Aufgabe 2: Kafka

Auch für diese Aufgabe benötigen Sie wieder einen VPN-Zugang. Der Kafka-Server der DHBW ist erreichbar unter 10.50.15.52:9092. Kafka ist eine weitere Möglichkeit, ereignisbasierte Datenströme zu speichern. Es bietet ebenfalls ein Subscriber-/Listener-Schema, das hier Producer/Consumer heißt.
Bitte geben Sie ein kurzes Protokoll der Bearbeitung der Aufgaben als PDF-Datei ab (mit Antworten auf die Fragen). Im Protokoll können Sie auch gerne aufge- tretene Probleme und deren Lösung beschreiben. Zusätzlich geben Sie bitte Ihre Quelldateien als Zip-Datei ab. Sie können dabei die gesamte Abgabe in einer einzigen Zip-Datei ablegen. Die Abgabe kann in Gruppen von 3-4 Personen im Moodle-Abgabecontainer erfolgen.

(a) Machen Sie sich mit Kafka vertraut.

- **Was sind Gemeinsamkeiten und Unterschiede zwischen Kafka und MQTT-Brokern?**
  - Beide Technologien/Protokolle nutzen das Publish-Subscribe-Pattern, um Daten zu kommunizieren.
  - Damit auch ein Broker-System, über das Nachrichten verteilt werden.
  - Im Gegensatz zu MQTT werden in Kafka einkommende Daten gespeichert. Es kann also auch im Nachinein noch auf sie zugegriffen werden. Bei MQTT können Daten nicht im Nachhinein angesehen werden, wenn sie nicht vom Client mitgeschnitten und abgespeichert wurden.
  - Producer und Consumer (bzw. Publisher und Subscriber) sind in beiden Technologien voneinander entkoppelt. Ein Producer muss also nicht auf einen Consumer warten, um seine Arbeit abzuschließen.
  - Ein Unterschied dabei ist allerdings, dass Kafka sicherstellt, dass Events nur exakt einmal geschickt werden und bei den Consumern, die auf das Event warten, dieses Event auch ankommt.
  - Topics sind sowohl in Kafka, als auch MQTT vorhanden.
- **Für welche Szenarien ist Kafka besser geeignet?**
  Kafka ist vor allem dann geeignet, wenn viele Datenmengen gleichzeitig geschrieben und gelesen werden müssen. Durch die verteilte Architektur kann Kafka sehr gut in die Breite skalieren und hat keinen Single-Point-Of-Failure (durch Partitionen).
  Außerdem ist Kafka dann geeignet, wenn auf Nachrichten in der Vergangenheit zugegriffen werden müssen. Dies ist mit typischen MQTT-Anwendungen nicht möglich, sofern der Client nicht bereits Teilnehmer ist und den gesamten Nachrichtenverlauf mitgeschnitten hat.

- **Man findet bei Vergleich öfters die Aussage, Kafka würde das Modell „Dumb Broker / Smart Consumer“ implementieren, während bei MQTT „Smart Broker / Dumb Consumer“ gilt. Was ist damit gemeint? Was muss ein Kafka-Consumer beachten?**

Kafkas Broker können als Dumb Broker bezeichnet werden, da sie nicht mit den Consumern interagieren. Sie wissen damit nicht, ob eine Nachricht konsumiert wurde oder nicht. Stattdessen halten sie alle Nachrichten für einen gewissen Zeitraum vor. Es ist also keine Logik implementiert, die sicherstellt, dass eine Nachricht auch von einem Consumer konsumiert wird. Diese Aufgabe wird bei Kafka dem Consumer überlassen, der schließlich die gewünschten Nachrichten aussuchen muss. MQTT hingegen schreibt Dumb Consumer vor. Sie können nicht mit den Brokern interagieren, sondern lediglich auf die Nachrichten hören, die gerade gepublished werden. Diese Daten kann der Client wiederum mitschneiden und abspeichern. Er erhält allerdings keine Möglichkeit, vergangene Daten zu beziehen. Andererseits stellt bei diesem Modell der Broker sicher, dass der Client Nachrichten einer unterschriebenen Topic auch erhält und muss die Nachricht dazu ggf. öfter versenden, wenn sie nicht ankommt.
Kafka-Consumer müssen beachten, dass sie beim Zugriff auf verschiedene Partitionen die „richtigen“ Nachrichten lesen.

- **Was sind Partitionen? Für was kann man sie neben Load-Balancing noch verwenden?**
  Kafka ist ein System, das über mehrere Systeme verteilt ist. Mit Partitionen können Topics in einzelne „Logs“ aufgeteilt werden, die wiederum auf verschiedenen Systemen im Kafka-Cluster liegen können. Dadurch kann das Schreiben und Lesen von Topics auf mehrere Systeme ausgelagert werden (Load-Balancing). Neben Load-Balancing können Kafka-Partitionen beispielsweise auch zur Erhöhung der Verfügbarkeit genutzt werden, indem Topics auf mehrere Nodes im Cluster repliziert werden. Fällt ein Node aus, so kann auf das/die Replicat/e zugegriffen werden.

(b) Die allseits beliebten Wetter-Daten mal wieder.

- **Implementieren und testen Sie ihren Consumer.**
  siehe Quelldateien

- **Geben Sie die Quellen mit in Ihrem Archiv ab.**
  siehe Abgabgecontainer

(c) Eigener Producer.

- **Implementieren und testen Sie ihren Producer.**
  siehe Quelldateien

- **Geben Sie die Quellen mit in Ihrem Archiv ab.**
  siehe Abgabgecontainer

- **Wozu dienen Key und Value beim Versenden von Kafka-Messages?**
  Im Value werden die tatsächlichen Daten versendet. Diese können in jeglicher Form vorliegen, bspw. in JSON, XML oder auch als Plaintext. Der Key wird

benötigt, um Nachrichten immer in der korrekten Reihenfolge der Nachrichten zu gewährleisten. Er ist einzigartig für jeden Record einer gesamten Topic.

(d) Kafka-Pipeline mit Grafana.

- **Schauen Sie sich das Tutorial zu Grafrana an.**

- **Benutzen Sie das DHBW-Grafana unter http://10.50.15.52:3000. Login bitte bei mir via Mail erfragen…**

- **Oder installieren Sie sich lokal eine Grafana-Instanz…**

- **Ziel ist, die Wetter-Daten via Grafana zu visualisieren für die 3 Standorte.**

- **Wie bekommen wir die Daten nun vom Broker ins Grafana?**
  Grafana bietet die Möglichkeit, verschiedene Datenquellen („Data sources“) einzubinden. Für jedes Panel kann dabei eine eigene Datenquelle ausgewählt werden, aus der Daten bezogen werden. Die unterstützten Datenquellen werden in der Dokumentation (https://grafana.com/docs/grafana/latest/datasources/) augelistet. Dabei handelt es sich um verschiedene Datenbanksysteme. Neue Datenquellen können über Plugins hinzugefügt werden. Anschließend kann über die für die Datenquelle typische Query-Language (für MySQL beispielsweise SQL) auf Daten in der Datenbank zugegriffen werden. Diese Daten werden anschließend im Dashboard im entsprechenden Panel visualisiert.
  Um Daten nun vom Broker in Grafana zu übertragen, müssen die Nachrichten eines Producers zunächst abgefragt werden (als Consumer) und anschließend in eine beliebige von Grafana unterstützte Datenbank geschrieben werden.
  Anschließend kann dann von Grafana aus auf die Nachrichten in der Datenbank zugegriffen werden.

Prinzipiell können Sie gerne auch einen anderen Weg gehen, als den in der Aufgabe vorgeschlagenen. Eine Alternative wäre beispielsweise ein eigener PostgreSQL- oder MySQL-Server, oder auch Prometheus.
Noch einfacher geht es allerdings mit Graphite, einer Datenbank für Zeitseriendaten.

- **Machen Sie sich mit Graphite vertraut.**

- **Schauen Sie sich insbesondere an, wie Daten in Graphite abgespeichert werden.**
  Daten werden in Metriken abgespeichert. Dabei kann Graphite als eine Art Key-Value-Store angesehen werden. Der Key kann hierarchisch aufgebaut werden, dazu werden einzelne Ebenen mit einem Punkt getrennt, z.B. „inf19b.weather.mosbach.tmpCurrent“. Der entsprechende Wert wird schließlich mit einem Zeitstempel versehen und unter diesem Key gespeichert. Auf diese

Weise legt Graphite eine Historie über die verschiedenen Werte an. Anders, als bei Key-Value-Stores wird also der Wert nicht überschrieben, sondern hinzugefügt.

- **Wir müssen umdenken! Einfach mal JSON reinfüttern wird nicht gehen.**

- **Überlegen Sie sich eine sinnvolle Namenshierarchie.**
  vlvs_inf19b.<user_id>.weather.<city>.<attribute>
  weobei vlvs_inf19b.<user_id>.weather der Präfix ist, unter der alle Wetterdaten gespeichert werden. In Grafana kann auf diese Daten pro Stadt zugegriffen werden.

- **Für erste Schritte können Sie gerne die auf dem Kafka-Server installierte Graphite-Instanz nehmen (http://10.50.15.52).**
  Ein Problem, das hierbei aufgetreten ist, war das Einspeisen der Daten in Graphite. Der übergebene Unix-Timestamp wird in Graphite in Sekunden benötigt, nicht etwa in Milli- oder Nanosekunden. Sonst werden Daten nicht richtig angezeigt.

Aufgabe: Schreiben Sie einen Kafka-Consumer, die die Daten an Ihre Graphite-Instanz übergibt.

- **Geben Sie den Code Ihres Kafka-Consumers ab.**
  siehe Quelldateien

- **Geben Sie ebenfalls einen Screenshot Ihres Grafana-Dashboards ab (mit Gruppen-Id, bzw. Matrikelnummern im Titel des Graphen).**
  siehe Quelldateien