
# 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**

18. November 2024

| | |
|---|---|
| **Kursbezeichnung** | Projekt: Data Engineering |
| **Matrikelnummer** | 92125193 |
| **Tutor** | Prof. Dr. Max Pumperla |

## 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.

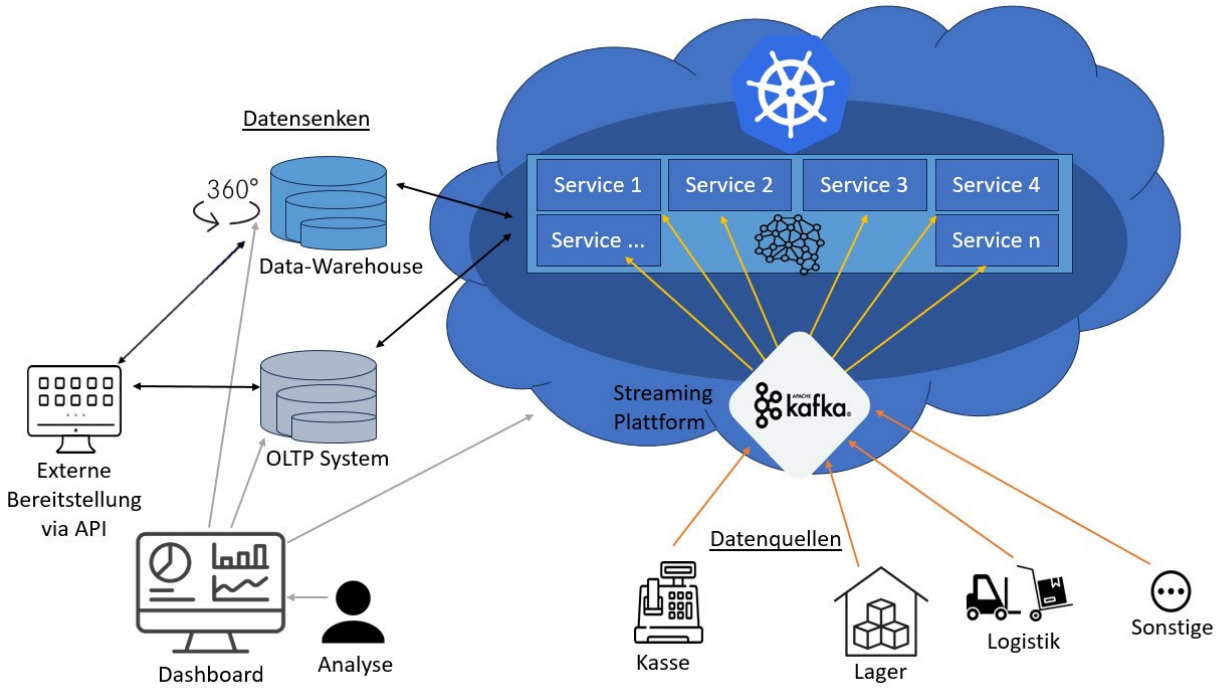


Abbildung 1: Architekturübersicht

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.

Ein zentraler Microservice übernimmt die Aufnahme von simulierten Datenströmen, die Verkaufs- und Bestandsdaten repräsentieren. Diese Daten werden direkt an Kafka Topics publiziert. Weitere spezialisierte Microservices konsumieren diese Daten und führen Vorverarbeitungen durch, wie etwa Aggregationen und das Extrahieren relevanter Informationen. Die Aggregationen erfolgen in zeitlich definierten Intervallen, um Echtzeitberichte bereitzustellen.

Die persistierten Daten werden in einer PostgreSQL-Datenbank gespeichert, die für eine effiziente Speicherung und Abfrage großer Datenmengen optimiert wurde. 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 erfolgt verschlüsselt, und rollenbasierte Zugriffskontrollen (RBAC) stellen sicher, dass nur autorisierte Entitäten Zugriff auf die Daten haben. Zudem werden alle Datenflüsse umfassend dokumentiert, um den Richtlinien der Datenschutz-Grundverordnung (DSGVO) zu entsprechen.

Die Entwicklung der Infrastruktur basiert auf synthetischen Datenströmen, die typische Anwendungsszenarien wie Verkaufs- und Lagerbestandsdaten simulieren. Diese Datenströme werden mithilfe von Python-Skripten generiert und kontinuierlich in Kafka Topics publiziert. Die simulierten Daten dienen als Testbasis, um die Funktionalität der gesamten Pipeline unter realistischen Bedingungen zu validieren.

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. Zusätzlich wird die Einhaltung bewährter Praktiken zur Versionskontrolle sichergestellt, um eine strukturierte Weiterentwicklung zu ermöglichen. 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 weiter zu verbessern.