Kafka Log Compaction

# Übung 6

# Was wir bis jetzt gesehen haben

- Starke Koppelung
  - JSON-over-HTTP mit dem REST Paradigma (Ressourcen Orientiert denken)
  - ProtoBuf mit RPC Style
- Entkoppelung
  - RabbitMQ: Message geht nach dem Konsumieren verloren (Konzept: Exchange, Topic, Binding)
  - Kafka: Messages stehen in einer Art „File“ und können immer von Beginn an gelesen werden (Retention Time = wann soll die „File“ aufgeräumt werden)

# Ziel der Übung

- Kafka für Ressourcen Abgleich
- Log Compaction
- Beispiel:
  - Wir haben Komponenten Personenverwaltung und Bestellabwicklung bzw. Verrechnung
  - Komponenten Personenverwaltung hat Person + Liefer- bzw. Rechnungsadresse
  - Weitere Komponenten
    - Lieferkomponente: benötigt immer die neuste Lieferadresse
    - Verechnungskomponte: benötigt immer die neueste Rechnungsadresse
- Ziel: Wir wollen die Daten von einer Komponente in die andere bekommen

# Erinnerung Vorlesung

- Wir könnten die Personenverwaltungskomponente pollen
  - Bei großen Datenmengen schwierig und belastet das System
- Wir können die Änderung per RabbitMQ schicken
  - Wenn eine neue Komponente kommt – wie bekommen wir den Ist-Stand? (→ Snapshot?)
- Die Personenverwaltungskomponente könnte die anderen Komponenten aufrufen, wenn sich etwas ändert
  - Was ist, wenn diese nicht verfügbar sind?

# Aufgabe 1: Topic anlegen (nur anschauen)

- Siehe docker-compose.yml
- Topic anlegen (wird per scripts/create-topic.sh automatisch gemacht):
  - Name: `persons`
  - Config: `{"cleanup.policy":"compact", "min.cleanable.dirty.ratio":"0.01", "segment.ms":"100", "delete.retention.ms":"100"}`
  - Config ist **nicht** praxistauglich!

# Aufgabe 2

- Nutzen des Projekts „KafkaCompaction“
- Ergänzen sie den Code an den vorgesehenen Stellen
- Anforderungen sind als Kommentar hinterlegt
- Grob: Wir simulieren Änderungen von Personendaten
- Ein bisschen Geduld mitbringen – Consumer braucht ab und zu 5 Sekunden, bis er zum Lesen beginnt …

# Aufgabe 3

- Beantworten sie folgenden Fragen in der Fragen.txt

1. Lassen Sie den Producer 1 Minute lang laufen. Brechen Sie mit Strg+C ab. Die Anzahl der generierten Events bzw. der letzte Status wird ausgegeben. Starten Sie kurz danach den Consumer. Was beobachten Sie?
   a) Wie viele Events kommen an? Kommen alle an?
   b) Welche Events kommen an? Mit anderen Worten: Wie funktioniert Log-Compaction? Was ist der Unterschied zur letzten Übung?
2. Starten Sie den Consumer 2 Mal neu (ohne den Producer erneut zu starten).
   a) Hat sich etwas zu Frage 1 geändert?
3. Wir speichern die Daten in Memory (für Übung). Normalerweise werden die Daten persistiert (nächste Übung). Erklären Sie in 1-2 Sätzen, warum wir e.g. das Outbox-Pattern (https://microservices.io/patterns/data/transactional-outbox.html) brauchen

# Weiterführende Themen

- Personendaten aufteilen: 1 Topic für Verrechnungsadressen und 1 Topic für Lieferadressen
  - https://docs.confluent.io/5.2.0/streams/quickstart.html
  - https://docs.confluent.io/platform/current/ksqldb/index.html