
- Very fast, very reliable
- Easy to use (API)
- Extremely durable, fits nicely in the aws landscape (SNS, Lambda)
- Built in DLQ
- No event replay
- No advanced routing
- Not a streaming platform
- Concept of "visibility window" can be distracting

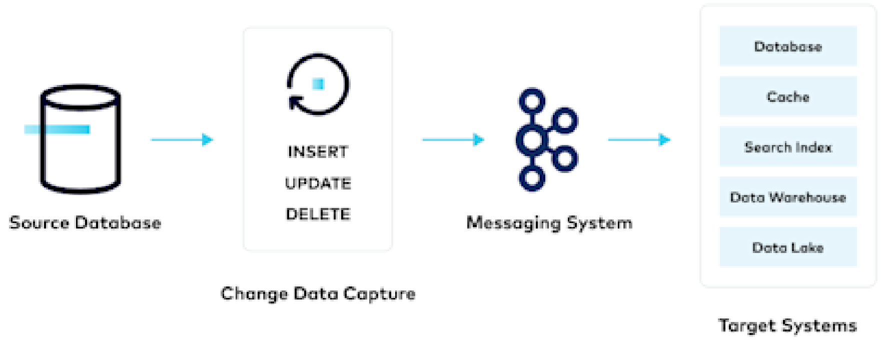
So, what is Kafka?

- Distributed streaming platform to handle large data volumes in real-time
  - Horizontal scaling
  - High throughput
- Publishers and subscribers are decoupled
- Messages are persisted to allow for multiple consumers

- Originated from work at LinkedIn
- Open source – commercial versions by confluent (www.confluent.io)

When to use Kafka

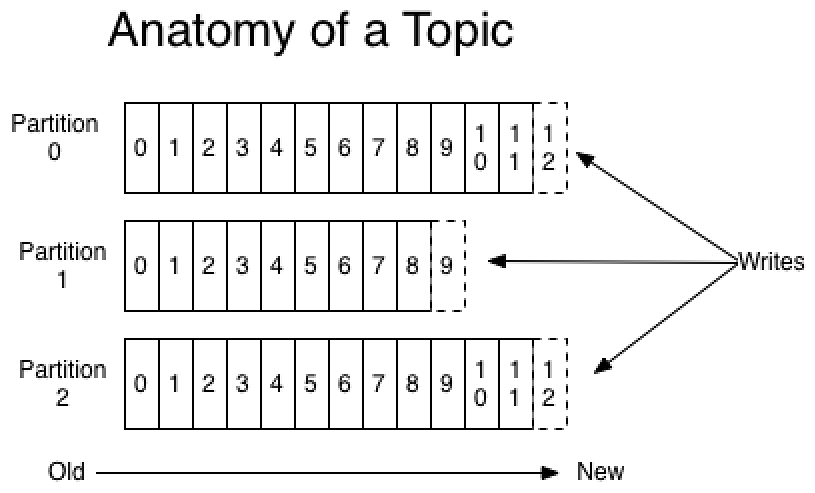
- When the additional features like Flink-Integration, kStreams, REST-Proxy, etc. matter
- Usually Large-scale environments - Tends to scale better horizontally
- Long-term storage of messages and replay to various receivers
- Often used for CDC (Change Data Capture)



https://www.confluent.io/en-gb/learn/change-data-capture/#:~:text=Change%20Data%20Capture%20(CDC)%20is,with%20microservices%2C%20and%20cloud%20adoption.

Topics

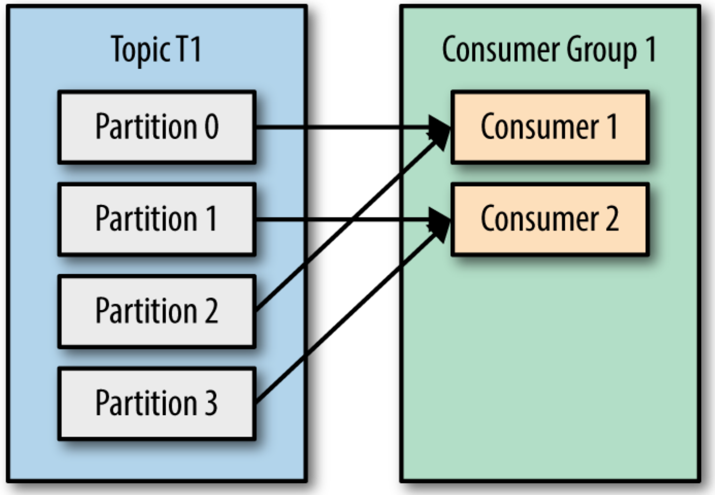
- Events are stored in topics
- A topic is the data pipeline between producer(s) and consumer(s)

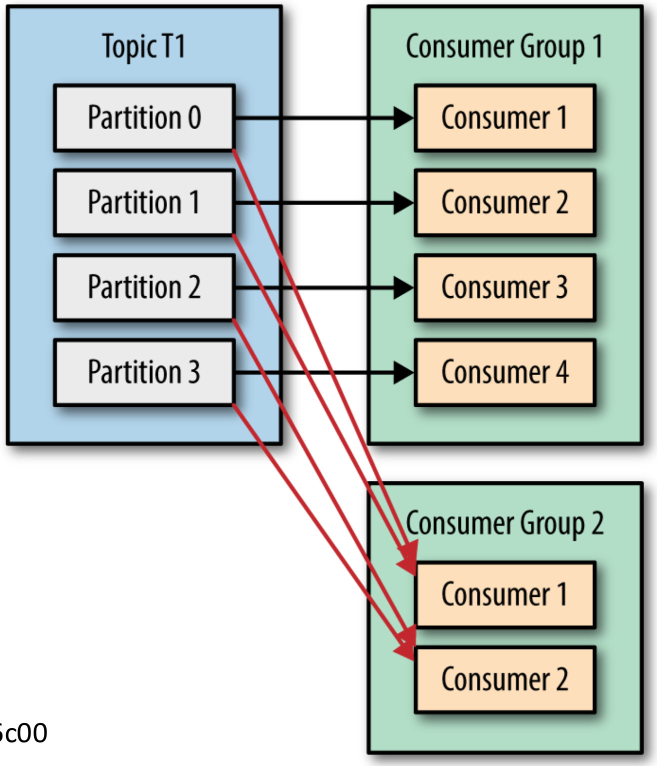


https://medium.com/javarevisited/kafka-partitions-and-consumer-groups-in-6-mins-9e0e336c6c00

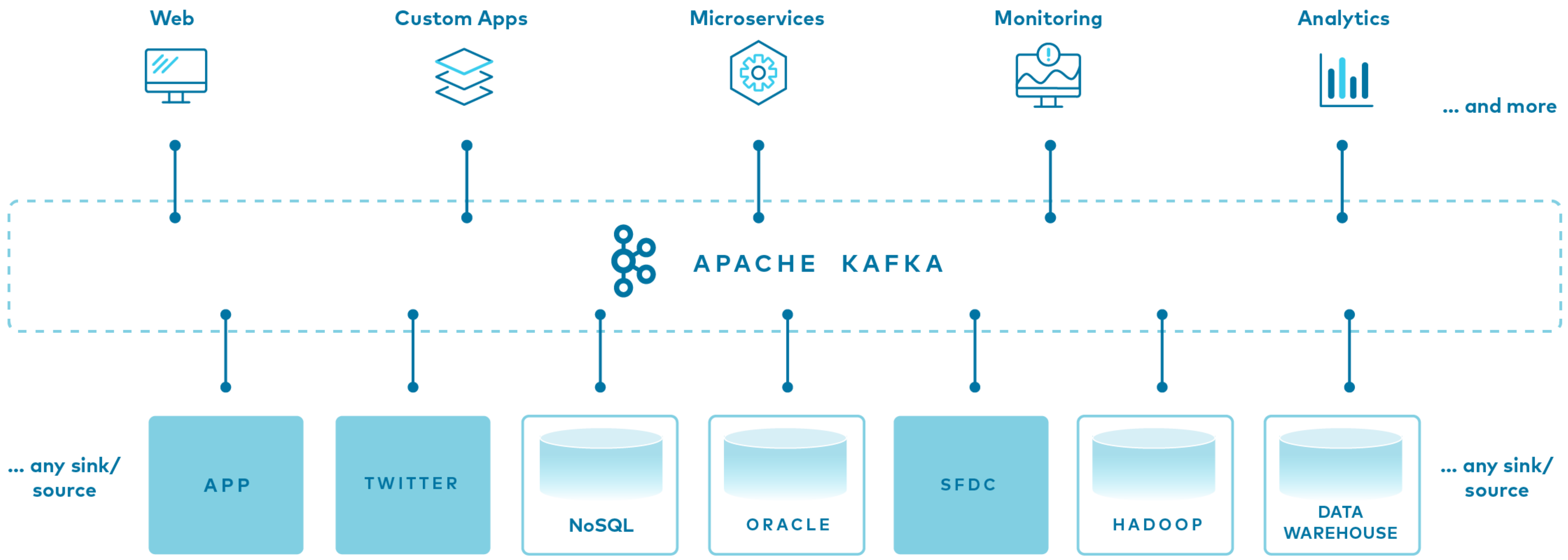
Partitions

- Partitions are used inside a topic to allow for parallelism
- Usually, one consumer per partition for maximum processing
- **Consumer groups** align consumers with partitions
  - Each group has its own offset (read position)





https://medium.com/javarevisited/kafka-partitions-and-consumer-groups-in-6-mins-9e0e336c6c00



For more info and many videos: https://developer.confluent.io/what-is-apache-kafka/

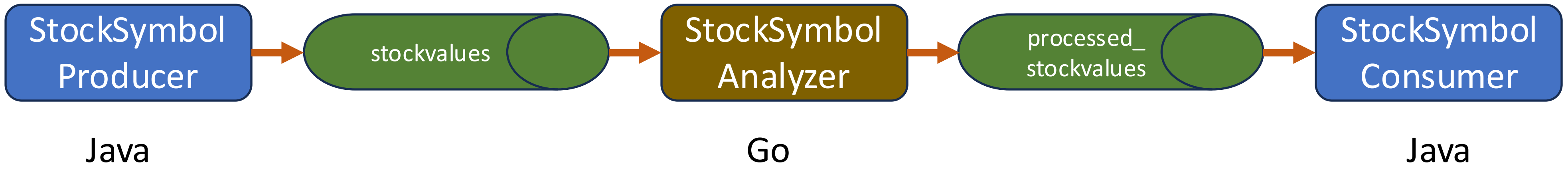
Message formats

- Usually JSON or AVRO
- Performance is similar, AVRO a bit smaller
- In larger scale, messages have to follow uniform pattern ("canonical message format") to allow common processing

  Often looks like
  - Outer packet (general & statistical data)
    - Inner packet (type of event, when, source)
      - Raw data for the event (specific to the event)

Our example environment (lecture + tutorial)

- 1 producer, 1 processor, 1 consumer



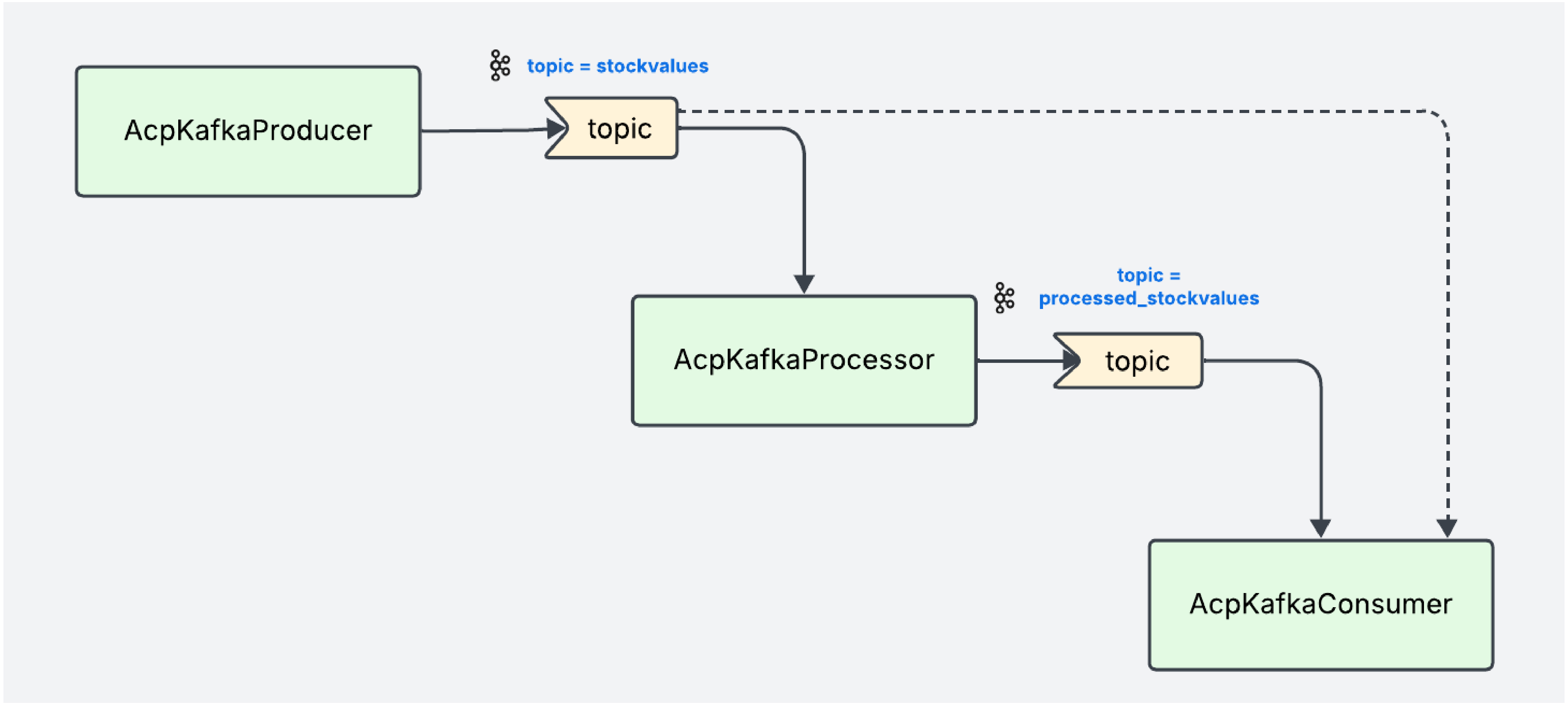
Some words about the upcoming assignment

- Essay
- Programming task

Git-Repos used for Lecture + Tutorial

- Producer:
- https://github.com/mglienecke/AcpKafkaProducer.git
- Processor:
- https://github.com/mglienecke/AcpKafkaProcessor.git
- Consumer:
- https://github.com/mglienecke/AcpKafkaConsumer.git

The big picture

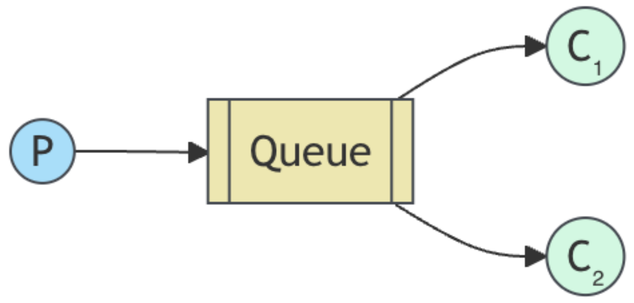


RabbitMQ

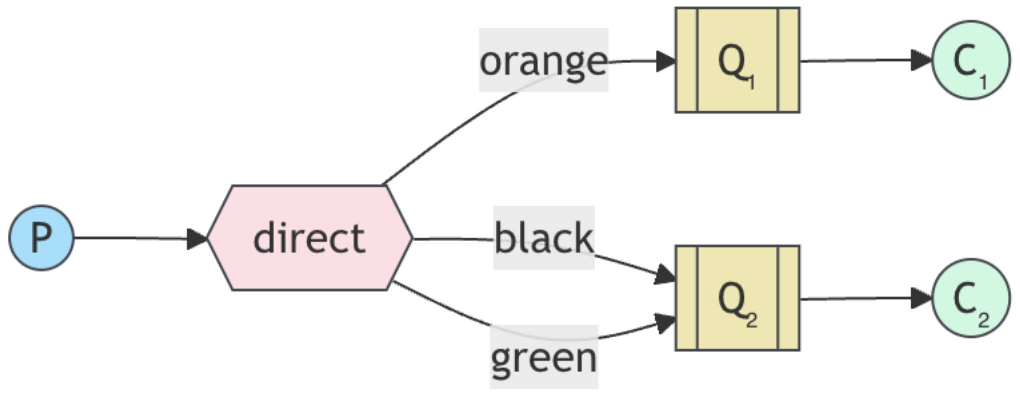
- Message broker with the option to have queues and streams
- Protocols:
  - AMQP
  - MQTT
  - STOMP (text on http)

- Can run in standalone, in docker or k8s – or cloud…

RabbitMQ – (most important) operation models



Direct queue



Exchange (routing / topics)

RabbitMQ – direct queues

- Either 1:1 communication
- Or distributing among receivers (only 1 receiver gets a unique message)

RabbitMQ – Exchange based

- Routing on keys (“red”, stock symbols, etc.)
- Routing on topics (xxx.yyy.zzz, uuu.vvv) by means of patterns
    - *.orange.*
    - *.inbound.*
    - #.data
    - outbound.#
    - * = 1 word, # = 0..n words
- The **exchange** does the magic…

RabbitMQ - installation

- https://www.rabbitmq.com/docs/download
- docker:
  - docker run -it --rm --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:4.0-management

  - Runs rabbitmq (5672) as well as the management UI (port 15672)