Kafka

Übung 5

Was wir bis jetzt wissen

- REST Architecture Style: Ressourcenorientiert denken
  - JSON-over-HTTP
- GRPC
  - Remote-Procedure-Call, Streaming
- Beide haben zeitliche Abhängigkeiten → andere Komponenten muss da sein
- Broker Style mit RabbitMQ
  - Vollständige Entkoppelung (Komponente Email und WhatsApp konnten irgendwann gestartet werden → Nachrichten wurden dann konsumiert → Nachricht ist verschwunden)

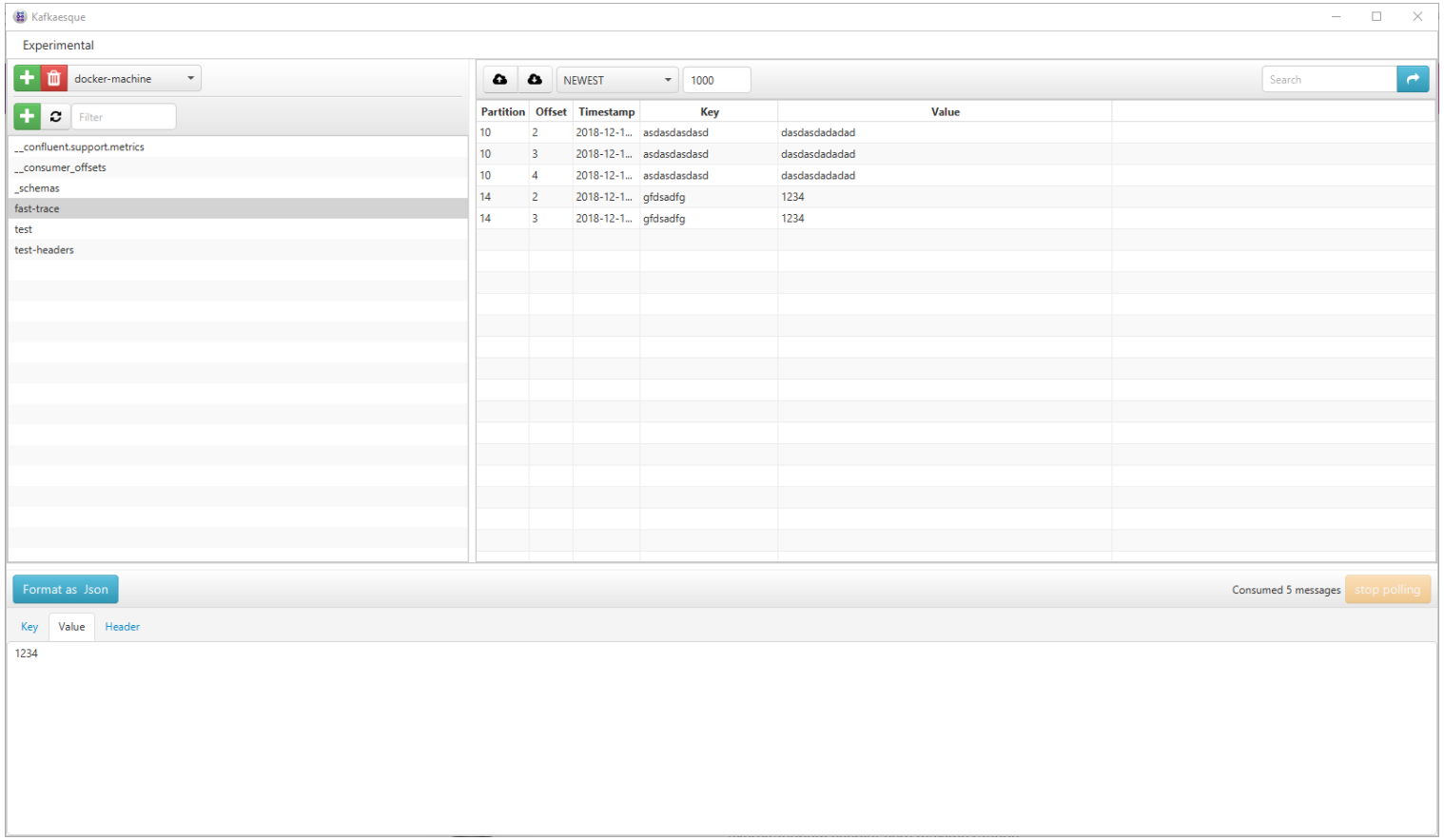
Ziel der Übung

- Wir wollen auf dem „Stream“ von Nachrichten jederzeit wieder zugreifen können … → Nachrichten sollen nicht beim Konsumieren verschwinden
- Sinnvoll für u.a. Auditing oder wenn eine neue Komponente auf den vorhanden Verkaufs-Events nochmals zugreifen will
- Oder wenn ich den State von einer Komponente in einer andere bringen
  - Beispiel: Unser Bestellsystem hat Produkte → in einer anderen Komponente brauche ich alle „ProduktIds“ die es gibt
  - Lösung: Polling auf REST, Streaming GRPC – oder Kafka (siehe nächste Übung)

Starten von Kafka

- Docker muss laufen
- Es laufen mehrere Services
  - Broker
  - Schema Registry (brauchn wir bei der Übung nicht)
  - Zookeeper

Topic anlegen (https://kafka.esque.at)



Achtung: Braucht Java 8

Aufgabe 1

- Solution: KafkaExercise
  - Hello world inkludiert → sollte funktionieren
- Anlegen einer
  - „stocks“ topic (Replication 1, Partition 1)
  - .NET Core Library: Common
  - .NET Core Console Application: StockGenerator
  - .NET Core Console Application: CsvGenerator

Module: Common

- Enthält eine Klasse `Stock`
  - `Name` : string
  - `Value` : double
- Serialisierung wieder mit protobuf

Module: StockGenerator

- Generieren für 3 Stocks („TSLA“, „GPRO“, „AON“)
- Generieren von Random Zahlenwerten (Stock.Value)
  - Generieren Sie alle 10-50msec Events → selektieren sie den Stock random (0…2)
  - In Summe 1000 Events
- POIs:
  - Nutzen sie entweder `producer.Produce("stocks", message, DeliveryHandler);`
  - Oder awaiten sie jedes Produce (`ProduceAsync`)
  - Ersteres sollte schneller sein

Module: CsvGenerator

- Erstellen Sie pro Stock die Summe von jeweils 10 Values und schreiben Sie das Ergebnis in ein CSV File und geben Sie den vollständigen Pfad aus: <stockname>.csv mit dem Inhalt <sum 0-9>;<sum 10-19>; ….
  - Nutzen Sie die Reactive Extension (siehe Demo)
  - Beginnen Sie immer vom Offset 0 zu lesen
- Vorlesung: GroupId – Kafka merkt sich die letzte konsumierte Position
  - `consumer.Assign(new TopicPartitionOffset("stocks", new Partition(0), Offset.Beginning));`
  - Beginnt wieder am Anfang

Beobachtung

- Was passiert wenn man
  1. den Producer nach dem Erzeugen der 1000 Events stoppt
  2. die CSV Files löscht
  3. den CsvGenerator erneut startet
  4. Und dann wieder bei Schritt 2 beginnt (2 Mal)
- Beobachtung
  - Obwohl der Producer nicht mehr läuft – Kafka merkt sich die Nachrichten (für eine gewisse Zeit → siehe retention time)
  - Überlegen Sie sich, wo das Sinn machen könnte (Stichwort: Auditing, Analyse der Events)
  - Antwort in Fragen.txt