

Big-Data-Technologien

Kapitel 16: Architektur

Hochschule Trier
Prof. Dr. Christoph Schmitz

Was ist Architektur?

1. Architektur =

Summe der Architekturentscheidungen

Architekturentscheidung =

Entscheidung mit weitreichenden Folgen

2. Architektur =

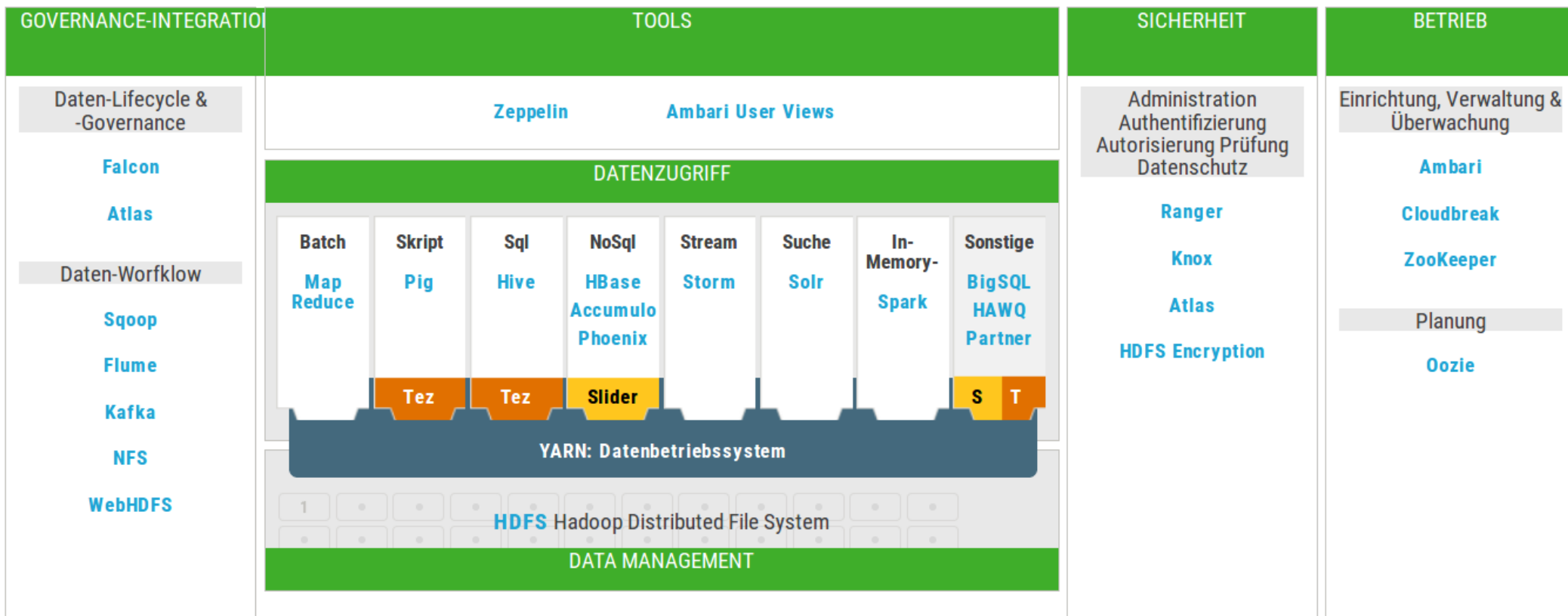
Grundlegende Komponenten und deren Zusammenspiel

Big-Data-Systemarchitektur

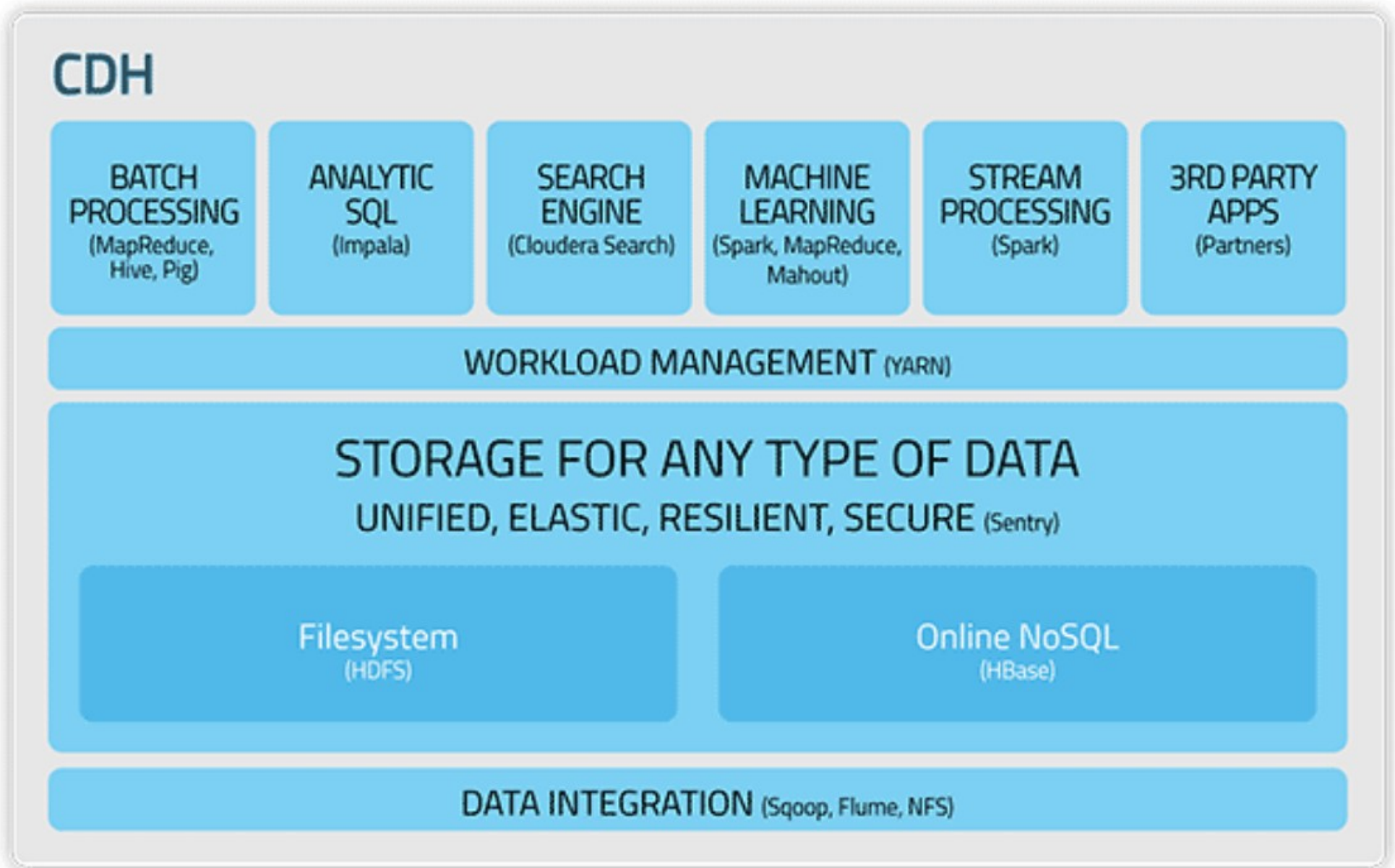


"Layer Cake"

Layer Cake



Layer Cake



Layer Cake

Home > **Products**



EXISTING ENTERPRISE
APPLICATIONS



BATCH & INTERACTIVE
ANALYTICS



INTELLIGENT
APPLICATIONS

CLOUD-SCALE
DATA STORE

ANALYTICS &
ML ENGINES

OPERATIONAL
DATABASE

GLOBAL EVENT
STREAMS

MAPR CONVERGED DATA PLATFORM

High Availability

Real Time

Unified Security

Multi-tenancy

Disaster Recovery

Global Namespace

ON-PREMISES, MULTI-CLOUD, EDGE



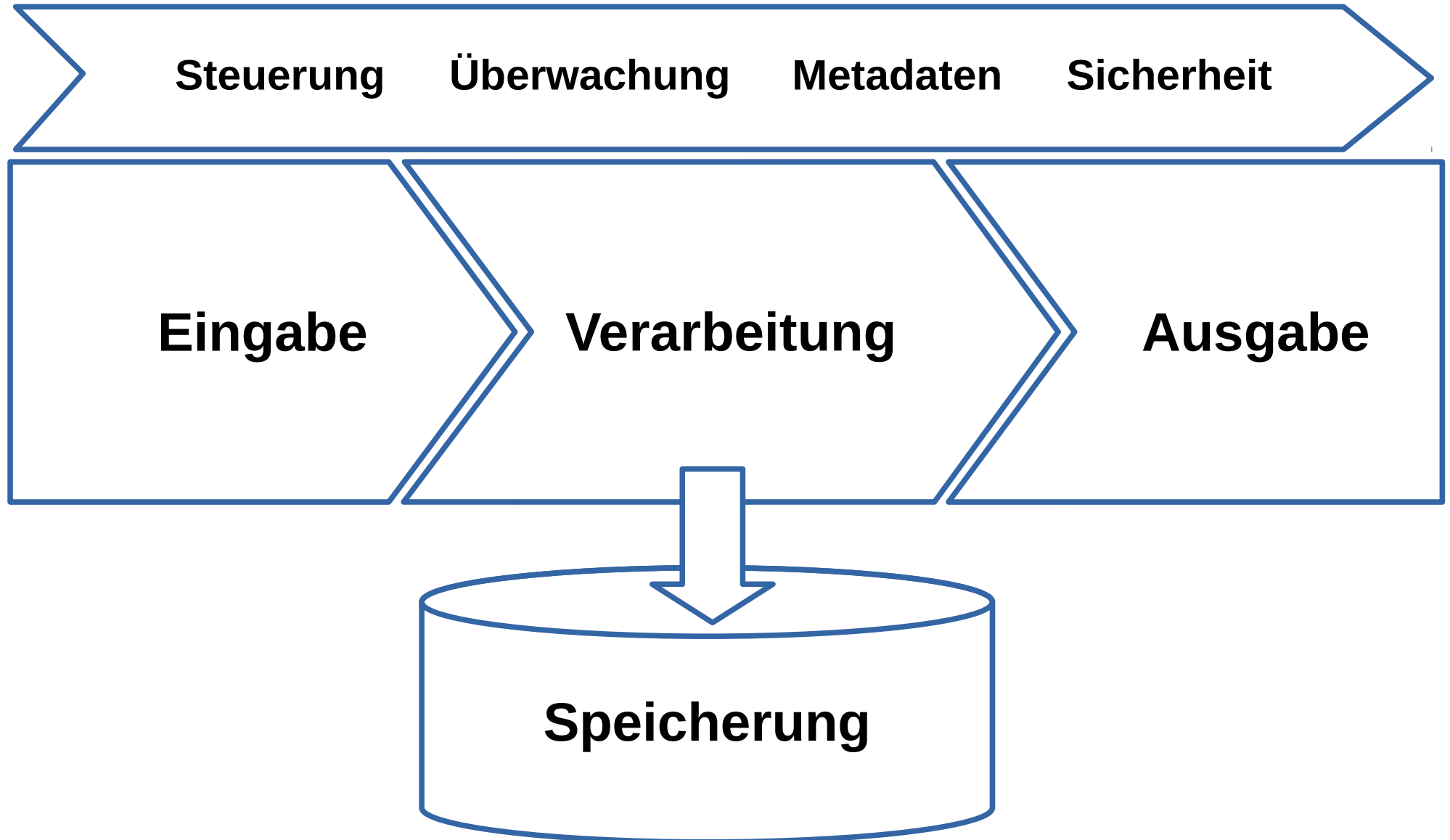
Architektur und Qualitätsziele

- **Qualitätsziele** (nonfunktionale Anforderungen) sind Haupttreiber für Architektur
- Skalierbarkeit
- Leistung und Effizienz
- Änderbarkeit
- Zuverlässigkeit
- Verständlichkeit
- Betreibbarkeit
- Benutzbarkeit
- ...

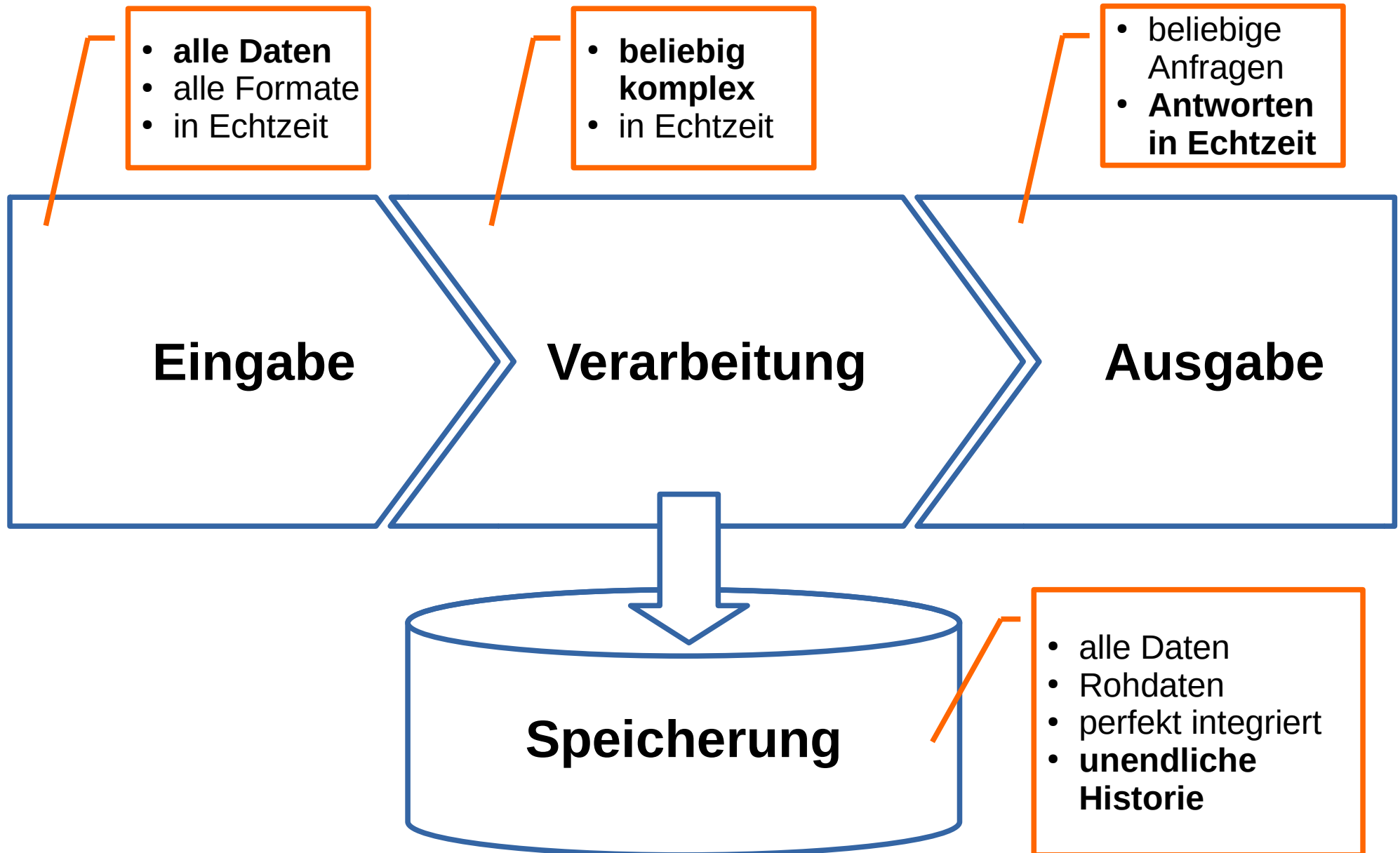
Typische Anwendungsfälle

- Entlasten von **relationalen Datenbanken** und Data Warehouses
- Langfristige **Aggregationen**
- Ad-Hoc-**Analysen**
- **Data Mining** und Machine Learning
- **Echtzeit-Verarbeitung** und -Analysen
- **Transaktionsverarbeitung** (OLTP)
- **Data Lake**

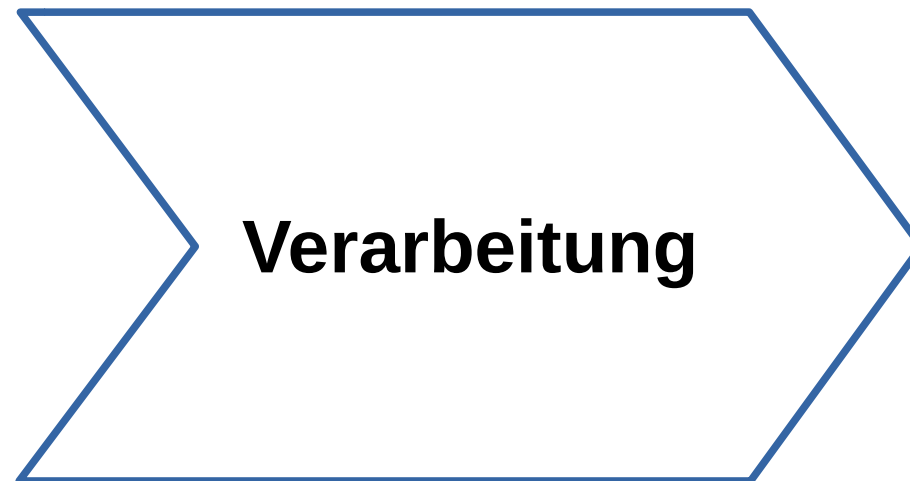
Das EVA-Prinzip



Das ideale Big-Data-System



Bausteine



- **MapReduce**
- **Spark**
- Hive
- **NoSQL-DB**
- **Kafka Streams**
- Spark Streaming
- Flink

Bausteine

- HDFS
- Cassandra
- HBase
- (MongoDB)



Bausteine



Eingabe

- **Import nach HDFS**
 - aus Dateien
 - aus DBMS
- **Import innerhalb des HDFS**
 - aus anderen Applikationen
- **Kafka**
 - Kafka Connect

Bausteine

- **Export aus HDFS**
 - in Dateien
 - in DBMS
- **Kafka**
 - Kafka Connect
- **Services**
basierend auf
NoSQL-DB
 - **Redis**
 - **MongoDB**
 - ...
- Anfrage-Schnittstelle
 - **SQL**



Bausteine

Steuerung

- Falcon
- Oozie
- Nifi
- Camel
- Workflow Engines

Überwachung

- Ambari
- Kibana

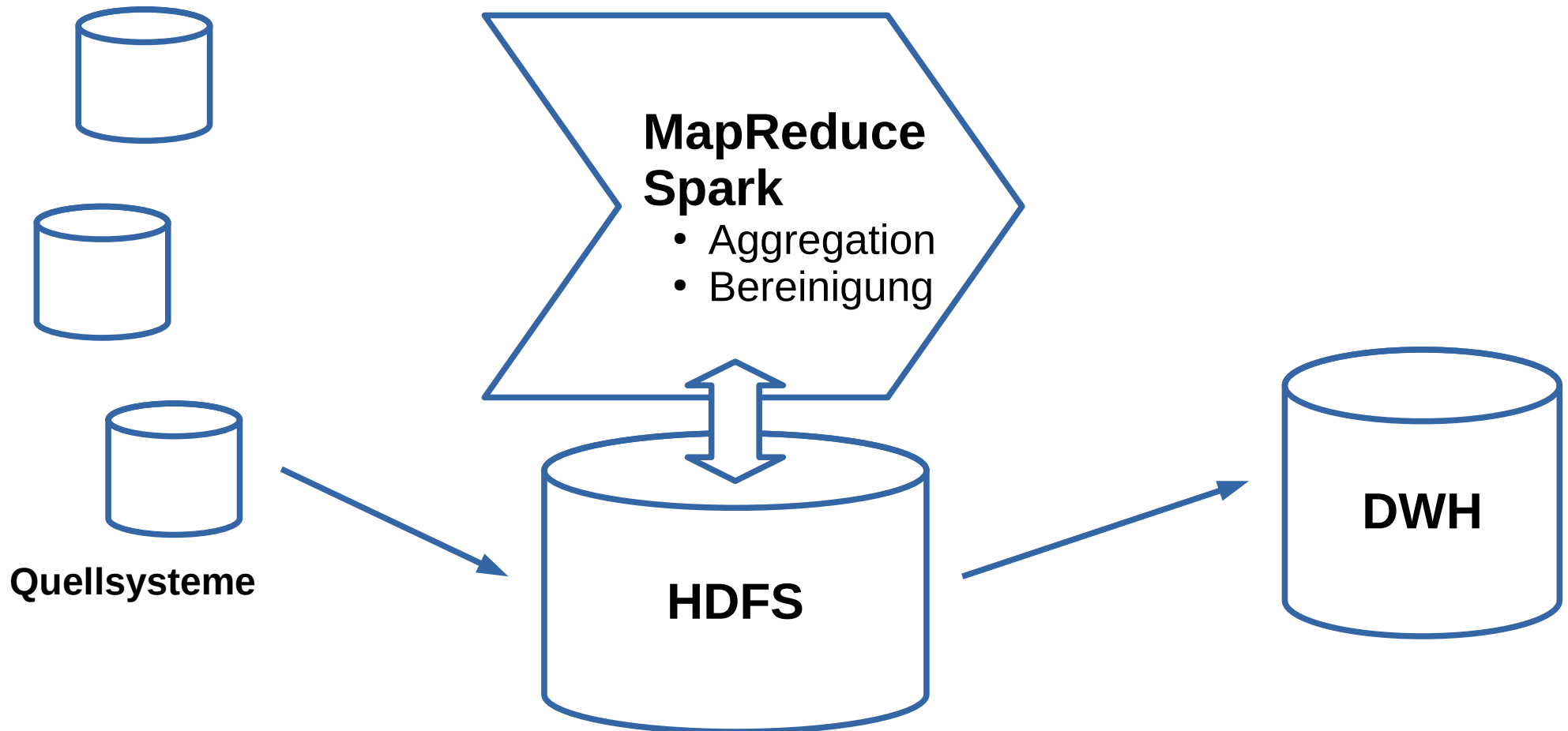
Metadaten

- Hive Metastore
- Falcon
- Cloudera Navigator

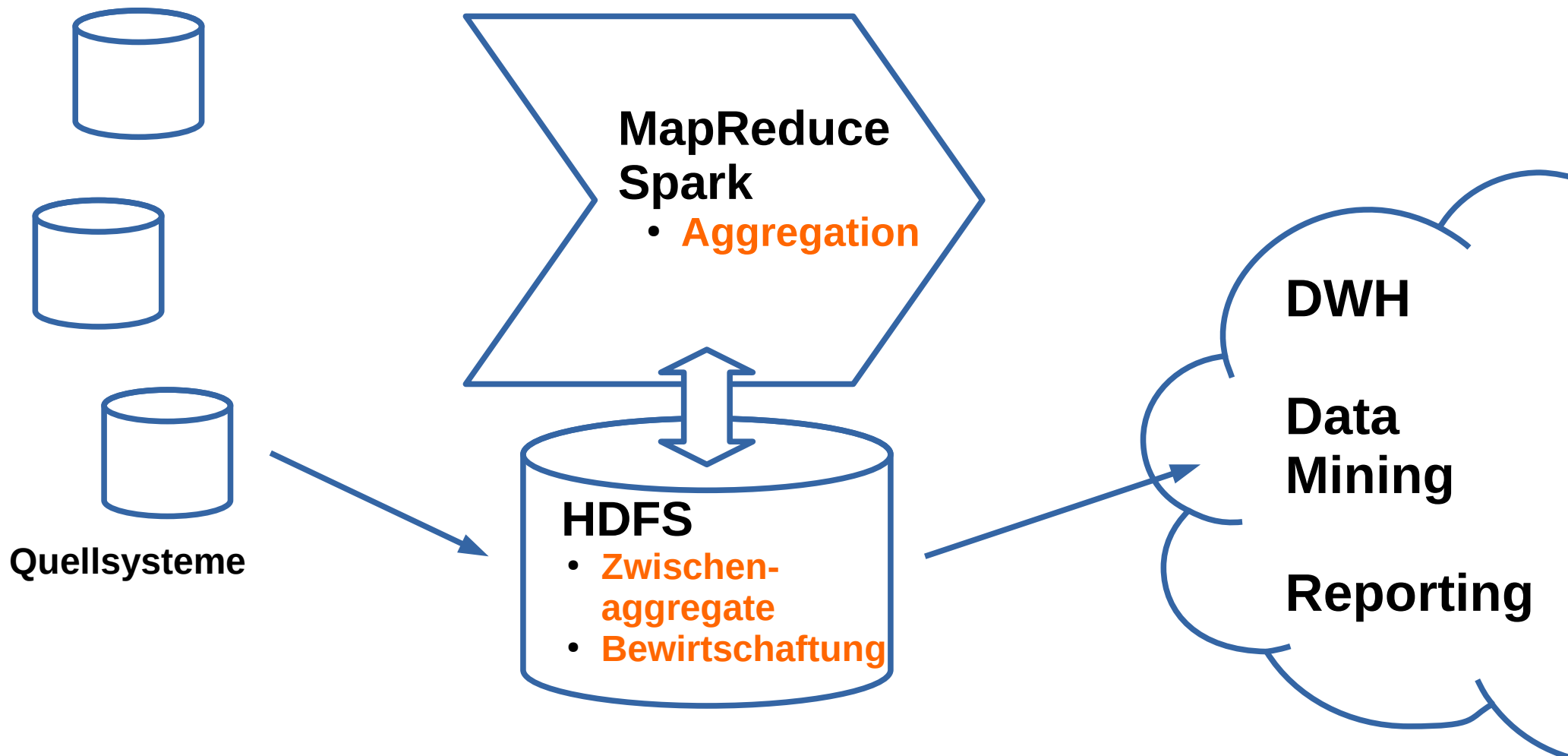
Sicherheit

- Ranger
- Knox

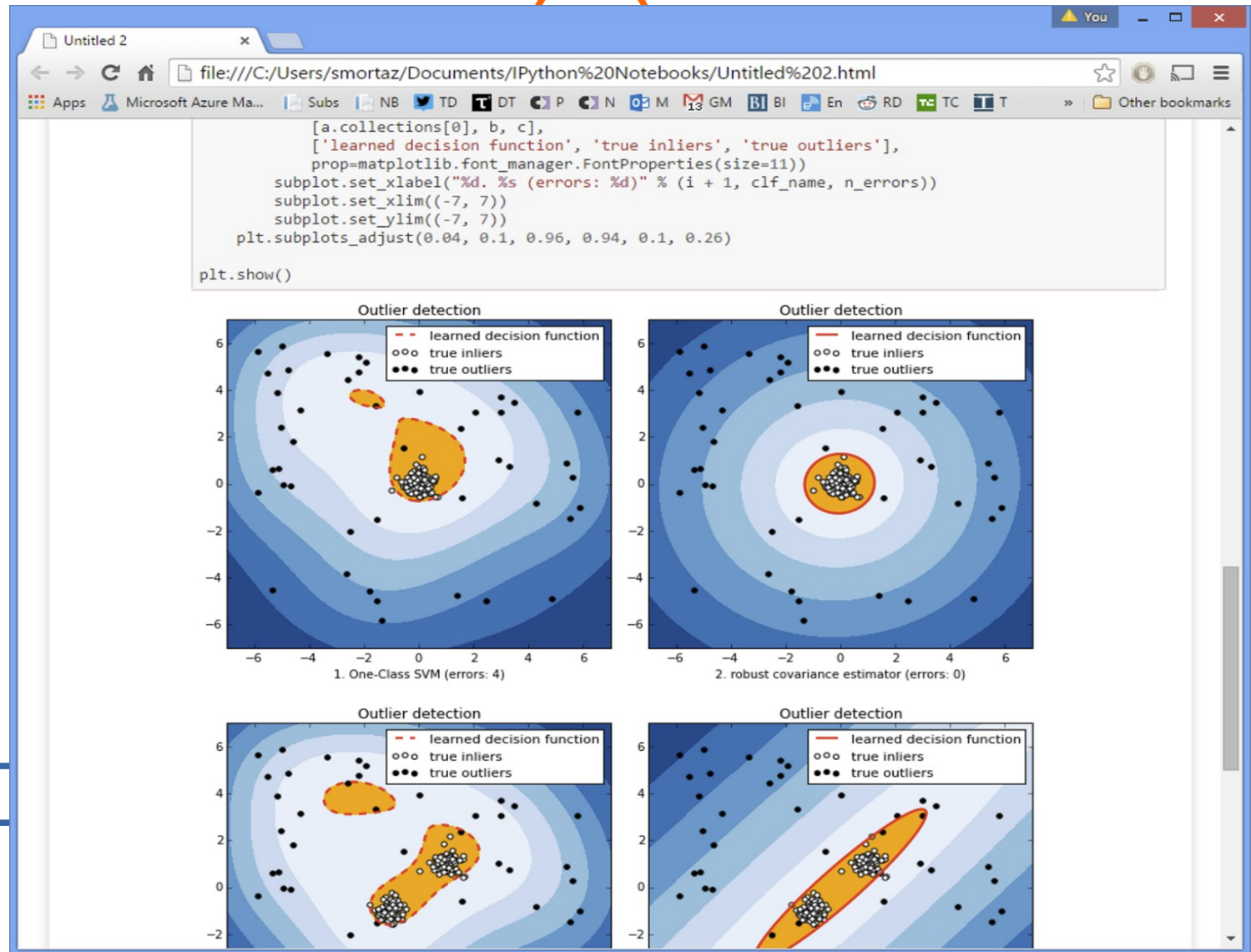
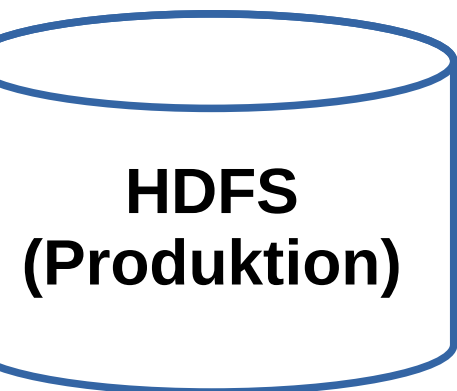
Anwendungsfall: DB/DWH-Entlastung



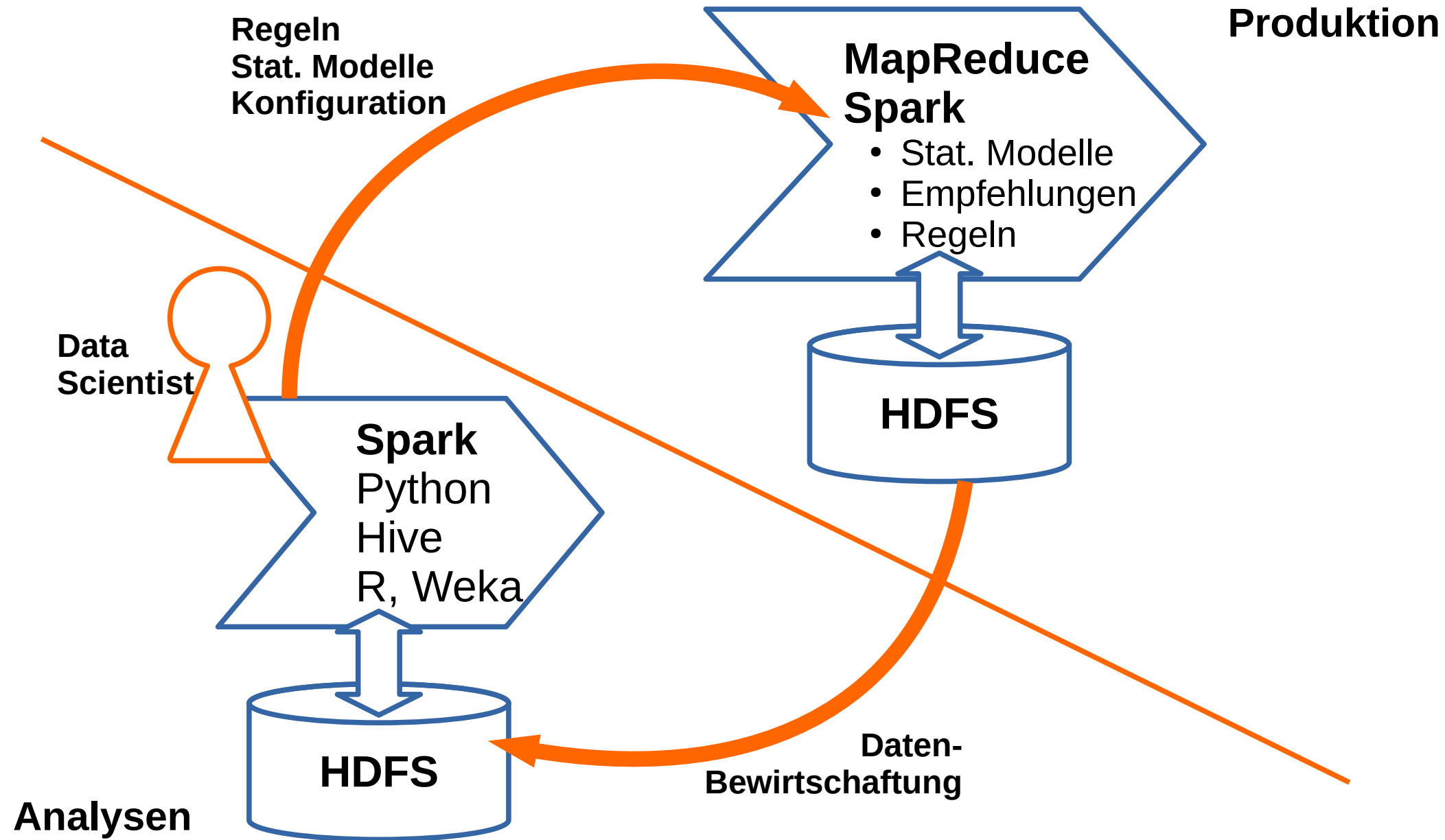
Anwendungsfall: Langfristige Aggregationen



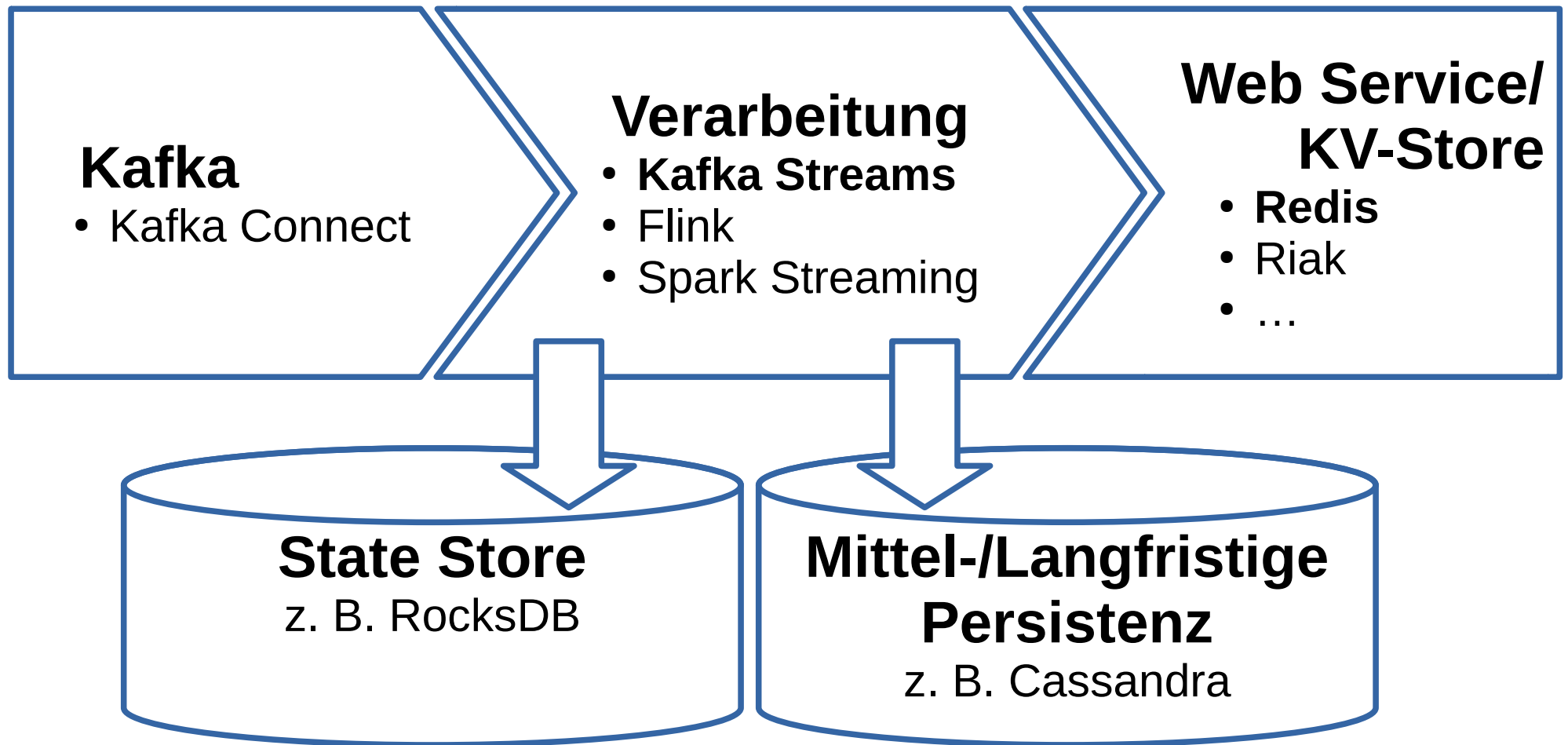
Anwendungsfall: Ad-Hoc-Analysen



Anwendungsfall: Data Mining und Machine Learning



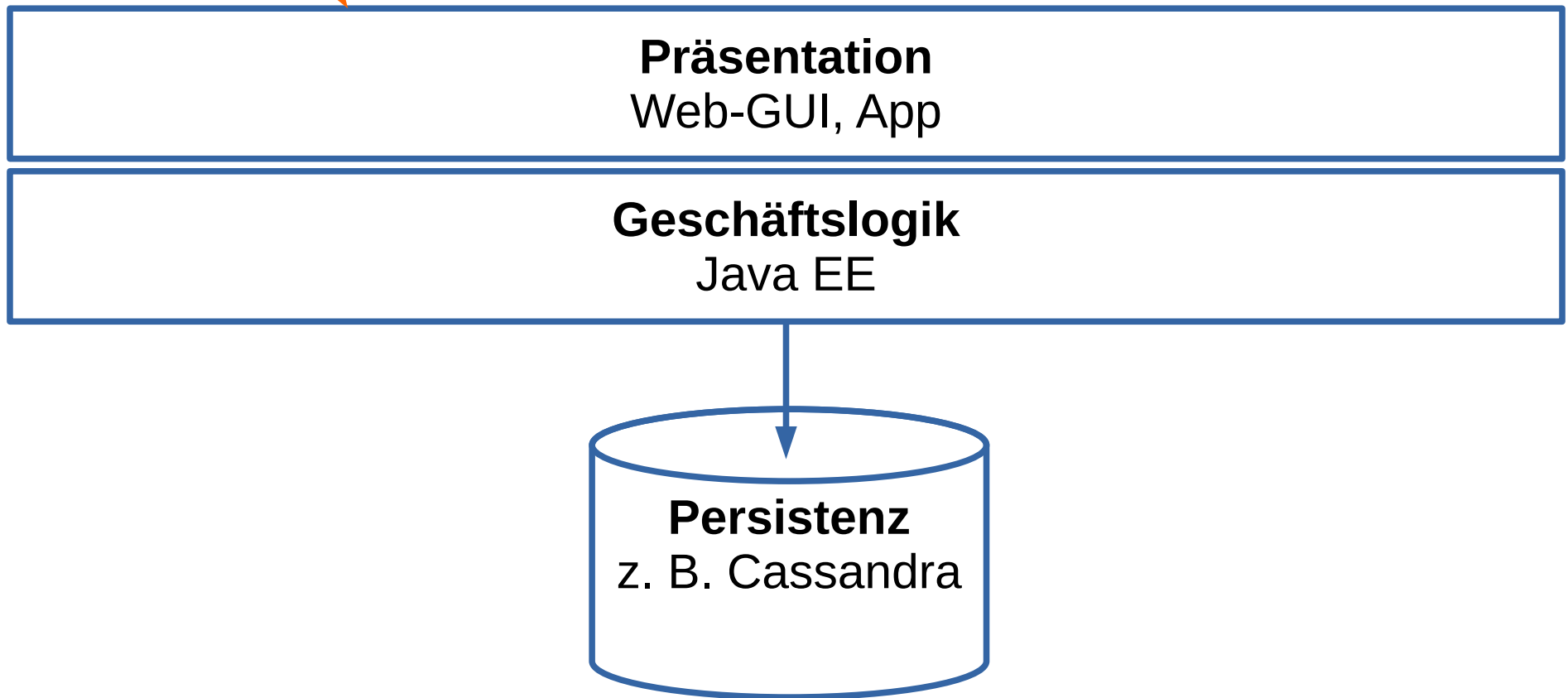
Anwendungsfall: Echtzeit-Verarbeitung



Anwendungsfall:

Transaktionsverarbeitung

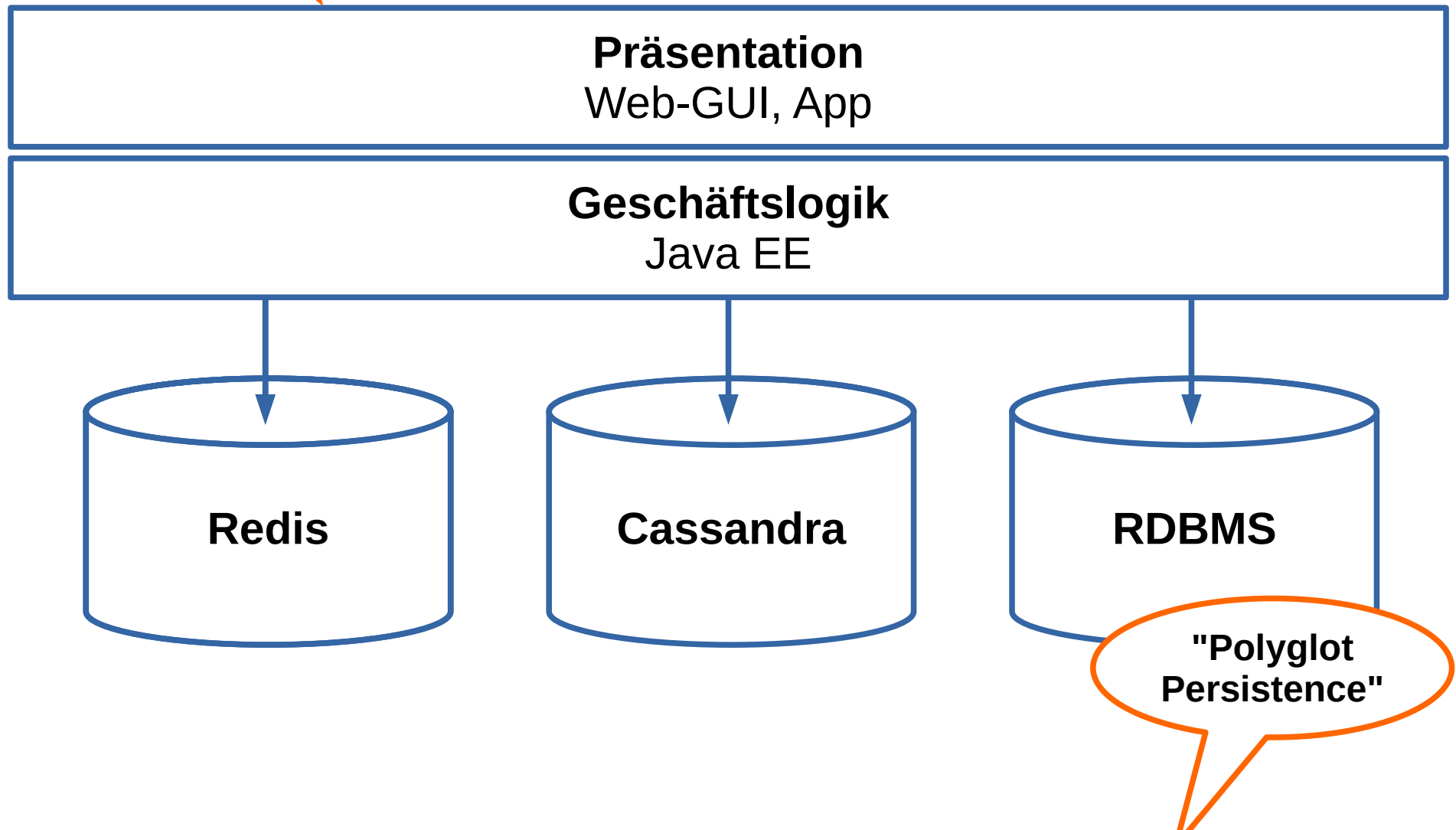
Clients



Anwendungsfall:

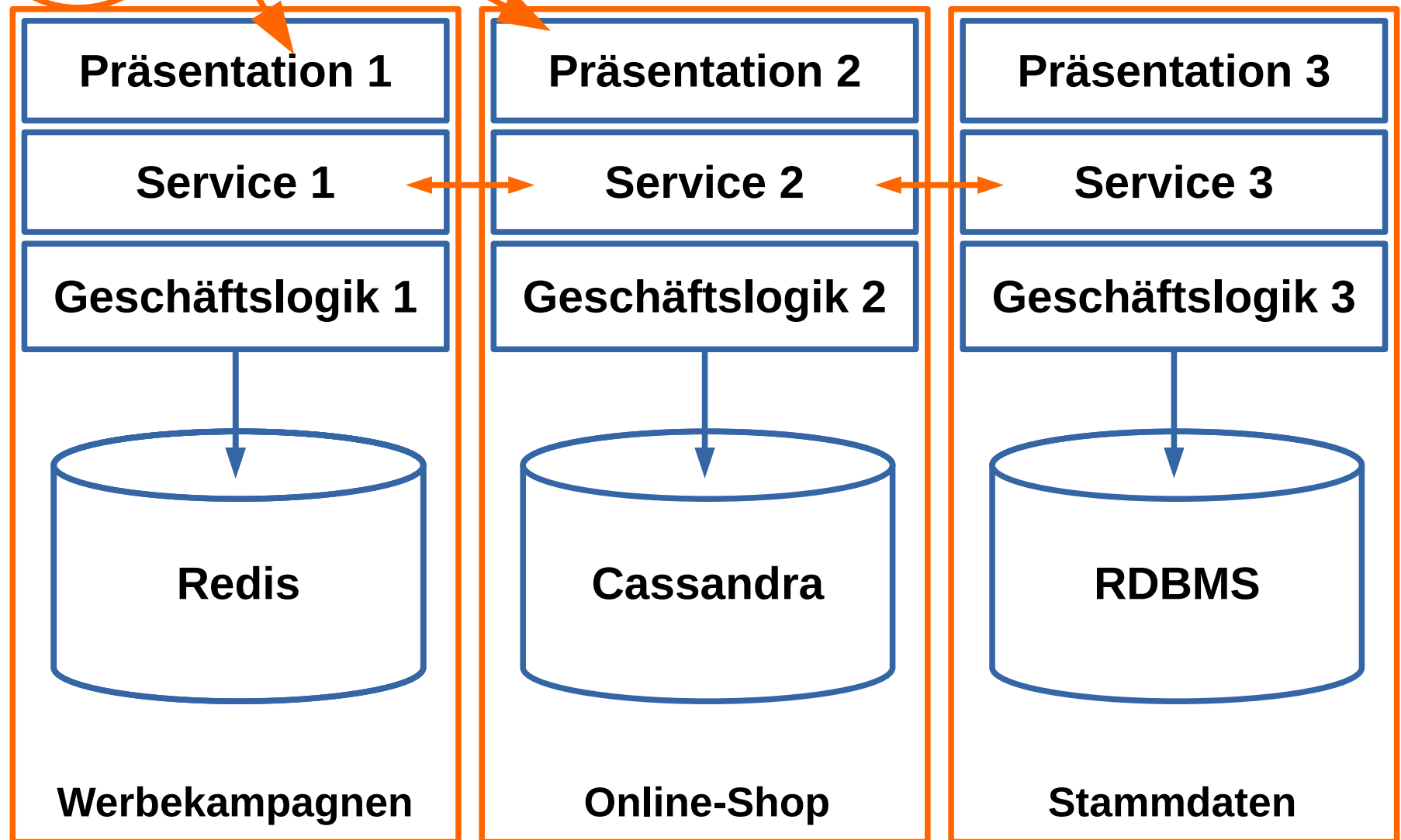
Transaktionsverarbeitung

Clients

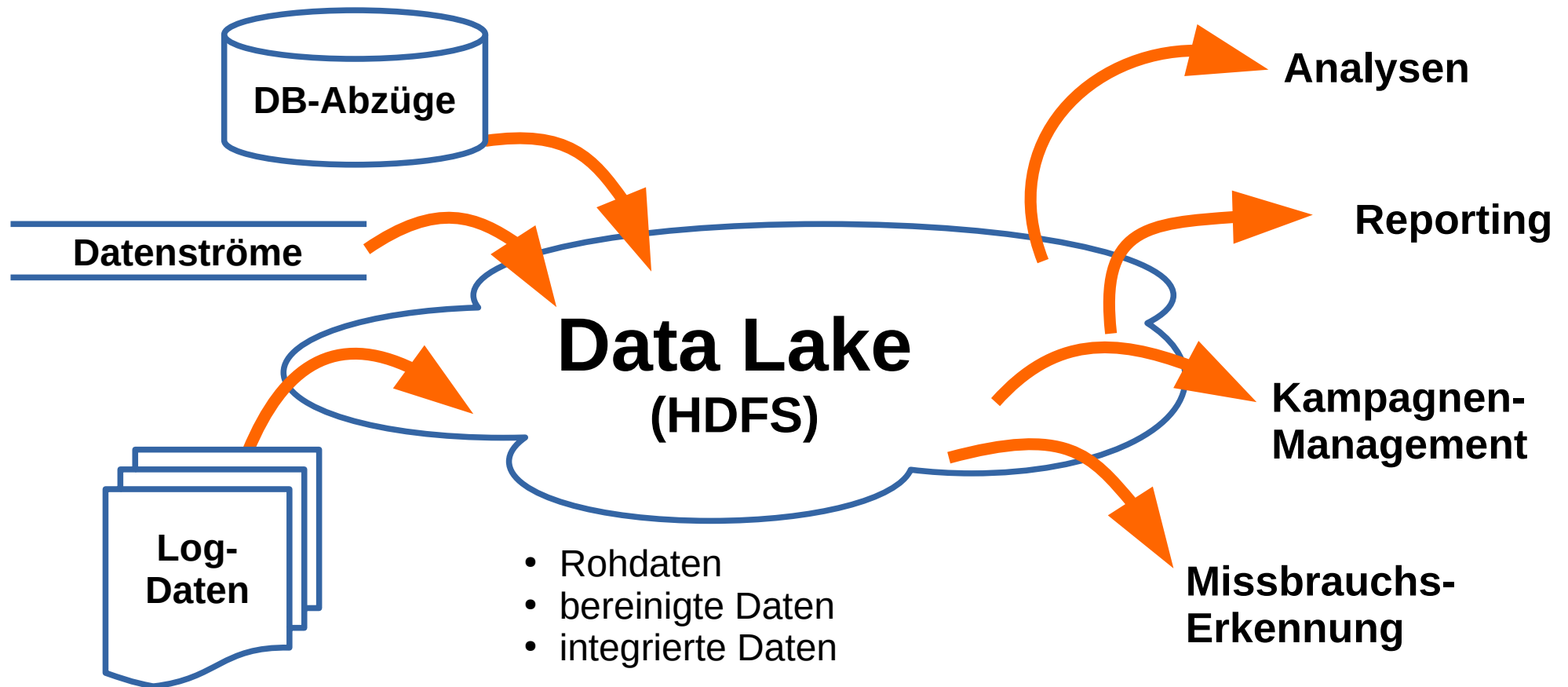


Transaktionsverarbeitung – Microservices

Clients



Anwendungsfall: Data Lake/Data Hub/...



Annäherung ans Ideal: Lambda-Architektur

- **Batch-Verarbeitung**
 - hoher Durchsatz
 - hohe Latenz
 - komplexe Verarbeitung möglich
- **Stream-Verarbeitung**
 - geringe Latenz
 - geringer(er) Durchsatz
 - Probleme mit Zustand und komplexer Verarbeitung



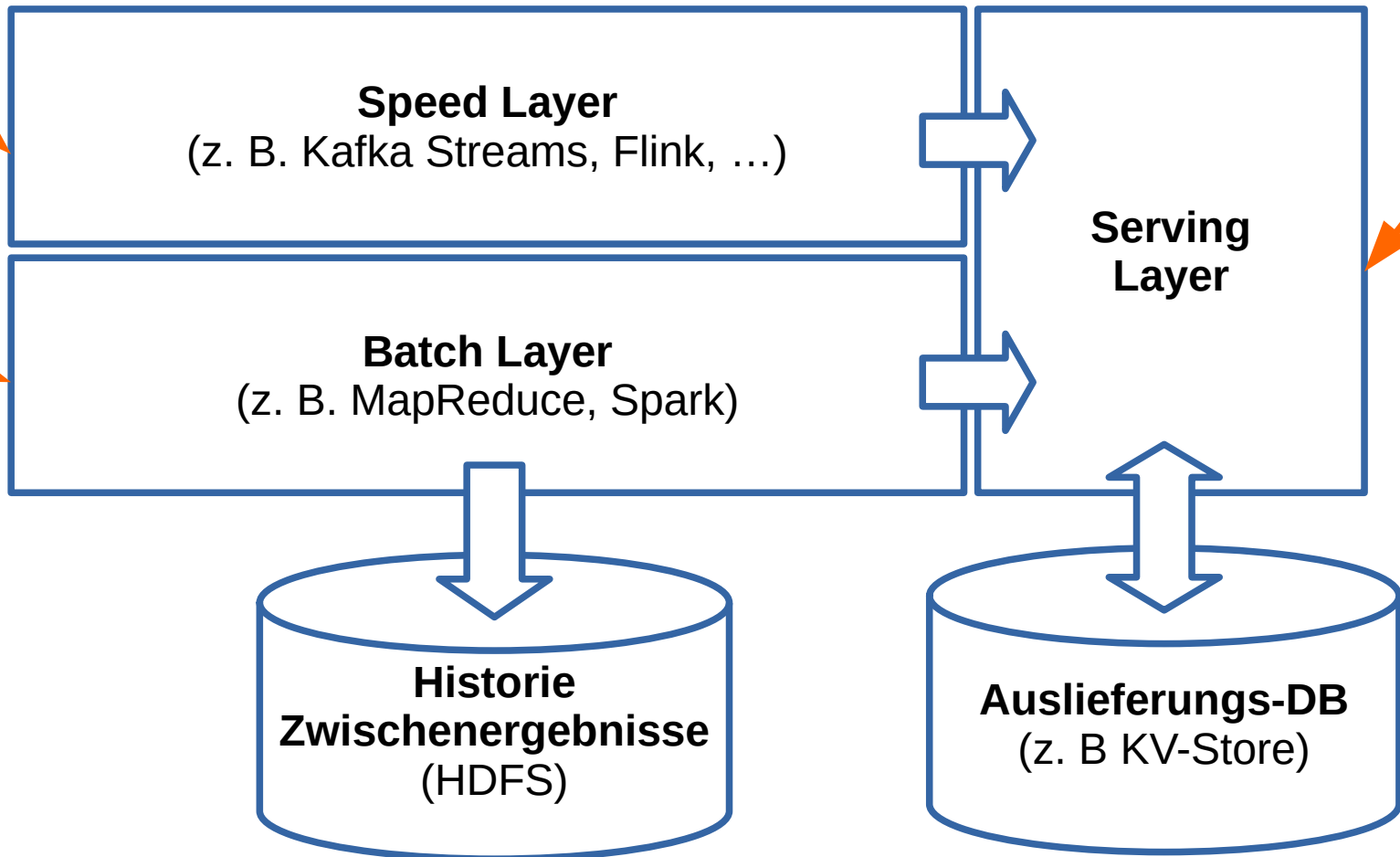
Kombination

- komplexe Verarbeitung
- lange Historie
 - **Batch**
- schnelle Verarbeitung
 - **Stream**

Lambda-Architektur

Datenquellen

Clients



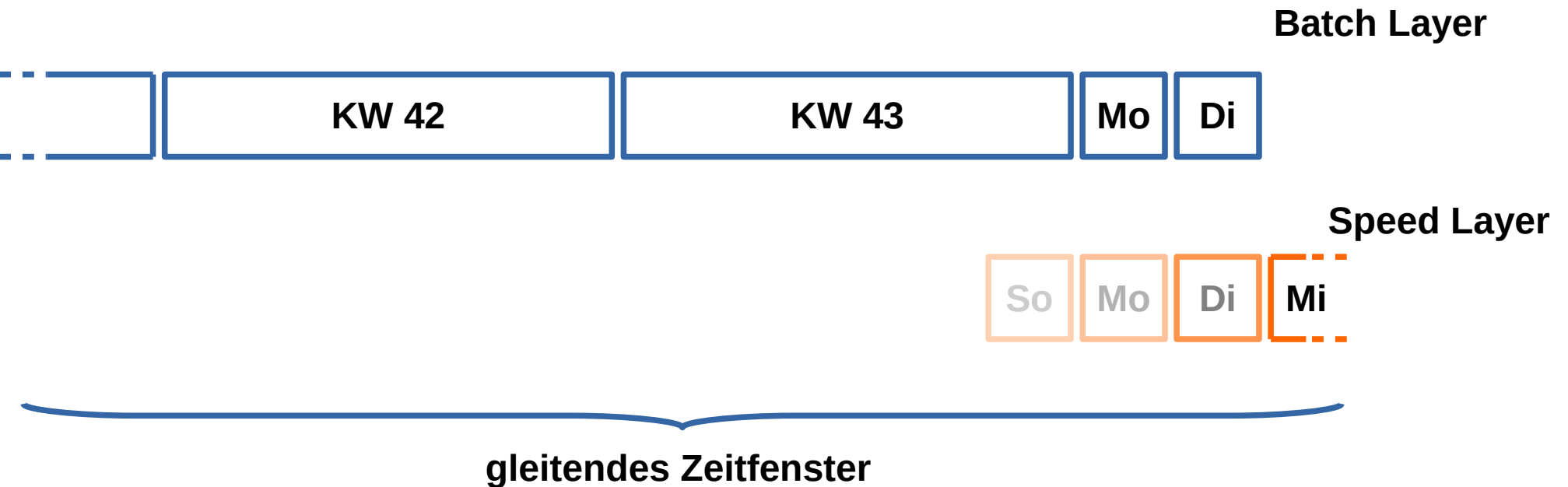
Lambda-Architektur



- **Batch Layer**
 - Langfristige Aggregationen (Monate)
 - Komplexe Logik
 - Historie
- **Speed Layer**
 - Kurzfristige Aggregationen (Stunden)
 - Schnelle Verarbeitung
- **Serving Layer**
 - Integrierte Sicht auf Batch Layer und Speed Layer

Lambda-Architektur

- Beispiel: Langfristige Aggregation



Lambda-Architektur

- **Pro**

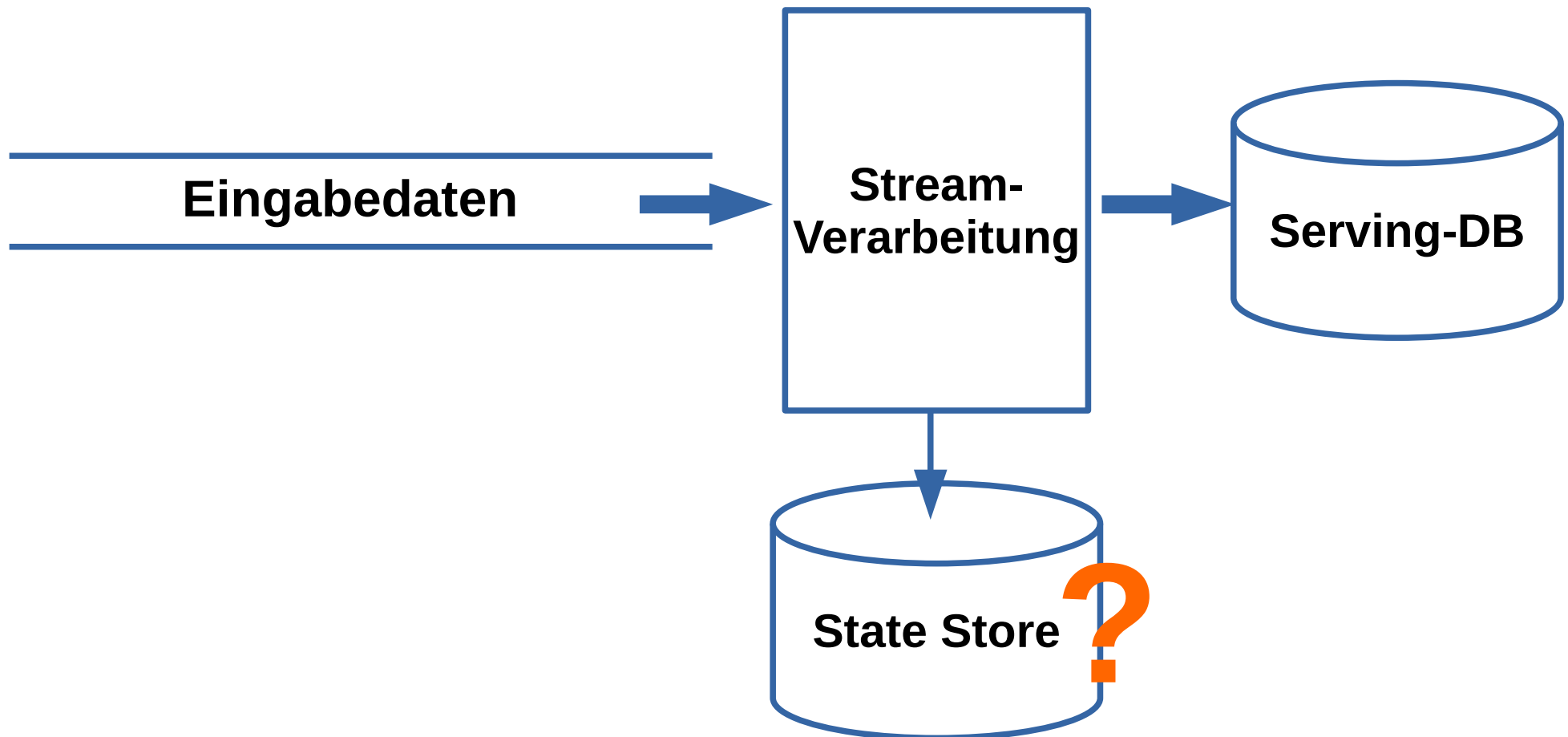
- kombiniert sehr hohen Durchsatz mit sehr geringer Latenz
- komplexe Verarbeitung in Echtzeit

- **Kontra**

- sehr aufwendig und fehleranfällig in der Implementierung:
Batch und Speed Layer müssen gleiche Logik implementieren!

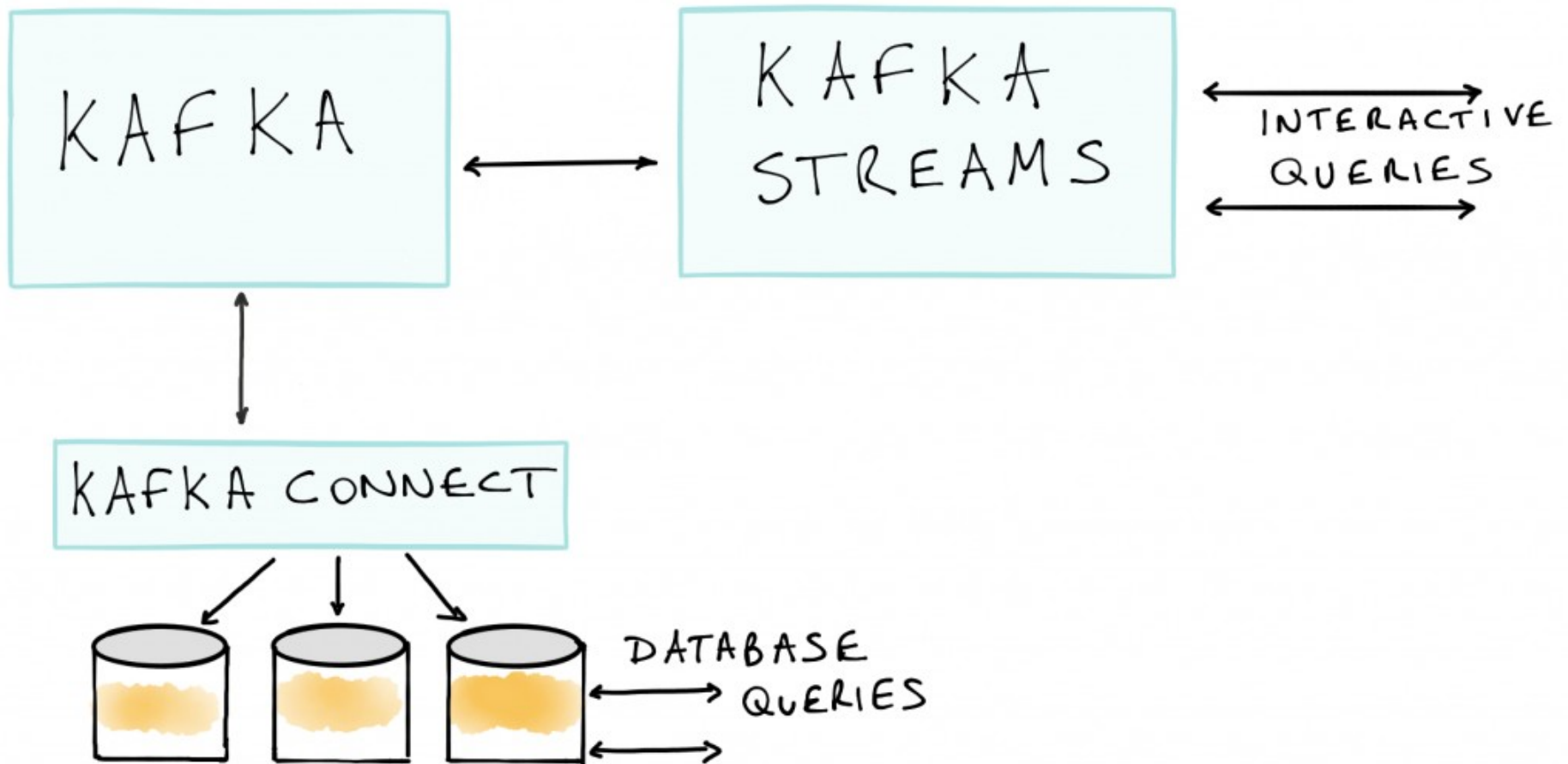
"Kappa-Architektur"

- Alles ist ein Stream!



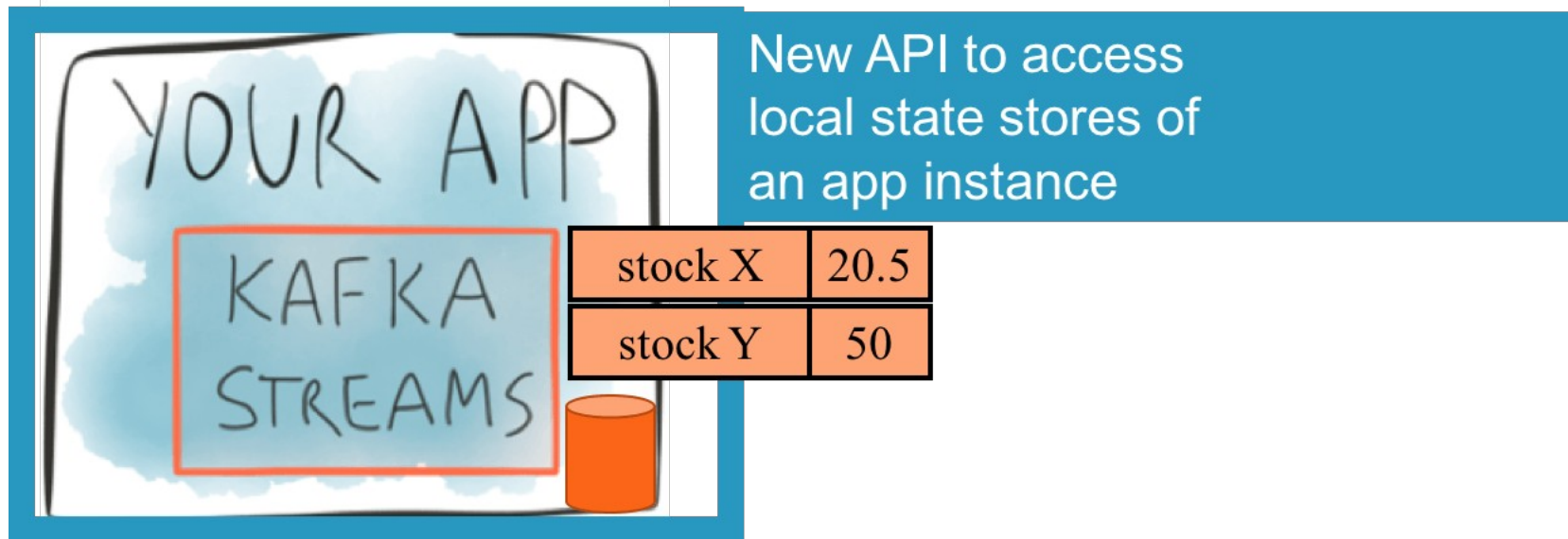
Kappa-Architektur

- Wo lebt der Zustand?



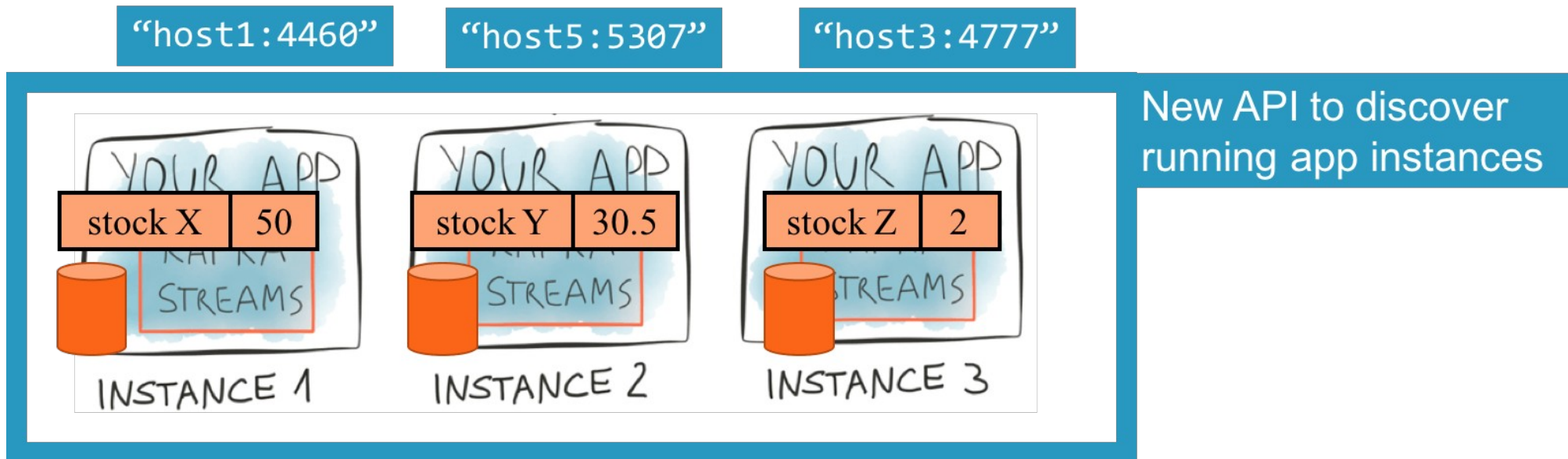
Kappa-Architektur

- Queryable Local State



Kappa-Architektur

- Queryable Local State



Kappa-Architektur


"Mind you, there are of course use cases where you **still need full-fledged external storage** or databases you know and trust such as HDFS, MySQL, or Cassandra.

The Kafka Streams API uses embedded databases to perform its processing, however querying them is optional, in that you **can always copy their data** to an external database first."

(Eno Thereska, Confluent)

Quelle: <https://www.confluent.io/blog/unifying-stream-processing-and-interactive-queries-in-apache-kafka/>

Zusammenfassung

- **Qualitätsziele/nonfunktionale Anforderungen** bestimmen die Architektur
- **Eingabe-Verarbeitung-Ausgabe**
 - ... plus flankierende Dienste
- Bausteine:
 - **Batch**
 - **Stream**
 - NoSQL-DB

jeweils austauschbar