ACP Kafka

Michael Glienecke, PhD

Welcome again

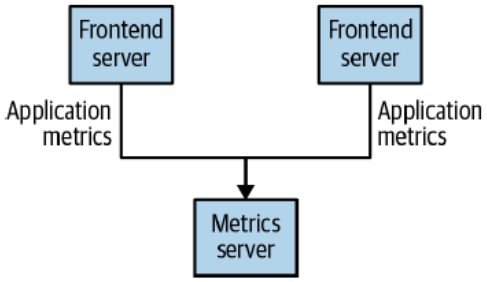
- The problem areas
- Kafka
- Topics
- Message formats (JSON, AVRO, etc.), Canonical formats
- Producer / Consumer
- chaining, flows
- event-driven applications, synchronization

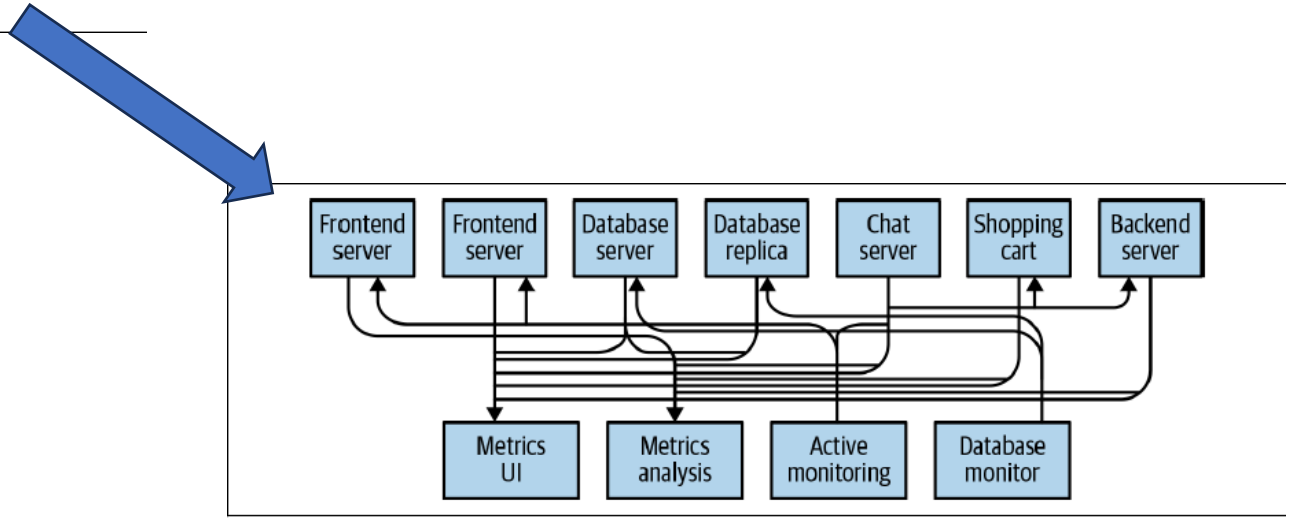
What is the problem?

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

Direct publishing

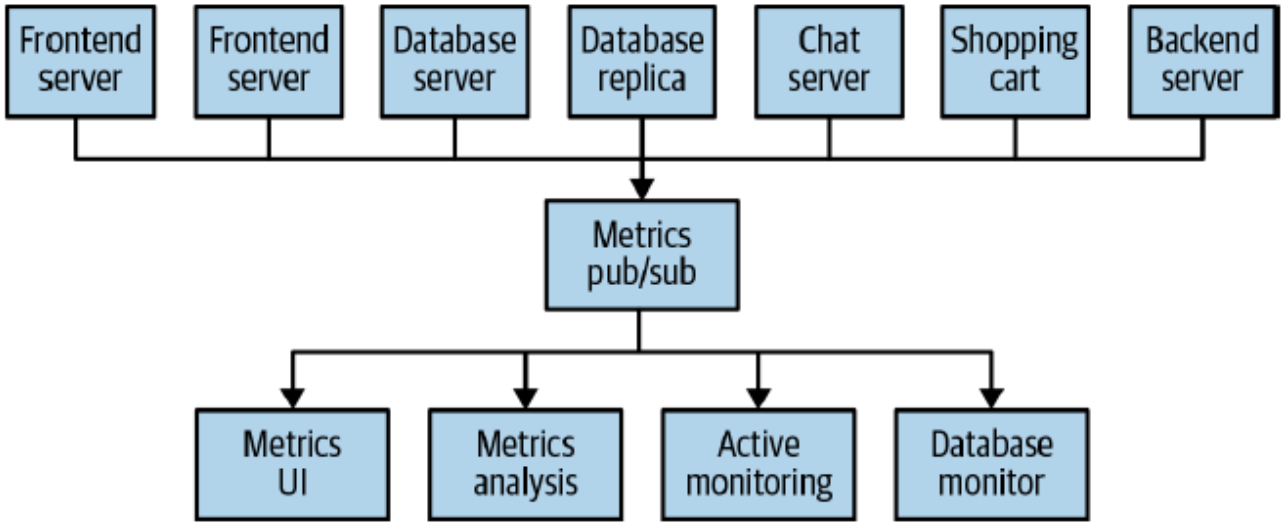
UDP can be used which is similar to telegram, where there is no guaranteed delivery but it is fast and anyone can listen to what is being broadcasted
Eg: Stock selling



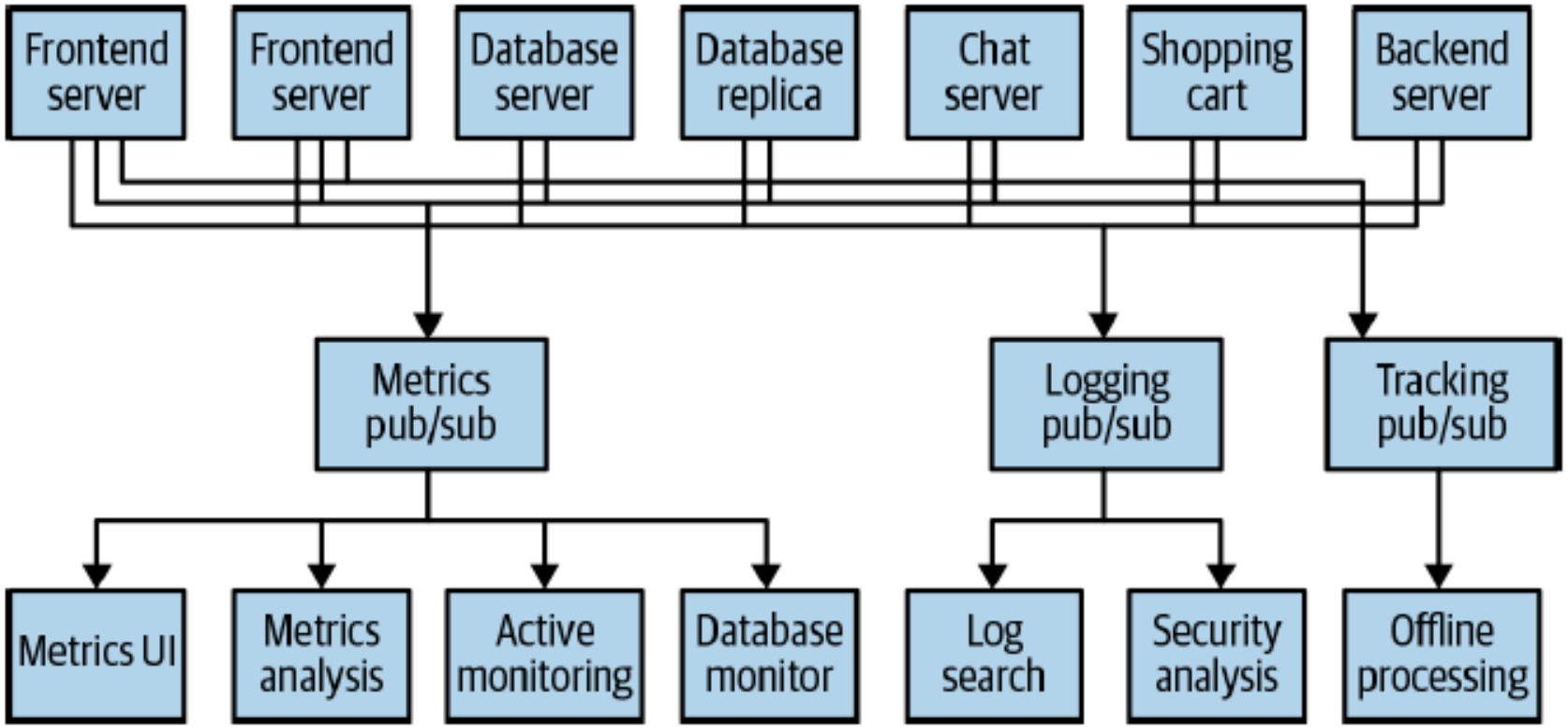


https://ebookcentral.proquest.com/lib/ed/detail.action?docID=6800597

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

Immutable

- 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)

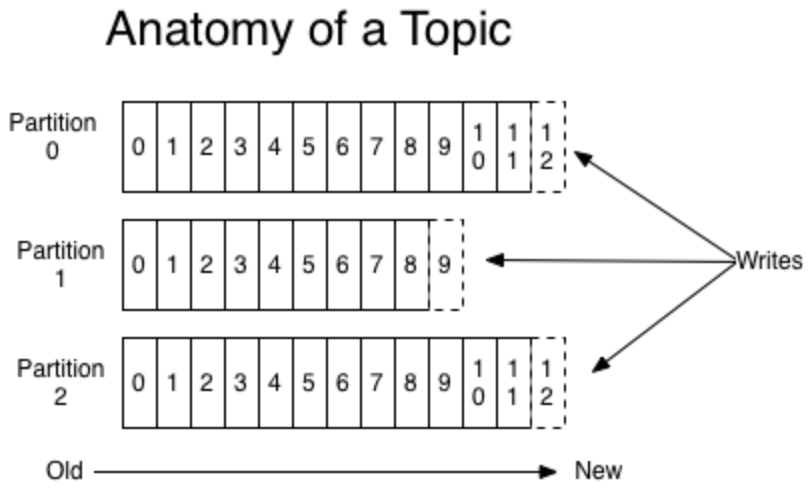
So, what is Kafka?

- 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)

Topics

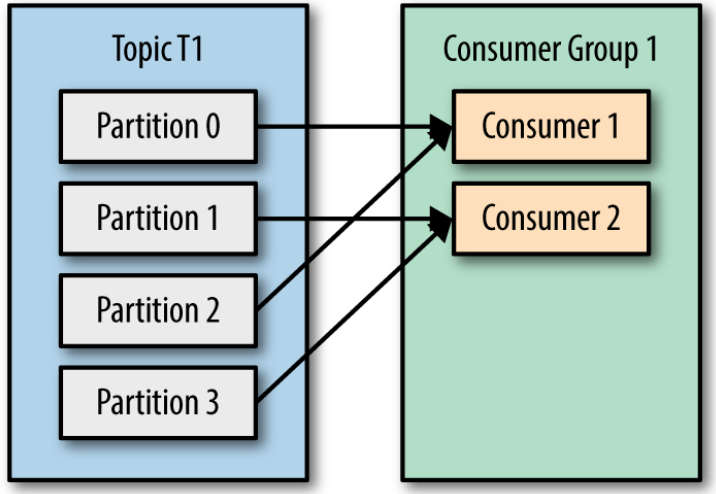
- Events are stored in topics
- A topic is the data pipeline between producer(s) and consumer(s)

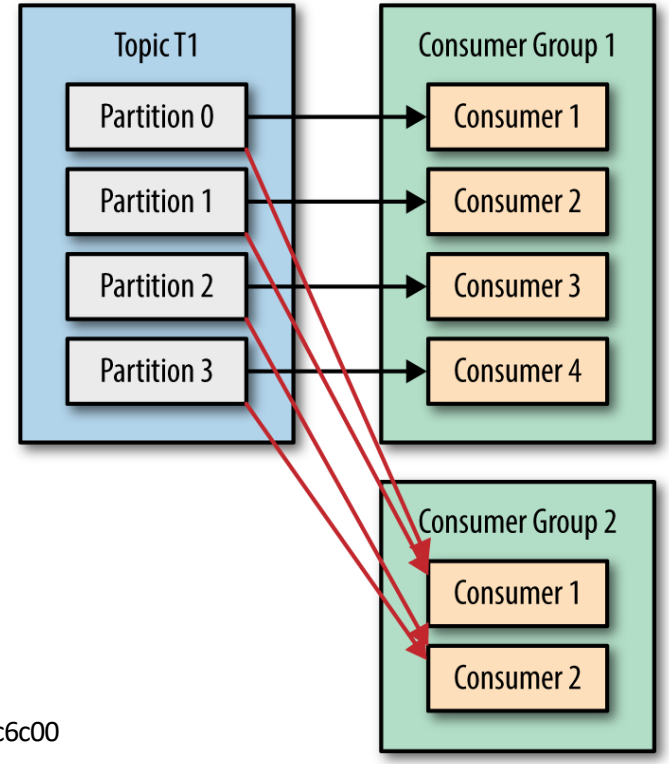


https://medium.com/javarevisited/kafka-partitions-and-consumer-groups-in-6-mins-9e0e336c6c00

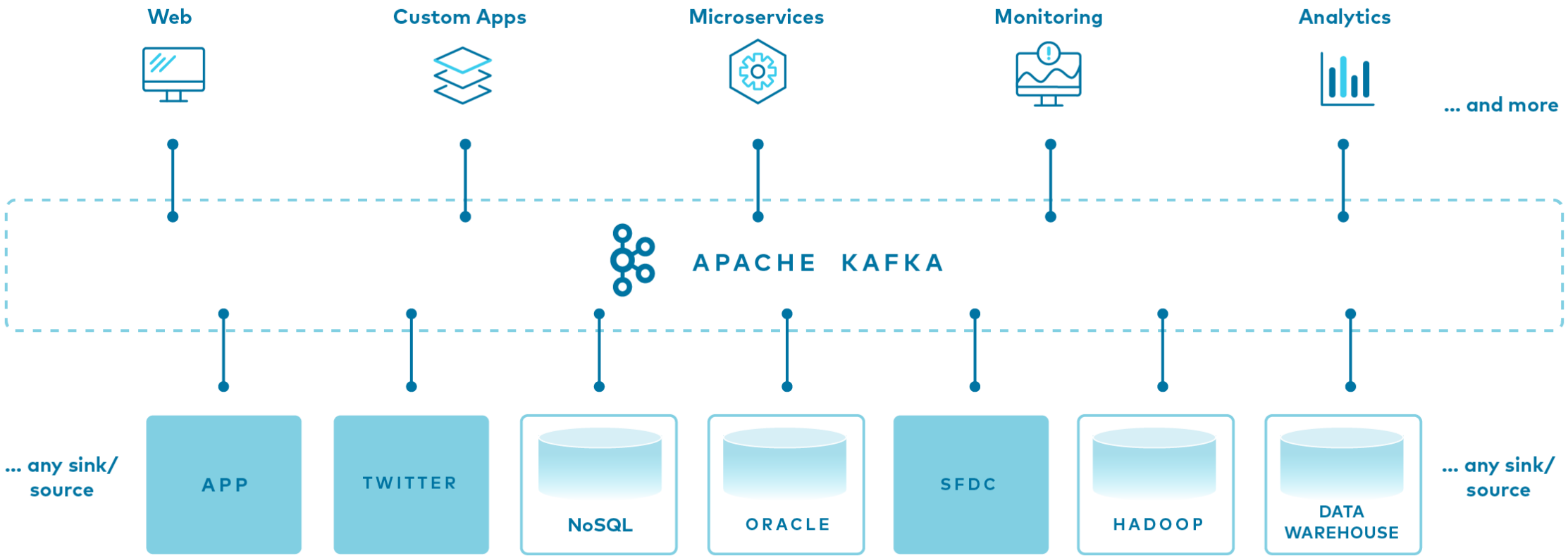
Partitions

- 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)





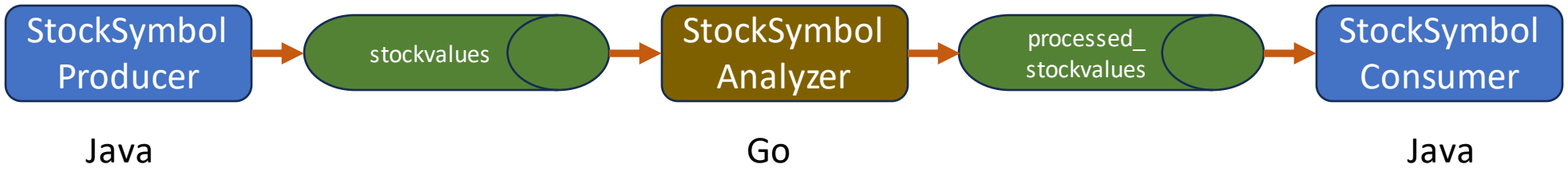
https://medium.com/javarevisited/kafka-partitions-and-consumer-groups-in-6-mins-9e0e336c6c00



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

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