Entwicklung einer skalierbaren Dateninfrastruktur für die Batch-Verarbeitung

Im Rahmen des Projekts, welches im Kontext des Kurses Data Engineering erarbeitet wurde, wurde eine robuste und skalierbare Dateninfrastruktur zur Batch-Verarbeitung von über einer Million in Deutschland erfasster Temperaturdaten entwickelt. Alle benötigten Komponenten wurden in einer containerisierten Umgebung mit Docker und Docker Compose bereitgestellt, was eine einfache Isolation, Portabilität und Wartung der Architektur ermöglicht. Die technische Umsetzung erfüllt alle Anforderungen hinsichtlich Zuverlässigkeit, Skalierbarkeit und Wartbarkeit. Der entwickelte Code wurde in einem Git-Repository versioniert, was eine lückenlose Nachverfolgung von Änderungen gewährleistet. Eine ausführliche Dokumentation, inklusive einer Installationsanleitung, befindet sich in der README-Datei des Repositories.

Der ***Data Ingestion Service***, welcher auf Apache Kafka basiert, ermöglicht eine zuverlässige und effiziente Aufnahme großer Datenmengen in Form von Batches. Die Eingangsdaten liegen im Verzeichnis „/input“ und werden durch den Kafka Producer, der in der Datei „kafka_producer.py“ implementiert ist, eingelesen und vorverarbeitet. Diese Daten werden anschließend in Batches von maximal 1 MB in die Kafka Topic „Temperature“ geschrieben. Danach verarbeitet der ***Data Processing Service***, mithilfe eines Kafka Consumers zum Auslesen der Kafka Topic und einem Apache Spark Service die Daten, um aggregierte Metriken wie bspw. Durchschnittstemperatur und Modus zu berechnen. Dabei sorgt die programmierte PySpark-Schnittstelle in der Datei „temperature_processor.py“ dafür, dass der Offset des Kafka-Consumers korrekt eingestellt wird, um sicherzustellen, dass bereits verarbeitete Daten nicht erneut eingelesen oder Datenbatches übersprungen werden. Der Spark-Service prüft in Intervallen von 10 Sekunden auf neue Daten im Kafka-Topic „Temperature“, um eine zeitnahe Verarbeitung sicherzustellen. Nach Abschluss der Verarbeitung eines Batches werden die Daten im ***Data Persistance Service*** mittels eines Hadoop Distributed File System (HDFS) gespeichert, welches durch die Möglichkeit des Einsatzes multipler DataNodes eine hohe Verfügbarkeit gewährleistet und gleichzeitig skalierbar ist. Zusätzlich werden nach der Verarbeitung Visualisierungen der aggregierten Daten in Form von Graphen erzeugt und in einer Ordnerstruktur im Verzeichnis „/output“ abgelegt. Die Strukturierung erfolgt dabei nach Wetterstationsnummer und Jahr, um die Nachvollziehbarkeit und Analyse zu vereinfachen.

Während der Implementierung traten zwei Hauptprobleme auf: Erstens, die Synchronisation der Verarbeitung zwischen den Microservices. Zweitens, Stabilitätsprobleme in der aktuellen Windows-Version von Docker Desktop, insbesondere in Verbindung mit sicherheitsrelevanten Best Practices, die die Nutzung administrativer Benutzerkonten einschränken. Die erste Problematik konnte durch eine sorgfältige Konfiguration der Docker-Compose-Datei und eine korrekte Implementierung der Microservices gelöst werden. Die zweite Problematik konnte jedoch vorerst nur durch einen Workaround umgangen werden. Docker Desktop muss im administrativen Kontext installiert werden, um die notwendigen Konfigurationen durchzuführen, Dienste zu starten und das benötigte Linux-Subsystem korrekt zu installieren. Nach einem Neustart des Clientsystems konnte allerdings die WSL

(Windows Subsystem for Linux) -Schnittstelle von Docker nicht mehr angesprochen werden, was darauf zurückzuführen ist, dass bei Login eines nicht administrativen Nutzers nicht alle benötigten Dienste für den Start des Linux Subsystems mit gestartet werden können. Ähnliche Probleme mit WSL inklusive derselben Symptomatik wurden von anderen Usern auf Stack Overflow berichtet[1]. Als kurzfristiger Workaround wurde sich dafür entschieden, Docker Desktop nach jedem Neustart des Systems zu reinstallieren. Für eine langfristige Lösung ist jedoch eine tiefergehende Ursachenforschung anzuraten. Ebenfalls sollte geprüft werden, ob eine neuere Version von Docker Desktop das Problem nachhaltig beheben kann. Ansonsten ist anzudenken auf ein Linux/UNIX-System umzusteigen. Diese Erfahrungen verdeutlichen, dass eine umfassendere Prüfung der Systemvoraussetzungen und der Einsatz von Docker Compose zur Automatisierung in zukünftigen Projekten sinnvoll sind, um die Effizienz und Stabilität der Infrastruktur zu steigern.

Ergänzend zu den bestehenden Datensicherheitsmaßnahmen, wie regelmäßigen Backups und die Implementation von Zugriffsrechten, könnten in zukünftigen Projekten weitergehende Maßnahmen wie die Integration von Datenverschlüsselung und eine detailliertere Überwachung der Data Governance implementiert werden. Rückblickend führten die gründliche Planung der Architektur, iterative Entwicklung und kontinuierlichen Tests zum Erfolg des Projekts. Zu den wichtigsten technischen Fähigkeiten, die in diesem Projekt erlernt wurden, zählen die Beherrschung von Docker und containerisierten Microservices, der effiziente Einsatz von Apache Kafka, Spark und HDFS sowie die Implementierung von Datenschutz, Datensicherheit und Data Governance-Prinzipien. Zudem wurden Soft Skills wie Problemlösungsfähigkeiten, Selbstorganisation und effektives Projektmanagement weiterentwickelt.

Eine mögliche Erweiterung des aktuellen Systems stellt die Integration einer zweiten Daten-Pipeline dar, die eine Echtzeitverarbeitung der Wetterdaten von deutschen Wetterstationen ermöglicht. Dies könnte durch den Einsatz von Stream-Prozessierungstechnologien wie Apache Kafka Streams oder Apache Spark Streaming erreicht werden. Dafür müssten eine oder mehrere API-Schnittstellen von Data Providern an den Service angebunden werden, um Daten nahe Echtzeit zu erhalten. Diese können von dem anschließenden Machine Learning Service verwendet werden, um kurzfristige Prognosen abzuleiten, die mittels einer Trendanalyse aus der Batchprozessierung der vergangenen Jahre untermauert werden kann. Dies würde die Flexibilität und Leistungsfähigkeit des Systems weiter steigern.

Zusammenfassend zeigt das Projekt, dass eine umfassende Microservice-Architektur auf Basis moderner Technologien wie Apache Kafka, Apache Spark und HDFS auch bei komplexen Anforderungen eine robuste, skalierbare und zukunftssichere Lösung bieten kann. Die gewonnenen Erkenntnisse und identifizierten Verbesserungspotenziale werden in zukünftige Projekte einfließen, um die Effizienz und Qualität weiter zu steigern.

[1]https://stackoverflow.com/questions/71238673/docker-desktop-starting-forever-on-windows, abgerufen am 14.08.2024