Bonjour tout le monde

Monde "Traditionnel"

Rabbit MQ → Montée en charge est "possible"

Activ MQ

+ puissant fonctionnellement que Kafka

Monde "Big Data"

Kafka → Montée en charge est conçue d'origine -

Structure souple dans tous les cas —

Big Data

Règle des 3 V

ETL

ELT

- Vitesse / Volacité ex: 100 To/sec débit — Le data dans Kafka
- Volume → Volume à un instant t — nécessaire
- Variabilité → schéma souple
  - Veracité (Volabilité) (Immuabilité)
  - Valeur ajoutée

Message → Information | structuré
- non "
- semi "

Appel Synchrone → Remote Procedure Call

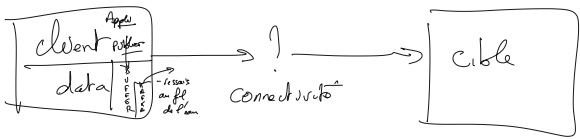
API en HA = objectif archi en microservices

→ nécessité que l'API appelée soit disponible (pas besoin de MQ)

Appels Asynchrones → en cas de connectivité fluctuante

→ Besoin de bufferiser les données

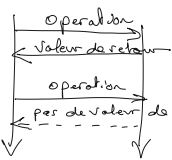
→ Transmission si la récupération de la ligne



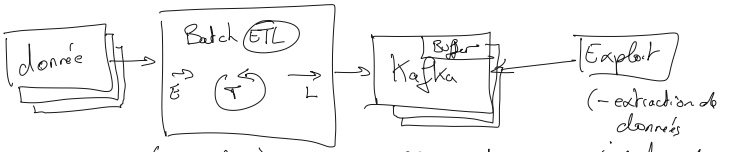
void operation (message)

message

=> Pas implémenté dans Kafka → Pas un Pb.



ETL vs ELT



(ex: Talend) ⇒ Scale Up (Mono instance souvent)

(- extraction de données
- ingestion de données, etc...

PBN des Batch:

- exécution planifiée (pas au fil de l'eau)
  - → Latence
  - → Durée d'exécution selon la charge

Montée en puissance: Mixé à l'échelle

Scale Up (Verticale)
+ - config/tuning - meta/esclave - Active MQ
- primaire/secondaire - Rabbit MQ
↳ BATCH - etc... - Mysql
↳ ETL

+ Scale Out - MongoDB - etc...
(Horizontale) Kafka

⇒ cluster | - Kubernetes - Hadoop
↳ Flux (Streams) ⇒ H.A.

Scale In -

Scale Down ↳ ELT

- Limite de Puissance
  - CPU - 2nde Machine slave en backup = coût 2x pour une machine en service
  - RAM

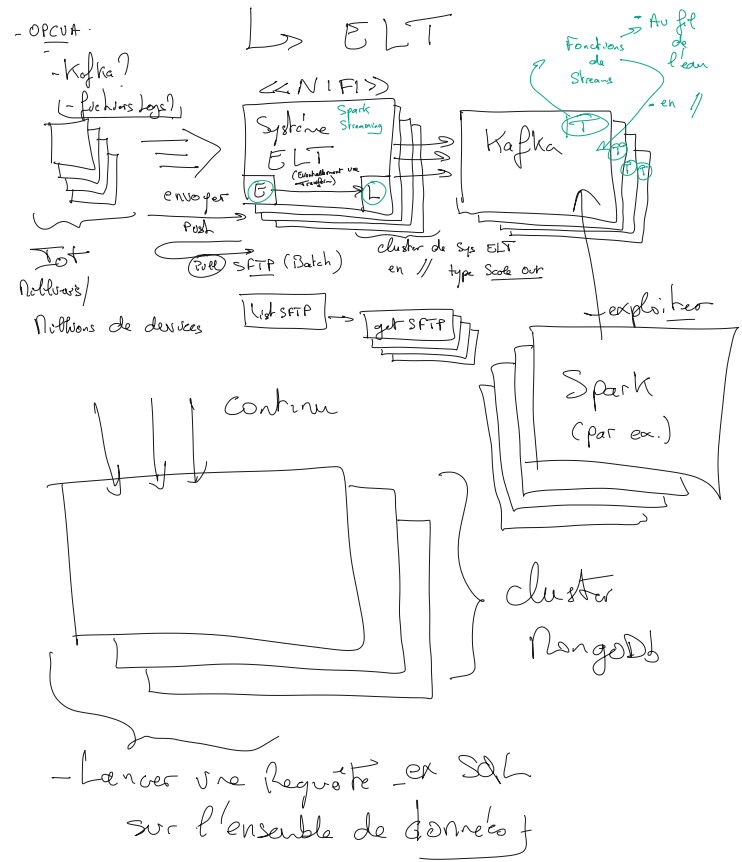
Scale Up = Machine + puissantes

Scale Out = Machine de base, ajout de Machines

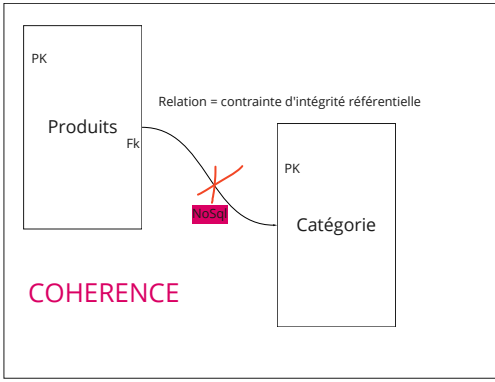
↳ Ajout de Machines et instance en
- fil de la Montée en charge
- Participe à la tolérance de PANNES

Haute Dispo HA.

~~EXL~~ SU Très gros Volumes



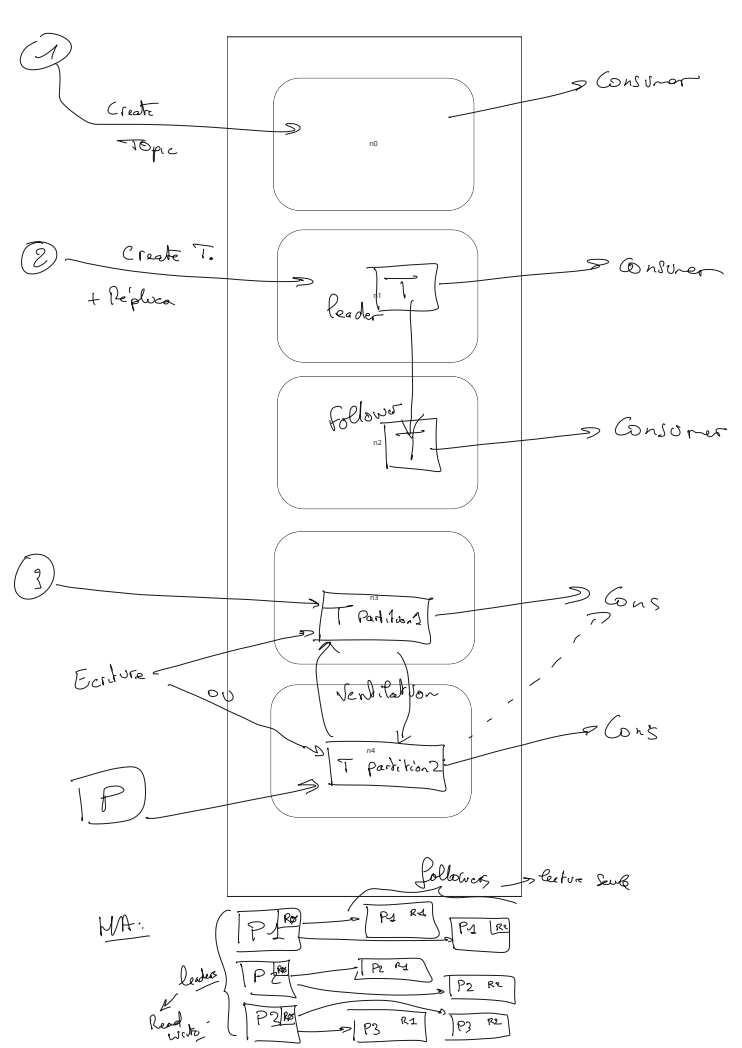
Relationnel

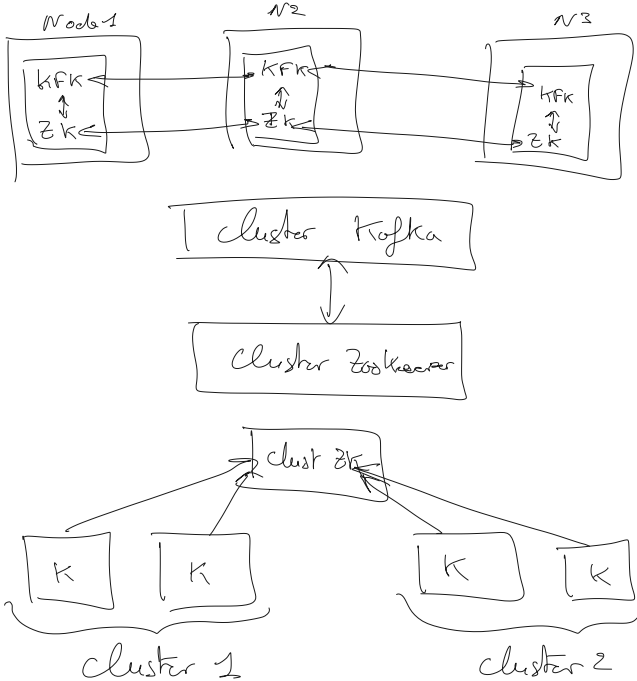


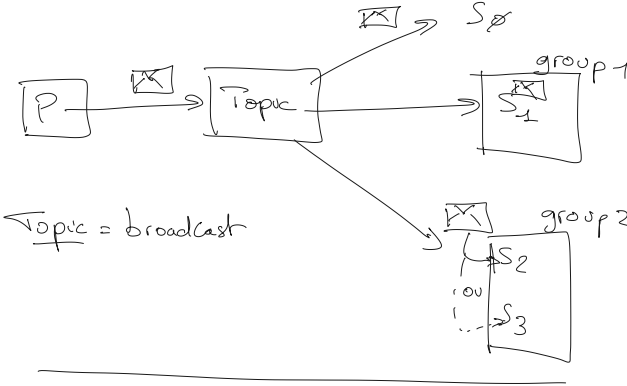
Relationnel = Tables avec des enregistrements (lignes)

NoSql Mongo = Collections avec des documents (id en guise de PK)

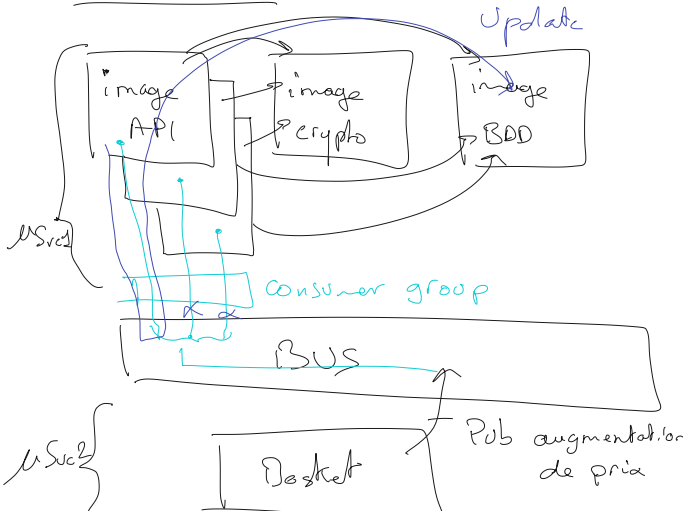
Le relationnel est une anti pattern (une mauvaise pratique) dans le domaine BigData


1
Create
Topic
n0
Consumer
2
Create T.
+ Réplica
T
leader
Consumer
follower
n2
Consumer
3
n3
T Partition1
Cons
Ecriture
OU
Verification
n4
T partition2
Cons
P
Followers
lecture seule
M/A:
P1 R0
P1 R1
P1 R2
leader
Read
Write
P2 R0
P2 R1
P2 R2
P3 R0
P3 R1
P3 R2


Node 1
N2
N3
KFK
ZK
KFK
ZK
KFK
ZK
cluster Kafka
Cluster Zookeeper
clust ZK
K
K
K
K
cluster 1
cluster 2

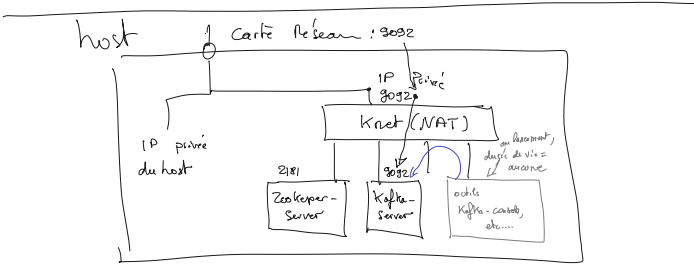


Microservices



```
sudo apt install docker.io
docker image pull bitnami/kafka
docker image inspect bitnami/kafka
...
"ExposedPorts": {
            "9092/tcp": {}
        },
...
docker image pull bitnami/zookeeper
docker image inspect bitnami/zookeeper
...
"ExposedPorts": {
            "2181/tcp": {},
            "2888/tcp": {},
            "3888/tcp": {},
            "8080/tcp": {}
        },
...
docker network create knet --driver bridge  # création du réseau
cat zoo.sh
docker run -d --name zookeeper-server --network knet -e
ALLOW_ANONYMOUS_LOGIN=yes bitnami/zookeeper:latest

cat ./kafka.sh
docker run -d --name kafka-server --network knet -e
ALLOW_PLAINTEXT_LISTENER=yes -e
KAFKA_CFG_ZOOKEEPER_CONNECT=zookeeper-server:2181 -p 9092:9092
bitnami/kafka:latest
```



Pour lancer un outil en ligne de commande dans le cluster
on peut lancer :
docker run -it --rm --network knet bitnami/kafka kafka-console-producer.sh
(lancement interactif, auto suppression, de
kafka-console-producer.sh
à partir d'une image kafka dans le réseau que l'on a créé)

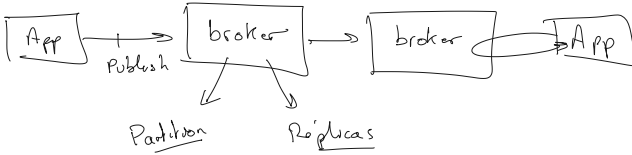
Creation d'un topic :
docker run -it --rm --network knet bitnami/kafka kafka-topics.sh --create --zookeeper zookeeper-server:2181 --replication-factor 1 --partitions 1 --topic demo

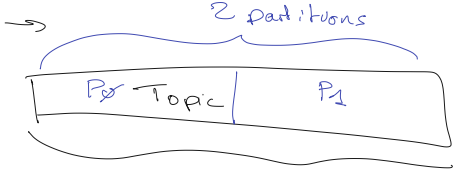
Lecture :
sudo docker run -it --rm --network knet bitnami/kafka kafka-console-consumer.sh --bootstrap-server kafka-server:9092 --topic test --from-beginning

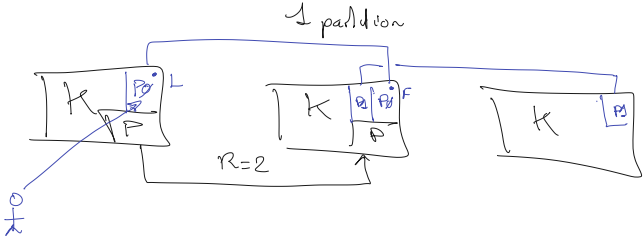
Envoi de messages en interactif :
docker run -it --rm --network knet bitnami/kafka kafka-console-producer.sh --broker-list kafka-server:9092 --topic test

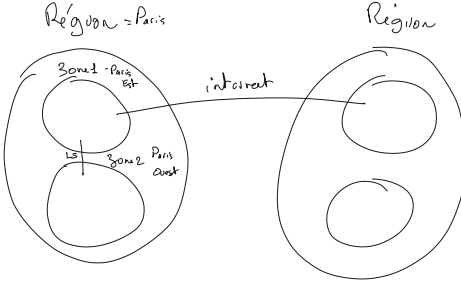
Producer :
--bootstrap-server <kafka:9092 par défaut> : **OK**
ou
--broker-list <liste de serveurs> : **Obsolète**

Consumer :
--bootstrap-server <serveur>

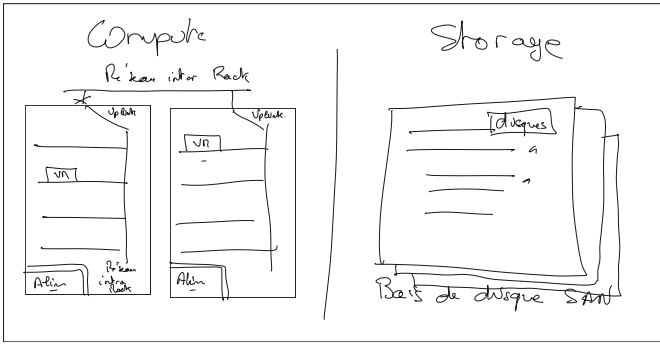

App
Publish
broker
broker
App
Partition
Réplicas

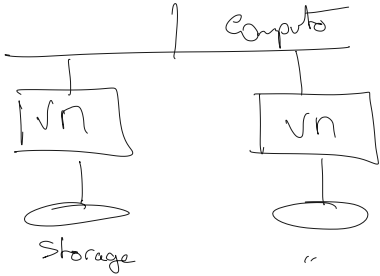

2 partitions
P0
Topic
P1


1 partition
L
F
R=2


Région = Paris
Région
Zone1 - Paris Est
intersect
LS
Zone2 Paris ouest

Dans 1 Zone / 1 Data Center



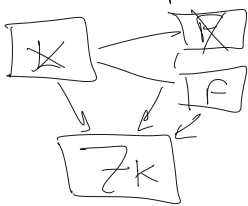


Recommandations = Disposer autant que possible des VMs Dans des Racks ≠

Et connaître la topologie pour Distribuer les Réplicas SUR DES RACKS ≠

Topic = Logique
Queue = "

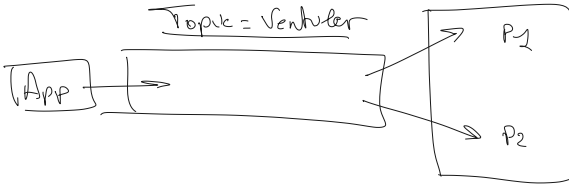
Réplica = haute dispo en Leader / Followers
=> 3 Recommandé -> pas d'avantage de Perf -
-> possible de Distribuer inter Site !



Perte du leader
= Réélection interne
Pour décider un
nouveau Leader

Partitions = Distribution de la charge
n Partitions sur n Serveurs
Distribution peut être désequilibrée

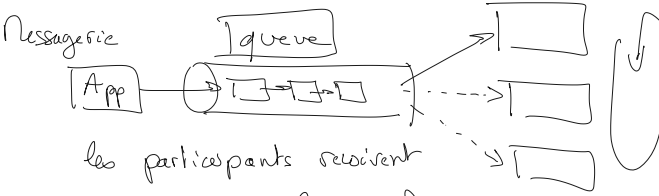
File = Topic ou Queue



Ventilation à tous

Modèle Pub/Sub → 1 msg à tous

⇒ TOPIC

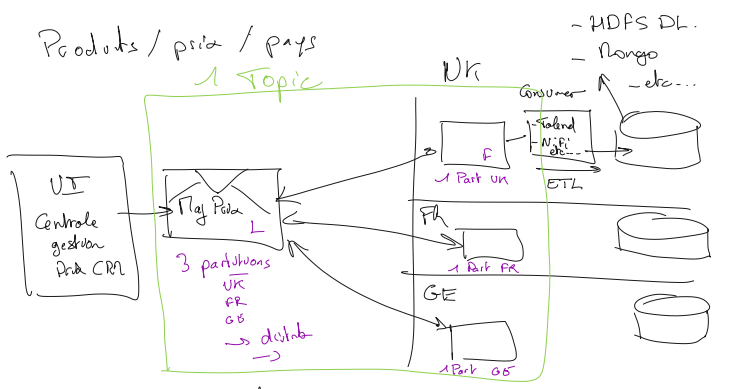


les participants reçoivent un message → Message Queue pour Traitement

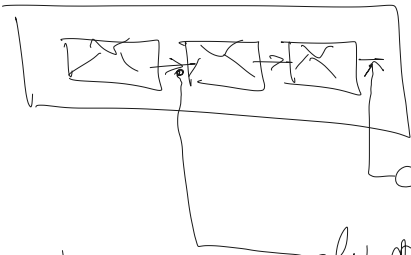
---

Kafka → Que de Topics

Queues possible avec le consumer group—



- Mettre les broker en réseau



→ Stockage des messages (L/F)

→ pointeur par client

⇒ les consumers doivent s'abonner (avec un GUID)

Mise en cluster :

Simple !

-> modifier server.properties

-> avoir un stockage par server

```
stagiaire@master:/opt/kafka_2.13-3.0.0/bin$ ./kafka-topics.sh --bootstrap-server srv3:9092 --list
__consumer_offsets
test
stagiaire@master:/opt/kafka_2.13-3.0.0/bin$ ./kafka-topics.sh --bootstrap-server srv3:9092 --topic test --describe

Topic: test TopicId: oAad6rMZRw568oEiw5W9DQ PartitionCount: 1 ReplicationFactor: 1 Configs: segment.bytes=1073741824

stagiaire@master:/opt/kafka_2.13-3.0.0/bin$ ./kafka-topics.sh --bootstrap-server srv3:9092 --topic topic1 --create --replication-factor 1 --partitions 1
Created topic topic1.
stagiaire@master:/opt/kafka_2.13-3.0.0/bin$ ./kafka-topics.sh --bootstrap-server srv3:9092 --topic topic3 --create --replication-factor 3 --partitions 1
Created topic topic3.

stagiaire@master:/opt/kafka_2.13-3.0.0/bin$ ./kafka-topics.sh --bootstrap-server srv3:9092 --topic topic3 --describe
Topic: topic3 TopicId: 3ILFlU0RSraMSfUStLa_xQ PartitionCount: 1 ReplicationFactor: 3 Configs: segment.bytes=1073741824
 Topic: topic3 Partition: 0 Leader: 2 Replicas: 2,3,1 Isr: 2,3,1

stagiaire@master:/opt/kafka_2.13-3.0.0/bin$ ./kafka-topics.sh --bootstrap-server srv3:9092 --topic topic3 --alter --partitions 10
stagiaire@master:/opt/kafka_2.13-3.0.0/bin$ ./kafka-topics.sh --bootstrap-server srv3:9092 --topic topic3 --describe
Topic: topic3 TopicId: 3ILFlU0RSraMSfUStLa_xQ PartitionCount: 10 ReplicationFactor: 3 Configs: segment.bytes=1073741824
 Topic: topic3 Partition: 0 Leader: 2 Replicas: 2,3,1 Isr: 2,3,1
 Topic: topic3 Partition: 1 Leader: 3 Replicas: 3,2,1 Isr: 3,2,1
 Topic: topic3 Partition: 2 Leader: 1 Replicas: 1,3,2 Isr: 1,3,2
 Topic: topic3 Partition: 3 Leader: 2 Replicas: 2,3,1 Isr: 2,3,1
 Topic: topic3 Partition: 4 Leader: 3 Replicas: 3,1,2 Isr: 3,1,2
 Topic: topic3 Partition: 5 Leader: 1 Replicas: 1,2,3 Isr: 1,2,3
 Topic: topic3 Partition: 6 Leader: 2 Replicas: 2,1,3 Isr: 2,1,3
 Topic: topic3 Partition: 7 Leader: 3 Replicas: 3,2,1 Isr: 3,2,1
 Topic: topic3 Partition: 8 Leader: 1 Replicas: 1,3,2 Isr: 1,3,2
 Topic: topic3 Partition: 9 Leader: 2 Replicas: 2,3,1 Isr: 2,3,1
```

Vers Kafka :

Dataframe Kafka :

```
df = spark \
  .readStream \
  .format("kafka") \
  .option("kafka.bootstrap.servers", "host1:port1,host2:port2") \
  .option("subscribe", "topic1") \
  .option("startingOffsets", "earliest") \
  .load()

df.printSchema() reveals the schema of our DataFrame.
root
 |-- key: binary (nullable = true)
 |-- value: binary (nullable = true)
 |-- topic: string (nullable = true)
 |-- partition: integer (nullable = true)
 |-- offset: long (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- timestampType: integer (nullable = true)
```

https://databricks.com/fr/blog/2017/04/26/processing-data-in-apache-kafka-with-structured-streaming-in-apache-spark-2-2.html

```
# Write key-value data from a DataFrame to a Kafka topic specified in an option
 query = df \
  .selectExpr("CAST(userId AS STRING) AS key", "to_json(struct(*)) AS value") \
  .writeStream \
  .format("kafka") \
.option("kafka.bootstrap.servers", "host1:port1,host2:port2") \
  .option("topic", "topic1") \
  .option("checkpointLocation", "/path/to/HDFS/dir") \
  .start()
```

Kafdrop (GUI pour Kafka)

```
sudo docker container run -d --name kafdrop -p 9000:9000 -e KAFKA_BROKERCONNECT=172.17.0.1:9092 -e JVM_OPTS="-Xms32M -Xmx64M" -e SERVER_SERVLET_CONTEXTPATH="/" obsidiandynamics/kafdrop

firefox http://localhost:9000 &
```

1) Créer un flux de description des topics que l'on souhaite déplacer.
cat ~/topics.json
{"topics":
 [{"topic": "topic3"}],
 "version":1
}

2) demander une suggestion
./kafka-reassign-partitions.sh --bootstrap-server master:9092 --generate --topics-to-move-json-file ~/topics.json --broker-list "0,1,2,3"

Proposed partition reassignment configuration
{"version":1,"partitions":
[{"topic":"topic3","partition":0,"replicas":[0,2,3],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":1,"replicas":[1,3,0],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":2,"replicas":[2,0,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":3,"replicas":[3,1,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":4,"replicas":[0,3,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":5,"replicas":[1,0,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":6,"replicas":[2,1,3],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":7,"replicas":[3,2,0],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":8,"replicas":[0,1,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":9,"replicas":[1,2,3],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":10,"replicas":[2,3,0],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":11,"replicas":[3,0,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":12,"replicas":[0,2,3],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":13,"replicas":[1,3,0],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":14,"replicas":[2,0,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":15,"replicas":[3,1,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":16,"replicas":[0,3,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":17,"replicas":[1,0,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":18,"replicas":[2,1,3],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":19,"replicas":[3,2,0],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":20,"replicas":[0,1,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":21,"replicas":[1,2,3],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":22,"replicas":[2,3,0],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":23,"replicas":[3,0,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":24,"replicas":[0,2,3],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":25,"replicas":[1,3,0],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":26,"replicas":[2,0,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":27,"replicas":[3,1,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":28,"replicas":[0,3,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":29,"replicas":[1,0,2],"log_dirs":["any","any","any"]}]}

3) envoyer ce flux dans un fichier
./kafka-reassign-partitions.sh --bootstrap-server master:9092 --execute --reassignment-json-file ~/partitions.json
Current partition replica assignment

{"version":1,"partitions":
[{"topic":"topic3","partition":0,"replicas":[2,3,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":1,"replicas":[3,2,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":2,"replicas":[1,3,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":3,"replicas":[2,3,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":4,"replicas":[3,1,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":5,"replicas":[1,2,3],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":6,"replicas":[2,1,3],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":7,"replicas":[3,2,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":8,"replicas":[1,3,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":9,"replicas":[2,3,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":10,"replicas":[0,3,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":11,"replicas":[1,0,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":12,"replicas":[2,3,0],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":13,"replicas":[3,0,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":14,"replicas":[0,1,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":15,"replicas":[1,0,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":16,"replicas":[2,3,0],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":17,"replicas":[3,0,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":18,"replicas":[0,1,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":19,"replicas":[1,2,3],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":20,"replicas":[2,0,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":21,"replicas":[3,1,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":22,"replicas":[0,2,3],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":23,"replicas":[1,3,0],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":24,"replicas":[2,1,3],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":25,"replicas":[3,2,0],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":26,"replicas":[0,3,1],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":27,"replicas":[1,0,2],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":28,"replicas":[2,3,0],"log_dirs":["any","any","any"]},{"topic":"topic3","partition":29,"replicas":[3,0,1],"log_dirs":["any","any","any"]}]}

Save this to use as the --reassignment-json-file option during rollback
Successfully started partition reassignments for
topic3-0,topic3-1,topic3-2,topic3-3,topic3-4,topic3-5,topic3-6,topic3-7,topic3-8,topic3-9,topic3-10,topic3-11,topic3-12,topic3-13,topic3-14,topic3-15,topic3-16,topic3-17,topic3-18,topic3-19,topic3-20,topic3-21,topic3-22,topic3-23,topic3-24,topic3-25,topic3-26,topic3-27,topic3-28,topic3-29

4) Verification pour supprimer le throttliing :
$ ./kafka-reassign-partitions.sh --bootstrap-server master:9092 --verify --reassignment-json-file ~/partitions.json

$ ./kafka-reassign-partitions.sh --bootstrap-server master:9092 --verify --reassignment-json-file ~/partitions.json
Status of partition reassignment:
Reassignment of partition topic3-0 is complete.
Reassignment of partition topic3-1 is complete.
Reassignment of partition topic3-2 is complete.
Reassignment of partition topic3-3 is complete.
Reassignment of partition topic3-4 is complete.
Reassignment of partition topic3-5 is complete.
Reassignment of partition topic3-6 is complete.
Reassignment of partition topic3-7 is complete.
Reassignment of partition topic3-8 is complete.
Reassignment of partition topic3-9 is complete.
Reassignment of partition topic3-10 is complete.
Reassignment of partition topic3-11 is complete.
Reassignment of partition topic3-12 is complete.
Reassignment of partition topic3-13 is complete.
Reassignment of partition topic3-14 is complete.
Reassignment of partition topic3-15 is complete.
Reassignment of partition topic3-16 is complete.
Reassignment of partition topic3-17 is complete.
Reassignment of partition topic3-18 is complete.
Reassignment of partition topic3-19 is complete.
Reassignment of partition topic3-20 is complete.
Reassignment of partition topic3-21 is complete.
Reassignment of partition topic3-22 is complete.
Reassignment of partition topic3-23 is complete.
Reassignment of partition topic3-24 is complete.
Reassignment of partition topic3-25 is complete.
Reassignment of partition topic3-26 is complete.
Reassignment of partition topic3-27 is complete.
Reassignment of partition topic3-28 is complete.
Reassignment of partition topic3-29 is complete.

Clearing broker-level throttles on brokers 0,1,2,3
Clearing topic-level throttles on topic topic3

./kafka-producer-perf-test.sh --topic topic3 --num-records 50000000 --record-size 100 --throughput -1 --producer-props acks=1 buffer.memory=67108864 batch.size=8196 bootstrap.servers=master:9092

./kafka-consumer-perf-test.sh --bootstrap-server master:9092 --messages 50000000 --topic topic3 --threads 1

Kafka Tools :
https://cwiki.apache.org/confluence/display/KAFKA/System+Tools

Exemple de code d'implémentation d'un comptage de mots avec Kafka Streams

```
import org.apache.kafka.common.serialization.Serdes;
import org.apache.kafka.common.utils.Bytes;
import org.apache.kafka.streams.KafkaStreams;
import org.apache.kafka.streams.StreamsBuilder;
import org.apache.kafka.streams.StreamsConfig;
import org.apache.kafka.streams.kstream.KStream;
import org.apache.kafka.streams.kstream.KTable;
import org.apache.kafka.streams.kstream.Materialized;
import org.apache.kafka.streams.kstream.Produced;
import org.apache.kafka.streams.state.KeyValueStore;

import java.util.Arrays;
import java.util.Properties;

public class WordCountApplication {

    public static void main(final String[] args) throws Exception {
        Properties props = new Properties();
        props.put(StreamsConfig.APPLICATION_ID_CONFIG, "wordcount-
application");
        props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka-
broker1:9092");
        props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG,
Serdes.String().getClass());
        props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG,
Serdes.String().getClass());

        StreamsBuilder builder = new StreamsBuilder();
        KStream<String, String> textLines = builder.stream("TextLinesTopic");
        KTable<String, Long> wordCounts = textLines
          .flatMapValues(textLine ->
Arrays.asList(textLine.toLowerCase().split("\\W+")))
           .groupBy((key, word) -> word)
           .count(Materialized.<String, Long, KeyValueStore<Bytes,
byte[]>>as("counts-store"));
        wordCounts.toStream().to("WordsWithCountsTopic",
Produced.with(Serdes.String(), Serdes.Long()));

        KafkaStreams streams = new KafkaStreams(builder.build(), props);
        streams.start();
    }

&nbs
```

- https://access.redhat.com/documentation/en-us/red_hat_amq/7.4/html/using_amq_streams_on_red_hat_enterprise_linux_rhel/overview-str