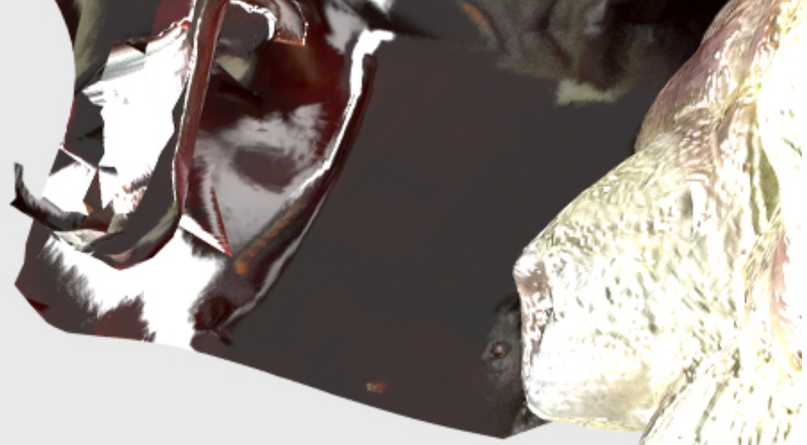


28.01.2020
Zürich / INNOQ Technology Night

Understanding Kafka

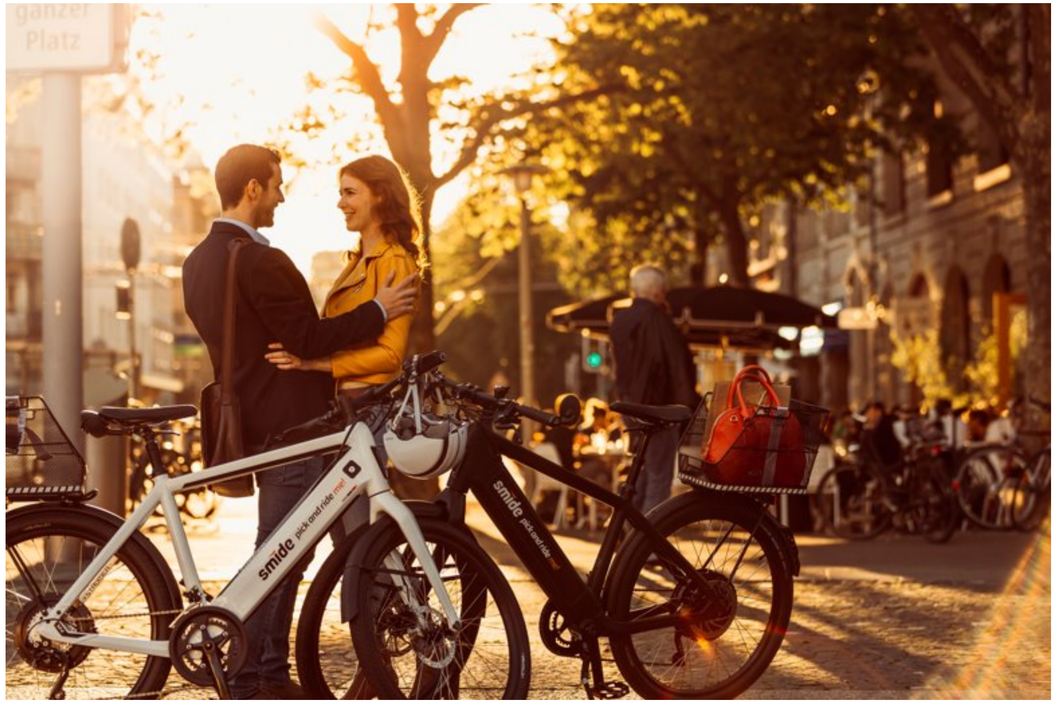
INNOQ

PATRICK KAUFMANN

Senior Consultant INNOQ Schweiz GmbH

patrick.kaufmann@innoq.com

Patrick Kaufmann ist seit 2016 Consultant bei INNOQ in der Schweiz. Er beschäftigt sich mit der Konzipierung, Umsetzung und Wartung komplexer fachlicher Microservice-Systeme beim Kunden, vorwiegend gebaut in Java, Spring Boot und Angular. Daneben begeistert er sich für Kafka, Event-getriebene Architekturen, funktionale Programmierung in Scala sowie skalierbare Softwaresysteme.

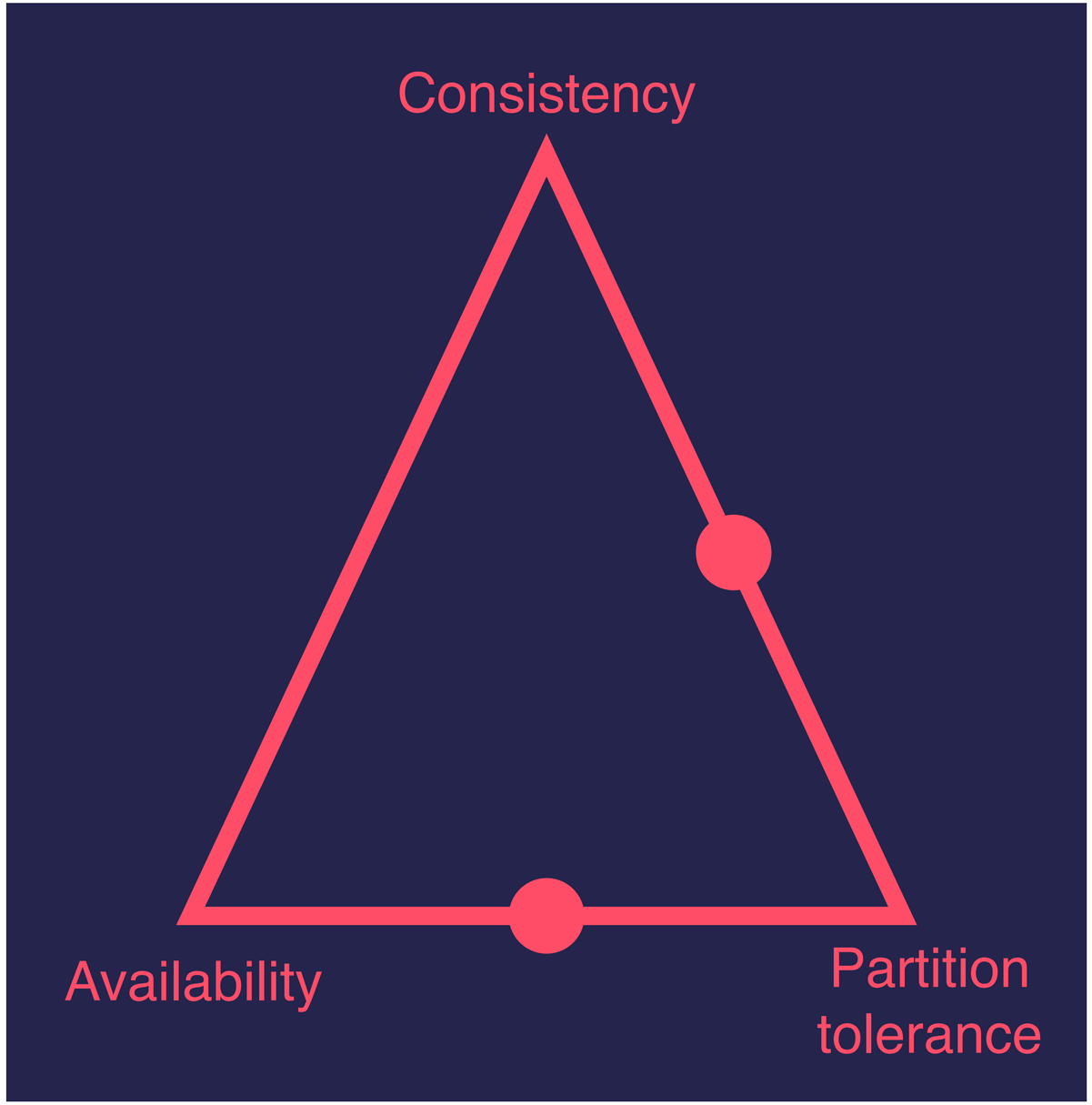


Was ist Kafka

- Verteilte Streaming-Plattform
  - Verteiltes Append-only Log
- Speichert Nachrichten in fehlertoleranter Architektur
- Erlaubt das Abarbeiten von Nachrichten in der Reihenfolge des Auftretens

Garantien

- Angenommene Nachrichten gehen nicht mehr verloren
- Reihenfolge der Nachrichten bleibt bestehen



Gewünschte Garantien

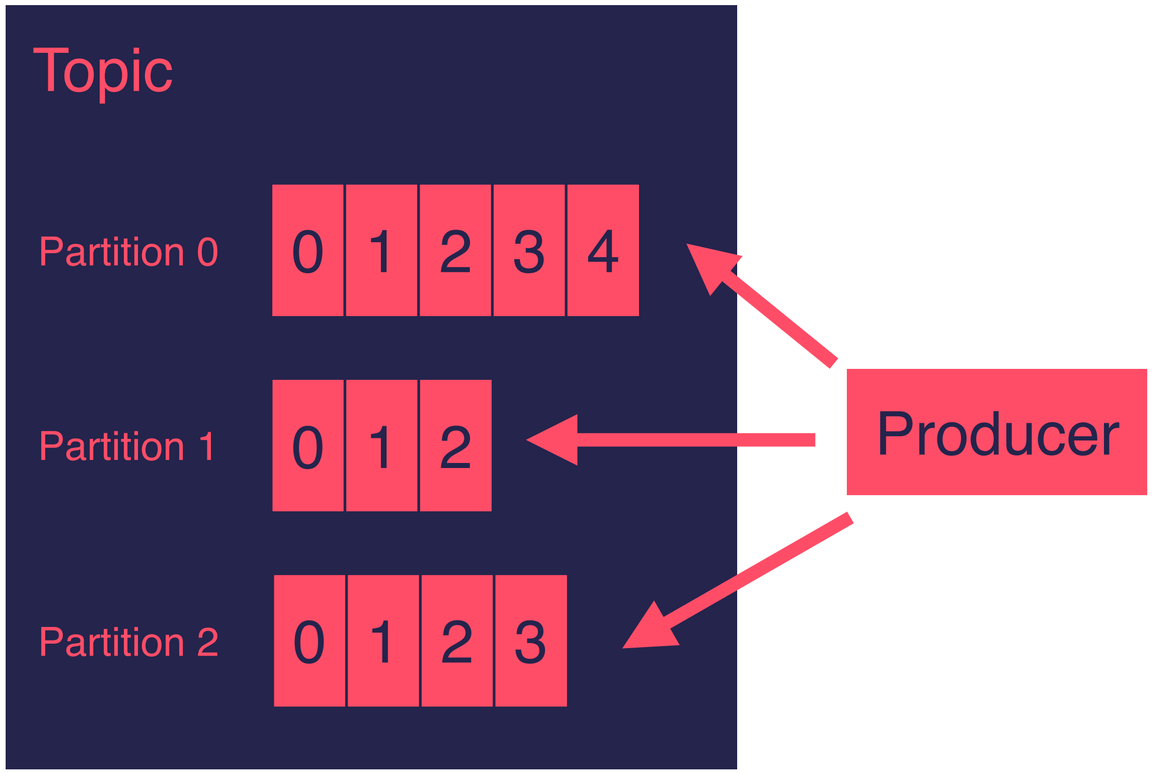
- Nicht alle Systeme brauchen CP-Garantien, AP kann zweckmässig sein
- Kafka bietet beide Modi an
  - Welche Parameter müssen konfiguriert werden um gewünschte Garantien zu gewährleisten?
  - Kafka 0.7: ~100 Parameter
    Kafka 2.4: ~400 Parameter
- Tradeoff Geschwindigkeit -> Sicherheit sowie Durchsatz -> Latenz

Topics und Partitionen

- 1 Topic enthält aus 1-n Partitionen
- Partition weisst jedem Record beim Schreiben einen Offset zu
- partitions = 1: Reihenfolge eindeutig gegeben
- Reihenfolge der Records nur pro Partition garantiert
- Partitionen bestimmen Parallelität

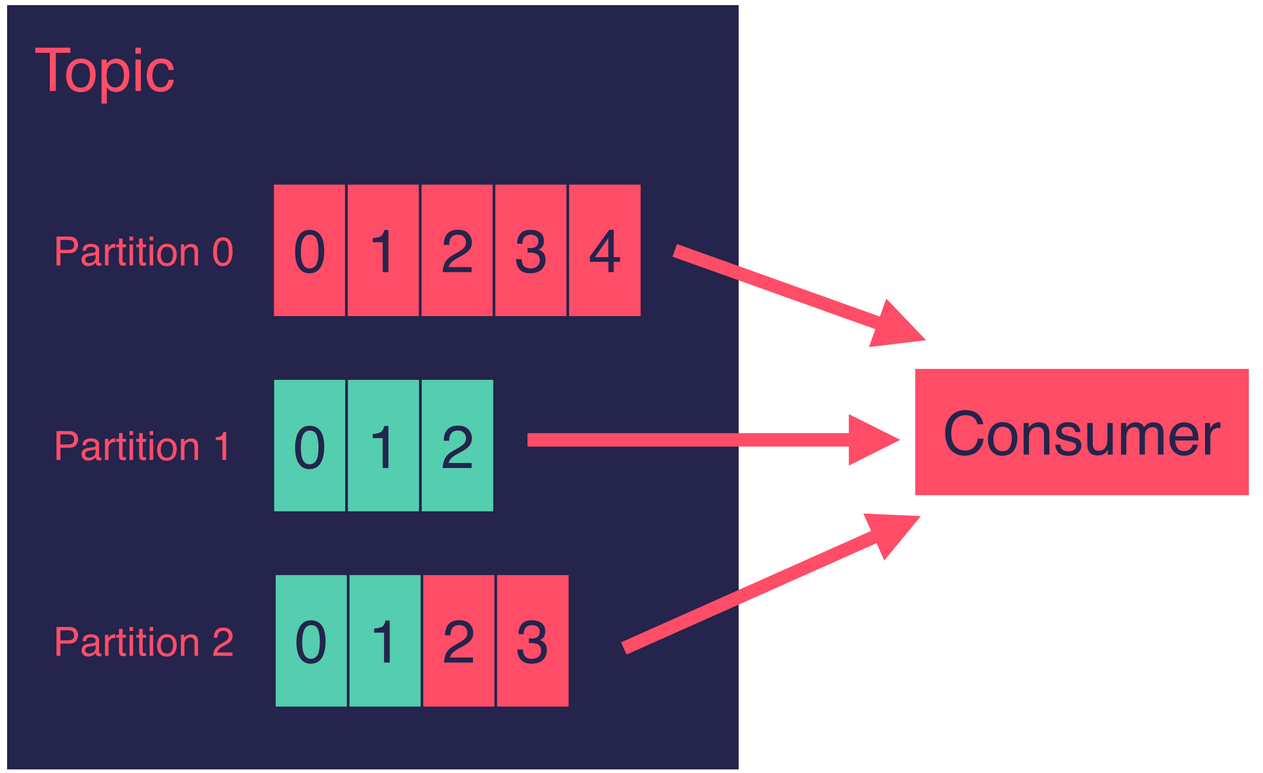
Producer

- Producer senden Records an Partition
- Records bestehen aus: Record-Key, Wert und Zeitstempel
- Bestimmt Partition des Records
  - Ohne Record-Key: Round-Robin
  - Mit Record-Key: Hash des Keys
- Record-Key != Partition-Key



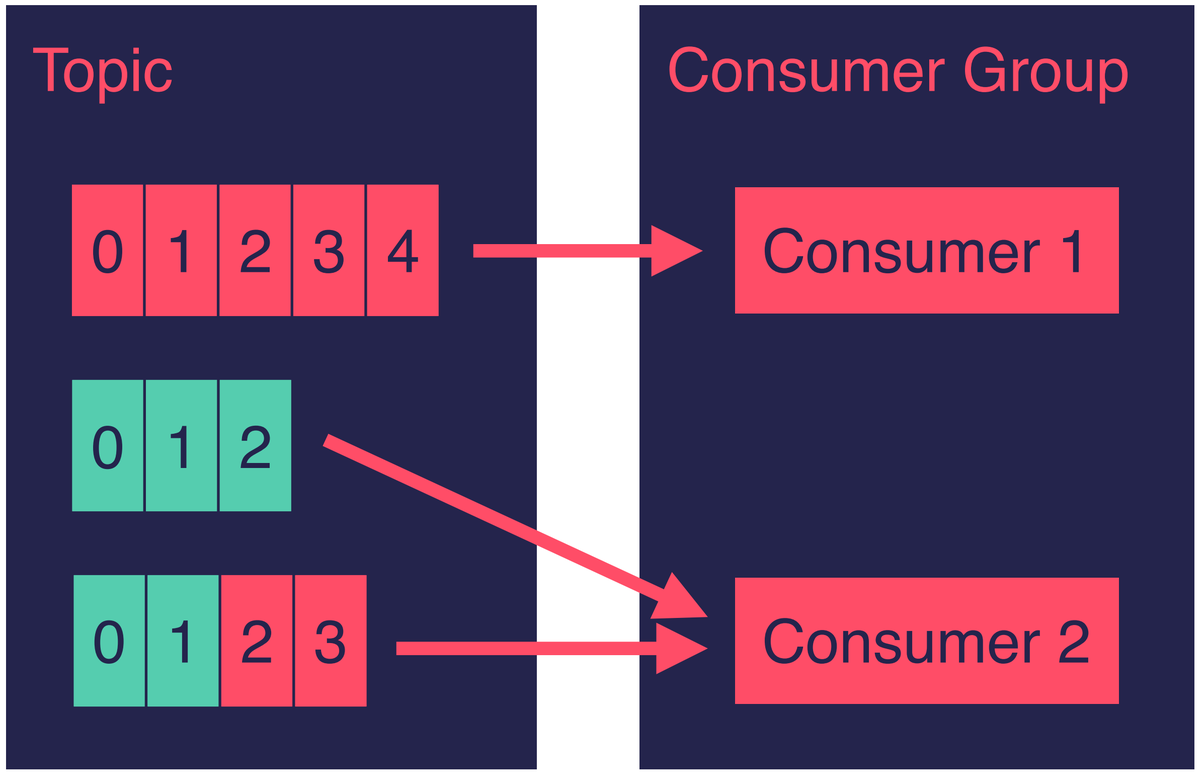
Consumer

- Consumer lesen Records aus 1-n Partitionen
- Pull-basiert
- Merkt sich Offset pro Partition
  - Entweder in Kafka
  - Oder lokal



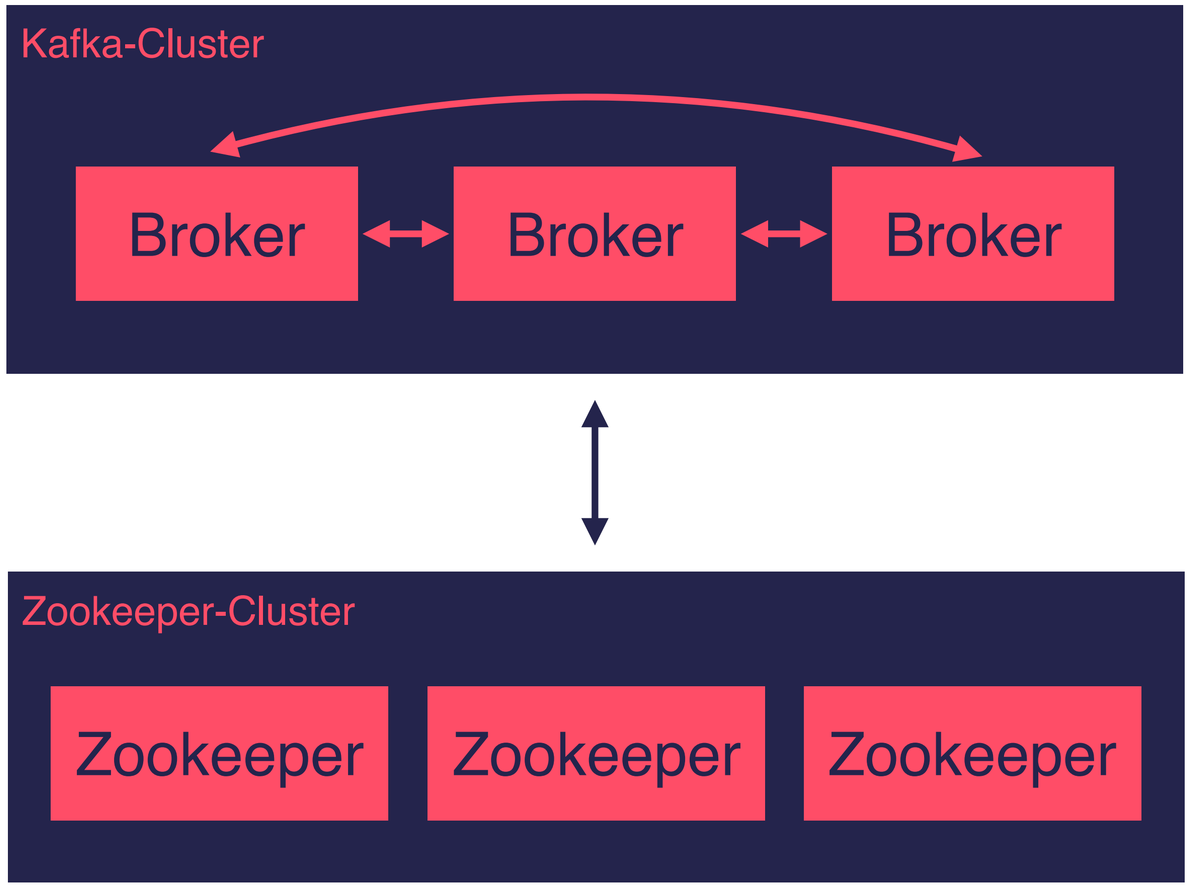
Consumer Groups

- Besteht aus 1-n Consumer
- Eine Partition kann Consumer Group nur von 1 Consumer gelesen werden
- Anzahl Partitionen = maximale Anzahl Consumer = Parallelität
- heartbeat.interval.ms = 3s, session.timeout.ms = 10s: Kann zu regelmässigen Consumergroup-Rebalances führen
- max.poll.interval.ms = 300s, max.poll.records = 500: Timeout bei langsamen Consumern



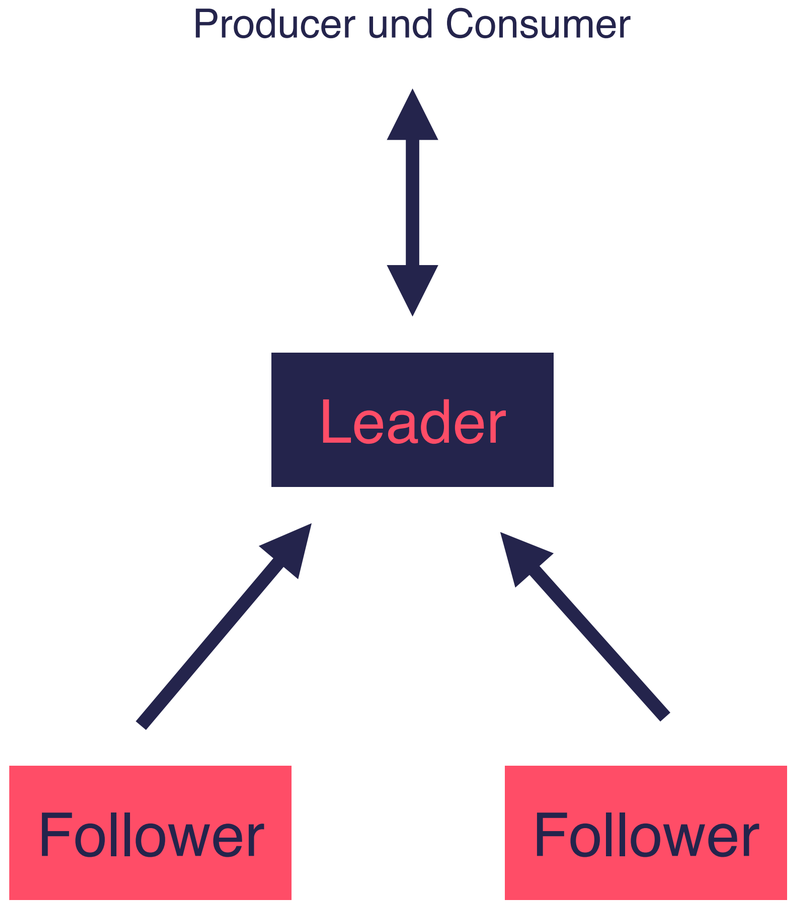
Broker

- Cluster besteht aus 1-n Brokern
- Zookeeper genutzt um Cluster zu verwalten
- Minimales Setup: 3 Kafka-Broker, 3 Zookeeper Nodes
- allow.auto.create.topics = true: Neue Topics werden automatisch beim ersten Zugriff generiert



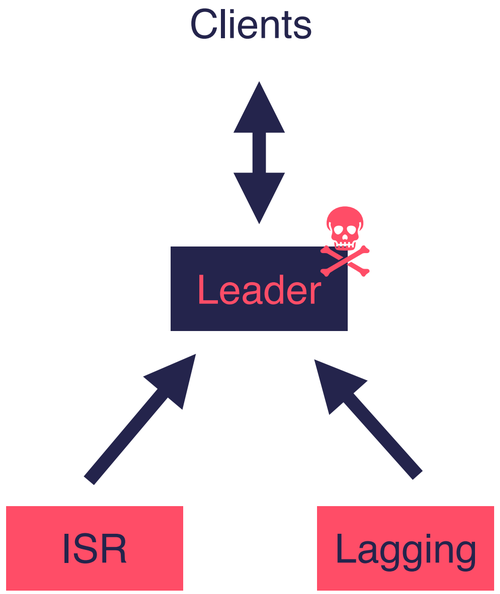
Replicas

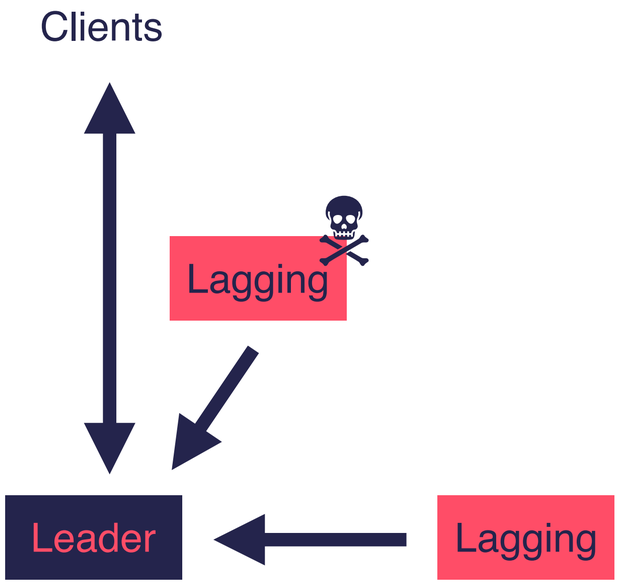
- flush.messages = MAX_LONG, flush.ms = MAX_LONG: Keine Garantie, dass auf die Disk geschrieben wurde
- Replikation von Partitionen verhindert Datenverlust
  - replication-factor = 1: Keine Replikation
- Leader handelt all Read und Write-Anfragen, Follower replizieren nur



ISR/Failover

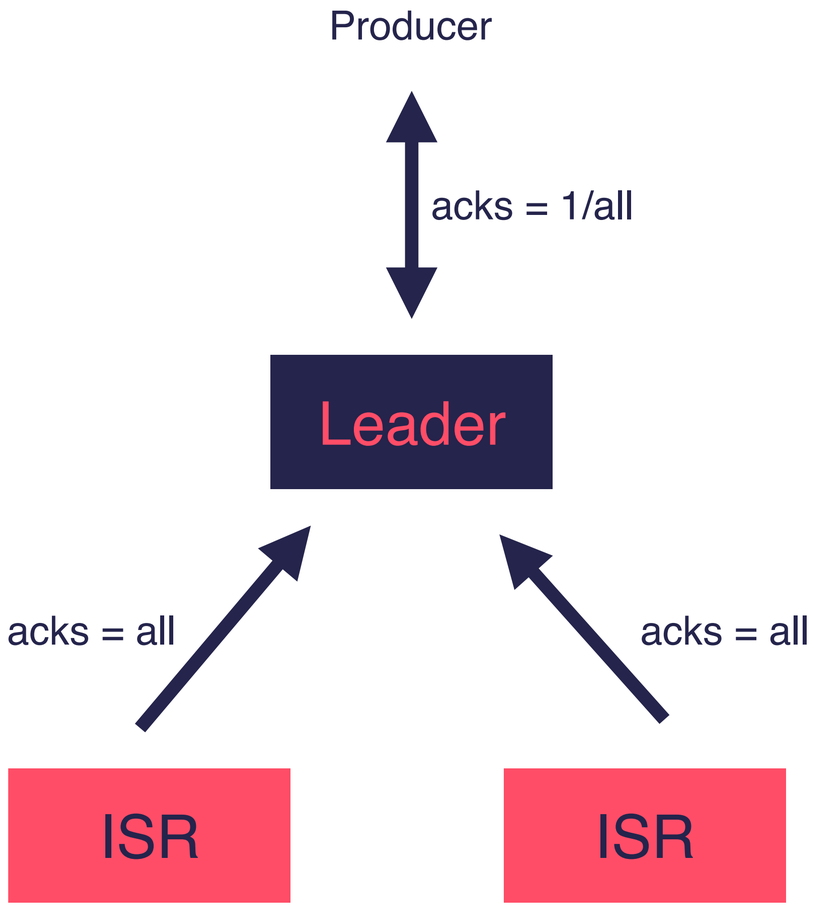
- Follower erhöhen Verfügbarkeit bei Ausfall von Brokern
- Follower sind In-Sync oder Lagging
  - replica.lag.time.max.ms = 10s: maximaler Lag
- Fällt Leader aus bestimmen ISR-Follower neuen Leader
  - unclean.leader.election.enable = false: nicht in-sync Follower kann Leader werden





Acknowledgments

- acks = 1: Nur Bestätigung des Leaders abwarten
- -1/all: garantiert, dass alle ISR die Nachricht erhalten haben
  - min.insync.replicas = 1: Bestimmt minimale Anzahl In-Sync Replicas für Writes
- 0: Fire-and-forget



Offset-Handling

- Consumer garantieren standardmässig nicht dass alle Nachrichten verarbeitet werden
- enable.auto.commit = true: Offset wird automatisch im Hintergrund committed
- auto.offset.reset = latest: Ohne validen Offset werden nur neue Nachrichten gelesen
  - offsets.retention.minutes = 7d: Offsets werden nach 7 Tagen gelöscht
  - Seit 2.1 für Consumer die immer Online sind nicht mehr so kritisch

Reihenfolge

- retries = INT_MAX, max.in.flight.requests.per.connection = 5: Erlaubt veränderte Reihenfolge bei Retries
  - max.in.flight.requests.per.connection = 1, enable.idempotence = true: Garantiert Reihenfolge, auch bei Retries
  - enable.idempotence = true erzwingt acks = all
- enable.idempotence: Gilt nur für die Dauer eines Prozesses

Transaktionen und Exactly-once

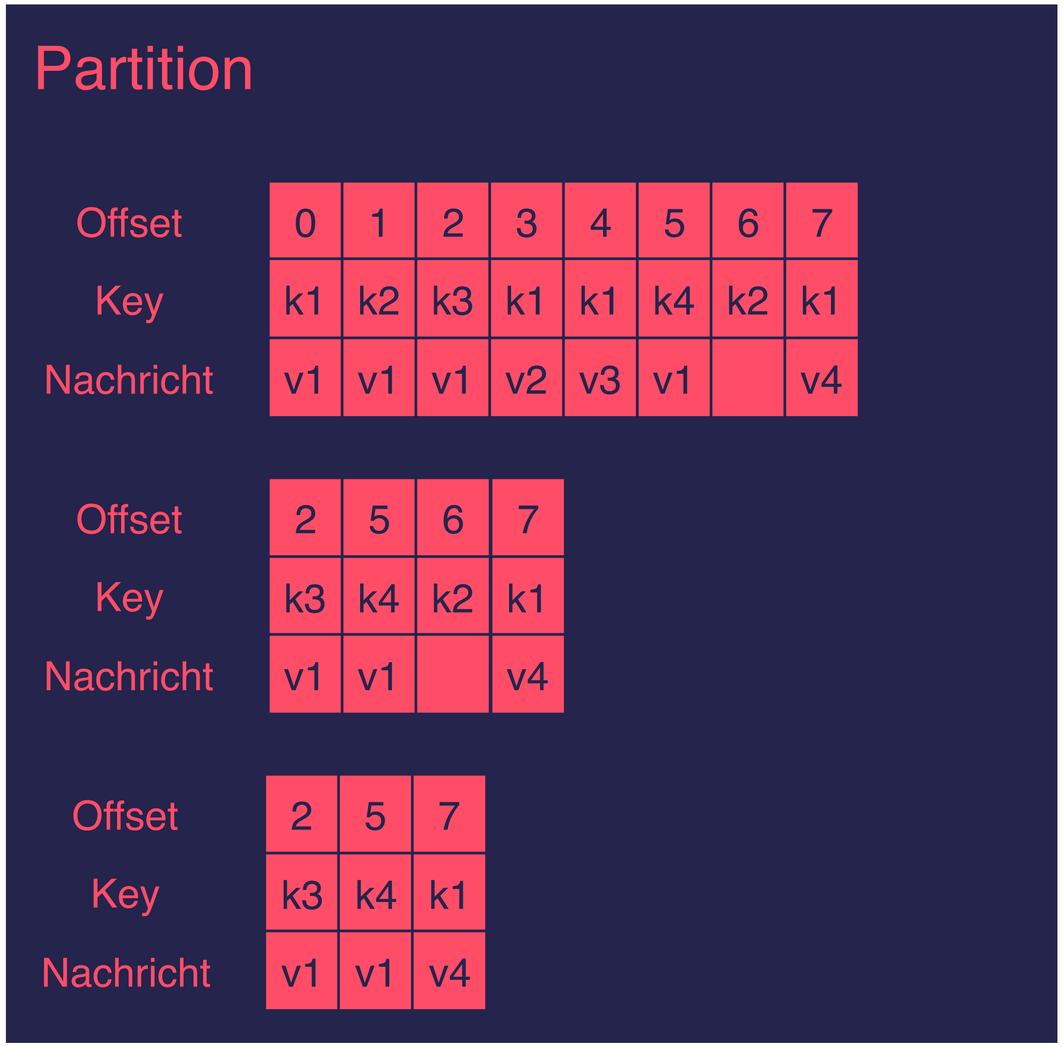
- Einsatzgebiet: Stream Processing
  - Committen von Offsets sowie schreiben des Resultats
- Transaktionen gelten nur für Kafka, nicht für ausgeführten Code
- isolation.level = read_uncommitted: Per default werden uncommittete Nachrichten gelesen
  - read_committed: Damit Consumer nur committete Transaktionen liest

Cleanup Policies

- Kafka speichert Daten in Segmenten
- segment.bytes = 1GB, log.roll.hours = 7d
  - Gilt pro Partition, nicht pro Topic
- Kafka bietet 2 Cleanup Policies: delete und compact
  - Per Default delete nach retention.ms = 7d
  - Gelöscht werden jeweils nur ganze Segmente

Log Compaction

- Löscht alte Nachrichten mit gleichem Message-Key
- Tombstones: Löschen Nachrichten aus Log
  - delete.retention.ms = 24h: bestimmt maximale Lesedauer für Consumer



Fazit

- Kafka nutzt verschiedene, untereinander abhängige Konzepte um die versprochenen Garantien zu gewährleisten
  - Defaults nicht für Produktion geeignet, egal ob für AP oder CP
- Definieren welche Garantieanforderungen man an Kafka hat und welche Tradeoffs man dafür in Kauf nimmt

Danke! Fragen?

www.innoq.com

Patrick Kaufmann
patrick.kaufmann@innoq.com

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Germany
+49 2173 3366-0

Ohlauer Str. 43
10999 Berlin
Germany
+49 2173 3366-0

Ludwigstr. 180E
63067 Offenbach
Germany
+49 2173 3366-0

Kreuzstr. 16
80331 München
Germany
+49 2173 3366-0

Hermannstrasse 13
20095 Hamburg
Germany
+49 2173 3366-0

innoQ Schweiz GmbH

Gewerbestr. 11
CH-6330 Cham
Switzerland
+41 41 743 0116