Kafka

Anotações de um estudo inicial que fiz de Kafka.

Kafka é um sistema de mensageria.

> ⚠️ Um sistema de mensageria que consegue trabalhar com paralelismo, distribuição, é totalmente preparado para falhas, com ele se um sistema falhar a mensagem não é perdida e sim armazenada por **x** dias que nós configuramos e quando o sistema voltar ele envia para o sistema a mensagem correta, com ele os sistemas não precisam se conhecer e nem depender do outro, cada um executa sua função e envia para o kafka a mensagem em questão sem esperar o outro, gerando assim um sistema mais performático e idenpendente. Se tivermos um cluster de kafka com mais de um kafka rodando acontece que se algum deles cair o outro vai ter as mensagens dele e trabalhará pelo outro que caiu, então um broker cai o outro começa a escutar e enviar mensagens do que caiu e também ele tinha um backup do que caiu, pois é assim que o kafka é configurado, quando temos mais de um broker acaba que eles escutam as mensagens uns dos outros para ter um backup em caso de falhas e conseguem escutar e enviar mensagens deles e do que caiu.

Zookeeper: Ele é um service descovery e ele ajuda o Kafka a distribuir seus nodes, como ele vai descobrir novas máquinas.

Tópico: É o local aonde nossas mensagens serão encaminhadas, então todos os dados que vão trafegar pelo Kafka vão ser trefegados através de um tópico, quando nós jogamos essa informação esse tópico pode ter algumas partições, ou seja partições é uma forma de dividirmos o tópico em partes e essas partições podem estar espalhadas em brokers diferentes (Máquinas diferentes)

Comandos:

> `docker-compose up -d` Sobe o docker da maneira que está configurada o arquivo do docker.

> `docker-compose ps` Lista os containers, inclusive do kafka

> `docker exec -it kafka-cluster_kafka-1-1 bash` Esse comandos faz com que entremos dentro de uma máquina chamada `kafka-cluster_kafka-1-1`

Conectado na máquina podemos executar comandos do Kafka:

> `kafka-topics --create --bootstrap-server localhost:29092 --replication-factor 3 --partitions 3 --topic meuTopico` Com esse comando estamos criando um tópico e falando qual a máquina que vamos nos conectar que tem o Kafka rodando que no caso está no localhot:29092, com 3 replication factor e 3 partições, o tópico também conta com o nome de “meuTopico”.

> `kafka-topics --list --bootstrap-server localhost:29092` Comando que lista os tópicos que temos na máquina que está em localhost:29092.

💡 **replication factor**: Nós vamos escolher quantas replicas vamos ter das partições desse tópico espalhadas em outras máquinas.

> `kafka-console-producer --broker-list localhost:29092 --topic meuTopico` Ele vai se conectar a este tópico e poderemos enviar mensagens para este tópico através dele.

> `kafka-console-consumer --bootstrap-server localhost:29092 --topic meuTopico` Ele vai se conectar a este tópico (meuTopico) e poderemos ouvir as mensagens que chegam neste tópico através dele.

> No Kafka não perdemos mensagens antigas, por padrão ele armazena em disco as mensagens então para consumir as mensagens antigas e as que ainda chegarão nós colocamos o comando da seguinte maneira `kafka-console-consumer --bootstrap-server localhost:29092 --topic meuTopico --from-beginning`

💡 Quando estamos no **Kafka** e estamos lendo um tópico somos um **client** (sistema que está lendo), mas muitas vezes esse tópico tem partições diferentes isso significa que poderíamos ter 3 máquinas rodando, lendo esse mesmo tópico e conseguiríamos ler e processar essas mensagens em paralelo, então seria **3x mais rápido**, pois fazemos com que cada máquina dessa leia uma partição de um tópico. Para fazer com que cada **Client** nosso leia uma partição especifica iremos usar os grupos do Kafka, então quando criamos um **Consumer Group** nós conseguimos criar vários **Clients** agrupados por grupo e o **Kafka** automaticamente vai organizar para nós em qual **Client** vai ler qual partição em determinado tópico.

> Comando para especificar o grupo na hora de consumir: `kafka-console-consumer --bootstrap-server localhost:29092 --topic meuTopico --from-beginning --group grupoTeste`

> `kafka-topics --describe --bootstrap-server localhost:29092 --topic meuTopico` Descreve o tópico e as configurações dele.

`kafka-consumer-groups –group groupTeste –bootstrap-server localhost:29092 –describe` Ele mostra como cada grupo (nesse caso groupTeste) está se dividindo para ler o nosso tópico

```
GROUP           TOPIC          PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG             CONSUMER-ID
                    HOST            CLIENT-ID
TesteA          WhatsApp       0          0               0               0               console-consumer-eceafaa7-f46b-4304-9891-43484318da56 /127.0.0.1     console-consumer
TesteA          WhatsApp       1          1               1               0               console-consumer-eceafaa7-f46b-4304-9891-43484318da56 /127.0.0.1     console-consumer
```

No caso ele está mostrando o **GROUP** o **TOPIC**, **Partição**, então temos 2 **clients** conectados nesse tópico neste momento e ele mostra como eles estão conectados no tópico.

CURRENT-OFFSET = cada partição tem um, ele é um número que indica quantas mensagens foram gravadas naquela partição.

LOG-END-OFFSET = Mostra qual é o último OFFSET, pois as vezes estamos processando, mas não chegamos na última mensagem.

LAG= Quantas mensagens pendentes temos para processar tudo.

CONSUMER-ID = Fala que é o cara conectado em cada um desses tópicos.

💡 Se desconectarmos 1 dos 2 acaba que o que ficar ligado deles vai ler as 2 partições.

⚠️ **Se tivermos uma mensagem sendo enviada para vários tópicos todos irão recebe-la, mas se enviarmos essa mensagem para ouvintes do mesmo tópico que são do mesmo grupo apenas 1 deles irá recebe-la, já que o grupo apenas um recebe, podemos direcionar isso através das partitions. Então não podemos ter ouvintes do mesmo grupo se temos apenas 1 partição, pois se não apenas 1 irá sempre receber as mensagens e o outro nunca conseguirá se inscrever em uma partição. O número máximo de paralização vai ser o número de partições. Eles se rebalanceiam sozinhos, se um está ligado e temos 3 partições acontece que ele fica ouvindo das 3 partições, mas se ligamos mais 2, ele rebalanceia e cada um irá ouvir apenas 1 partição, tudo isso automáticamente.**
Mas para o Kafka decidir para qual partição enviar a mensagem temos de passar uma chave e essa chave que ele olhará e através dela irá analisar e decidir para qual partição mandar, a chave pode ser o id do usuário, numero aleatório, etc. Desde que não seja fixo, pois se não ele ficará enviando sempre para a mesma partição.

`kafka-consumer-groups --all-groups --bootstrap-server localhost:29092 --describe` **Usamos esse comando para descrever os grupos de consumidores.**

Podemos destacar também que normalmente, ainda mais em um projeto grande que tem milhares de mensagens e precisamos saber se elas foram concluídas e etc, a gente irá precisar modificar a configuração do **POOL** que é um commit também que vai informar ao **Kafka** se a mensagem já foi lida ou não e daí que ele forma o describe.

```
properties.setProperty(ConsumerConfig.MAX_POLL_RECORDS_CONFIG, "1");
```

Com esse comando de configuração a cada mensagem enviada ele irá notificar o kafka disso e assim o Kafka pode atualizar o describe da maneira que queremos.

Armazenamento:

O **Kafka** e o **Zookeeper** eles armazenam por padrão seus arquivos de log na pasta temp (tmp), sabemos que os arquivos que ficam nessa pasta podem sumir a qualquer momento, se reiniciarmos a máquina, etc. Para evitar isso nós guardamos os LOGS dos mesmos em um diretório que criamos para isso que nós conseguimos controlar.

Para configurar isso editamos os arquivos de configuração que fica na pasta **config** e se chama server.properties para o **Kafka**, **Caminho**: kafka/config/server.properties e mudamos o valor do **log.dirs** para o path do diretório que queremos armazenar. No **Zookeeper** fica em config/zookeeper.properties e mudamos o valor de **dataDir** para o path do diretório que queremos.

Lembrando que os **LOGS** não são apenas para nós vermos, mas para o Kafka é um registro e ele precisa deles para verificar o tempo ou tamanho em disco (configurações do arquivo).