**Apache Kafka**

**producer vs consumer**

-producer = aplikasi yg ngepublish data ke message broker (apache kafka)

-consumer = aplikasi yang ngambil data dari message broker (apache kafka)

A diagram of a cluster of applications

Description automatically generated

1.0 gambar arsitektur kafka

Publish = publish data ke aplikasi

Subscribe = aplikasi yg subcribe bakal nerima data

Stream processor = data yg mengalir terus kalo subcriber

Connector = integrasi dengan database, jadi setiap kali ada perubahan data, datanya di kirim ke apache kafka

Kakfa cluster = cluster adalah kumpulan dari beberapa aplikasi kafka, kafka bakal di instal ganjil karena ada network split brain engga bermasalah

A diagram of a cluster of data

Description automatically generated

1.1 gambar internal cluster kafka

Kafka butuh sistem/ aplikasi **ZOOKEEPER, ZOOKEPER** bakal running di dalam sebuah cluster, artinya ada banyak aplikasi ZOOKEEPER. Saat donwload kafa, aplikasi zookeper tidak perlu di instal lagi **(HANYA BERLAKU UNTUK DEVELOPMENT, PADA PROSES DEPLOYMENT BUTUH INSTAL ZOOKEEPER MANUAL BIKIN CLUSTER SENDIRI)**

Kafka engga bisa manage clusternya sendiri, jadi butuh si ZOOKEEPER buat manage.

**Topic (data)**

-data disimpan di dalam topic

-data ditopic tidak bisa diubah (Konsep nya buat ngirim event aja)

A diagram of a diagram

Description automatically generated

1.2 topic partition

Angapan topic itu 1 buah file, dan kita mau bagi topic itu bagi-bagi jadi beberapa file. Kenapa dibutuhkan partition karena terlalu besar kalo engga partition. Lalu untuk TOPIC PARTITION kenapa dilakuin partition, karena dalam 1 partition itu cuman bisa dipakai oleh 1 consumer saja, jadi kalo misal ada 3 aplikasi jadi kita harus nyediain 3 partition juga.

Offset = posisi terakhir consumer membaca data, kenapa butuh offset? Karena kalo aplikasi mati otomatis engga reload dari awal offsetnya(maksudnya baca dari 0 ke N) tapi langsung lompat dari offset ke N. Offset secara default sudah di setting oleh kafka.

**A screenshot of a computer

Description automatically generated**

1.3 gambar replication

Tujuannya replication buat replikasi jadi kalo server mati tinggal ambil replikanya aja. Jadi contohnya diatas ada 3 server kafka, saat kita bikin 1 topic lalu kita bikin 2 partisi yaitu p1 dan p2. P artinya primary (data utama) lalu kita replikasi sebanyak 1. Lalu setiap partisi bakal di replika, p1 jadi r1 dan p2 jadi r2. Letak penempatan di atur oleh kafka, intinya 1 server engga mungkin ada primary dan replikasinya, pasti di tempatin di server lain.

A diagram of a group

Description automatically generated

1.4 gambar consumer group

Konsep consumer group = buat pisahin, misal ada 1 server isinya p1 dan p2. Lalu ada aplikasi A dan aplikasi B. secara logic berati kita sambungin 2 aplikasi ke masing masing server, misal p1 ke aplikasi A dan p2 ke aplikasi A, lalu p1 ke B dan p2 ke B. tapi gimana kalo misalnya ada satu order nih, lalu diterusin ke server dan tiap aplikasi nerima orderan tersebut, lalu disuruh payment. Maka yang terjadi kedua aplikasi tersebut melakukan payment 2 kali. Hal ini dihindari makanya dibuat consumer group.

A screenshot of a computer

Description automatically generated

1.5 contoh consumer group

Kalo udah dijadiin consumer group, aplikasi A ga mungkin disambungin ke p2 dan sebaliknya karena consumer group nya sama.

A close-up of a white background

Description automatically generated

1.6 tipe retention policy

-Retention Policy = data dihapus secara otomatis

-Log Retention Time = jeda waktu seberapa lama waktu yang kita ingi datanya di keep, by default data di keep 7 hari (yg dihapus data yg udah lewaat 7 hari aja)

-Log Retention Bytes = berapa maksimum ukuran partition yang diperbolehkan, misal makx 30 gb, lalu ukuran partition udah mau penuh, lalu yg baru masuk ke kafka bakal dihapus karena maks nya 30gb

-Offset Retention Time = seberapa lama data offset di simpan di kafka, by default kalo engga ada pergerakan kafka atau offsetnya sama terus lebih dari 7 minggu bakal direset.