Konzeption – 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

16. Oktober 2024

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

1 Szenario

Die Frischmarkt AG, eine Kette von Lebensmittelgeschäften, steht vor der Aufgabe, datengestützte Entscheidungsfindung in ihre Betriebsprozesse zu integrieren, um Effizienz zu steigern, Kosten zu senken und die Kundenzufriedenheit zu verbessern. Die derzeitige Infrastruktur, bestehend aus Kassensystemen, Lagerverwaltung und Kundendatenbanken, wird hauptsächlich zur Datenspeicherung genutzt, jedoch nicht für fortschrittliche Analysen oder Echtzeit-Entscheidungen.

Das Ziel des Projekts ist der Aufbau einer Dateninfrastruktur, die in der Lage ist, kontinuierlich große Mengen an Echtzeitdaten zu verarbeiten und automatisierte Prozesse zu ermöglichen. Ein Schwerpunkt liegt auf der automatisierten Bestandsverwaltung, dynamischen Preisgestaltung und personalisierten Angeboten. Automatisierte Nachbestellungen sollen auf Grundlage von Verkaufsdaten, Lagerbeständen und saisonalen Schwankungen ausgelöst werden.

2 Strategie

Um diese Ziele zu erreichen, verfolgt die Frischmarkt AG eine mehrstufige Strategie zur Entwicklung einer effizienten Datenverarbeitungsarchitektur.

Ein zentrales Prinzip der Strategie ist der Aufbau einer modularen und skalierbaren Architektur auf Basis einer Microservice-Architektur. Kubernetes und Helm werden als Werkzeuge für Infrastructure as Code (IaC) verwendet, um die Skalierbarkeit und Wartbarkeit des Systems zu gewährleisten. Kubernetes orchestriert die Microservices, um ihre Skalierung und Fehlertoleranz zu gewährleisten.

Für die Containerisierung wird Podman anstelle der mehr bekannten Docker Container Engine verwendet, da Podman keinerlei proprietäre Einschränkungen wie Docker hat und außerdem einige Vorteile gegenüber Docker bietet, wie z. B. Rootless Containers, welche die Sicherheit erhöhen. Podman kann problemlos im Zusammenhang mit Kubernetes-Umgebungen genutzt werden und unterstützt ebenfalls die wesentlichen Features, die Docker bietet.

Die Architektur teilt das System in unabhängige Komponenten auf, z.B. Datenaufnahme und -verarbeitung, die von Apache Kafka als Nachrichten-Streaming-Plattform übernommen wird. Diese Modularität erleichtert die Skalierbarkeit des Systems, da einzelne Services unabhängig skaliert oder aktualisiert werden können.

Ein zentraler Aspekt des Systems ist die automatisierte Bestands- und Verkaufserfassung. Verkaufs- und Lagerbestandsdaten werden kontinuierlich über Apache Kafka gesammelt und in Echtzeit verarbeitet.

Das Sammeln der Daten wird in den Filialen oder anderen Einrichtungen des Unternehmens von unabhängigen kleinen Rechnern übernommen, um möglichst flexibel mit den Gegebenheiten vor Ort umgehen zu können. Ohne zusätzlichen Computer ist das direkte Senden der Daten aus der Kasse direkt zu Kafka technisch schwer realisierbar, da es keine einfache Möglichkeit gibt containerisierte Microservices darauf auszuführen. Daher werden die Microservices auf seperaten Rechnern ausgeführt und es werden die Schnittstellen zu den Systemen vor Ort, wie u. a. die Kassensysteme genutzt. So lassen sich für unterschiedliche Betriebsstätten auch verschiedene Microservices einsetzen, die ihre vorberarbeiteten Daten an eine zentrale Stelle senden. Da die Datenvorprozessierung und Aggregation bereits außerhalb des eigentlichen Rechenzentrums stattfindet handelt es sich bei dem Konzept u. a. auch um Edge Computing.

Während der Entwicklungsphase werden Fakes verwendet, welche diese Rechner dann nachstellen, um Datenströme zu simulieren und die automatisierte Bestandsverwaltung zu testen, bevor das System mit realen Daten verknüpft wird. Die zusendenden Daten sollen auf eine Minute zusammengefasst sein. Die Einführung dieser Verwaltung würde nach Abschluss schrittweise in Pilotfilialen erfolgen, um die Effizienz und Anpassungsfähigkeit an regionale und saisonale Unterschiede zu testen. Kubernetes ermöglicht die flexible Skalierung der Infrastruktur, um die Datenanforderungen zu bewältigen. Im Rahmen der Entwicklung wird das Kernsystem nur auf ein Single-Node Cluster ausgerollt und die Edge-Computer Microservices in einem eigenen Namespace. Später im Produktivbetrieb lässt sich dann ohne zusätzlichen Aufwand auf ein Multi-Node Cluster oder ein High Availability (HA) Cluster bestehend aus mehreren anderen Clustern umsteigen und die Edge-Computer Microservices lassen sich auf kleinere Rechner vor Ort ausrollen.

Die datengestützte Entscheidungsfindung wird durch Echtzeit-Datenanalysen ermöglicht, die es dem Management erlauben, auf aktuelle Entwicklungen in den Filialen zu reagieren. Mithilfe von Machine Learning-Algorithmen werden Verkaufsprognosen erstellt, die u. a. dynamische Preisanpassungen und personalisierte Angebote unterstützen. Diese Algorithmen werden kontinuierlich mit aktuellen Verkaufs- und Lagerdaten sowie weiteren relevanten Daten trainiert, die über Kafka in das System eingespeist werden. In der Entwicklungsphase wird dies mit simulierten Daten getestet.

Obwohl Machine Learning eine zentrale Rolle in der zukünftigen Entwicklung spielt, liegt der Fokus dieses Projekts auf der Implementierung der zugrunde liegenden Dateninfrastruktur. Die konkrete Implementierung der Machine Learning Algorithmen wird nicht abgehandelt.

Die nahtlose Integration bestehender Systeme ist ein weiterer strategischer Fokus. Die Kassensysteme und CRM-Systeme der Frischmarkt AG sollen in die neue Dateninfrastruktur eingebunden werden, wobei Apache Kafka als zentrale Plattform für die Datenaufnahme fungiert.

Da der Zugang zu produktiven Datenquellen in der Entwicklungsphase noch nicht gegeben ist, werden wie bereits erläutert Fakes verwendet, um Kassendaten, Lagerbestände und andere Daten zu simulieren. Diese Fakes publizieren vorprozessierte Daten an Kafka Topics, um den realen Betrieb zu simulieren und die Datenpipeline unter realistischen Bedingungen zu testen. Die Fakes übernehmen damit auch die Aufgabe der Vorprozessierung, die im Produktivbetrieb von dem Edge-Computer und dessen Microservice(s) gemacht werden würden. Anschließend werden die publizierten Daten aus den Topics von verschiedenen Microservices konsumiert und für die verschiedenen Anwendungsfälle verarbeitet und anschließend in den Datensenken bereitgestellt.

Die Ziel-Architektur ist in nachfolgender Abbildung 1 vereinfacht dargestellt.

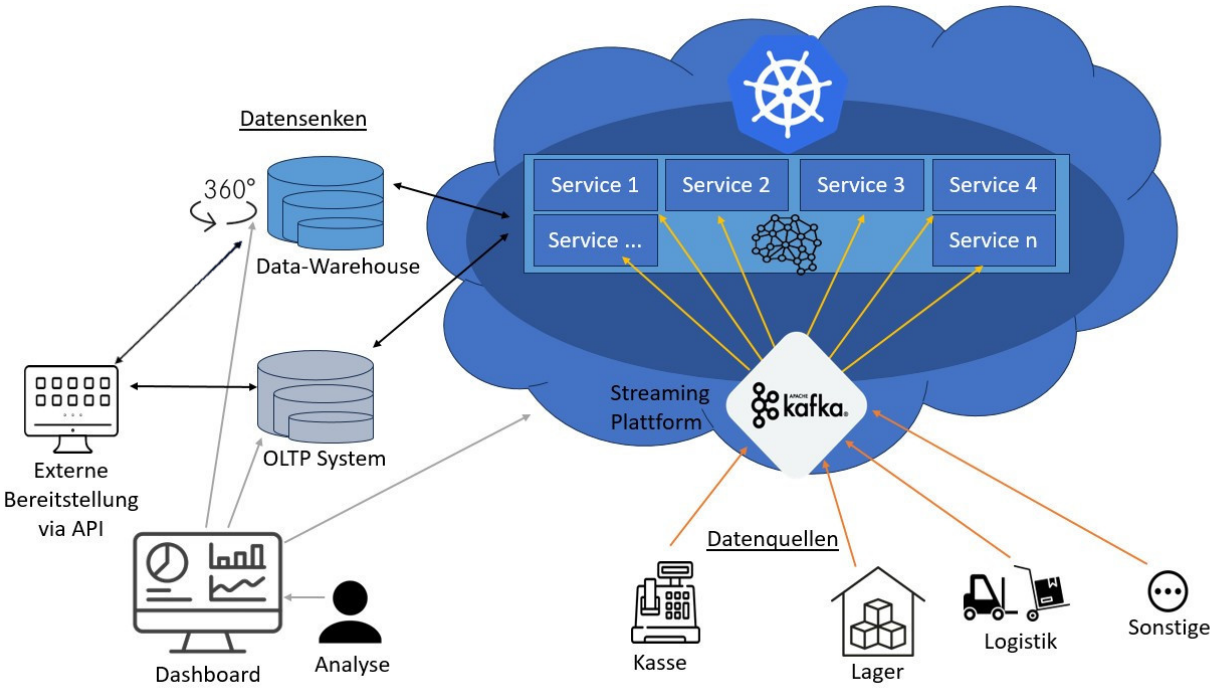

Abbildung 1: Architekturübersicht

Die Exklusion der API und der Datensenken aus dem Kubernetes Cluster in der Abbildung 1 ist lediglich implizit. Das bedeutet, dass sie nicht unbedingt im Kubernetes-Cluster betrieben werden müssen, da sie keine unbedingt skalierbaren Microservices darstellen. Im Rahmen der Entwicklung werden sie jedoch ebenfalls im Cluster bereitgestellt.

Apache Kafka und andere Softwarebestandteile, die keine Eigenentwicklung sind, werden über Helm Packages installiert. Darin sind ebenfalls auch die „richtigen Docker-Images“ hinterlegt. Ein gut geeignetes Helm Package für Kafka ist z. B. folgendes: `https://artifacthub.io/packages/helm/bitnami/kafka`.

Die Entscheidung für diese Microservice-Architektur bietet mehrere wesentliche Vorteile:

- **Skalierbarkeit durch Microservices**: Die Nutzung von Kubernetes ermöglicht eine flexible und effiziente Skalierung der einzelnen Services.
- **Flexibilität durch Kafka**: Kafka entkoppelt Datenquellen und -senken, was eine flexible Integration neuer Datenquellen ermöglicht.
- **Echtzeit-Datenverarbeitung**: Kafka bietet nahezu sofortige Datenweitergabe, was in Echtzeit Entscheidungen ermöglicht.
- **Zentralisierte Datenverarbeitung**: OLTP-Systeme und Data-Warehouses ermöglichen parallele Nutzung für operative und analytische Zwecke.
- **Modularität und Wartbarkeit**: Jeder Dienst kann unabhängig entwickelt und gewartet werden, was die Agilität erhöht.

Es gibt jedoch auch einige Nachteile, die mit dieser Architektur verbunden sind:

- **Komplexität der Integration**: Die Integration von Kafka, Kubernetes und Microservices erfordert erhebliche technische Expertise.
- **Erhöhter Wartungsaufwand**: Die verteilte Architektur erfordert eine kontinuierliche Überwachung und Pflege.
- **Datenkonsistenz**: In einer verteilten, asynchronen Umgebung besteht ein Risiko für inkonsistente Daten zwischen verschiedenen Systemen.

Datensicherheit und Datenschutz sind ebenfalls zentrale Bestandteile. Erfasste Kundendaten werden gemäß der Datenschutz-Grundverordnung (DSGVO) verarbeitet. Die Speicherung und Verarbeitung der Daten erfolgt verschlüsselt, und es werden strenge Zugangskontrollen implementiert.

Um die Einhaltung der Datenschutzanforderungen sicherzustellen, werden Sicherheitsmechanismen wie verschlüsselte Datenübertragungen und in Zukunft regelmäßige Sicherheitsaudits in die Infrastruktur integriert. Ein Data Governance-Framework regelt den Umgang mit den Daten und gewährleistet Transparenz.

Die Zusammenarbeit zwischen den internen IT-Teams der Frischmarkt AG und externen Experten ist entscheidend für den Erfolg des Projekts. Git in Verbindung mit GitHub als Plattform wird für die Versionskontrolle genutzt, um eine klare Rückverfolgbarkeit der Änderungen und eine kollaborative Entwicklung in verteilten Teams zu ermöglichen.

Die Strategie zur Entwicklung der Dateninfrastruktur für die Frischmarkt AG basiert auf einer modularen, skalierbaren Architektur, die Kubernetes, Helm und Podman verwendet, um die Skalierbarkeit, Wartbarkeit und Verfügbarkeit des Systems sicherzustellen. Apache Kafka wird als zentrale Plattform für die Datenaufnahme und -verarbeitung eingesetzt. Simulierte Datenquellen werden genutzt, um die Datenpipeline während der Entwicklungsphase zu testen. Datensicherheit und Datenschutz haben höchste Priorität, um den gesetzlichen Anforderungen gerecht zu werden.