streaming architectures

adam-bien.com

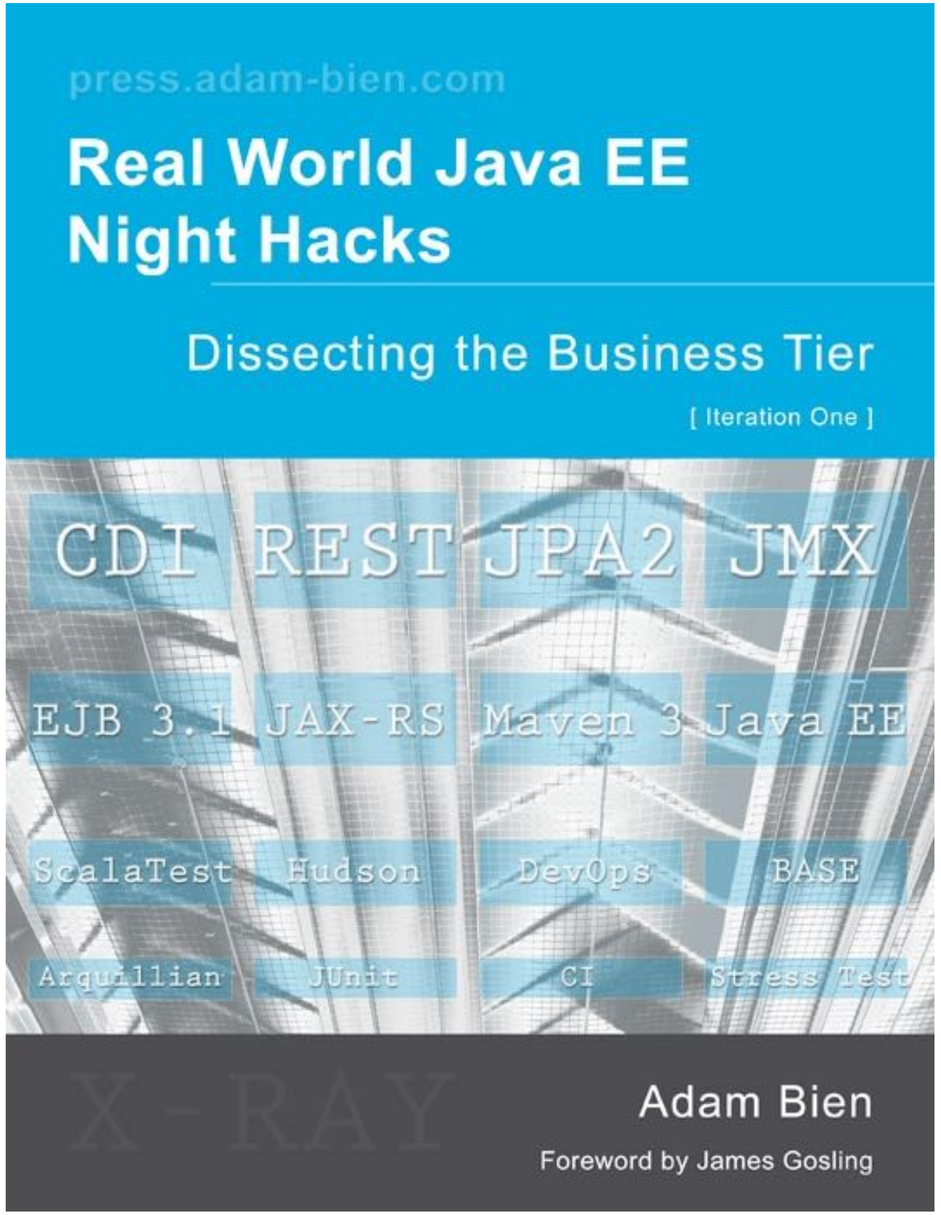
"It's not work if you like it"
…so I never worked. #java

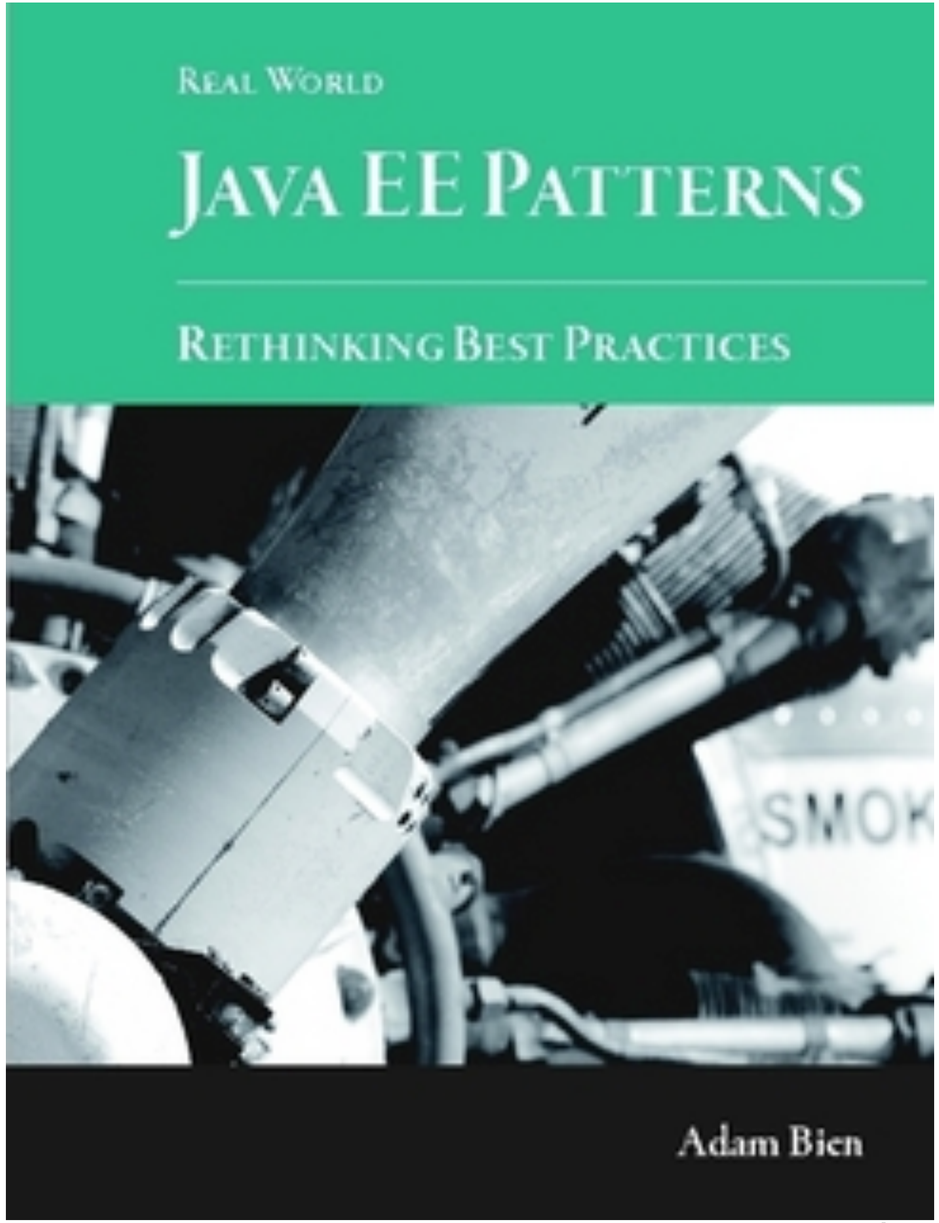
adam-bien.com

learn once,
never migrate

adambien.blog
airhacks.io
airhacks.tv
airhacks.news

airhacks.fm





adam-bien.com

airhacks.TV

adam-bien.com

Munich (MUC) Airport Web Workshops for Java Developers, Summer 2020:

WebComponents Kickstarter, June 30th, 2020

partially also available as: [Streaming / Download Edition]

Building Apps with WebComponents, July 1st, 2020

available as: [Streaming / Download Edition]

Munich (MUC) Airport Workshops, Winter 2019:

Jakarta EE / MicroProfile Microservices, December 10th, 2019

also available as: [Streaming / Download Edition]

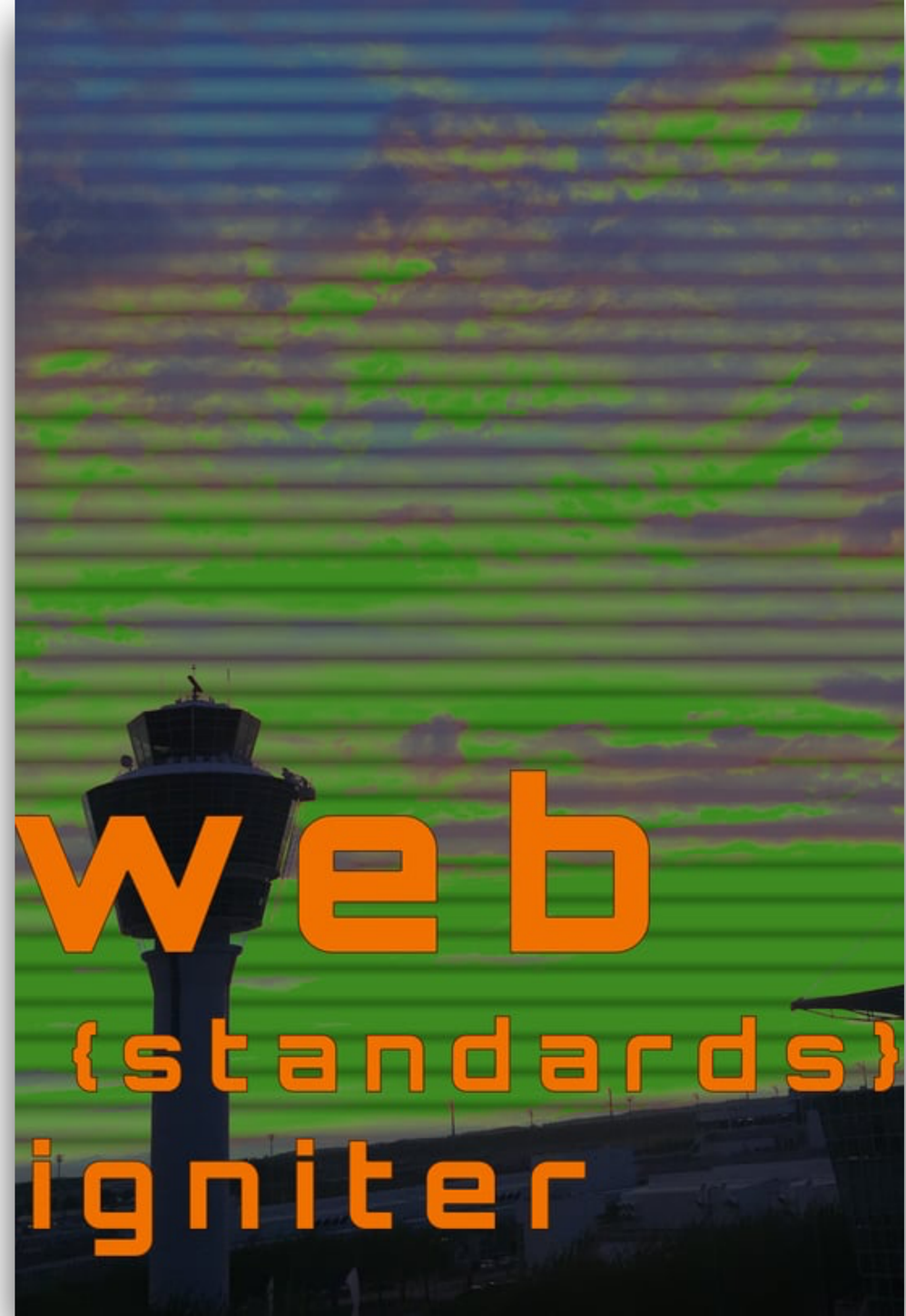
Jakarta EE, Microprofile and Clouds, December 11th, 2019

Beyond Boring Jakarta EE and JDK 12--From NoSQL, over Reactive to Fibers, December 12th, 2019

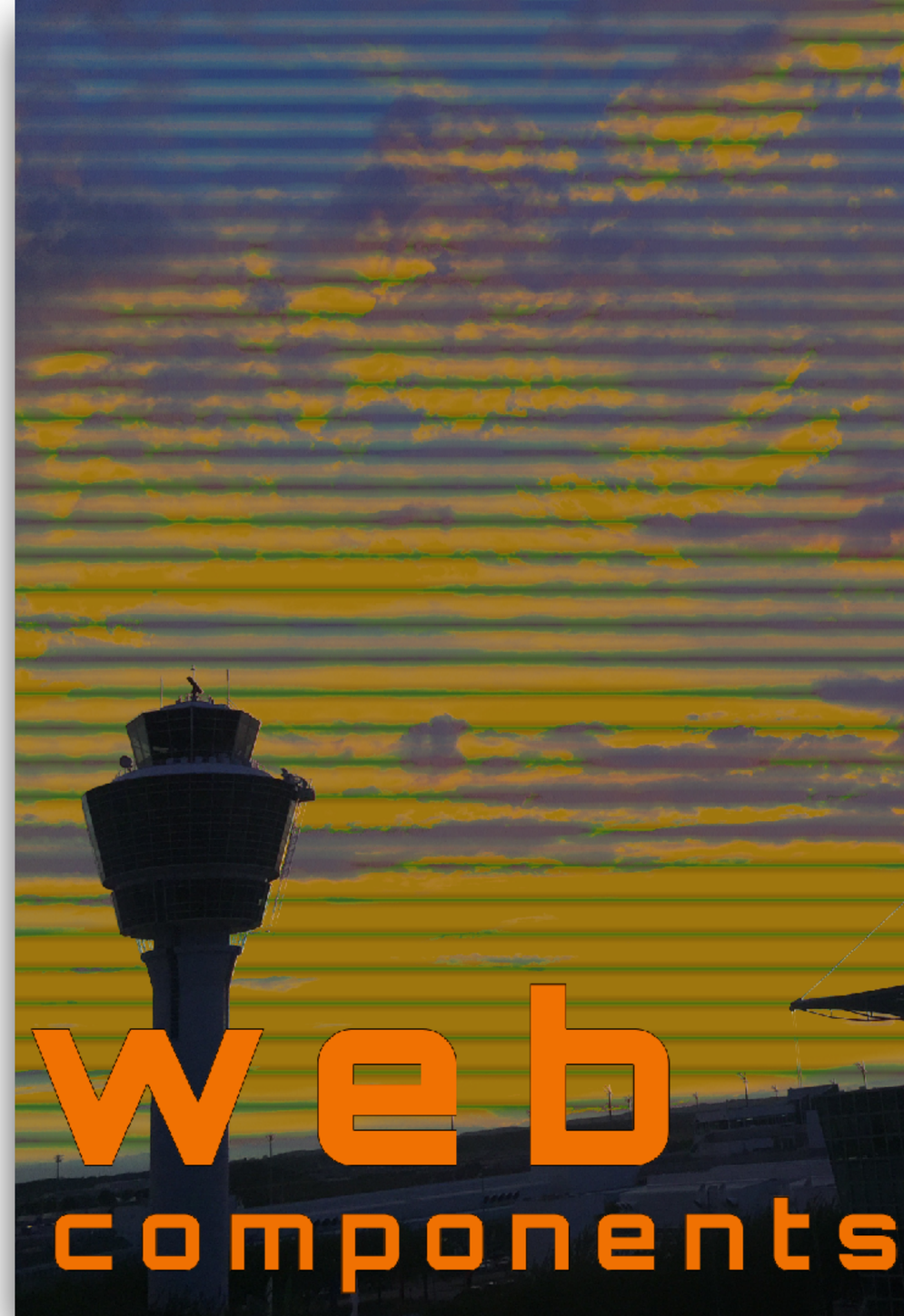
NEW:

Streaming Architectures, December 13th, 2019

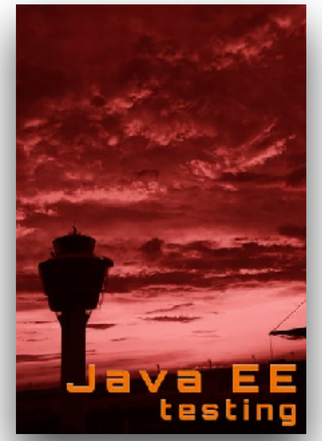
http://webstandards.training

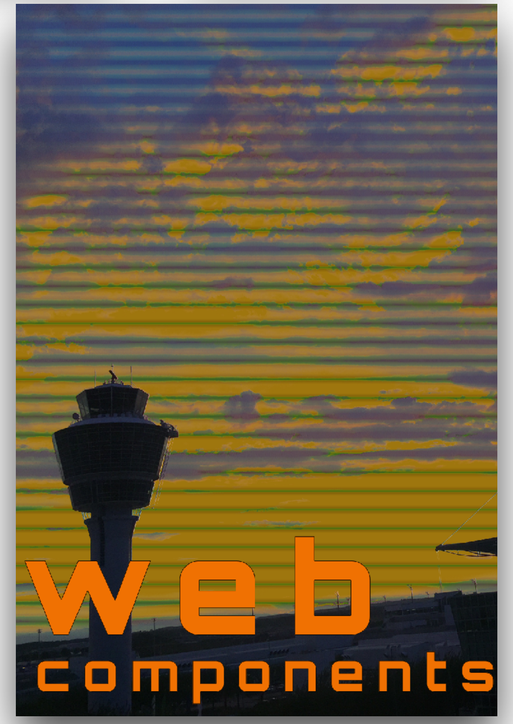


http://webcomponents.training



http://effectiveweb.training





airhacks.io

javaeemicro.services

microservices

Causality

Consensus

Consensus

- Agreement (=> nodes agree on state)
- Validity (=> agreed state is taken)
- Progress (=> no deadlocks)

Messaging

Messaging

- Publish Subscribe (Topics)
- Point to Point (Queues)

Delivery

- Persistent
- Non Persistent
- Redelivery
- Transacted

Persistence

- Durable Subscriptions
- Persistent Queues

Protocols vs. API

- AMQP
- Stomp
- MQTT

Microservice Communication

Message Driven vs. Event Driven

A message is an item of data that is sent to a specific destination. An event is a signal emitted by a component upon reaching a given state. In a message-driven system addressable recipients await the arrival of messages and react to them, otherwise lying dormant. In an event-driven system notification listeners are attached to the sources of events such that they are invoked when the event is emitted. This means that an event-driven system focuses on addressable event sources while a message-driven system concentrates on addressable recipients. A message can contain an encoded event as its payload.

https://www.reactivemanifesto.org/glossary#Message-Driven

Resilience is more difficult to achieve in an event-driven system due to the short-lived nature of event consumption chains: when processing is set in motion and listeners are attached in order to react to and transform the result, these listeners typically handle success or failure directly and in the sense of reporting back to the original client. Responding to the failure of a component in order to restore its proper function, on the other hand, requires a treatment of these failures that is not tied to ephemeral client requests, but that responds to the overall component health state.

https://www.reactivemanifesto.org/glossary#Message-Driven

could mean…

- A message is a DTO addressed to a specific destination
- An event describes something from the real world without being addressed to anyone

Stream Processing

Stream processing is a computer programming paradigm, equivalent to dataflow programming, event stream processing, and reactive programming,[1] that allows some applications to more easily exploit a limited form of parallel processing.

https://en.wikipedia.org/wiki/Stream_processing

Event stream processing

Event stream processing, or ESP, is a set of technologies designed to assist the construction of event-driven information systems. ESP technologies include event visualization, event databases, event-driven middleware, and event processing languages, or complex event processing (CEP). In practice, the terms ESP and CEP are often used interchangeably. ESP deals with the task of processing streams of event data with the goal of identifying the meaningful pattern within those streams, employing techniques such as detection of relationships between multiple events, event correlation, event hierarchies, and other aspects such as causality, membership and timing.

https://en.wikipedia.org/wiki/Event_stream_processing

Event Sourcing

Capture all changes to an application state as a sequence of events.

https://martinfowler.com/eaaDev/EventSourcing.html

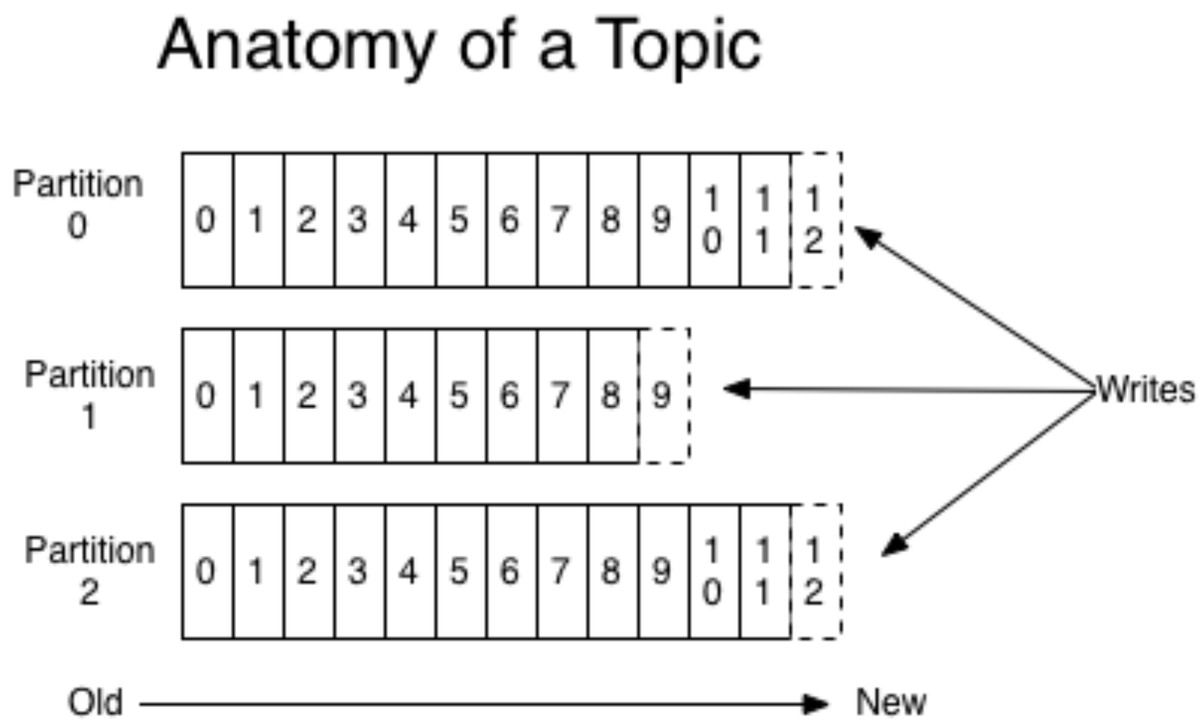
Event Stores

Kafka

Naming

- Subscriber => Consumer
- Publisher => Producer
- Stream => Topic
- Event => Message

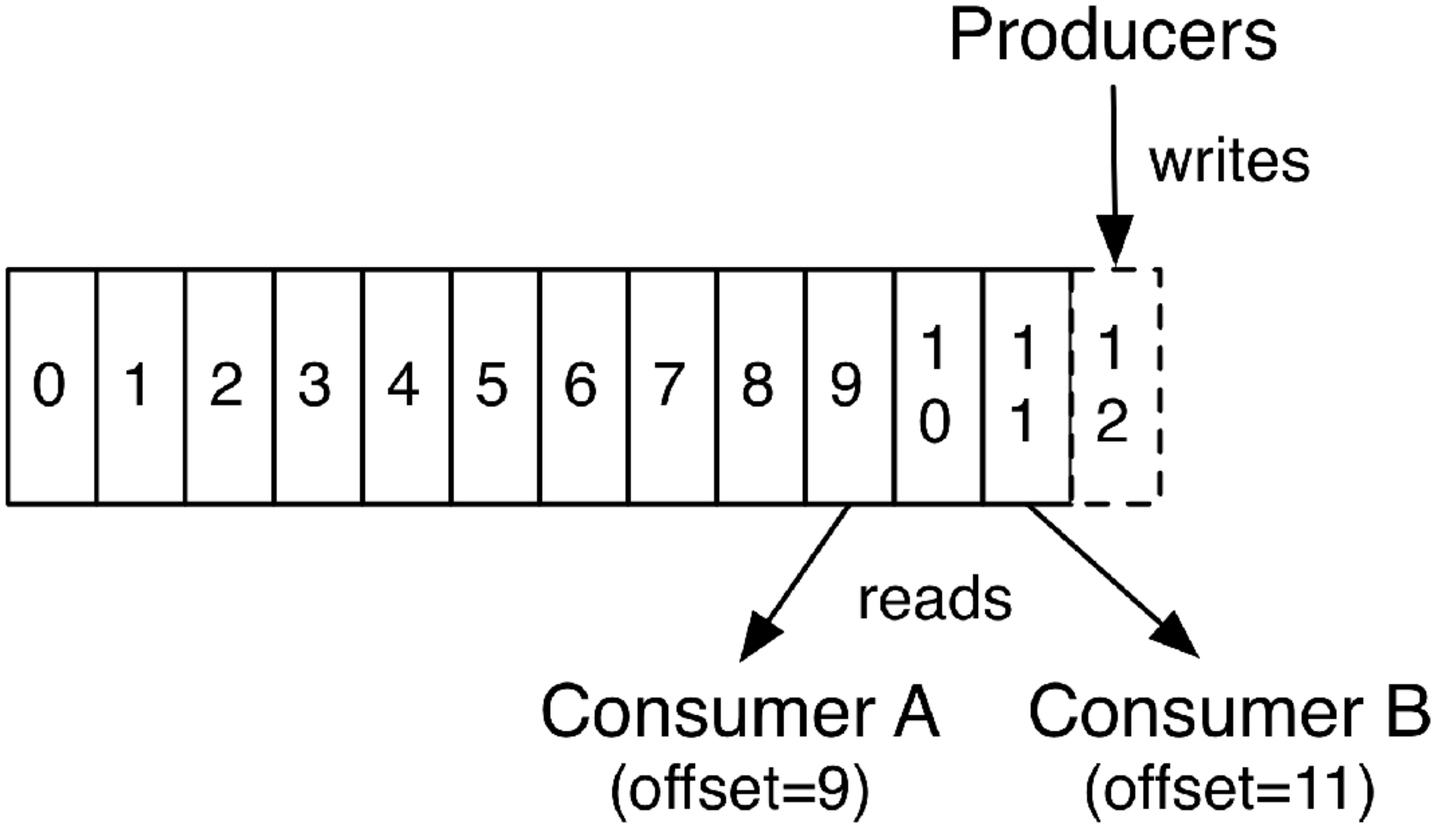
Partitions


Anatomy of a Topic
Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 12
Partition 1
0 1 2 3 4 5 6 7 8 9
Partition 2
0 1 2 3 4 5 6 7 8 9 10 11 12
Writes
Old
New

Distribution

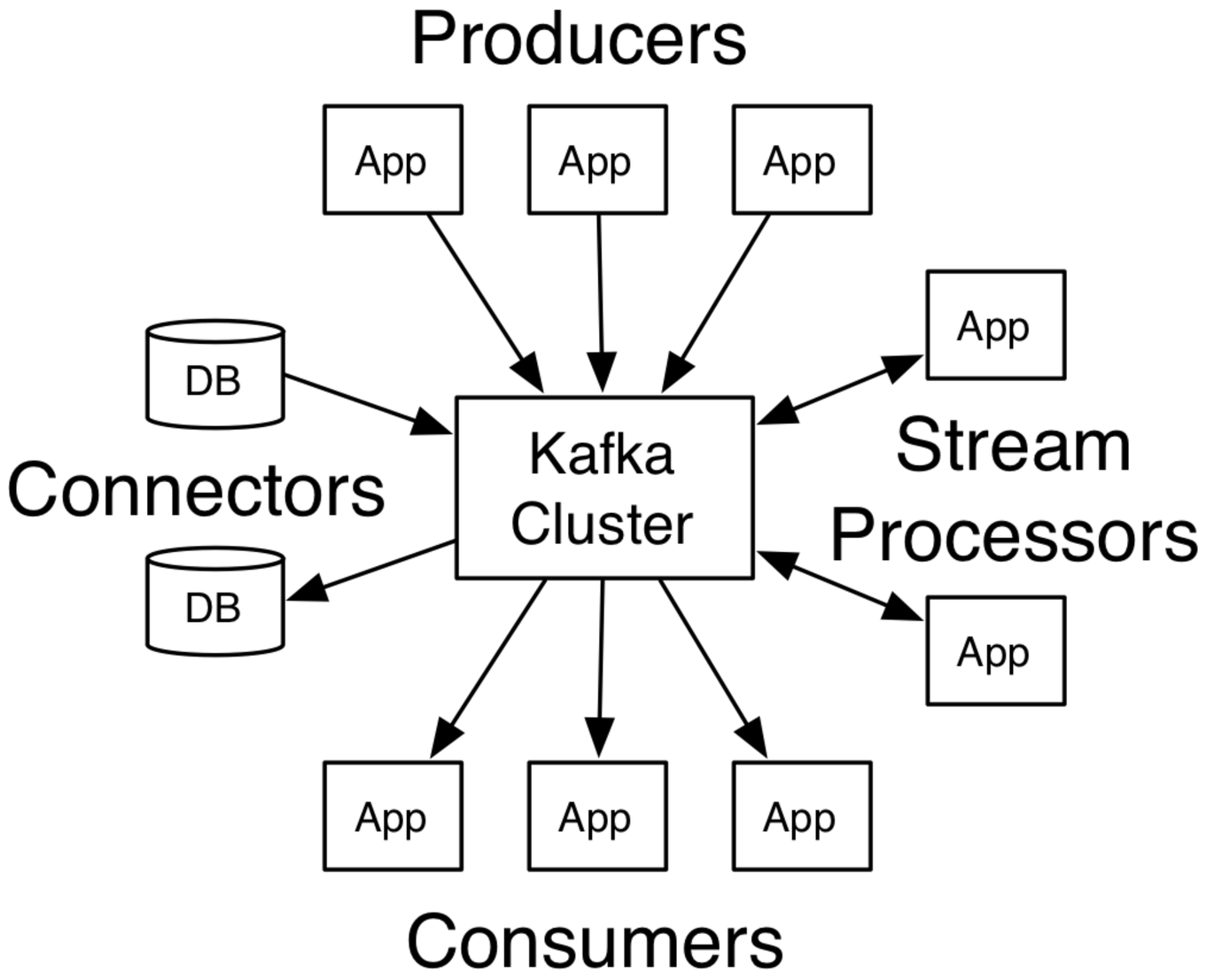
- Each partition has one leader (server) zero or more followers
- The leader handles all read and write requests and the followers passively replicate the leader
- On failure a re-election happens
- Each server acts as a leader for some of its partitions and a follower for others

Consumers


Producers
writes
0
1
2
3
4
5
6
7
8
9
10
11
12
reads
Consumer A
(offset=9)
Consumer B
(offset=11)

Consumers

- Consumer groups
- Load balancing
- Queues vs. Topics



https://kafka.apache.org/intro

Challenges

- Transactions
- Isolation Levels
- Message Order
- Delivery Semantics
- Duplicates
- Idempotency
- Latency

connectors

CDC

debezium

clouds

Munich (MUC) Airport Web Workshops for Java Developers, Summer 2020:

WebComponents Kickstarter, June 30th, 2020

partially also available as: [Streaming / Download Edition]

Building Apps with WebComponents, July 1st, 2020

available as: [Streaming / Download Edition]

Munich (MUC) Airport Workshops, Winter 2019:

Jakarta EE / MicroProfile Microservices, December 10th, 2019

also available as: [Streaming / Download Edition]

Jakarta EE, Microprofile and Clouds, December 11th, 2019

Beyond Boring Jakarta EE and JDK 12--From NoSQL, over Reactive to Fibers, December 12th, 2019

NEW:

Streaming Architectures, December 13th, 2019

Thank You!

blog.adam-bien.com
twitter.com/AdamBien