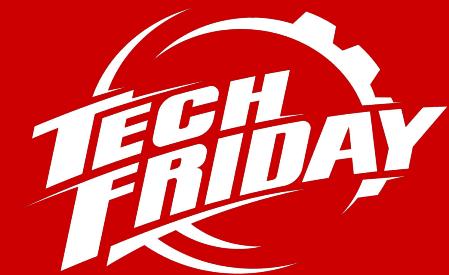


 tarent solutions GmbH - tech friday 2020 - hannes rohde

Daten schaufeln ohne Code mit Kafka Connect



Agenda

1. Kafka als Stream Data Platform
2. Typischer erster Kontakt mit Kafka
3. Kafka Connect
4. Architektur von Kafka Connect
5. Einsatzbeispiele (Live Demo)
6. Zusammenfassung





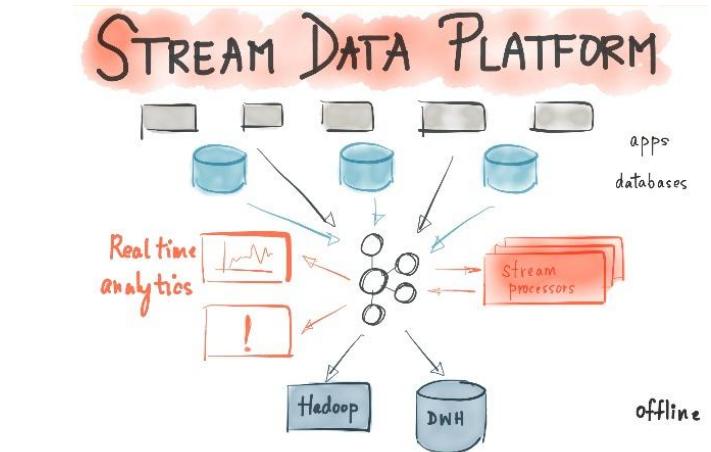
Kafka als Stream Data Platform



Datenströme als Backbone des Unternehmens

Diverse Systeme produzieren oder konsumieren Daten

- Relationale Datenbanken
- Key-Value Stores
- Suche/Indexierung
- Web-Applikationen
- Business-Logik
- Analytics
- Sensoren
- externe APIs
- ...





Kafka als Stream Data Platform

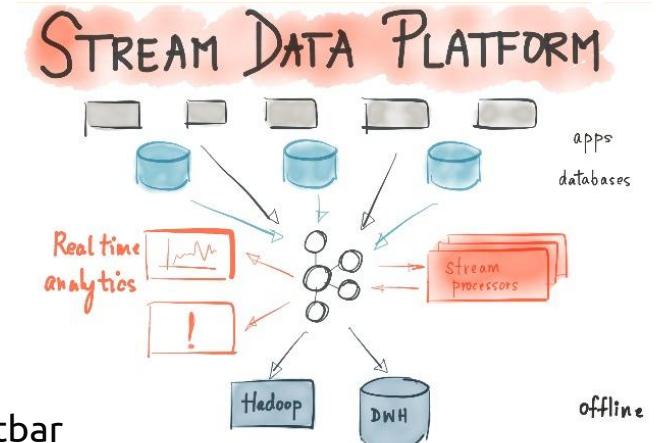
Kontinuierliche Verarbeitung in Echtzeit

Daten werden

- kontinuierlich produziert
- kontinuierlich verarbeitet
- kontinuierlich konsumiert

Anforderungen:

- große Datenmengen verarbeitbar
- Transformation von heterogenen Daten
- Hochverfügbarkeit, Skalierbarkeit



Typischer erster Kontakt mit Kafka

Spring Boot @KafkaListener selber bauen

```
class MyConsumer {  
    @Autowired  
    private Database db;  
  
    @KafkaListener (topics = "my-topic", id = "my-consumer")  
    public void onMessage (  
        @Header (RECEIVED_MESSAGE_KEY) String key,  
        @Payload String message,  
        Acknowledgement ack  
    ) {  
        if (message != null) {  
            // persist item in database  
            db.upsert (key, message);  
        } else /* tombstone message */ {  
            db.delete (key);  
        }  
        ack.acknowledge ();  
    }  
}
```



Typischer erster Kontakt mit Kafka

Spring Boot @KafkaListener selber bauen

Dazu kommen dann noch:

- JPA-Entities, -Repository
- Mapping / Transformation
- Schema-Verwaltung
- Error Handling

Nachteile:

- viel Boilerplate
- eher schwergewichtig
- manche Quellen/Senken auf diesem Weg nicht gut anbindbar!
- Skalierbarkeit, Administrierbarkeit



→ Kafka Connect



Vorgefertigte Connectoren für viele Standard-UseCases

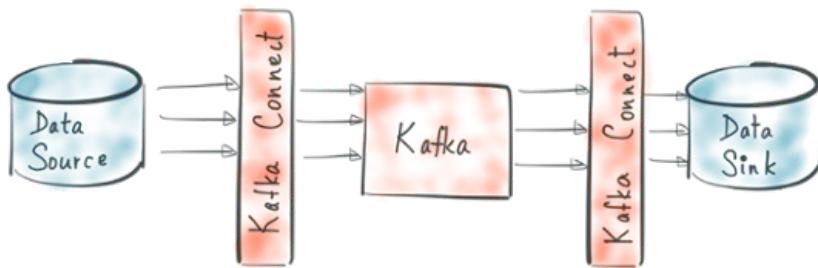


- eingeführt mit Kafka 0.9.0 (2016)
- Framework für Anbindung verschiedenster Systeme an Kafka
- Vorgefertigte Connectoren zum Streamen von Daten zu und aus Kafka
- rein konfigurativ können viele Standard-UseCases abgedeckt werden

Kafka Connect



KAFKA CONNECT



Sources und Sinks

Kafka Connect kennt zwei Arten von Connectoren:

Source

Datenquelle → Kafka
(= Producer)

Sink

Kafka → Datensenke
(= Consumer)

Was kann man per Kafka Connect anbinden?

Connector-Plugins für viele Standard-Technologien

Initialer Release mit Connectoren für HDFS- und JDBC

Aufruf an Community: *entwickelt Connectoren!*

Inzwischen großes Ökosystem:

auf [Confluent Hub](#) > 170 Connector-Plugins

- Datenbanken
- Messaging-Systeme
- Storage
- ...



Was kann man per Kafka Connect anbinden?

Verfügbare Connector Plugins

Datenbanken:

- JDBC, ArangoDB, HBase, AWS DynamoDB, Cassandra, CockroachDB, Couchbase, Debezium, Google BigTable, Google Firebase, InfluxDB, MongoDB, Neo4j, OrientDB, Redis, Splunk, Teradata, Yugabyte

Messaging:

- JMS, RabbitMQ

Storage:

- Azure Blob Storage, File Stream, HDFS, Sftp, SpoolDir, S3, Syslog

Social:

- Email, IRC, PagerDuty, Reddit, RSS, Telegram, Twitter

Misc:

- Http, Shell, Sound



Architektur von Kafka Connect

Connectors / Tasks / Workers

Connector

- Definition eines logischen Jobs
- Grundlage für Tasks

Tasks

- eigentliches Übertragen der Daten
- gemanaged durch Kafka Connect
- stateless, parallelisierbar

Worker

- Container zur Ausführung von Connectoren und Tasks
- Mehrere Worker bilden einen Cluster und verteilen Tasks automatisch



Architektur von Kafka Connect

Standalone oder verteilt

Standalone

- alles in einer Maschine
- speichert in lokalem Filesystem
- keine Fehlertoleranz

verteilt

- mehrere Worker laufen verteilt als "Connect Cluster"
- skalierbar, fehlertolerant
- Bei Ausfall eines Workers werden Tasks neu verteilt
- Datenhaltung in Kafka:
 - Konfiguration
 - Source- und Sink-Offsets
 - Status





Vorverarbeitung der Daten

Konvertierung / Transformation

Konvertierung in verschiedene Übertragungsformate:

- Binärformate: Avro, Protobuf, Byte Array
- JSON mit und ohne Schema
- String

Transformation:

- Typecast, Umwandlung von Datum/Timestamp-Feldern
- Filterung anhand von Prädikaten
- Manipulation / Löschen / Einfügen einzelner Felder
- Ver- / Entpacken hierarchischer Daten

Steuerung und Konfiguration

Zentrale Administration von Kafka Connect

per [REST-API](#):

- Connectoren anlegen/bearbeiten/löschen
- Connectoren und Tasks starten/stoppen
- z.B. GET /connectors/hdfs-sink-connector/config

→ verträgt sich nicht gut mit CI/CD

Alternativ: [strimzi](#)-Operator auf Kubernetes

- eigene CRD “kafkaconnector”
 - Operator übernimmt Ansteuerung von Kafka Connect API
- Management wie bei allen anderen Ressourcen im Cluster



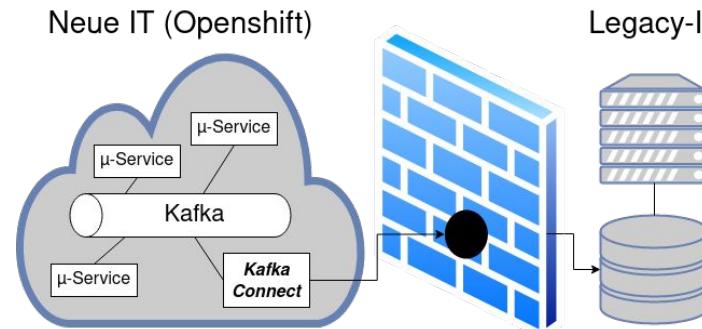


Einsatzbeispiele - Live Demo

Szenario 1: SQL-Datenbank → Kafka

Aufgabe: Datenbestand aus Legacy Application nach Kafka spiegeln

- Source Connector
- JDBC-Connector Plugin + JDBC-Treiber
- Periodisch ausgeführte SQL-Query
- Erkennung neuer Daten: per ID oder Timestamp





Einsatzbeispiele - Live Demo

Szenario 1: SQL-Datenbank → Kafka

Beschränkungen von Kafka Connect JDBC

- Datenbank-Tabellen müssen laufende id oder Timestamps enthalten
- Keine Erkennung von gelöschten Zeilen
- periodisches Polling verursacht DB-Last

→ ggfs. "richtiges" CDC z.B. mit Debezium besser geeignet



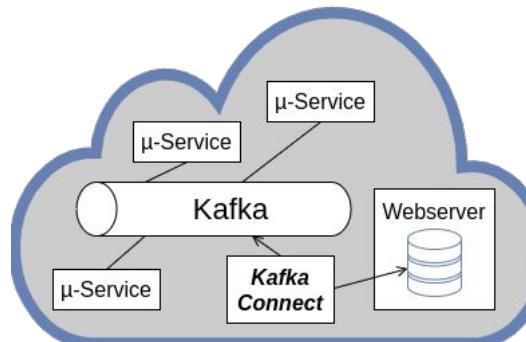


Einsatzbeispiele - Live Demo

Szenario 2: Logfile → Kafka

Aufgabe: Log-Ausgabe eines Servers (z.B. Webserver, Security-Appliance, IoT) in Kafka spiegeln

- Source Connector
- FileStream (enthalten in Basis-Image)
- Alternative Plugins: Sftp, SpoolDir, FileSystem



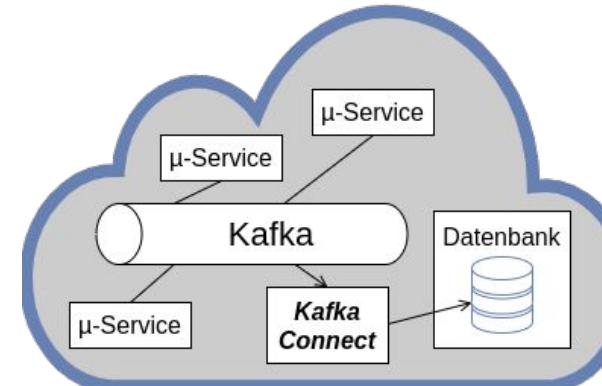


Einsatzbeispiele - Live Demo

Szenario 3: Kafka → SQL-Datenbank

Aufgabe: Daten aus Kafka-Topic in SQL-Datenbank persistieren

- Sink Connector
- JDBC-Connector Plugin + JDBC-Treiber
- Daten in geeignete Form transformieren



Zusammenfassung

Kafka Connect...

- bindet verschiedenste Systeme als Quelle oder Senke an Kafka an
- deckt viele Standard-UseCases ab
- ist leicht* konfigurierbar und administrierbar
- ist wichtiger Bestandteil einer Stream-Data-Platform auf Basis von Kafka
- hat ein paar kleine Macken
- = "Daten schaufeln ohne Code" 😊





tarent solutions GmbH

Vielen Dank!

Hannes Rohde
Software Developer

h.rohde@tarent.de

Rochusstraße 2-4
53123 Bonn

Telefon: 0228 54881-0
Telefax: 0228 54881-235

info@tarent.de
www.tarent.de

Folien und Beispielcode:
<https://github.com/hannesrohde/daten-schaufeln-mit-kafka-connect>

