APACHE KAFKA

Kafka Brokers: nodos de guardem a data de uma partition, a o conceito de Leader Broker que é um nodo que será seguido(armazenará a data do nodo que está seguindo) por outros nodos, quando o Leader Broker crashar, será eleito um novo Leader Broker, é assim que o Kafka guarda a informação das Partitions e faz com que o sistema seja "Fault-Tolerant".

Como saber qual partition é a Leader, para saber qual Read/Write. Para isso é usado o service Zookeeper.

Zookeeper é um sistema distribuído de key-value, assim quando uma mudança é feita (um Leader crasha e é eleito um novo leader), a data será guardada pelo Zookeeper, assim os nodos sabem qual Partition seguir.
Todos o Brokers estão sincronizados com o Zookeeper, para saber quem é o leader, quando um Producer quer escrever um mensagem em uma Partition e nao sabe qual ele vai pedir para qualquer Broker quem é o Leader e vai ser enviado uma resposta contendo qual o Partition Leader está sendo usado no momento.

Três tipos de status para data em streaming: urgent, not-so-urgent, flexible:

- **Urgent**: deve ser tratado em real-time
- **Not-so-urgent**: pode ser tratado em segundos
- **Flexible**: segundos, horas, dias, semanas.

Kafka streams: framework para trabalhar com dados em streaming com Kafka.
Comandos:

- Analisar logs zookeeper: docker-compose logs zookeeper | grep -i binding
- Analisar logs kafka: docker-compose logs kafka | grep -i started

ACKS_CONFIG - Critério para considerar uma request como completa.
RETRIES_CONFIG - Número de vezes que vai tentar enviar novamente se a request falhar
BATCH_SIZE_CONFIG - Tamanho do buffer que armazena registros ainda não enviados
LINGER_MS_CONFIG - Tempo que espera até enviar uma request (em milisegundos)
BUFFER_MEMORY_CONFIG - The buffer.memory controls the total amount of memory available to the producer for buffering.

REFERÊNCIAS

https://kafka.apache.org/documentation/streams/

https://www.confluent.io/blog/introducing-kafka-streams-stream-processing-made-simple/

https://docs.microsoft.com/pt-br/azure/hdinsight/kafka/apache-kafka-streams-api

https://dzone.com/articles/kafka-streams-more-than-just-dumb-storage

https://github.com/confluentinc/kafka-streams-examples

https://medium.com/@stephane.maarek/the-kafka-api-battle-producer-vs-consumer-vs-kafka-connect-vs-kafka-streams-vs-ksql-ef584274c1e

https://docs.confluent.io/current/quickstart/ce-docker-quickstart.html#getting-started-with-docker-compose

https://medium.com/trainingcenter/apache-kafka-838882261e83

https://medium.com/trainingcenter/apache-kafka-codifica%C3%A7%C3%A3o-na-pratica-9c6a4142a08f

How to easily run Kafka with Docker for Development:
https://www.e4developer.com/2018/05/20/how-to-easily-run-kafka-with-docker-for-development/

Kafka and Zookeeper with Docker:
https://medium.com/rahasak/kafka-and-zookeeper-with-docker-65cff2c2c34f