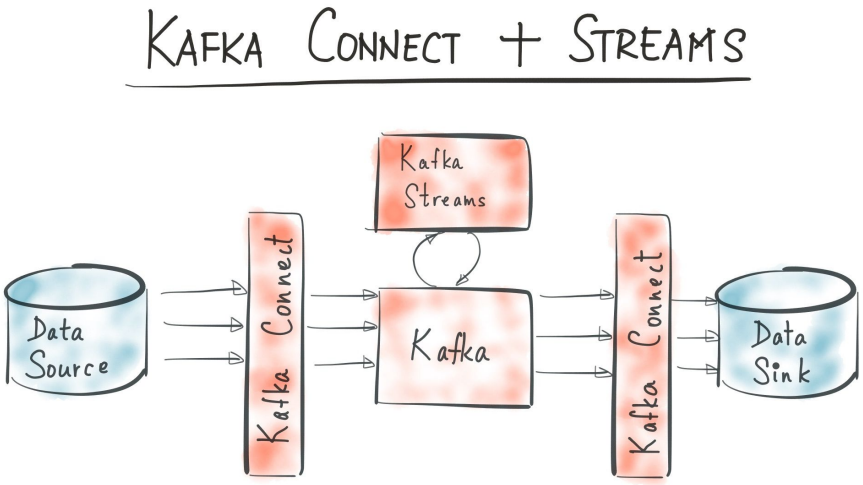
Kafka Streams

Kafka Streams - uvod

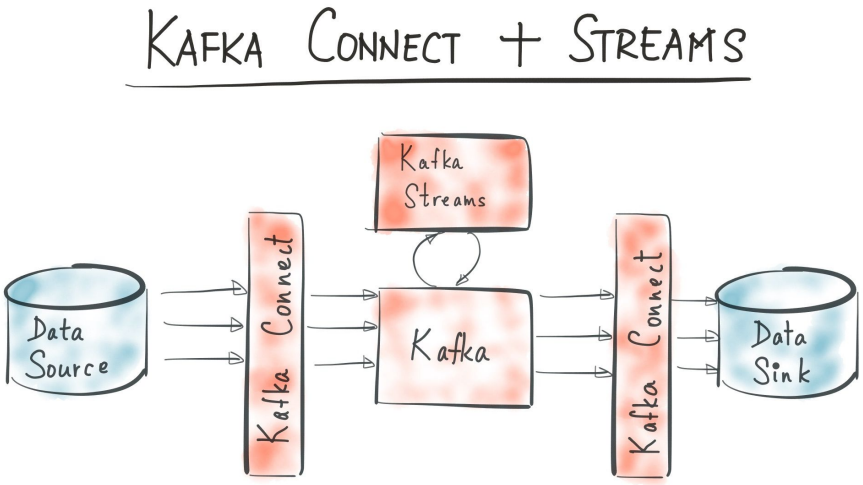
- Jednostavna i “lagana” klijentska biblioteka za kreiranje servisa za obradu tokova podataka.
- Ulazni i izlazni podaci se skladište na *Kafka* klasteru.
- Umesto da se pišu namenski obrađivači toka podataka, koriste se Kafka *producer*-i i *consumer*-i.
  - To znači da će paralelizam, distribuirana koordinacija i otpornost na otkaze biti nativno podržani.



source: https://www.confluent.io/blog/hello-world-kafka-connect-kafka-streams/

Kafka Streams - prednosti

- Nije potreban zaseban klaster za obradu podataka.
- Podržana *record-at-a-time* obrada (ne vrši se micro-batching).
- Sve prednosti kafke nativno podržane:
  - uređivanje, particionisanje, skalabilnost, otpornost na otkaze…
- Lako rukuje zakasnelim i out-of-order podacima.

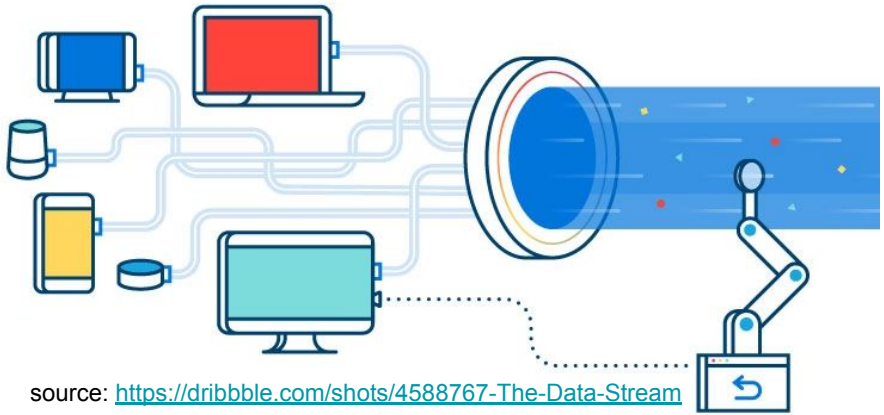


source:
https://www.confluent.io/blog/hello-world-kafka-connect-kafka-streams/

Key Idea: **Outsource hard problems to Kafka**

Kafka Streams - osnovni koncepti

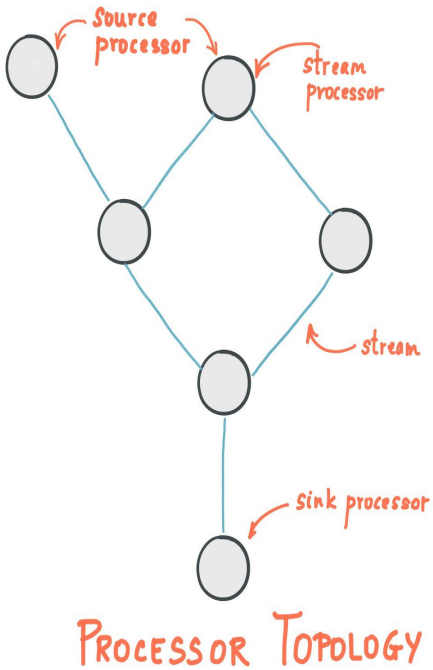
- *Data Stream*
  - neograničeni, kontinualno ažurirani skup podataka;
  - sekvenca nepromenljivih zapisa podataka (*immutable data records*) koja je uređena, *replayable* i otporna na otkaze .
- *Data record* - par ključ-vrednost
- Topologija za obradu toka podataka
  - usmereni aciklični graf izvora i obrađivača, povezanih u cilju analize i obrade podataka



source: https://dribbble.com/shots/4588767-The-Data-Stream

Kafka Streams - topologija za obradu toka podataka

- Svaki program koji koristi Kafka Streams biblioteku zapravo definiše logiku za izvršavanje kroz jednu ili više topologija obrađivača
  - gde je topologija obrađivača usmereni aciklični graf obrađivača toka (čvorova) povezanih tokovima podataka (ivicama).
- Obrađivači toka (*Stream processors*) - čvorovi u topologiji obrađivača; predstavljaju korak obrađivanja kojim se podaci toka transformišu na sledeći način:
  - ulazni podaci se prihvataju *record-at-a-time*;
  - nad njima se vrše specificirane operacije obrađivača;
  - ukoliko ima potrebe, transformisani podaci se šalju nizvodno, na obrađivanje od strane drugih obrađivača.
  - Podvrste - *Source processor* i *Sink processor*



source: https://kafka.apache.org/23/images/streams-architecture-topology.jpg

Kafka Streams - Kafka Streams DSL

- Kako bi se definisala topologija za obradu toka podataka, može se koristiti *Kafka Streams DSL* (*domain specific language*)
  - apstrakcija nad *Stream Processor API*-jem, koja omogućava izražavanje operacija obrađivača na deklarativan način uz korišćenje funkcionalnog programiranja.
  - Pruža mogućnost korišćenja *KStream*, *KTable* i *GlobalKTable* apstrakcija za tokove podataka i tabele.
  - Podržava i *stateless* (npr. mapiranje, filtriranje) i *stateful* (agregacije, spajanja, klizni okviri...) transformacije.

Kafka Streams - primeri

- Klaster - *docker-compose.yaml*
- Skup podataka - letovi između Pekinga i Šangaja, sortirani hronološki, uz podatke o cenama, tipovima letelica, šiframa leta itd.
  - https://www.kaggle.com/lpisallerl/air-tickets-between-shanghai-and-beijing/downloads/air-tickets-between-shanghai-and-beijing.zip/1
- *Producer* - Java Maven aplikacija koja na svakih 300 milisekundi čita podatke iz CSV fajla (*pek-sha.csv*) i odašilje ih na zadati *topic*: *pek-sha*
- *Consumers* - Java Maven aplikacije sa primerima upotrebe *Kafka Streams API-ja:* grananje, filtriranje, mapiranje, grupisanje, agregiranje (*count, average*)...

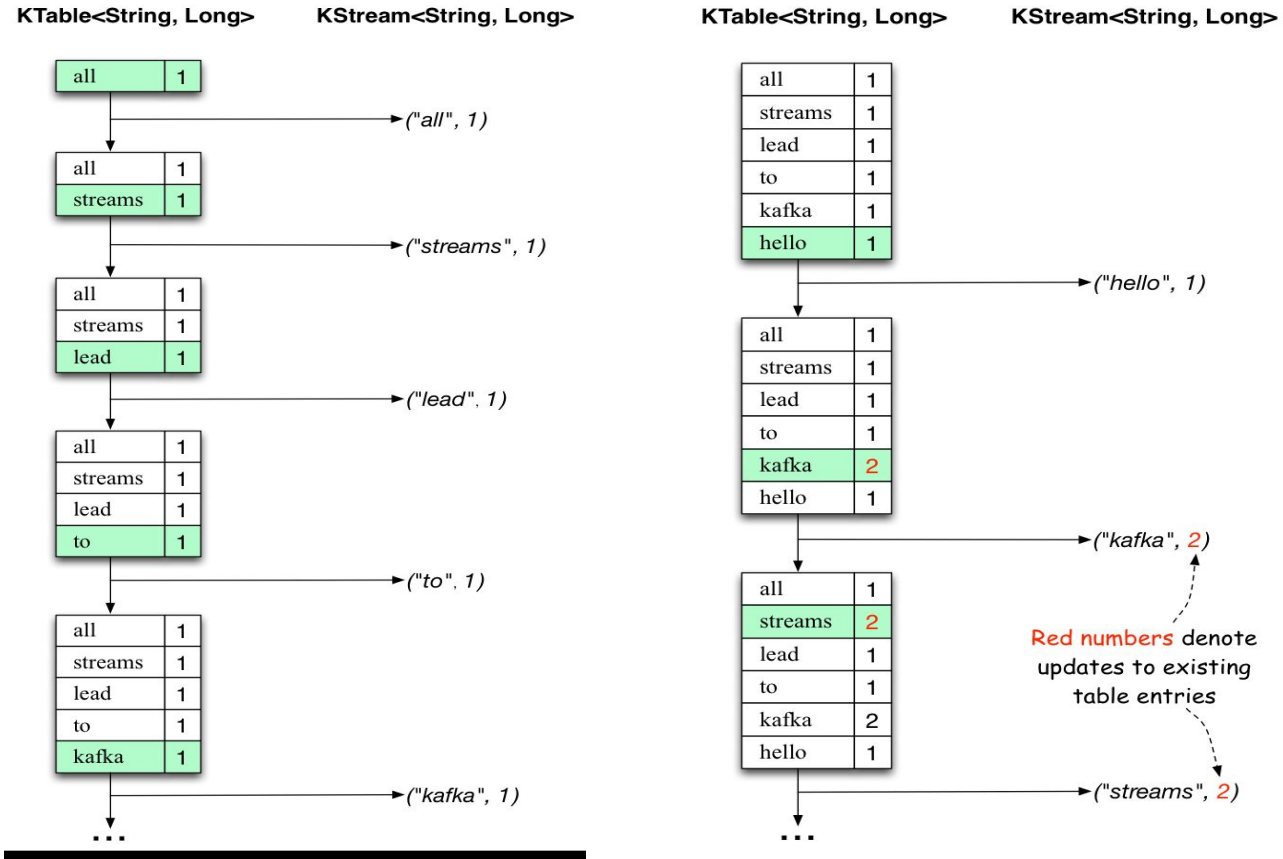
Kafka Streams - FlightNumberCount primer

- *Consumer* prihvata podatke o pojedinačnim letovima, te vrši grupisanje i prebrajanje po broju leta.
- Pokretanje primera:
  - Povezivanje na kontejner:
    - **`sudo docker exec -it producer_and_consumers bash`**
  - Pokretanje aplikacije:
    - **`java -cp /usr/consumers/kafka-streams-examples-0.0.1-SNAPSHOT.jar com.example.kafka_streams_examples.FlightNumberCounterExample`**

Kafka Streams - tokovi podataka i tabele

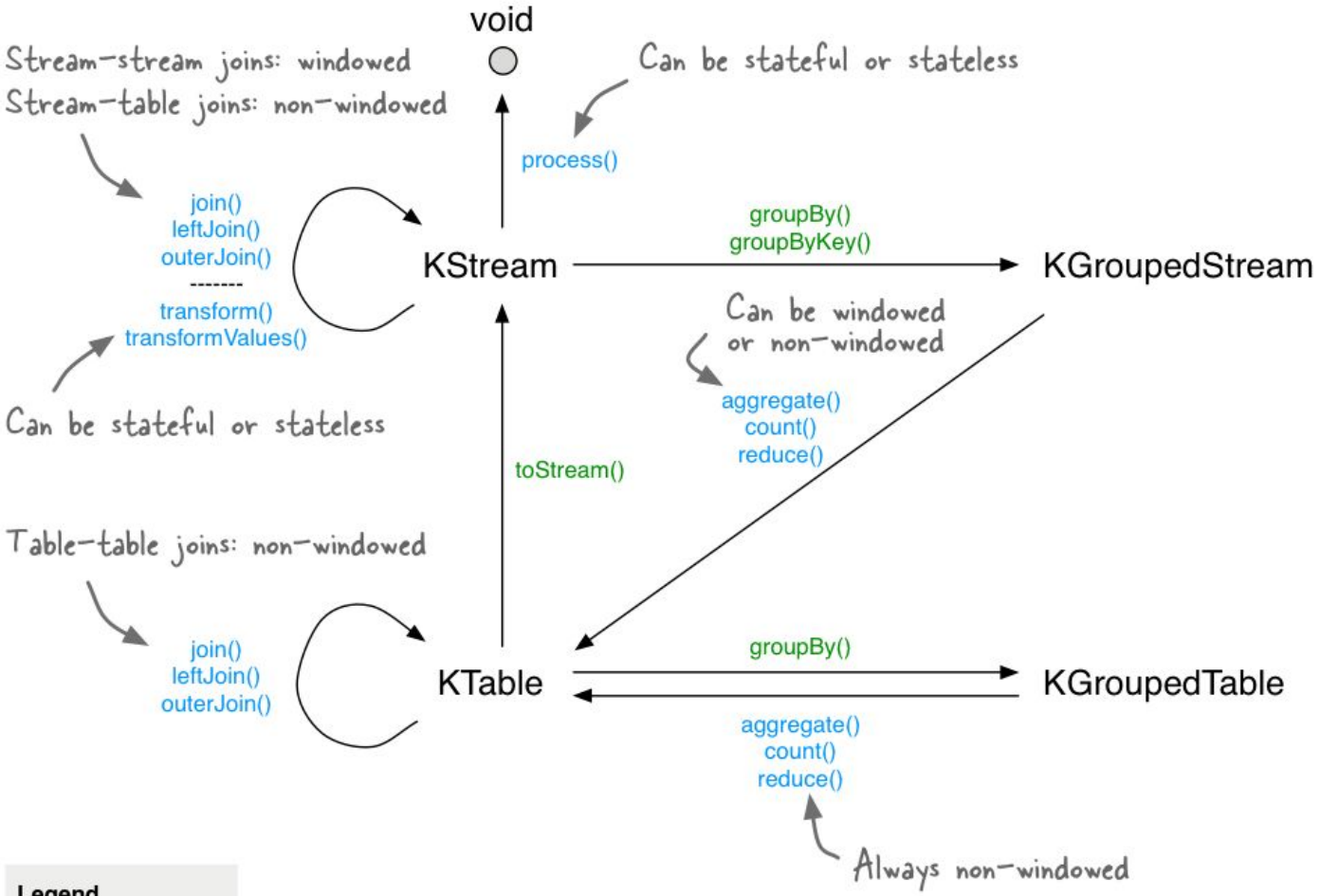
- *KStream* - predstavlja apstrakciju particionisanog toka zapisa (*record stream*), gde svaki zapis predstavlja pojedinačni podatak u neograničenom skupu podataka.
  - Podaci se u *KStream* mogu samo dodavati.
- *KTable* - predstavlja apstrakciju particionisanog toka zapisa promena (*changelog stream*), gde se svaki zapis tumači kao ažuriranje postojeće vrednosti u tabeli sa istom vrednošću ključa, ukoliko postoji (*upsert*).
  - *Null* vrednosti se tumače kao operacija brisanja zapisa iz tabele.
- *GlobalKTable* - kao *KTable*, ali će svaka instanca aplikacije dobijati podatke iz svih particija *topic*-a.

Kafka Streams - tokovi podataka i tabele



Kafka Streams - transformacije

- Izvori tokova podataka
  - input topics → KStream
  - input topics → KTable
  - input topics → GlobalKTable
- Transformacije nad tokovima podataka
  - Stateless
    - Branch
    - Filter
    - Inverse Filter
    - FlatMap
    - FlatMap (values only)
    - Foreach
    - GroupByKey
    - GroupBy
    - Map
    - Map (values only)
    - Merge
    - Peek
    - Print
    - SelectKey
    - Table to Stream
  - Stateful
    - Aggregate
    - Count
    - Reduce
    - Inner Join
    - Left Join
    - Outer Join



source: https://kafka.apache.org/23/documentation/streams/developer-guide/dsl-api.html

Kafka Streams - primeri

- *BranchByCraftExample* - deljenje originalnog toka podataka na osnovu vrste letelice.
- *JoinExample* - primer upotrebe operacije spajanja
  - Nakačiti se na broker i pokrenuti: `usr/bin/kafka-console-producer --broker-list localhost:19092 --topic codes --property "parse.key=true" --property "key.separator=:"`
  - Proslediti vrednost za neki broj leta, npr: HO1252:12345
- *PriceClassMeanExample* - primer upotrebe agregacije
  - oslanja se na upotrebu namenskog (de)serijalizatora
- *TraFilterExample* - primer filtriranja

Kafka Streams - zadaci

- Koristeći *pek-sha topic* iz prethodnih primera, kreirati sledeće aplikacije za obradu toka podataka:
  - aplikacija koja mapira svaku vrednost zapisa na broj leta, dužinu leta i cenu, te šalje mapirane vrednosti na novi *topic pek-sha-dur*
  - aplikacija koja filtrira vrednosti koje imaju vreme poletanja između ponoći (0AM) i 7 ujutro (7AM), i šalju ih na novi *topic pek-sha-night*
  - aplikacija koja prebraja broj letova po danu i štampa rezultate na *System.out.*
- Kreirati aplikaciju koja vrši prebrajanje reči *reddit* komentara.