Konzeptvorstellung der Dateninfrastruktur

Ziel dieses Projekts ist die Entwicklung einer skalierbaren und wartbaren Dateninfrastruktur zur Batch-Prozessierung einer großen Anzahl von Temperaturdaten (>1.000.000), welche in Deutschland ermittelt wurden (Kleine, o.D.).

Zur Umsetzung werden drei Microservices implementiert: der ***Data Ingestion Service*** auf Basis von Apache Kafka (The Apache Software Foundation, 2024), der ***Data Processing Service*** auf Basis von Apache Spark (The Apache Software Foundation, 2018), und der ***Data Persistence Service*** welcher auf HDFS, dem Hadoop Distributed File System (The Apache Software Foundation, *Apache Hadoop: HDFS Architecture*, 2023) basiert. Für jeden einzelnen Microservice wird, durch die Nutzung von Docker, die Isolation und Portabilität gewährleistet. Die Implementierung soll mittels Docker Compose erfolgen.

Der Prozess beginnt mit der Datenaufnahme durch einen Kafka-Microservice. Kafka ermöglicht die zuverlässige und skalierbare Aufnahme großer Datenmengen in Echtzeit und dient als Puffer, um eingehende Datenströme zu speichern und zu verwalten, bis sie in Batches zur weiteren Verarbeitung bereit sind (Amazon Web Services, Inc., o.D.).

Nach der Ingestion werden die Daten an einen Spark-Microservice zur Batch-Verarbeitung und Aggregation weitergeleitet. Spark bietet sich an, da dieser effizient große Datenmengen aufgrund seiner In-Memory-Computing Funktion verarbeiten kann (IONOS Redaktion, 2023). Der Spark-Service soll über die Python Schnittstelle „PySpark“ (The Apache Software Foundation, *PySpark Overview,* 2024) folgende aggregierte Werte berechnen:

- **Durchschnittstemperatur pro Monat**: Mittelwert der Temperaturwerte.
- **Temperatur-Schwankungsbreite pro Monat**: Differenz zwischen der höchsten und niedrigsten Temperatur.
- **Mediantemperatur pro Monat**: Mittlerer Wert der Temperaturverteilung, um Ausreißer zu minimieren.
- **Modus der Temperaturen pro Monat**: Häufigster Temperaturwert.

Diese Aggregationen sind nützlich, da sie mit Hilfe von Machine Learning komplexe Muster und Anomalien in den Temperaturdaten offenbaren können.

Die verarbeiteten Daten werden schließlich in einem HDFS gespeichert. HDFS bietet eine skalierbare und zuverlässige Möglichkeit zur Speicherung großer Datenmengen und stellt sicher, dass die Daten für die spätere Nutzung durch die Machine Learning Applikation zur

Verfügung stehen (The Apache Software Foundation, *Apache Hadoop: HDFS Architecture*, 2023).

Die Architektur ist so konzipiert, dass sie durch den Einsatz containerisierter Microservices (Kafka, Spark, HDFS) skalierbar, zuverlässig und wartbar ist. Datensicherheit, Datenschutz und Data Governance werden durch entsprechende Zugriffrechte (The Apache Software Foundation, *HDFS Permissions Guide*, 2023) und regelmäßige Backups gewährleistet. Eine Versionskontrolle wird durch GitHub realisiert, während die Reproduzierbarkeit durch Infrastructure as Code (IaC) zur Automatisierung der Bereitstellung und Konfiguration der Infrastruktur ermöglicht wird. Dies stellt sicher, dass das System einerseits nach Ausfällen schnell wiederhergestellt und andererseits kontinuierlich weiterentwickelt werden kann, ohne zu Unterbrechungen im Regelbetrieb zu führen.

Dennoch gibt es auch einige Nachteile. Die Implementierung und Verwaltung der konzipierten Microservice-Architektur erfordert ein hohes Maß an Fachwissen und kann somit komplexer sein als ein monolithischer Ansatz. Dies kann den initialen Aufwand erheblich erhöhen und erfordert eine sorgfältige Planung und Umsetzung. Darüber hinaus können Netzwerk-Latenzen und die Notwendigkeit zur Synchronisierung zwischen den Microservices potenzielle Herausforderungen darstellen. Schließlich ist die Sicherstellung von Datensicherheit und Governance in einer verteilten Architektur aufwändiger, da sie eine gründliche Implementierung von Zugriffskontrollen und Verschlüsselung erfordert.

Zusammenfassend bietet die konzipierte Microservice-Architektur eine skalierbare, zuverlässige und wartbare Lösung zur Batch-Prozessierung von Temperaturdaten in Deutschland. Durch den Einsatz von Apache Kafka, Apache Spark und HDFS sowie die Nutzung von Docker wird eine flexible und effiziente Infrastruktur geschaffen. Trotz der höheren Komplexität und des initialen Aufwands im Vergleich zu monolithischen Ansätzen, überwiegen die Vorteile hinsichtlich Skalierbarkeit, Datensicherheit und kontinuierlicher Weiterentwicklung, was diese Architektur zukunftssicher und robust macht.

Literaturverzeichnis

Amazon Web Services, Inc. (o.D.). *Was ist Kafka?* Amazon Web Services, Inc.. Abgerufen am 24. Juli 2024, von https://aws.amazon.com/de/what-is/apache-kafka/

IONOS Redaktion. (21. Juni 2023). *Apache Spark: Definition und Funktionen*. IONOS SE. Abgerufen am 24. Juli 2024, von https://www.ionos.de/digitalguide/server/knowhow/was-ist-apache-spark/

Kleine, M. (o.D.). *German temperature data 1990-2021: Official german temperature data from 513 weather stations from 1990 to 2021*. Kaggle. Abgerufen am 23. Juli 2024, von https://www.kaggle.com/datasets/matthiaskleine/german-temperature-data-1990-2021/data

The Apache Software Foundation. (2018). *Unified engine for large-scale data analytics*. The Apache Software Foundation. Abgerufen am 23. Juli 2024, von https://spark.apache.org/

The Apache Software Foundation. (18. Juni 2023). *Apache Hadoop: HDFS Architecture*. The Apache Software Foundation. Abgerufen am 23. Juli 2024, von https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

The Apache Software Foundation. (18. Juni 2023). *HDFS Permissions Guide*. The Apache Software Foundation. Abgerufen am 24. Juli 2024, von https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsPermissionsGuide.html

The Apache Software Foundation. (2024). *Apache Kafka*. The Apache Software Foundation. Abgerufen am 23. Juli 2024, von https://kafka.apache.org

The Apache Software Foundation. (Februar 2024). *PySpark Overview*. The Apache Software Foundation. Abgerufen am 24. Juli 2024, von https://spark.apache.org/docs/latest/api/python/index.html