Mat. Nr: 2627585, 2858031,3106335,3326612

2 a)

- Was sind Gemeinsamkeiten und Unterschiede zwischen Kafka und MQTT-Brokern?
    - Gemeinsamkeiten
        - Topics für verschiedene Themenkategorien
        - Textfreiheit -> Interoperabilität frei wählbar
        - Pub/Sub-Architektur
    - Unterschiede
        - MQTT besitzt keine Partitionen
        - Kafka ist store-and-forward, MQTT ist fire-and-forget
        - Kafka ist auf Skalierbarkeit ausgelegt (z.B. Cluster mit mehreren Nodes)
        - MQTT ist ein Protokoll, Kafka ist eine Event-Streaming-Plattform (mit eigenem Protokoll)
        - MQTT ist schlank für Clients ausgelegt (bsp .für IoT-Geräte), wobei der Kafka-Client mehrere TCP Verbindungen gleichzeitig aufrecht erhalten kann und viel komplexer ist
- Für welche Szenarien ist Kafka besser geeignet?
    - Falls Zustellung der Nachrichten sichergestellt sein muss -> Zahlungsverkehr, Finanztransaktionen
    - Große Datendurchsätze (bsp. zwischen nativen Cloud-Servern)
- Man findet bei Vergleichen 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?
    - Bei Kafka speichert der Broker nicht welche Nachrichten bereits von einem Consumer verarbeitet wurden, daher muss der Consumer selbst darauf achten Informationen nicht mehrfach zu verarbeiten (bzw. nicht mehrfach abzufragen).
    - Bei MQTT liefert der Broker die Nachrichten nicht mehrfach an den gleichen Subscriber aus (außer Retained Nachrichten; diese jedoch erst nach einem Reconnect des Subscribers), daher kann der Client alle Nachrichten die erhalten werden ohne Prüfung verarbeiten
- Was sind Partitionen? Für was kann man sie neben Load-Balancing noch verwenden?
    - Aufteilung eines topics in mehrere Teile/Partitionen
    - Partition: Kleinste Speichereinheit
    - Partitionen können im Kafka-Cluster auf unterschiedlichen nodes laufen (bessere Performance)
    - Ermöglichen, dass ein Topic von mehreren Consumers parallel konsumiert werden kann
    - Mehrere Instanzen desselben Consumers könne sich mit Partitionen unterschiedlicher nodes verbinden (höherer Message-Durchsatz)
    - Kafka benutzt Partitionen für Redundanz

b)

• Implementieren und testen Sie ihren Consumer.

• Geben Sie die Quellen mit in Ihrem Archiv ab

c)
• Implementieren und testen Sie ihren Producer.
• Geben Sie die Quellen mit in Ihrem Archiv ab.
• Wozu dienen Key und Value beim Versenden von Kafka-Messages?
- Der Key dient dem Kafka-Cluster, gleiche Keys auf gleiche Partitionen unterzubringen
- Der Value ist der eigentliche Payload

d)
• Geben Sie den Code Ihres Kafka-Consumers ab.
• Geben Sie ebenfalls einen Screenshot Ihres Grafana-Dashboards ab (mit Gruppen-Id, bzw. Matrikelnummern im Titel des Graphen).