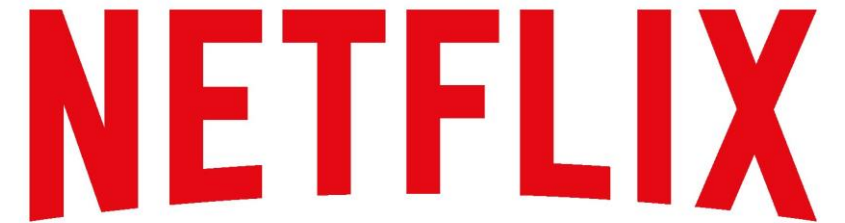




Echtzeit-Datenverarbeitung mit Apache Kafka und Apache Spark

Eric Metternich, Niklas Wilhelm, Timo Feindor

**Netflix generiert in Echtzeit neue
Empfehlungen für TV-Shows**

The Netflix logo, consisting of the word "NETFLIX" in a bold, red, sans-serif font.

**PayPal überprüft in Echtzeit
Transaktionen auf möglichen Betrug**

The PayPal logo, featuring the word "Pay" in a dark blue font and "Pal" in a light blue font, both in a bold, sans-serif font.

1. Grundlagen von Apache Kafka



Kleine Pause

2. Apache Kafka: Hands-On



Pause

3. Grundlagen von Apache Spark

4. Einführung in Databricks



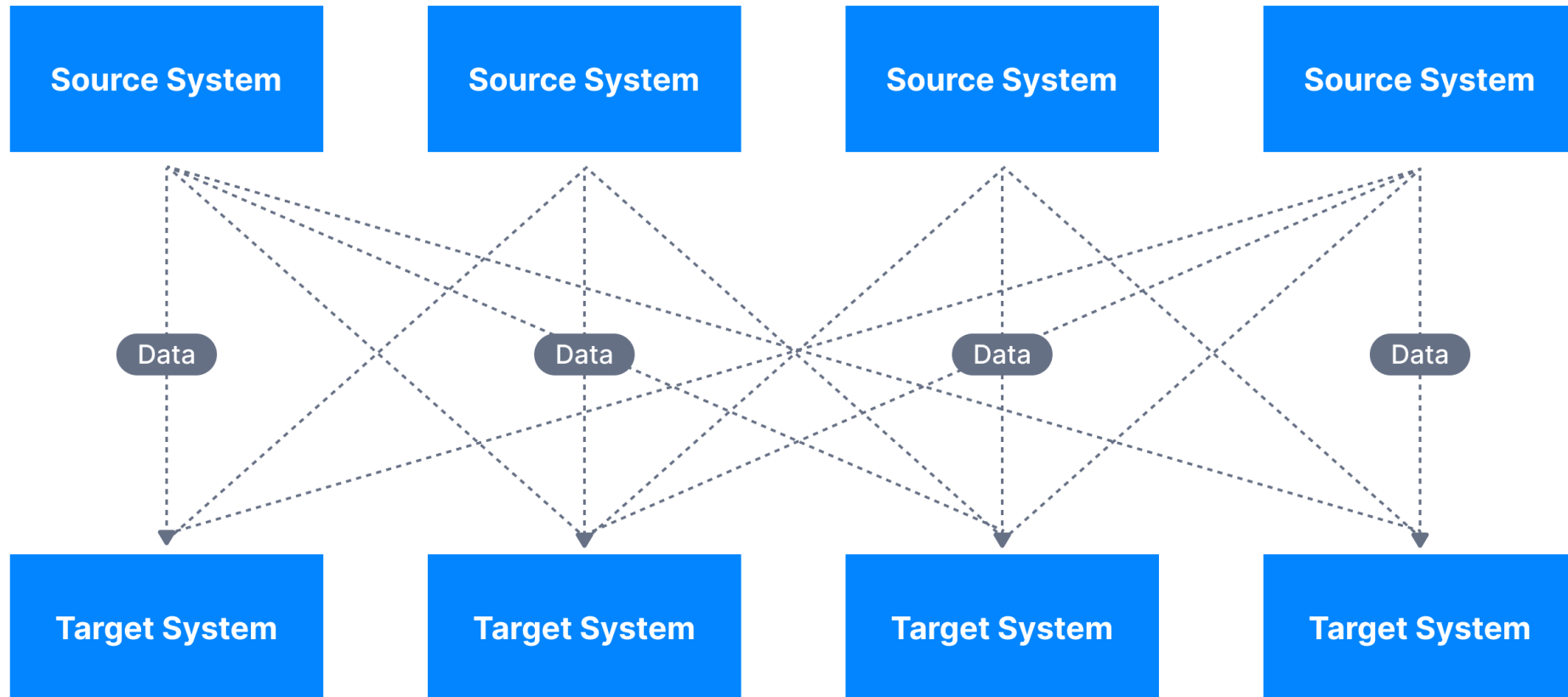
Kleine Pause

5. Apache Spark mit Databricks: Hands-On



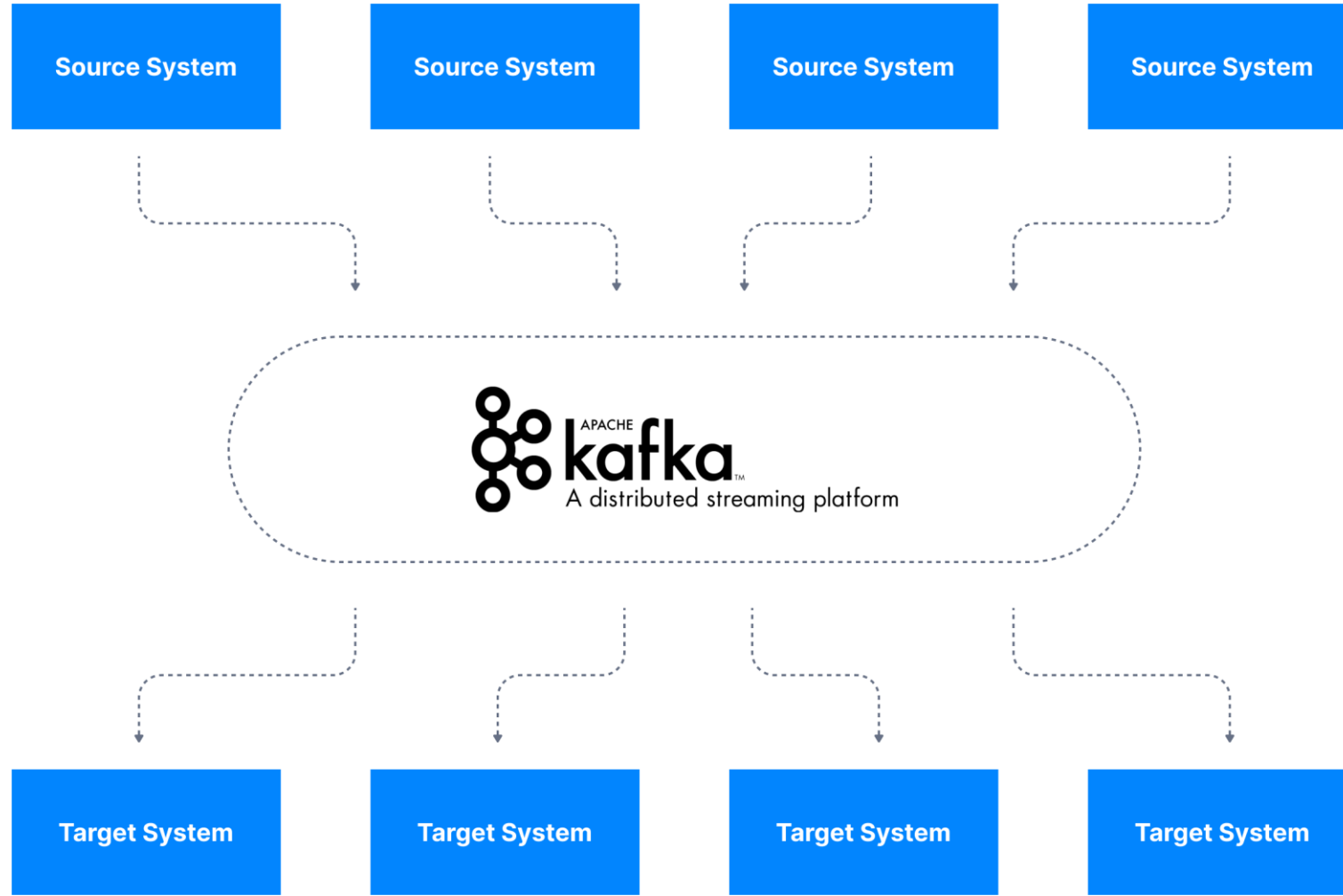
Apache Kafka

Ausgangssituation in Unternehmen



Sehr kompliziert!

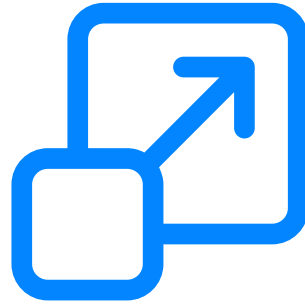
Apache Kafka zur Rettung



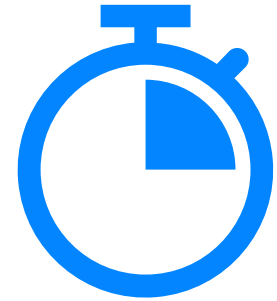
Argumente für Apache Kafka



**Fehlertolerante
Architektur**



**Horizontale
Skalierbarkeit**



Geringe Latenz

Topic: eine Sammlung von Nachrichten mit einem eindeutigen Namen

- Eine neue Nachricht wird immer an das Ende des Topics geschrieben (append-only)
- In das Topic geschriebene Nachrichten sind unveränderbar (immutable)
- Keine Vorgabe, welches Format der Inhalt einer Nachricht haben muss
- Producer erzeugen und Consumer konsumieren Nachrichten

Kafka Cluster

logs

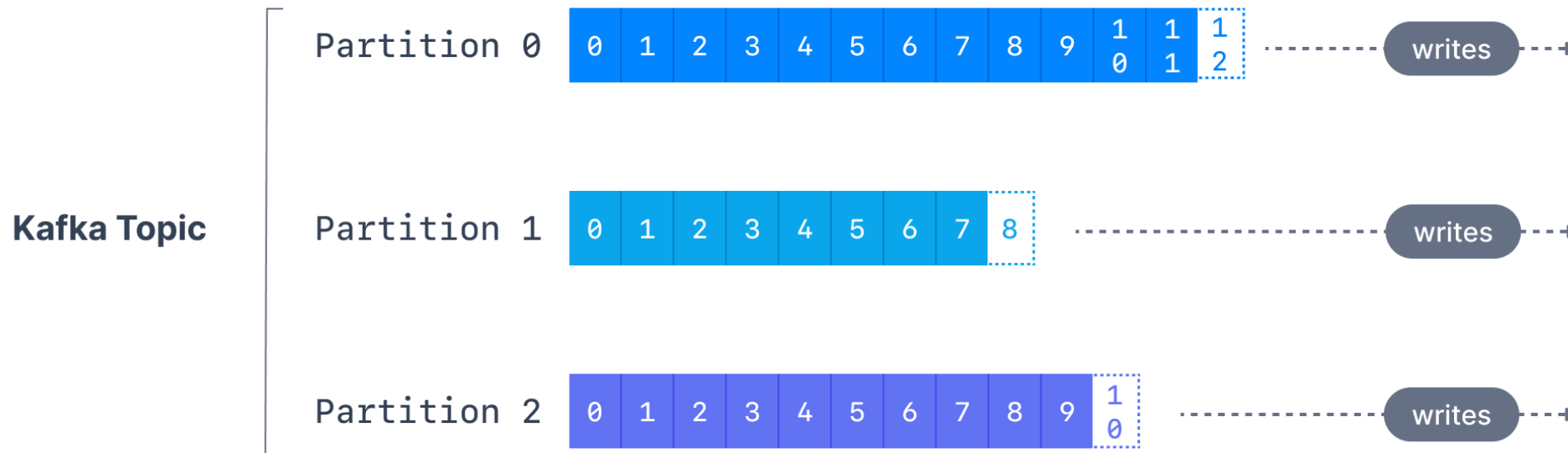
purchases

twitter_tweets

trucks_gps

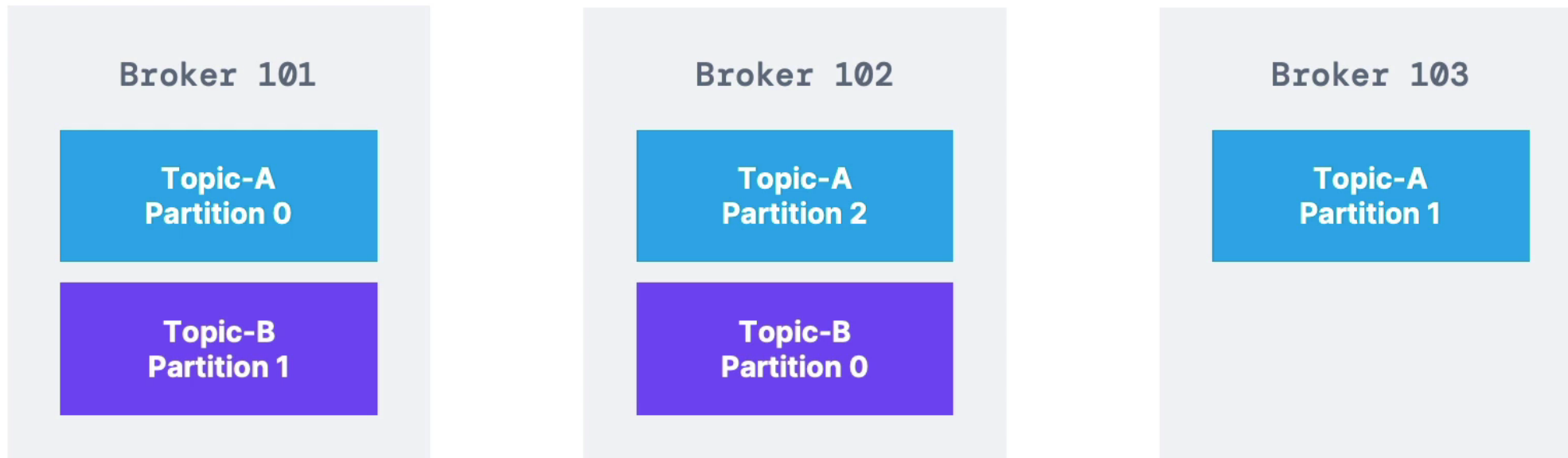
Partitionen und Offsets

- Ein Topic wird in mehrere Partitionen aufgeteilt
- Jede Nachricht innerhalb einer Partition erhält eine inkrementelle ID (Offset)
- Die korrekte Reihenfolge der Nachrichten ist nur innerhalb einer Partition gewährleistet



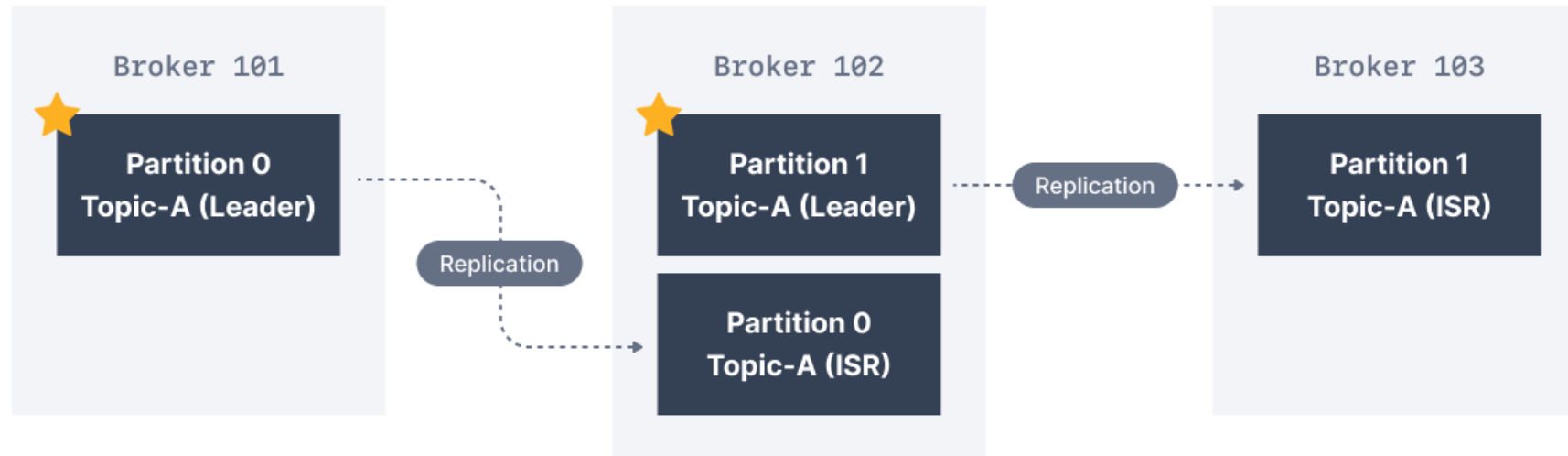
Broker: Ein einzelner Server in einem Kafka-Cluster

- Jeder Broker enthält bestimmte Partitionen eines Topics
- Broker dienen der horizontalen Skalierung bei höherer Rechenlast



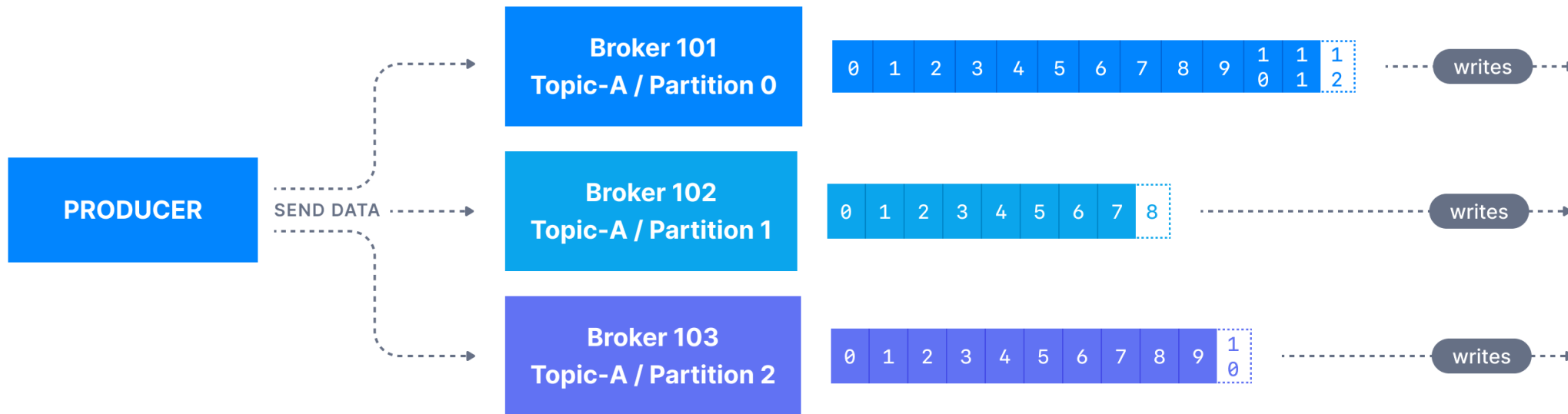
Topic Replication

- In Produktionsumgebungen sollten Topics auf mehreren Brokern gespeichert werden
- Meistens wird ein Replication Factor von 3 verwendet
- **Leader:** Broker, der für die Partition zuständig ist
- **In-Sync Replica:** Broker, der die Partition repliziert und bei einem Ausfall übernehmen kann



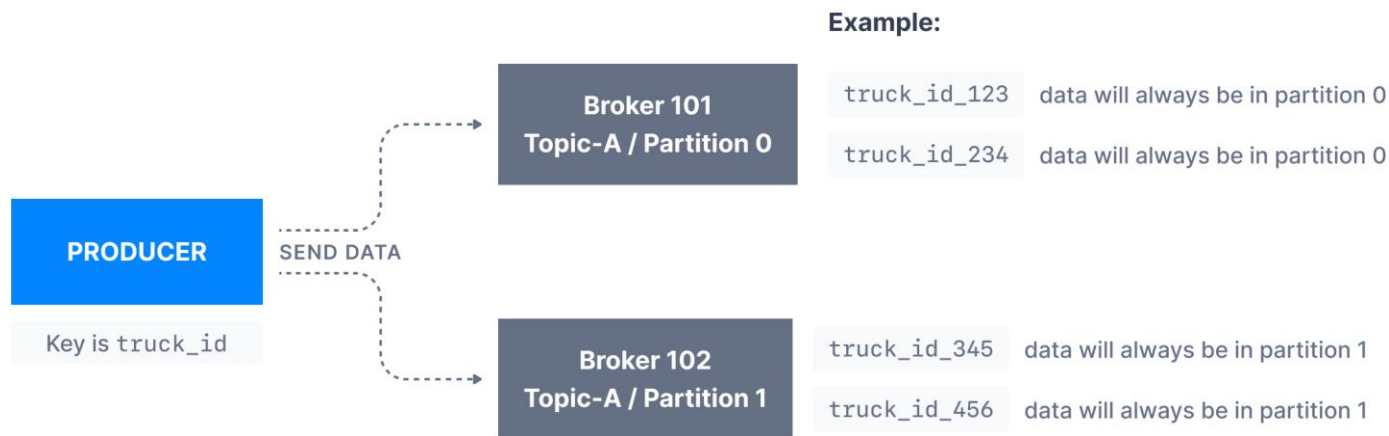
Producer: eine Anwendung, die Nachrichten in ein Kafka-Topic schreibt

- Producer verwenden i.d.R. eine Kafka-Library, um mit dem Cluster zu kommunizieren
- Kafka-Libraries existieren für fast alle Programmiersprachen (Java, Python, Go, ...)



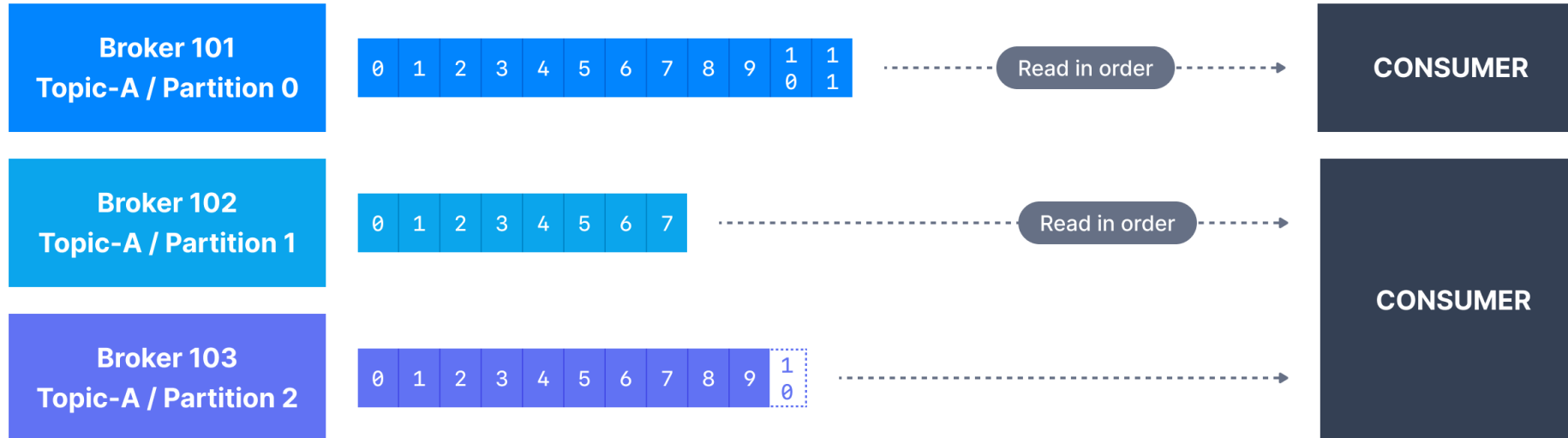
Nachrichten-Keys

- Producer können eine Nachricht mit einem Key versehen
- *Key=null* → Nachrichten werden nach dem Round-Robin-Prinzip den Partitionen zugeordnet (Default-Partitioner)
- *Key!=null* → Nachrichten werden entsprechend des Key-Hashes den Partitionen zugeordnet
- Ein Key wird verwendet, wenn die Reihenfolge der Nachrichten relevant ist



Consumer: eine Anwendung, die Nachrichten von einem Kafka-Topic liest

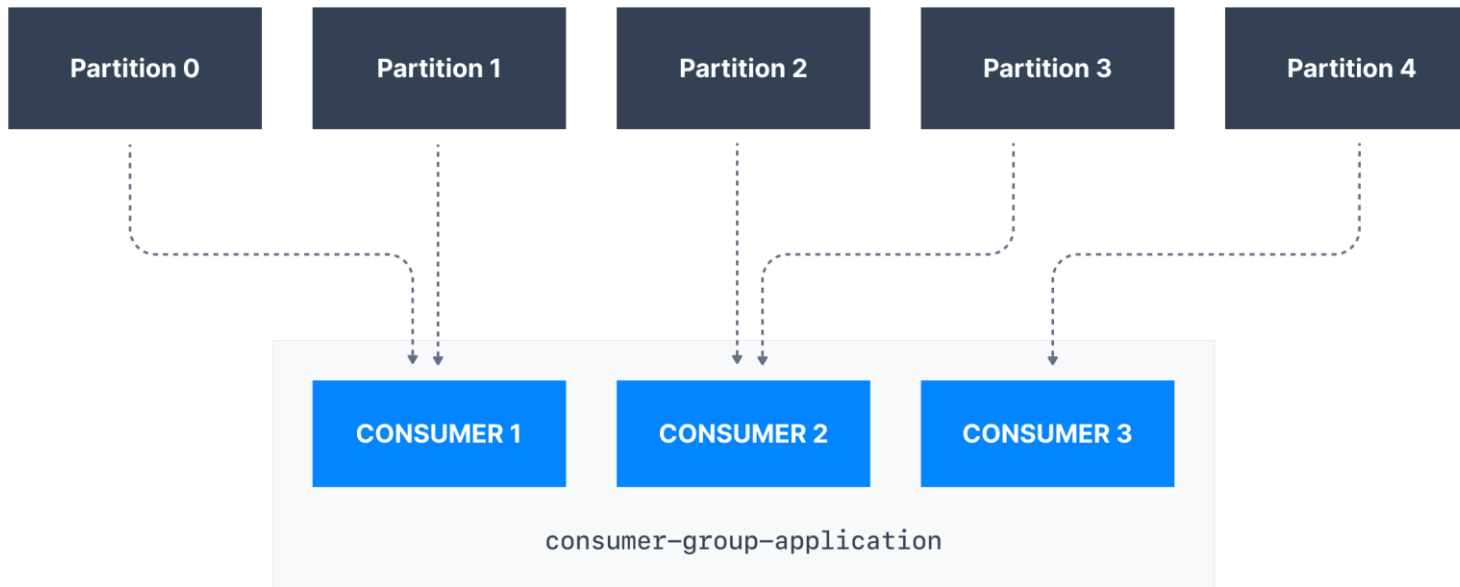
- Pull-Modell: Consumer müssen die Nachrichten selbst abrufen
- Consumer verwenden wie die Producer i.d.R. eine Kafka-Library, um mit dem Cluster zu kommunizieren



Consumer Groups

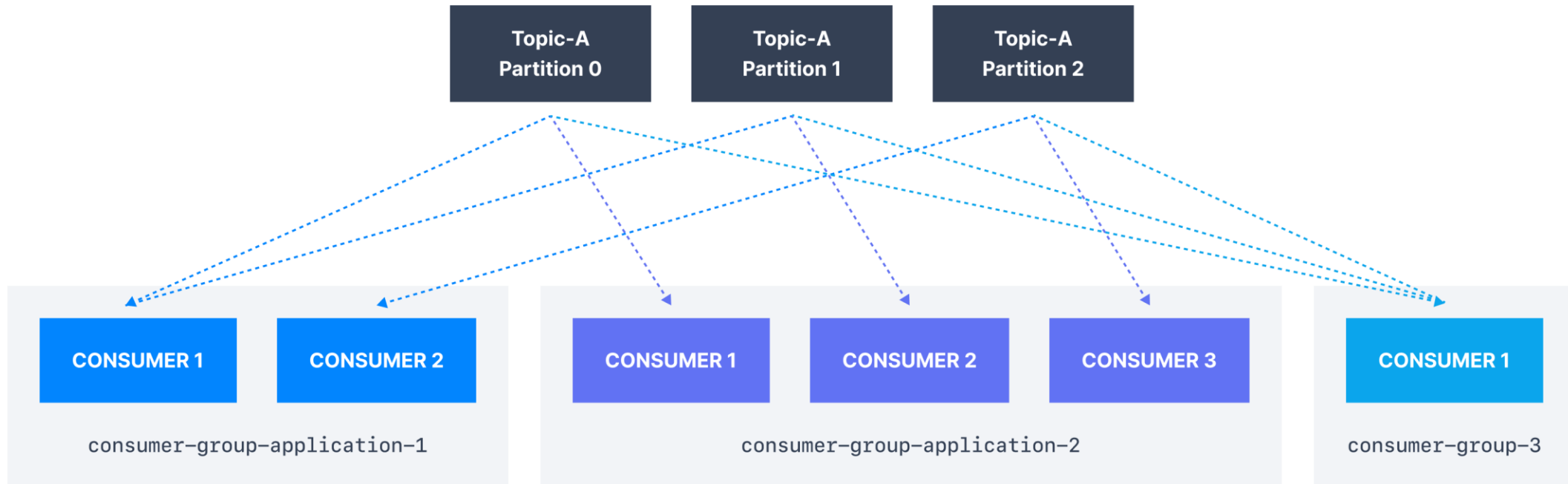
Consumer Group: eine Gruppe von Consumern, die den gleichen Job verrichten und sich die Arbeit aufteilen

- Jede Partition wird nur von einem Consumer innerhalb der Gruppe verarbeitet
- Ein Consumer innerhalb der Gruppe kann für mehrere Partitionen zuständig sein



Zusammenspiel von Consumer Groups

Mehrere Consumer Groups können das gleiche Topic verarbeiten



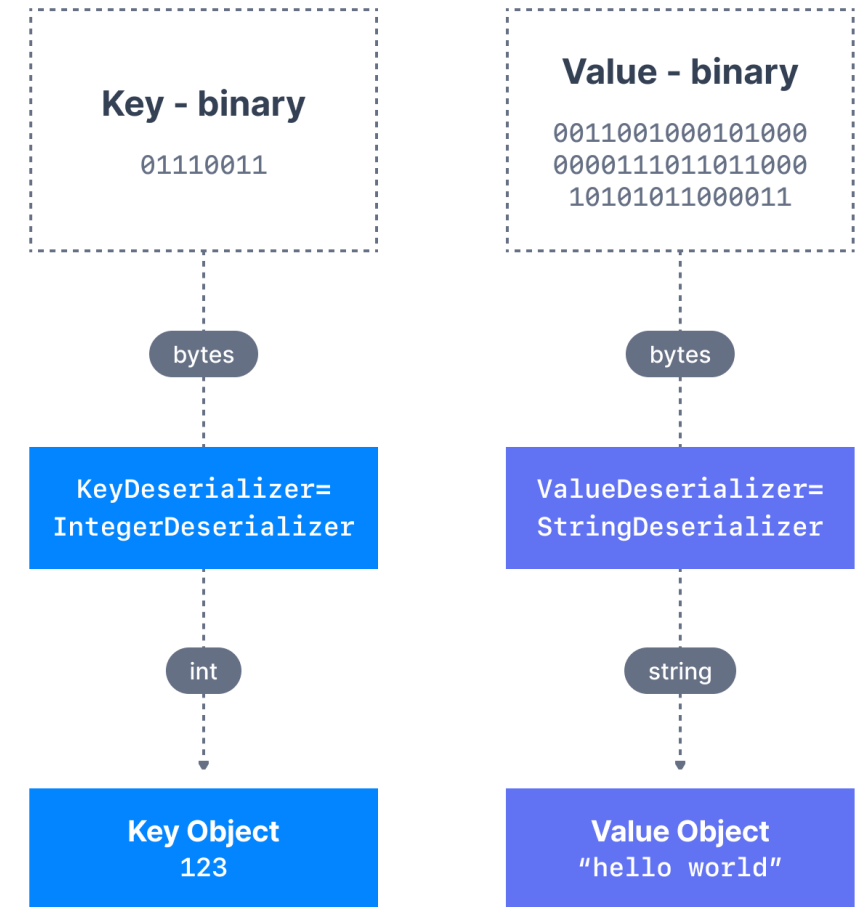
Consumer Offsets

- Consumer sollten das zuletzt von ihnen verarbeitete Offset committen
- Stürzt der Consumer ab, kann ein anderer Consumer der gleichen Gruppe mit der Verarbeitung der Daten fortfahren
- Kafka bietet unterschiedliche Commit-Typen an:
 - At-Least-Once
 - At-Most-Once
 - Exactly-Once



Serializer und Deserializer

- Kafka-Broker verarbeiten die Nachrichten im Binärformat
- Producer und Consumer müssen für die Umwandlung von Nachrichten Serializer bzw. Deserializer verwenden
- Die Kafka-Libraries stellen für einige Datentypen bereits Serializer und Deserializer zur Verfügung (Integer, String, Float, Protobuf, ...)
- Für komplexe Datentypen können benutzerdefinierte Serializer und Deserializer implementiert werden



1. Aufsetzen eines Kafka-Clusters

URL zur Cloudplattform: *<https://upstash.com>*

2. Implementierung eines Wikimedia-Producers

URL zur Datenquelle: *<https://stream.wikimedia.org/v2/stream/recentchange>*

URL zur Producer-Vorlage: *<https://github.com/Daumel/data-engineering>*



Apache Spark

Batch-Verarbeitung

- Große Gruppen von Daten/Transaktionen werden in einem Durchgang verarbeitet.
- Jobs laufen ohne manuelle Eingriffe ab.
- Alle Daten sind vorab ausgewählt und werden über Kommandozeilenparameter und Skripte eingespeist.
- Wird verwendet, um mehrere Operationen auszuführen, schwere Datenlasten zu bewältigen, Berichterstattung und Offline-Daten-Workflow.
- **Beispiel:** Regelmäßige Berichte, die Entscheidungsfindung erfordern

Echtzeit-Verarbeitung

- Datenverarbeitung erfolgt sofort bei Dateneingabe oder Erhalt.
- Muss innerhalb strenger Zeitbeschränkungen ausgeführt werden.
- **Beispiel:** Betrugserkennung

MapReduce Limitations



Ungeeignet für Echtzeitverarbeitung

Da es sich um ein batchorientiertes System handelt, dauert es Minuten, Jobs abhängig von der Datenmenge und der Anzahl der Knoten im Cluster auszuführen.



Ungeeignet für triviale Operationen

Bei Operationen wie Filter und Joins könnte es nötig sein, die Jobs umzuschreiben, was aufgrund des Schlüssel-Wert-Musters komplex wird.



Unpassend für große Datenmengen im Netzwerk

Es funktioniert nach dem Prinzip der Datenlokalität, kann jedoch nicht gut eine große Menge an Daten verarbeiten, die über das Netzwerk verschoben werden müssen.

MapReduce Limitations



Ungeeignet für OLTP (Online Transaction Processing)

OLTP erfordert eine große Anzahl kurzer Transaktionen, was aufgrund des Batch-orientierten Frameworks nicht umsetzbar ist.



Ungeeignet zur Verarbeitung von Graphen

Die Apache Giraph-Bibliothek verarbeitet Graphen und fügt zusätzliche Komplexität oben auf MapReduce hinzu.

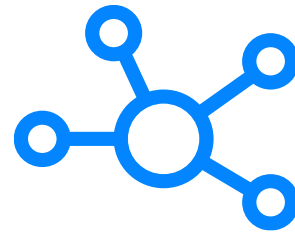


Ungeeignet für iterative Ausführung

Als zustandslose Ausführung passt MapReduce nicht zu Anwendungsfällen wie Kmeans, die iterative Ausführung benötigen.



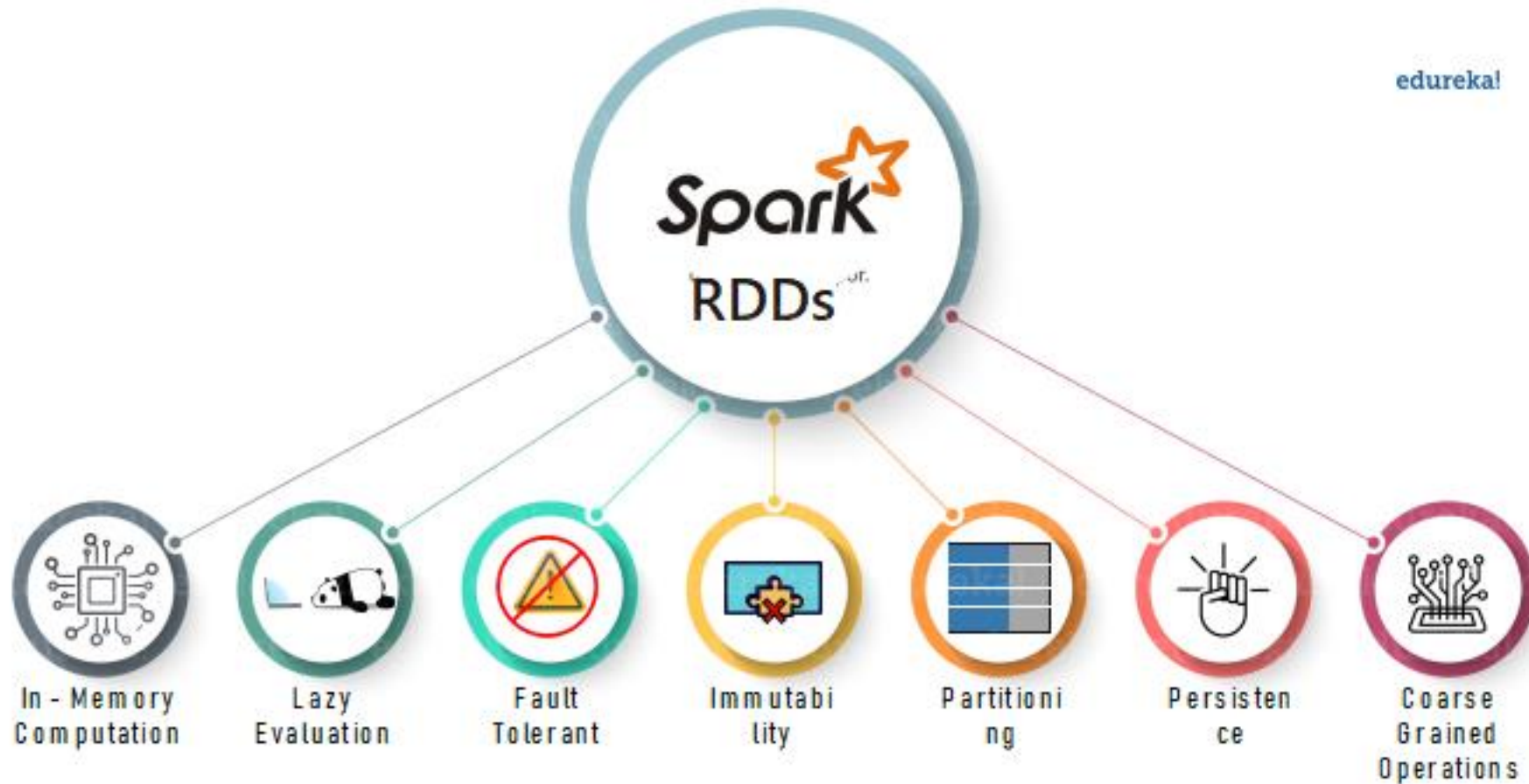
Resilient



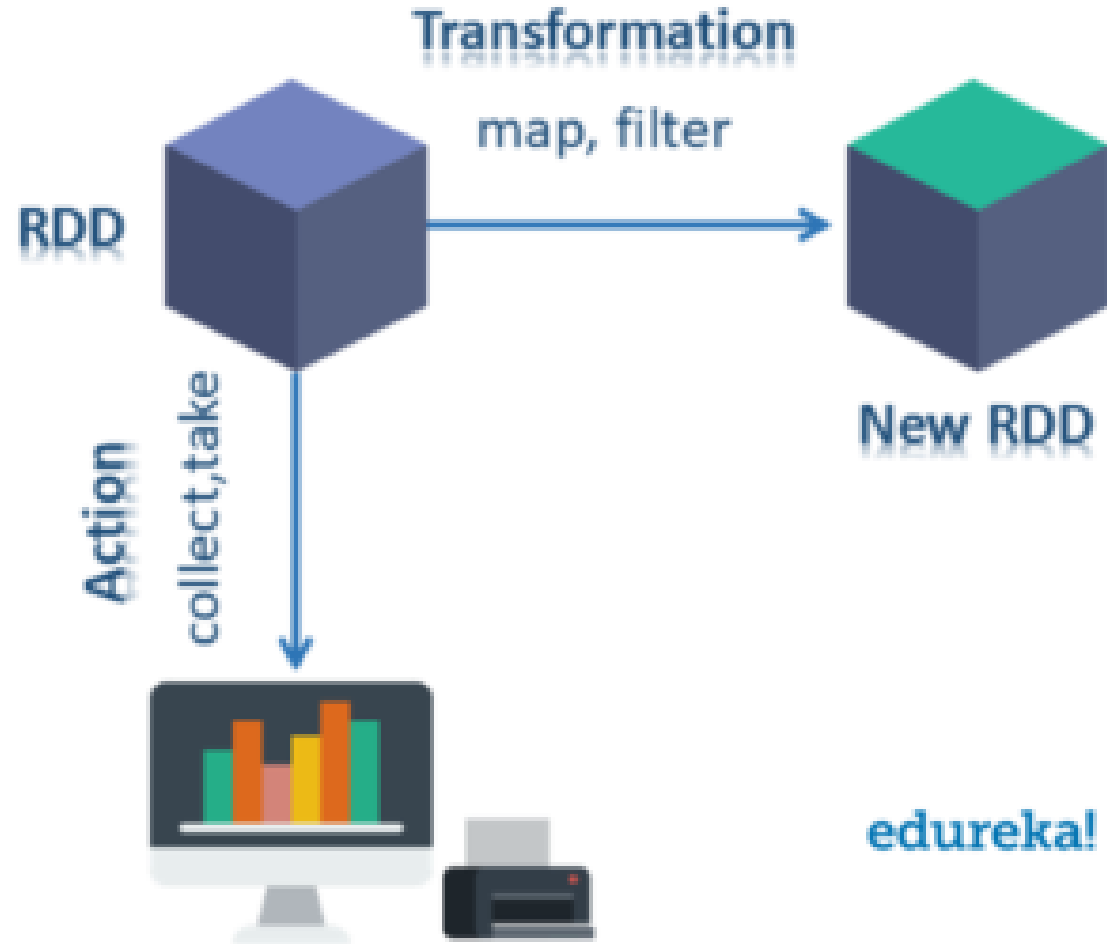
Distributed



Dataset



<https://www.edureka.co/blog/rdd-using-spark/>



Dataframe and Dataset

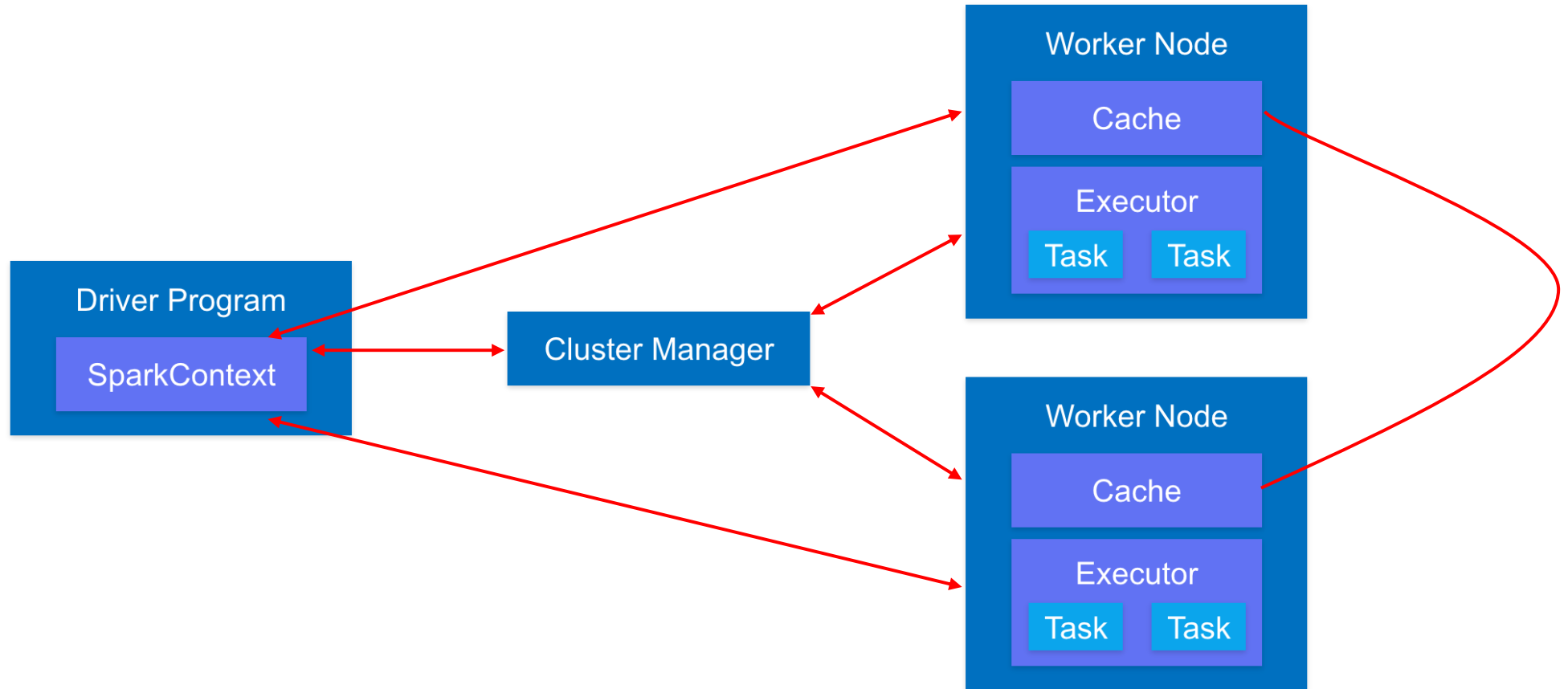


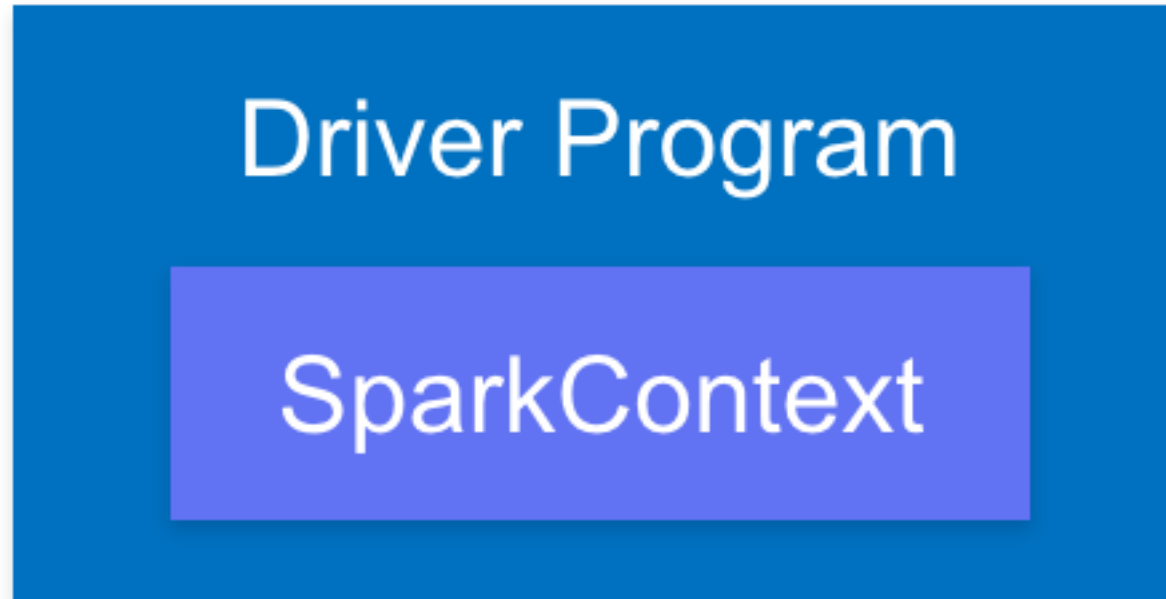
Abstraktion von RDD

Daten ähnlich zu RDBMS organisiert

Einfacher zu verwenden, aber weniger Kontrolle

Kombiniert Vorteile von RDD und DF

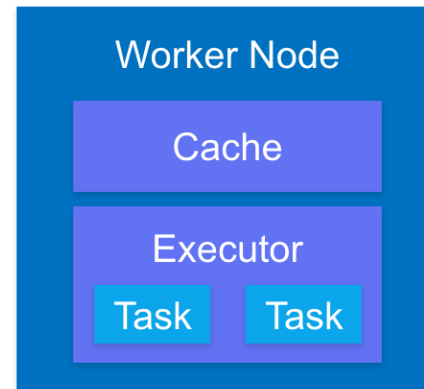
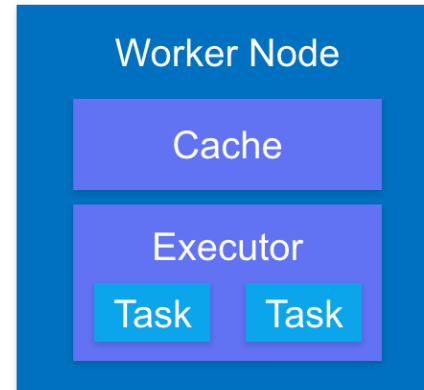




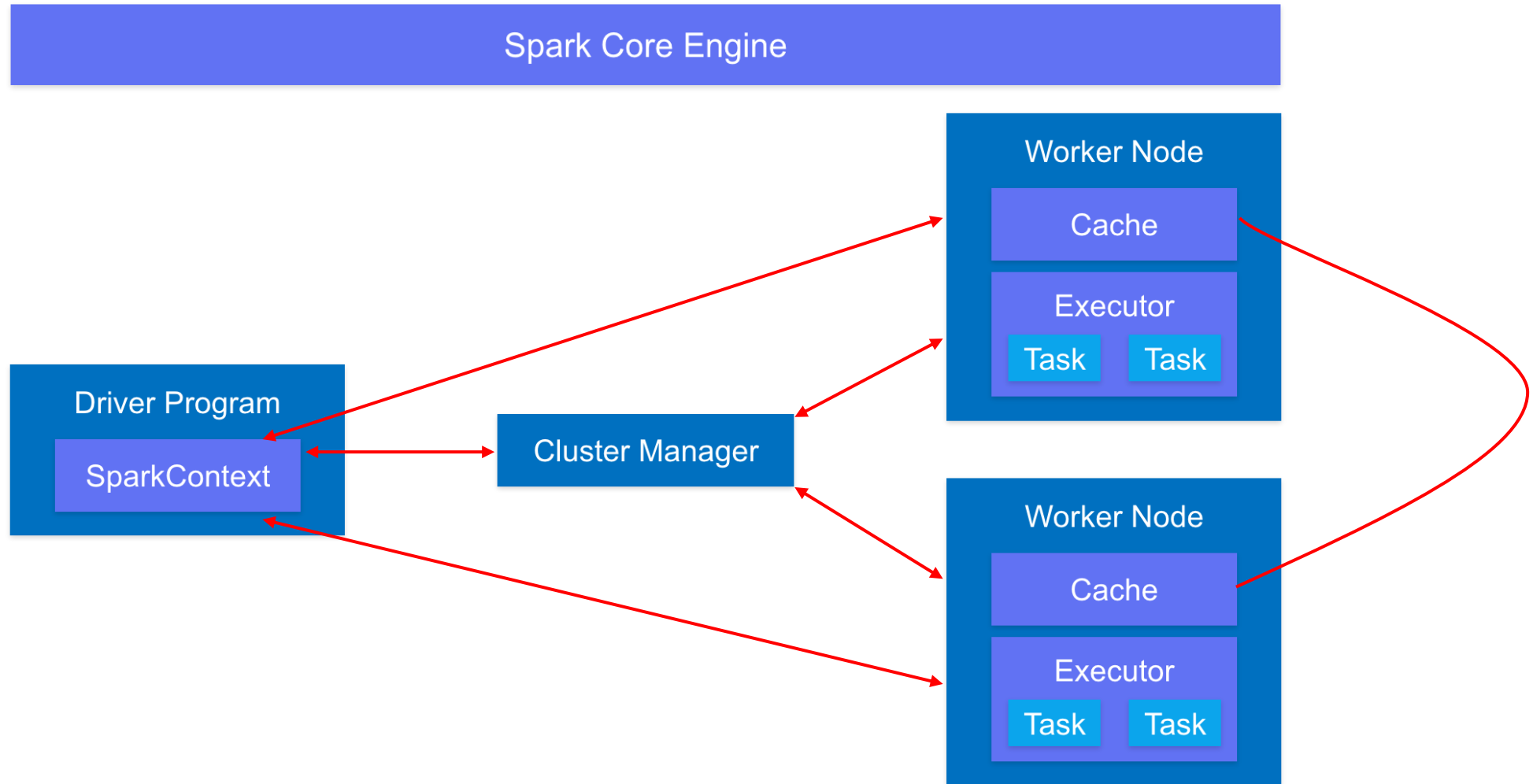
Architecture: Cluster Manager

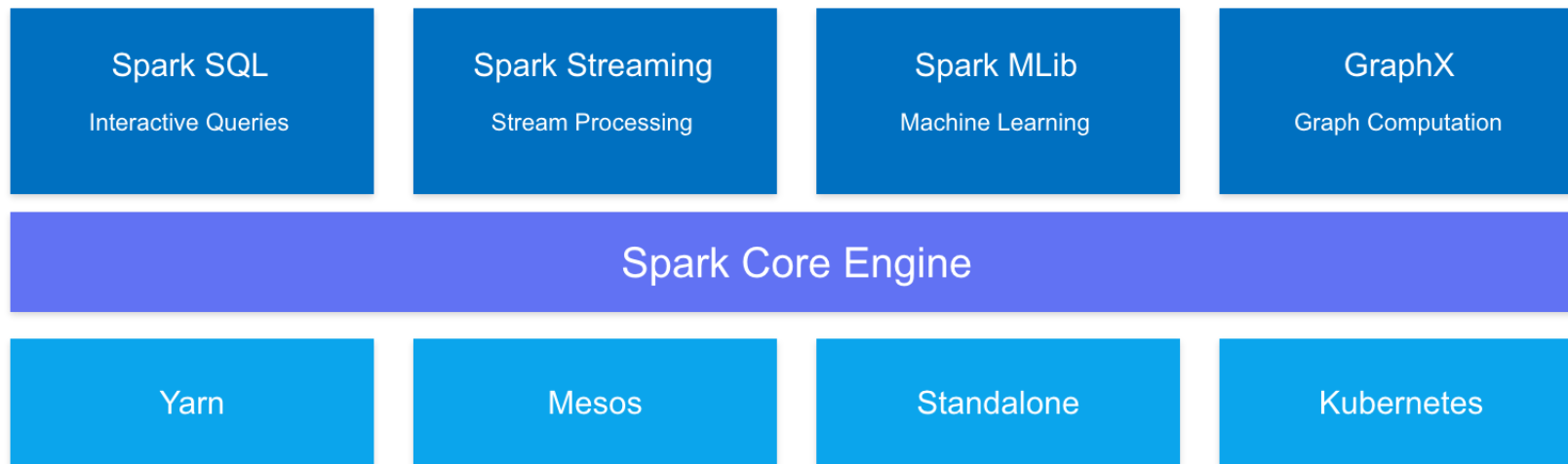
Cluster Manager

Architecture: Worker



Architecture



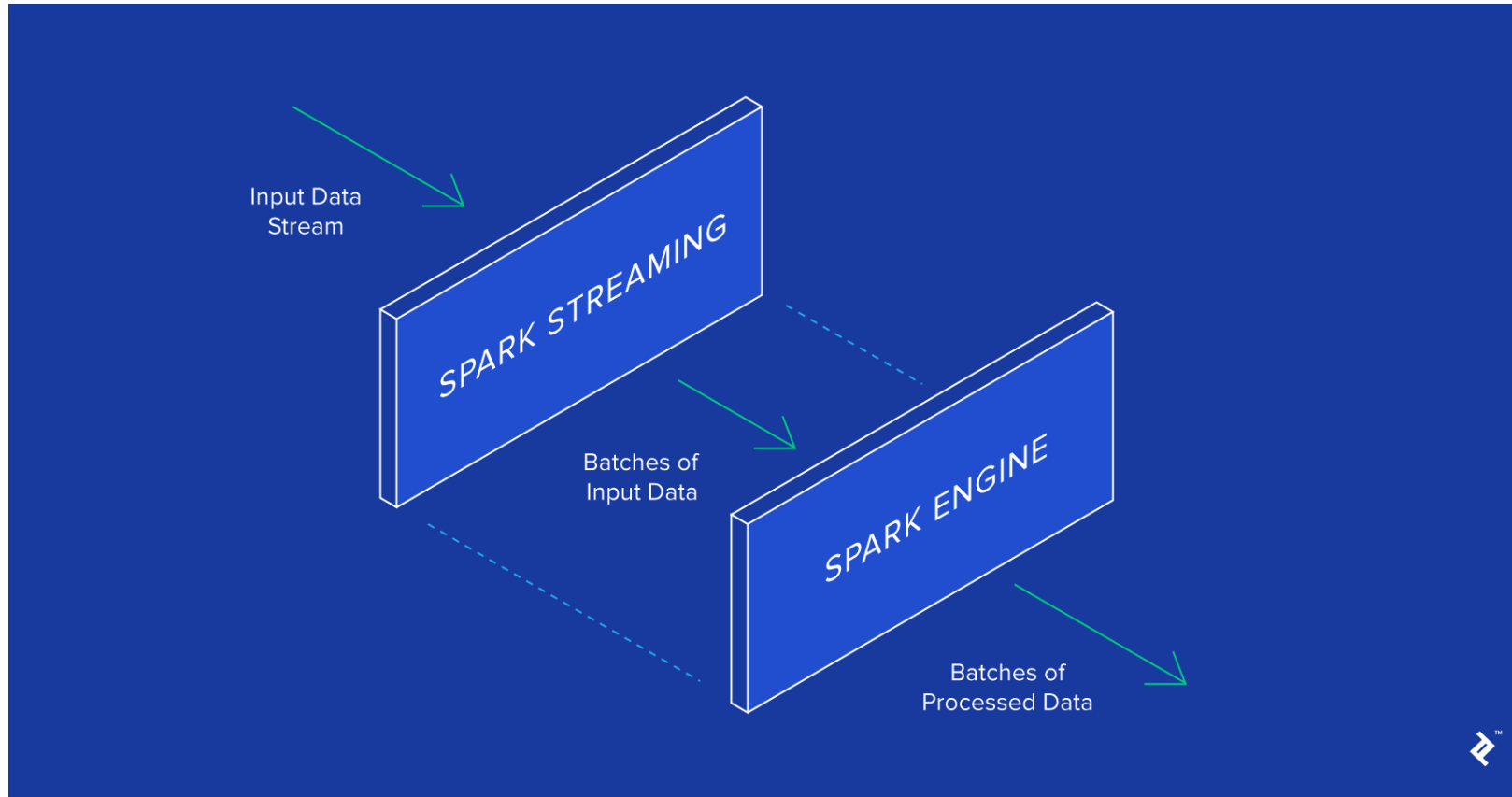


Ermöglicht die Verwendung von SQL-Befehlen wie Select, Where, Group By, Join, etc.

```
// DataFrame API where()
df.select("country", "city", "zipcode", "state")
  .where("state == 'AZ'")
  .show(5)
```

```
// sorting
df.select("country", "city", "zipcode", "state")
  .where("state in ('PR', 'AZ', 'FL')")
  .orderBy("state")
  .show(10)
```

Spark Streaming



Code: RDD in Python

```
>>> textFile = spark.read.text("README.md")
```

Code: RDD Actions in Python

```
>>> textFile.count() # Number of rows in this DataFrame
126

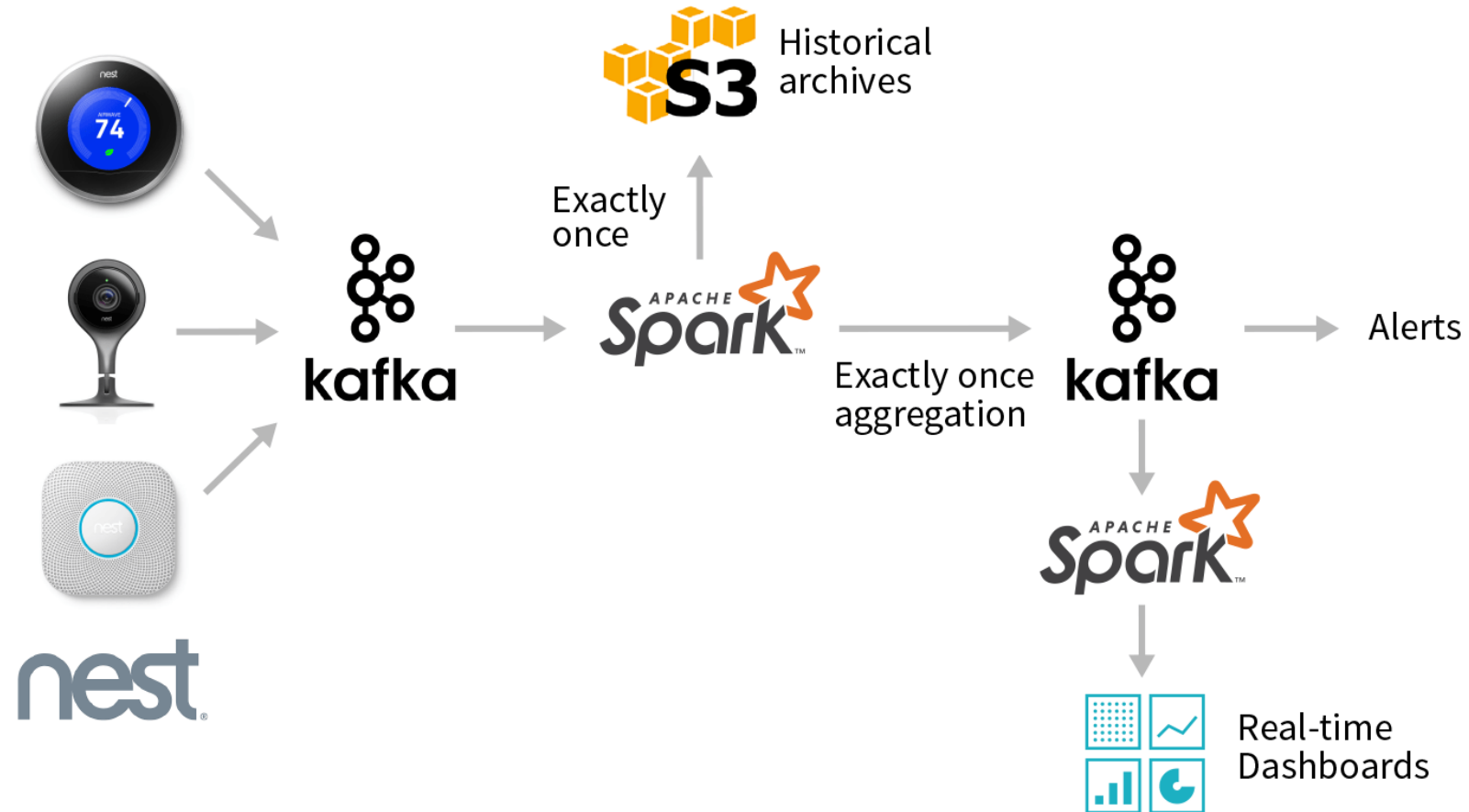
>>> textFile.first() # First row in this DataFrame
Row(value=u'# Apache Spark')
```

Code: RDD Transformation in Python

```
>>> linesWithSpark = textFile.filter(textFile.value.contains("Spark"))
```

```
>>> from pyspark.sql import functions as sf
>>> textFile.select(sf.size(sf.split(textFile.value,
"\s+")).name("numWords")).agg(sf.max(sf.col("numWords"))).collect()
[Row(max(numWords)=15)]
```

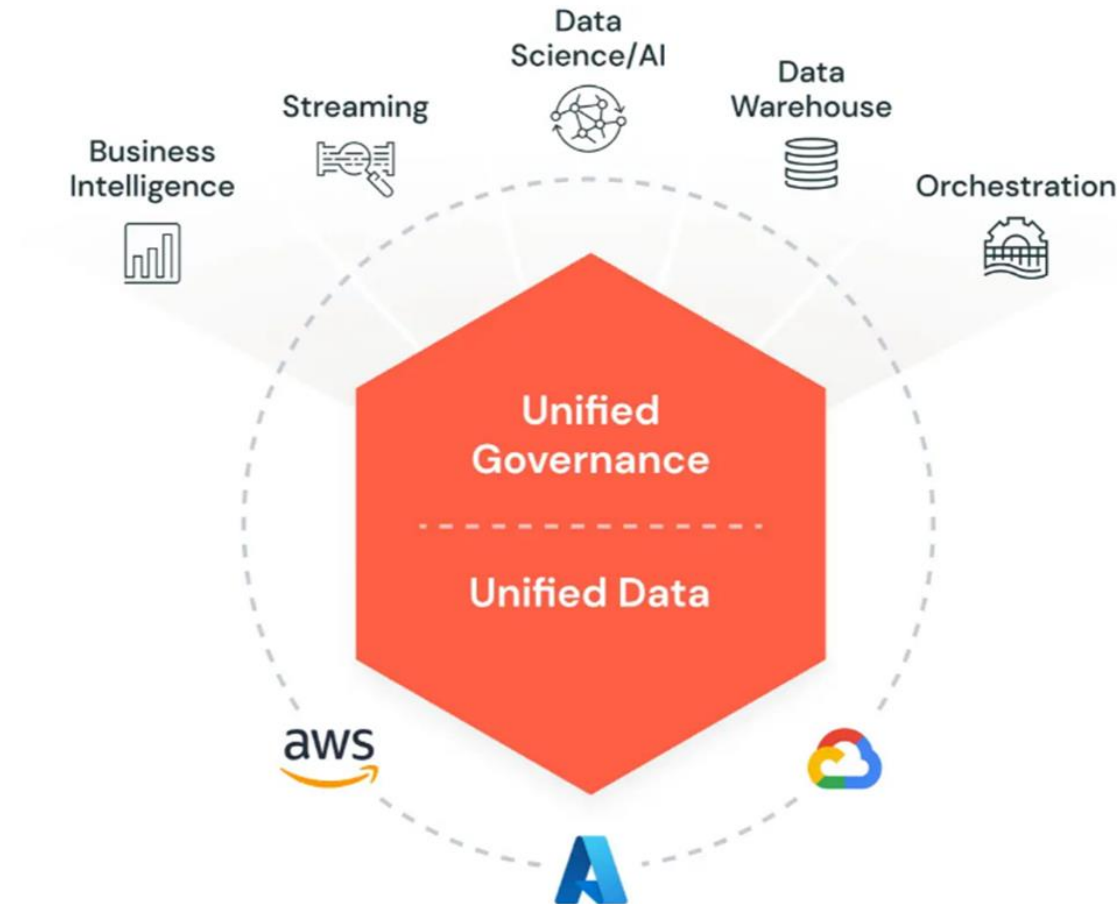
Integration von Apache Kafka und Apache Spark



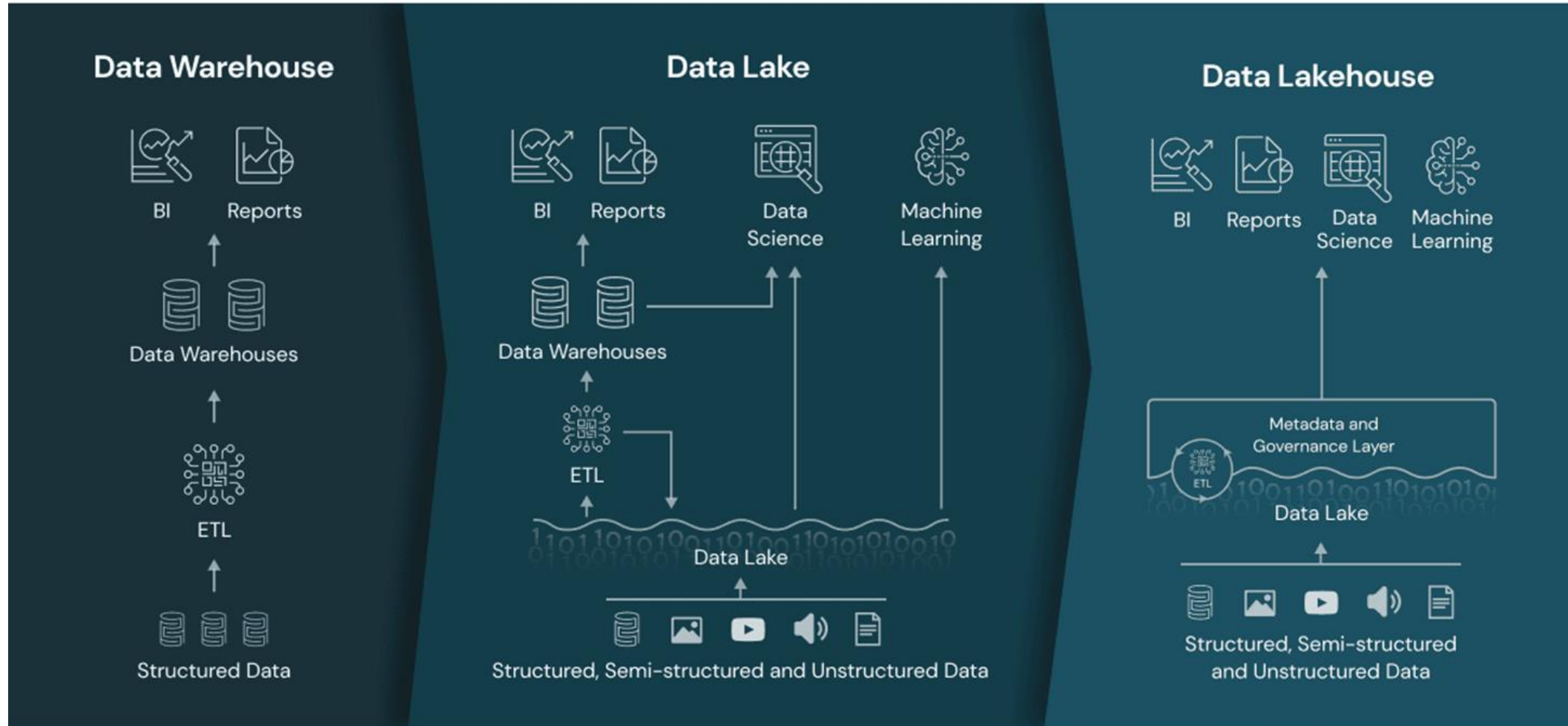


Databricks

Was ist Databricks?



Data Lakehouse Lösung



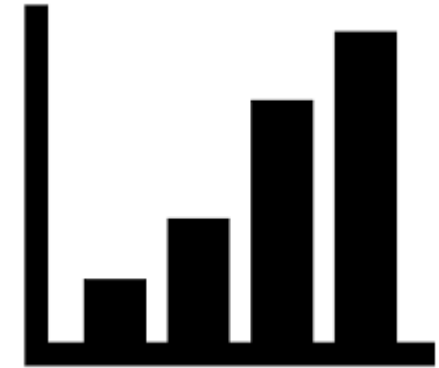
Argumente für Databricks



**Data Lakehouse
Lösung**



**Spezialisiert auf
Spark**



**Visuelle
Darstellung der
Daten**

S SQL

+ Create

SQL Editor

Workspace

Queries

Dashboards

Alerts

Search

Data

SQL Warehouses

Workflows

Query History

Partner Connect

1/3 Tasks Completed

Help

Settings

SQL



SQL query editor
Create a new query and explore your data in a SQL editor.

[Create a query](#)



Sample dashboards
Explore sample dashboards containing rich visualizations and queries.

[Visit gallery](#)



Sample data
Analyze a collection of pre-loaded data samples.

[Open the Data Explorer](#)



Partner Connect
[Fivetran, dbt Cloud](#)
[Tableau, Power BI](#)
[View all partners](#)

Recents

Name

Last viewed

Favorites

1. Registrieren bei Databricks

URL: <https://www.databricks.com/try-databricks>

Es ist wichtig, die Community-Edition bei der Registrierung auszuwählen!

2. Auswerten der Wikimedia-Daten

URL zum Databricks-Notebook: <https://github.com/Daumel/data-engineering>

