

Erarbeitungs-/Reflexionsphase – Entwicklung eines real-time Backends für eine datenintensive Applikation

Seminararbeit

Prüfungsleistung für den

Master of Science

des Studiengangs Data Science

an der Internationalen Hochschule

von

Jasper Bremenkamp

25. November 2024

1 Implementierung

Die Umsetzung der Dateninfrastruktur basiert auf einer modularen Microservice-Architektur, die für die Echtzeitverarbeitung großer Datenmengen entwickelt wurde. Die Architektur fokussiert sich auf Zuverlässigkeit, Skalierbarkeit und Wartbarkeit, während sie gleichzeitig datenschutzrechtliche Anforderungen erfüllt.

Eine Visualisierung ist in nachfolgender Abbildung 1 zu sehen.

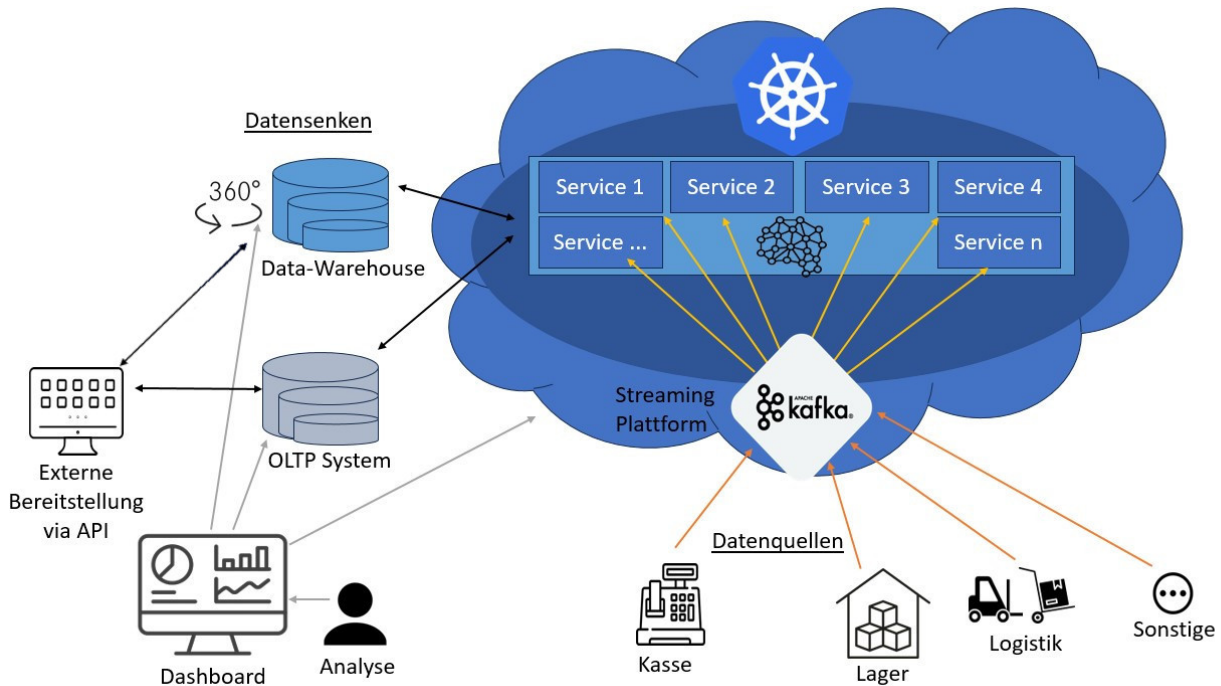


Abbildung 1: Architekturübersicht

Zur Datenaufnahme, Verarbeitung und Kommunikation zwischen den Komponenten werden ausschließlich Python-Microservices eingesetzt. Jeder Microservice ist so konzipiert, dass er eine klar definierte Aufgabe erfüllt und unabhängig von anderen Services agieren kann. Die Kommunikation erfolgt über Apache Kafka, das als Message-Broker verwendet wird. Die Daten werden in Kafka Topics geschrieben, von den zuständigen Microservices konsumiert und anschließend verarbeitet. Dabei wird keine zusätzliche Middleware wie Kafka Streams genutzt, sondern die gesamte Datenlogik ist direkt in den Python-Microservices implementiert.

Mehrere Microservices übernehmen die Aufgabe Datenströme zu simulieren, die Verkaufs- und Bestandsdaten repräsentieren. Im Produktivbetrieb würden diese Microservices echte Daten aus den Filialen und Lagerhäusern auf Edge Nodes verarbeiten und an Kafka übergeben. Diese Daten werden direkt an Kafka Topics publiziert. Die simulierten Daten dienen als Testbasis, um die Funktionalität der gesamten Pipeline unter realistischen Bedingungen zu validieren. Weitere spezialisierte Microservices konsumieren diese Daten und führen Transformationen durch, wie etwa Aggregationen und das Extrahieren relevanter Informationen. Die Aggregationen erfolgen in zeitlich definierten Intervallen, um Echtzeitberichte bereitzustellen.

Die stark konsolidierten und zu persistierenden Daten werden in einer PostgreSQL-Datenbank gespeichert, die für eine effiziente Speicherung und Abfrage großer Datenmengen optimiert ist. Die Übergabe der Daten an die

Datenbank erfolgt durch einen dedizierten Microservice, der die Daten aus Kafka konsumiert, verarbeitet und in die entsprechenden Tabellen einfügt. Diese Struktur ermöglicht eine klare Trennung der Verantwortlichkeiten zwischen Datenaufnahme, Verarbeitung und Speicherung.

Kubernetes wird zur Orchestrierung der Microservices genutzt. Die Bereitstellung und Konfiguration der Services erfolgt über Helm-Charts, die eine einfache Verwaltung und Skalierung der Infrastruktur gewährleisten. Dies ermöglicht es, die Infrastruktur flexibel an veränderte Anforderungen, wie etwa größere Datenvolumina, anzupassen.

Besonderer Wert wird auf Datensicherheit und Datenschutz gelegt. Die Kommunikation zwischen den Microservices wird im Produktivbetrieb verschlüsselt erfolgen, und rollenbasierte Zugriffskontrollen (RBAC) stellen sicher, dass nur autorisierte Entitäten Zugriff auf die Daten haben. Schützenswerte Daten, die nicht persistiert werden müssen, werden durch die Kafka Retention automatisch innerhalb kürzerer Zeit gelöscht und müssen nicht gesondert im Rahmen der DSGVO behandelt werden. Zudem werden alle Datenflüsse umfassend dokumentiert, um den Richtlinien der Datenschutz-Grundverordnung (DSGVO) zu entsprechen. Für die persistenten Daten und die Einhaltung der DSGVO gibt es speziell einen Microservice, der die Aufgabe der Einhaltung übernimmt.

Der Quellcode der Infrastruktur, der Applikationen inkl. der Dokumentation wird in einem Git-Repository versioniert, um die Nachvollziehbarkeit und Reproduzierbarkeit sicherzustellen. Das Repository enthält alle relevanten Konfigurationsdateien, Python-Skripte und Helm-Charts, die für den Aufbau der Infrastruktur notwendig sind.

Das Repository ist über folgenden Link zugänglich: https://github.com/jasper-bk/Projekt_DataEngineering_WiSe2024-2025.

Erste Tests zeigen, dass das System in der Lage ist, große Datenmengen effizient zu verarbeiten und in Echtzeit bereitzustellen. Die Architektur erfüllt die Anforderungen an Zuverlässigkeit, Skalierbarkeit und Wartbarkeit und stellt eine robuste Grundlage für zukünftige Erweiterungen dar. Das Optimierungspotenzial liegt insbesondere in der Ressourcennutzung des Kubernetes-Clusters, um die Performance zu verbessern und zukünftige Kosten zu senken.

Zur Einrichtung der Umgebung wird ein Kubernetes-Cluster, die Software Kubectl und Helm benötigt. Zur lokalen Einrichtung bietet sich die einfach einzurichtende Kubernetes Open-Source Lösung K3S von SUSE an. Sind Kubernetes-Cluster, Kubectl und Helm eingerichtet, lässt sich die gesamte Software mit zwei Befehlen aus dem Repository-Root als Arbeitsverzeichnis wie folgt ausrollen. Die Befehle zur Installation sehen wie folgt aus:

```
helm dependency update app/helm-chart-deployment /
helm install my-release app/helm-chart-deployment /
```

Der erste Befehl lädt die Abhängigkeiten runter und der zweite Befehl rollt die Applikationen auf das verbundene Cluster aus. Nach Ausführung ist die gesamte Infrastruktur verfügbar und einsatzbereit. Mit dem Kafka-Broker kann über Port 9094 von außen kommuniziert werden, um von außen Daten zu publizieren oder zu konsumieren. Außerdem können Kafka, dessen Topics und die Daten über Port 8080 mithilfe des Tools AKHQ im Browser inspiziert und überwacht werden. Damit lässt