Devoxx France 2018

De la musique collaborative avec Kafka

@Im_flog

@vmaleze

@pboutes

Coucou c'est nous

fgarcia@ippon.fr

@Im_flog

- Backend lover
- Speaker

- Tracing distribué
- Kafka
- Docker

vmaleze@ippon.fr

@vmaleze

- Architecte
- Codeur fou

- Spring
- React
- DevOps

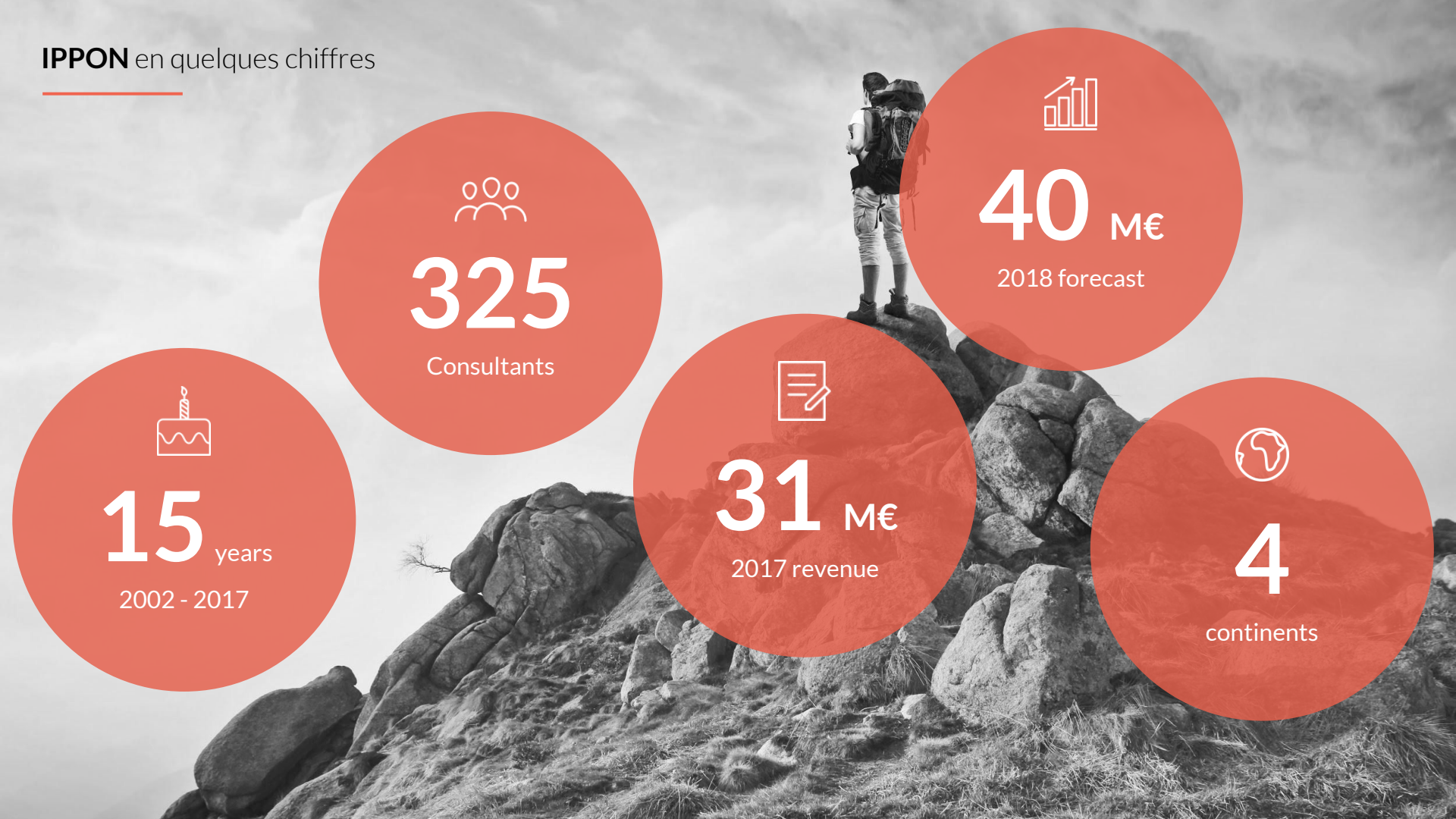
pboutes@ippon.fr

@pboutes

- Craftman
- Musicien

- Prog fonctionnelle
- Systèmes distribués
- Musique

IPPON en quelques chiffres



325
Consultants

40 M€
2018 forecast

15 years
2002 - 2017

31 M€
2017 revenue

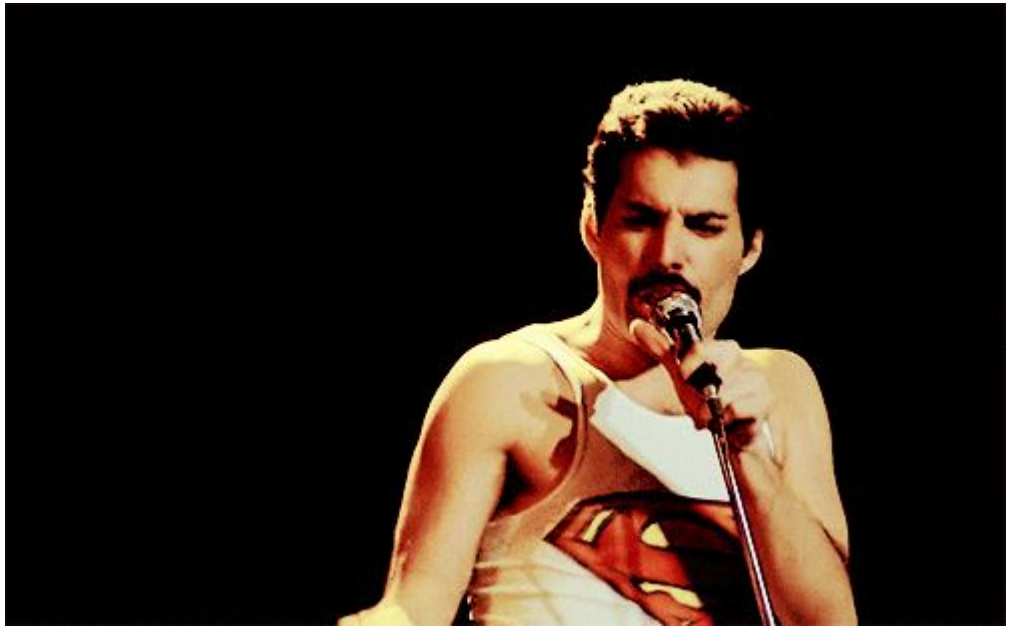
4
continents

3h c’est long

Posez des questions !

Qu’est ce qu’on va faire aujourd’hui ?!

- Quoi ?
- Comment ?
- Pourquoi ?



En détail

Consommation de tweets

Traitements intermédiaires

Analyses

Front pour la musique ♡

Les transformations

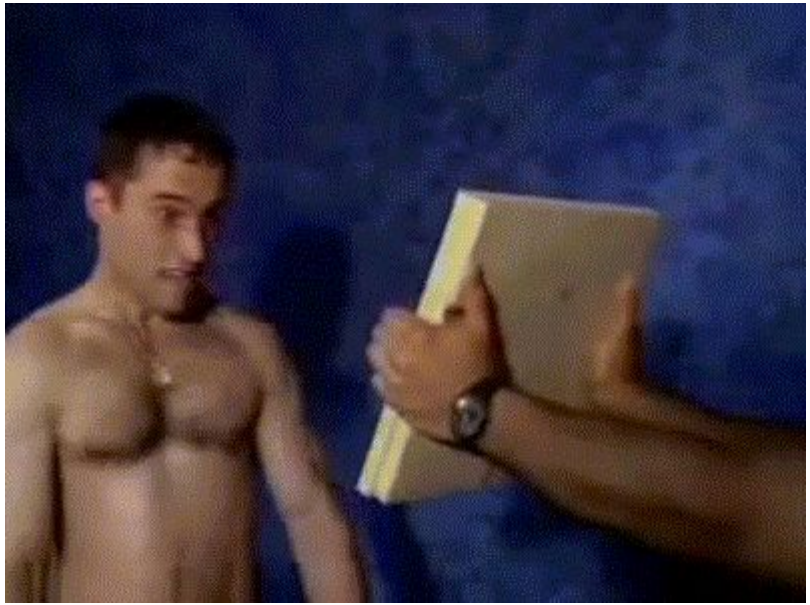
On veut:

- Un count de tweet par utilisateurs
- Le top 5 des tweets par fenêtres de 30s
- L’affichage du top 5 actuel

Bonus : La personne qui twittera le plus lors de la démo recevra un cadeau !

Problématiques

- Temps réel
- Volume de données
- Disponibilité
- Reprise sur erreur
- Multi-langages



Let’s GO !

Plan (plan)

- Architectures envisagées
- Fondamentaux
- Construction
- Party time
- Analyse
- La production

Architectures

The Glorious Monolith

READ + TRANSFORM + COUNT + ANALYTICS

DB

The Glorious Monolith

Facile à maintenir

Faible latence

Disponibilité

Simplicité

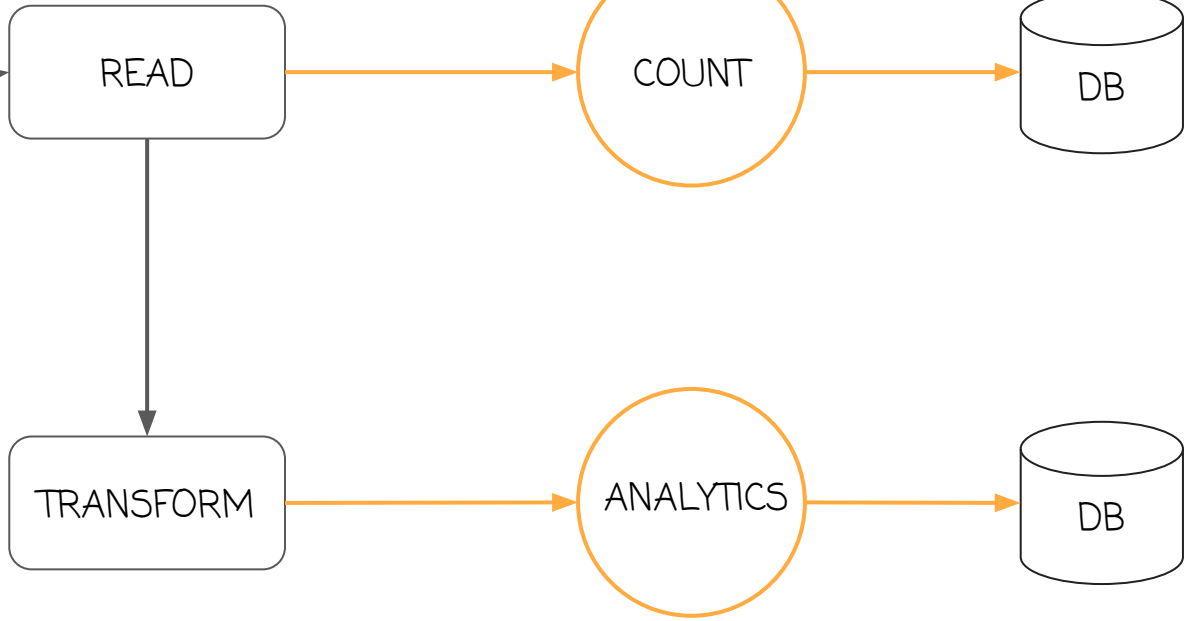
Scalabilité verticale

Polyglot tu ne seras pas

Reprise sur erreur

Maintenance (Panne réseau / Leadership)

The Microservices Myth



Microservices

Scalabilité

Polyglot

Disponibilité

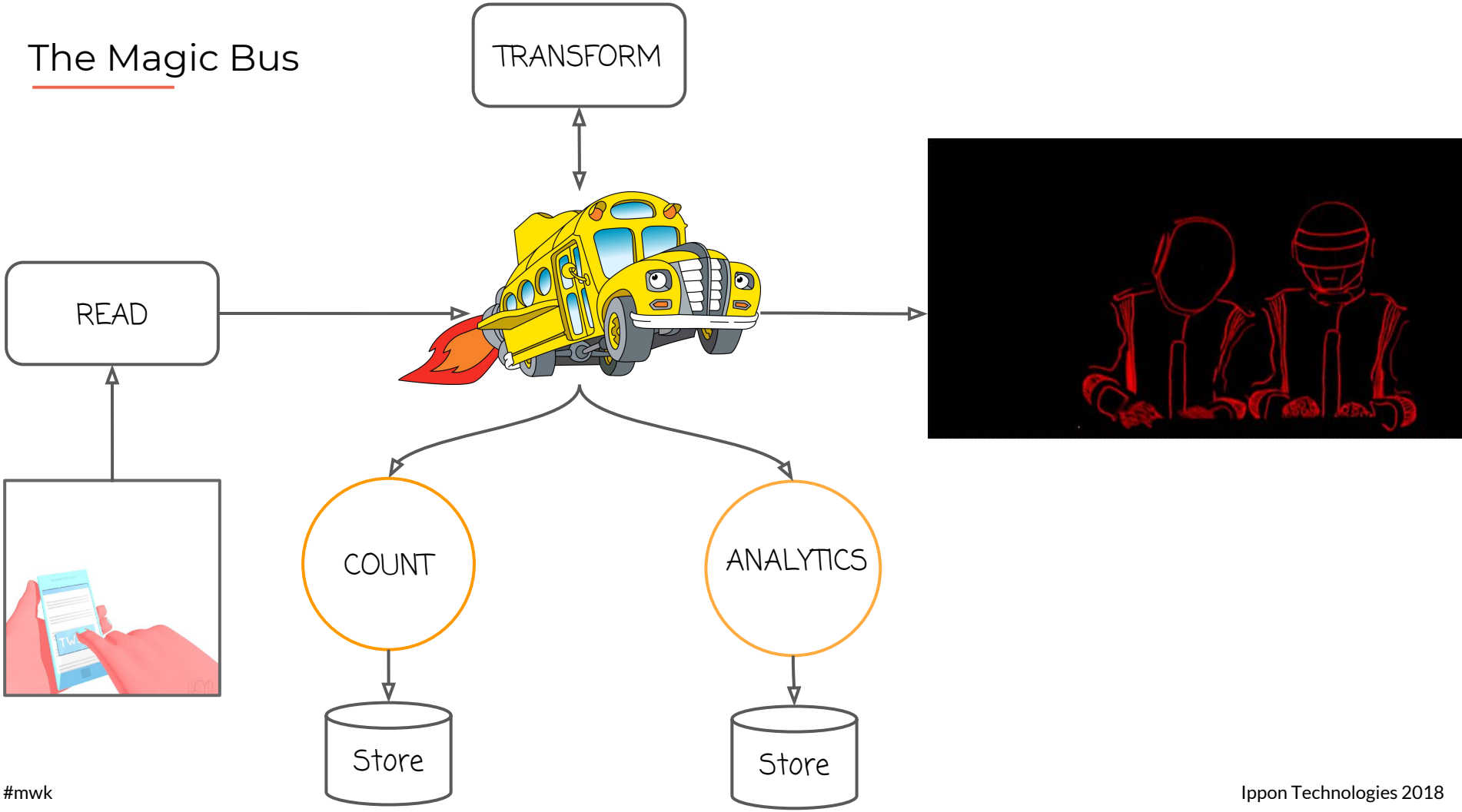
Reprise sur erreur

Latences réseau

Mises en place (Discovery / Infra)

Maintenance (Panne réseau / Leadership)

The Magic Bus



Bus

Scalabilité

Polyglot

Reprise sur erreur

Disponibilité

Découplage

Système distribué à gérer

Minimum de connaissance

Mais quel bus ?

	Kafka	RabbitMQ
Stockage	Longue durée	Consommation = suppression
Routage	Basique	Flexible
Multidiffusion	Par groupe de consommateurs	Matérialisé des échangeurs de type topic
Transaction	Oui	Non
Ordre	Par partition	Un consommateur par file
Performances	Très haute	Haute
Streaming	Oui (built-in)	Non (externe)
Ecosystème	Riche	Riche

Framework de Streaming ?

	Infra en plus ?	Performance	Accessibilité	Latence*
Kafka Streams	Non	Moyenne	Simple	Faible
Apache Flink	Oui	Haute	Moyenne	Faible
Spark Streaming	Oui	Haute	Complexe	Haute

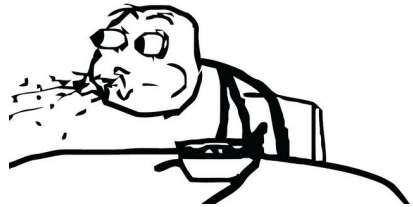
* http://blog.ippon.fr/2018/04/09/latence_kafka/

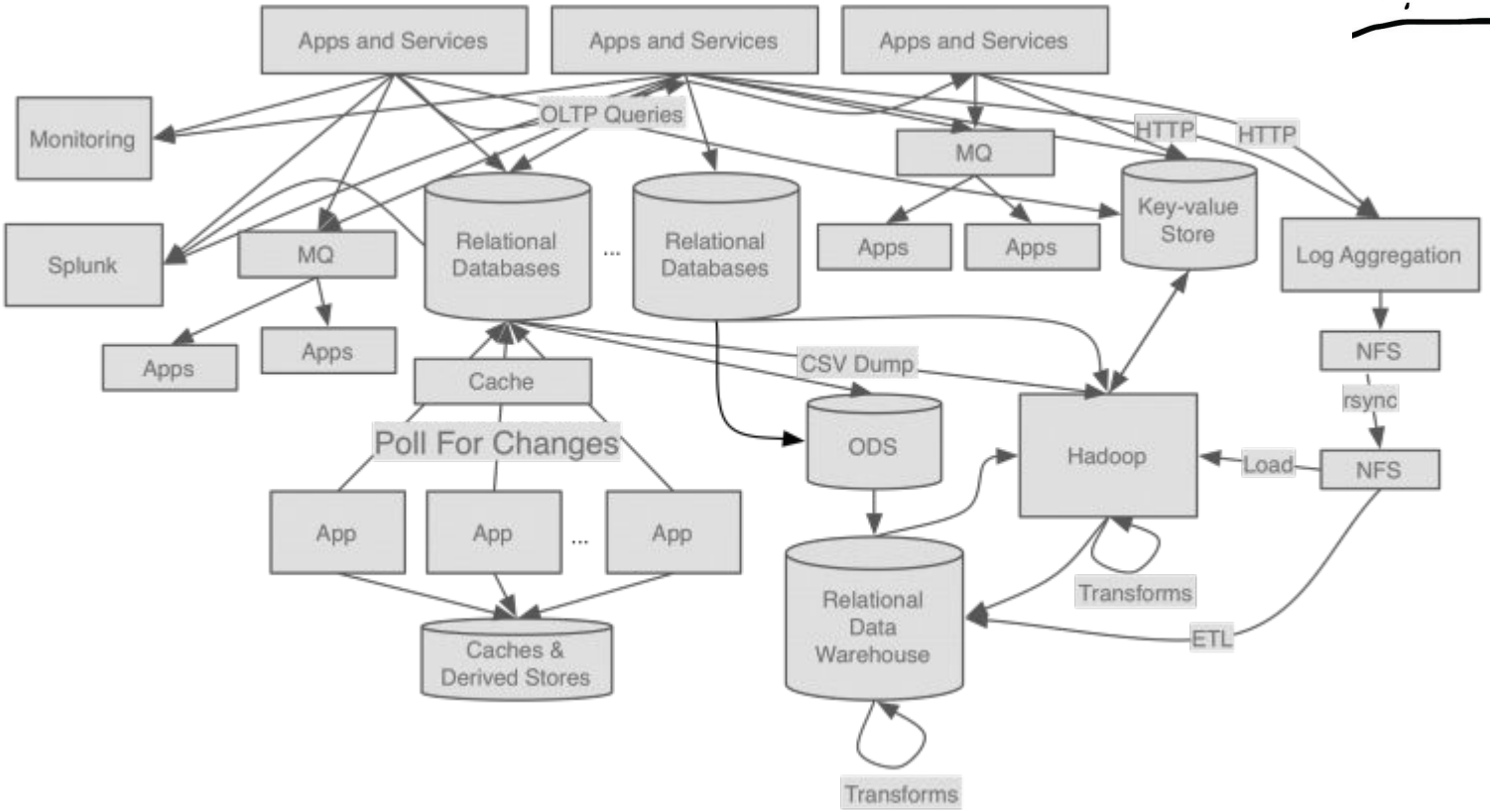
Et du coup ?

- On fait du kafka (duh !)
- Kafka Connect pour lire les tweets
- Kafka Stream pour traiter les données
- React pour le front
- KSQL pour l'analyse

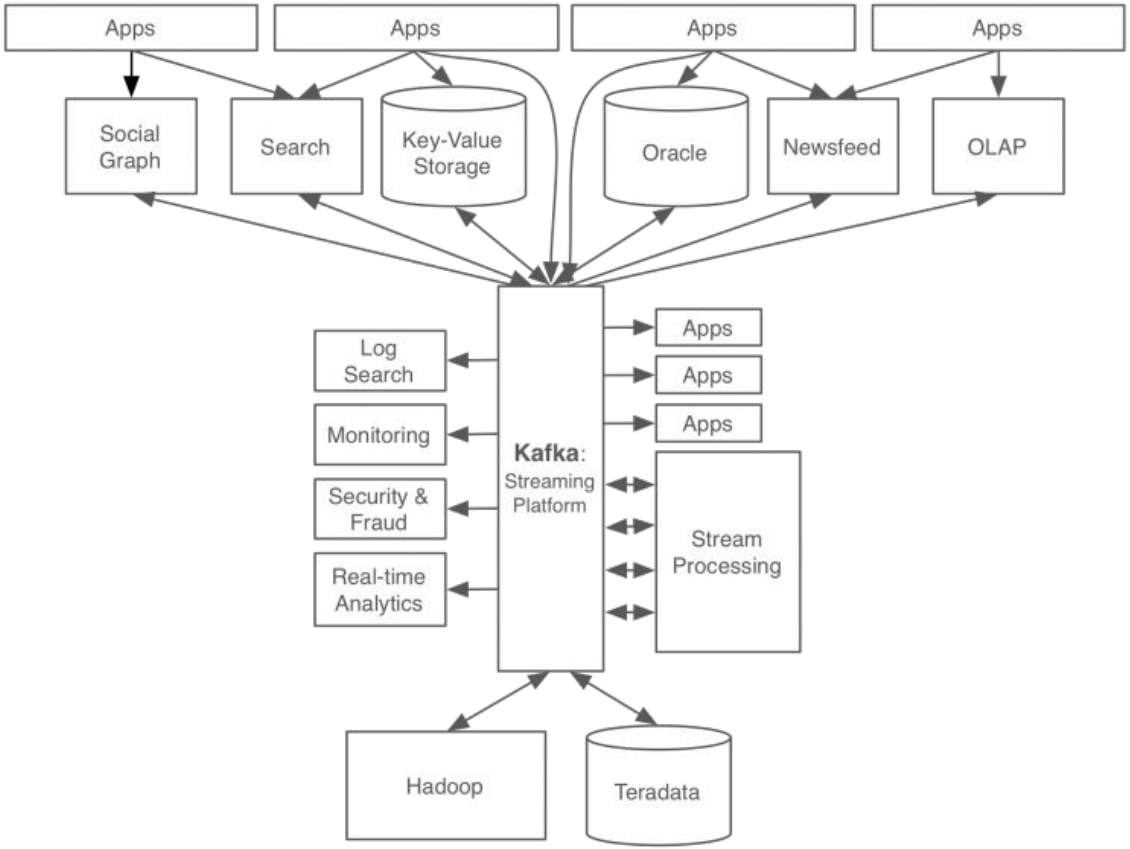
Fondamentaux Kafka

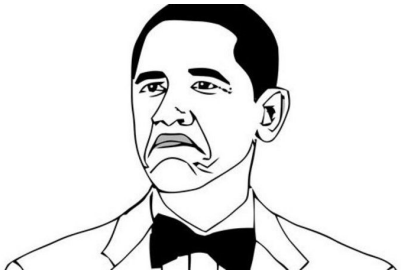
LinkedIn before Kafka





Linkedin after Kafka

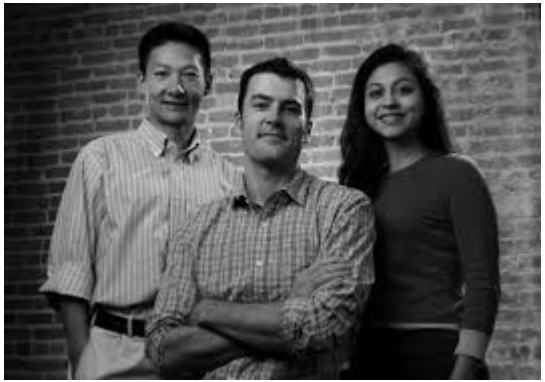


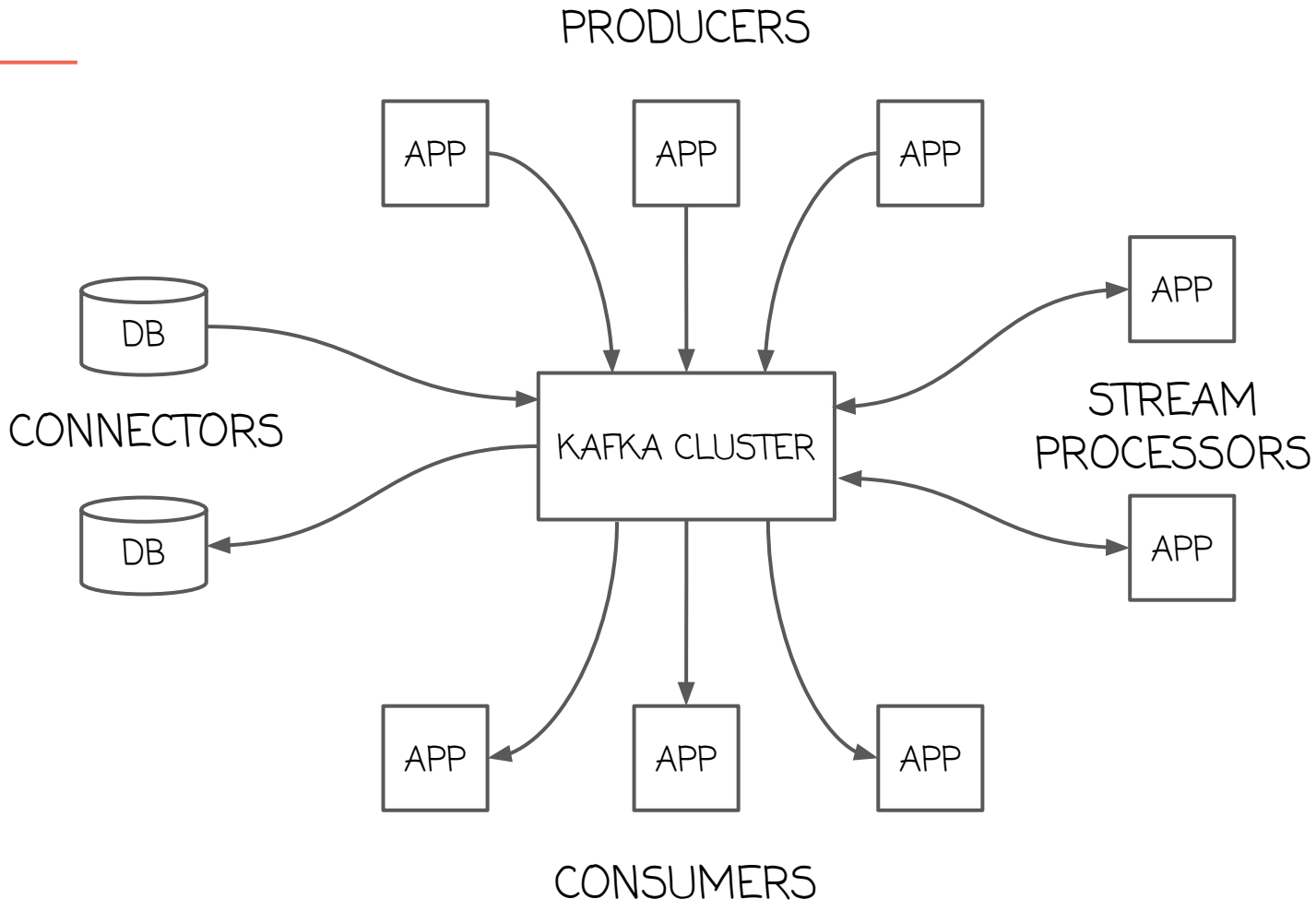


What about today ?

- Open source depuis 2011
- Soutenu par la fondation Apache
  - Très actif
  - Version actuelle 1.1.X

- Compagnie Confluent créée en 2014
  - Jay Kreps, Neda Narkhede et Jun Rao
  - Développe la plateforme Confluent
  - Ont levé plusieurs millions de dollars




PRODUCERS
APP
APP
APP
DB
CONNECTORS
KAFKA CLUSTER
APP
STREAM
PROCESSORS
DB
APP
APP
APP
APP
CONSUMERS

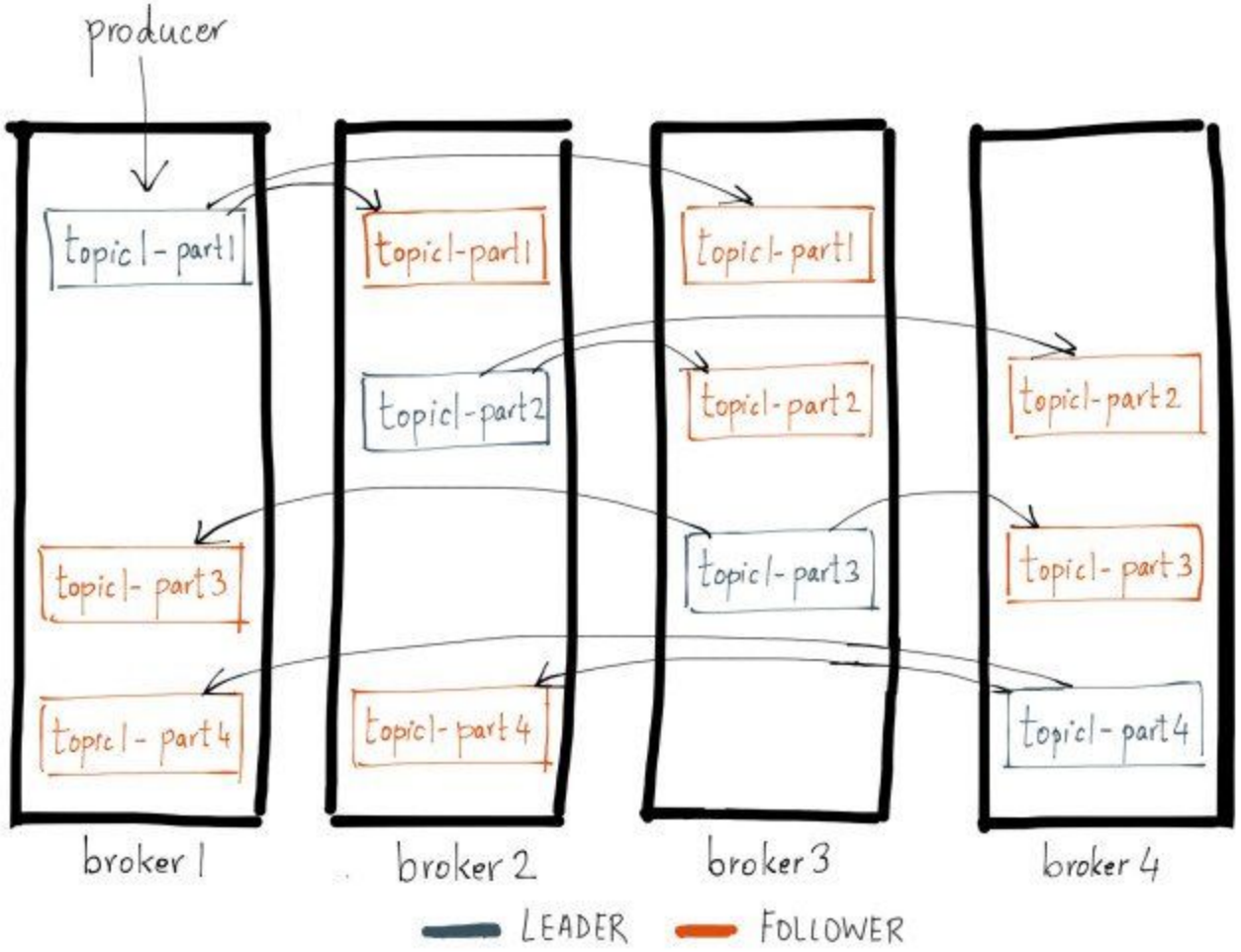
Broker

- Un service Kafka sur un serveur
- Rôle
  - Lecture / écriture sur disque
  - Interface consumer / producer
  - Group balancing / Leader election
- Utilise Zookeeper pour la coordination

Topic et partitions

- Équivalent d’une file
- Divisés en partitions
  - ➔ Répliqués
  - ➔ Distribués sur les brokers
  - ➔ 1 leader et N followers
  - ➔ Concept d’ISR pour l'élection de leader
- Durée de vie de message paramétré par topic (rétention)

Partitions

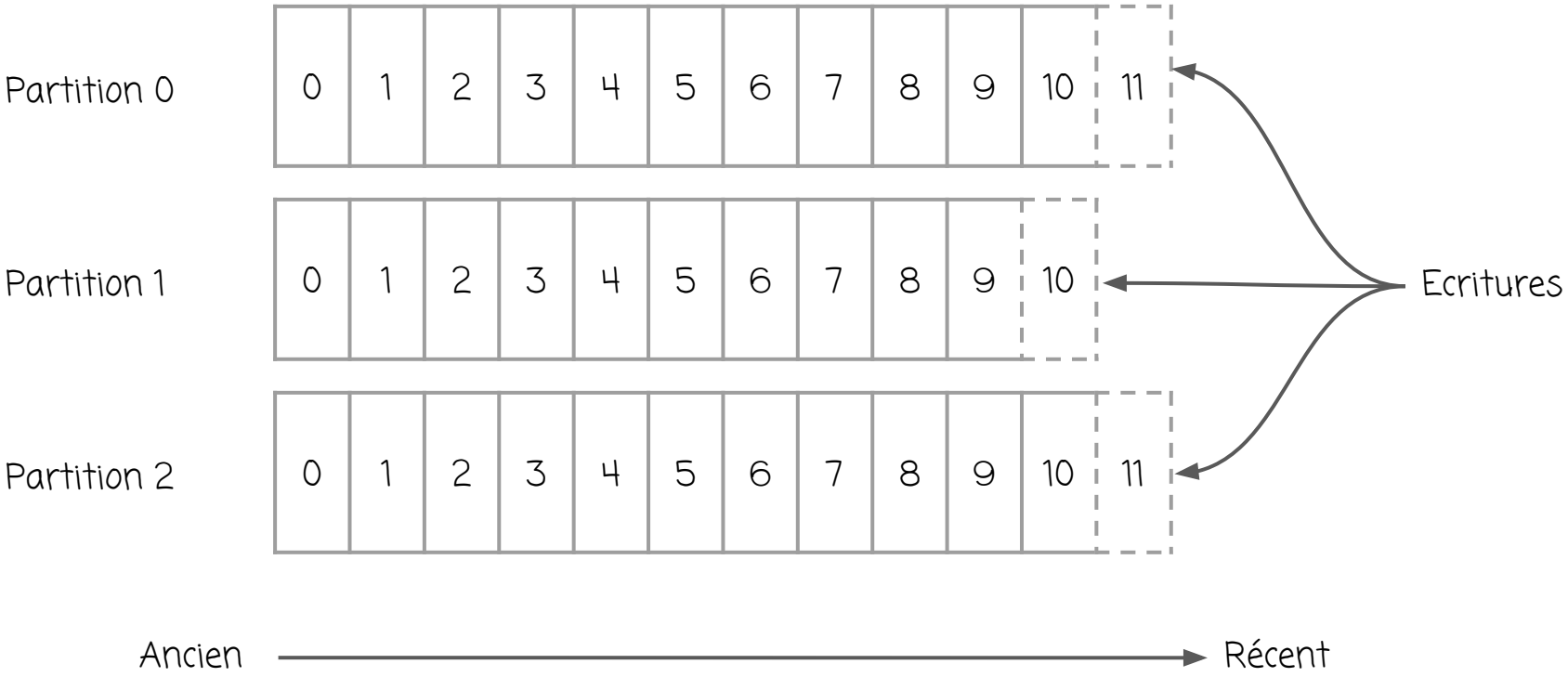

producer
topic1-part1
topic1-part2
topic1-part3
topic1-part4
topic1-part1
topic1-part2
topic1-part4
topic1-part1
topic1-part2
topic1-part3
topic1-part2
topic1-part3
topic1-part4
broker 1
broker 2
broker 3
broker 4
LEADER
FOLLOWER

Contenu des messages

- RecordBatch
  - FirstOffset & LastOffsetDelta
  - FirstTimestamp & MaxTimestamp
  - MagicByte = 2 (pour les versions > 0.11.X)
  - List<Record>

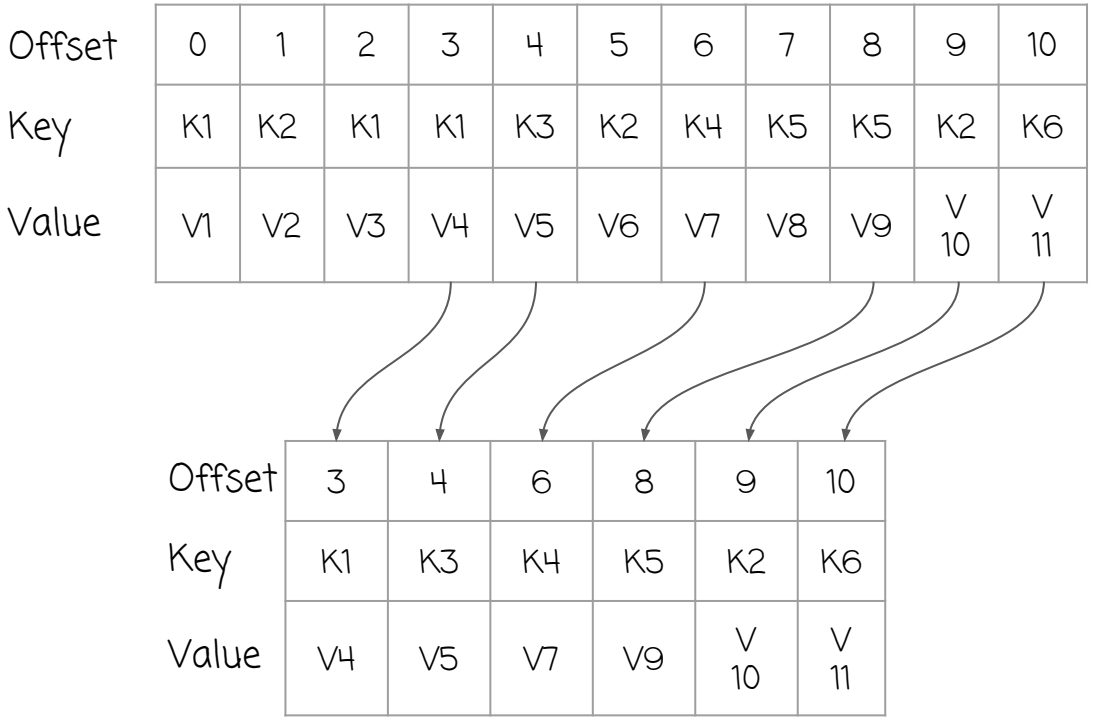
- Record
  - Timestamp & Offset deltas
  - Key
  - Value
  - Headers

Log distribué



Compaction

- Garder uniquement la dernière valeur pour une même clé
- Lors de la création d'un nouveau segment par le "log cleaner"



Les clients

Librairies bas niveau

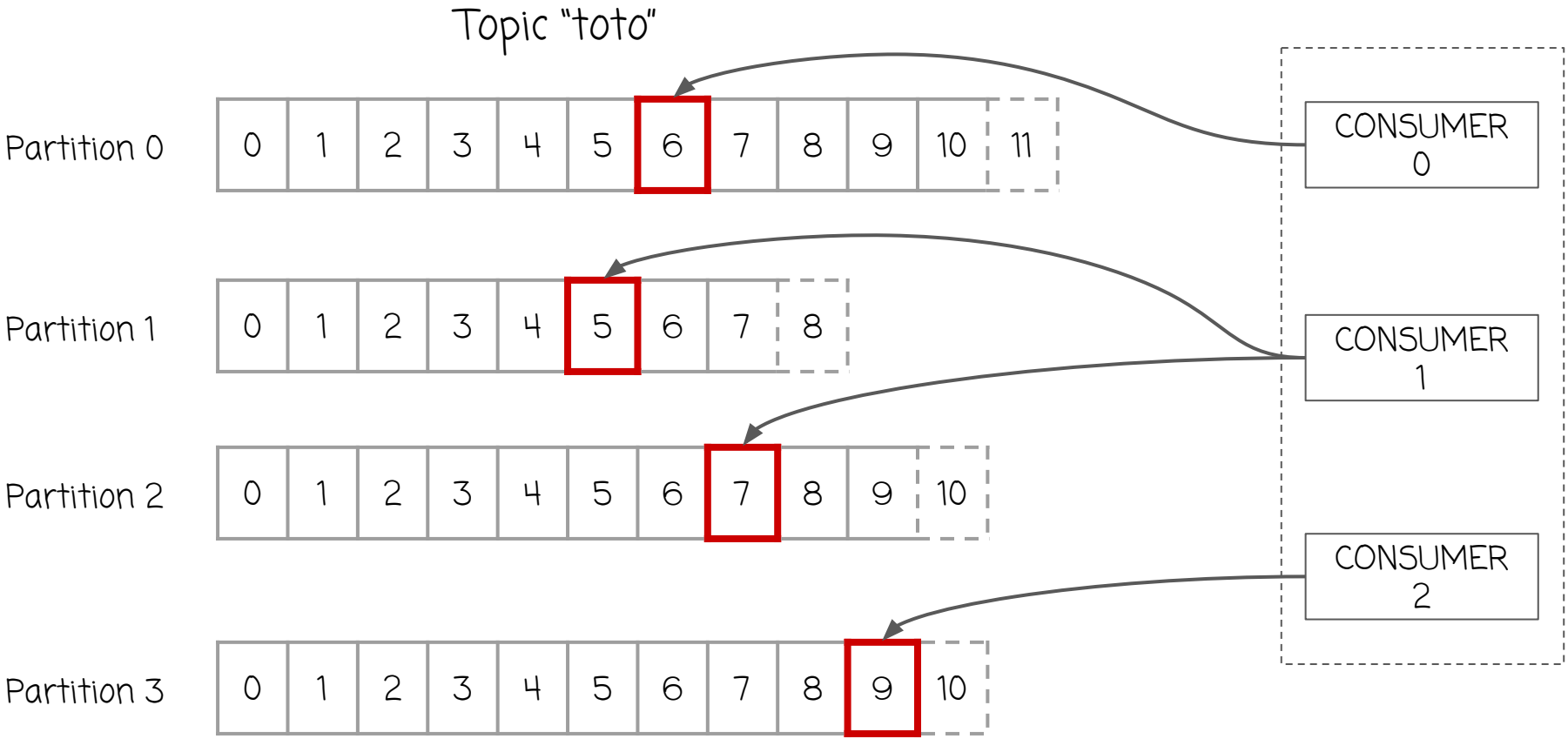
Utilisateurs de librd

[Liste complète]

Production

- Envoyer de la donnée dans Kafka
- Contenu binaire
- La clé définit la partition
- Asynchrone
  - Bufferise les records en attente
  - Batch les records pour efficacité
- Possibilités de retry
  - Modifie les garanties

Consommation



Quelles garanties ?

Understanding Streaming Semantics

At most once	At least once	Exactly once
Message pulled once	Message pulled one or more times; processed each time	Message pulled one or more times; processed once
May or may not be received	Receipt guaranteed	Receipt guaranteed
No duplicates	Likely duplicates	No duplicates
Possible missing data	No missing data	No missing data

5

MEMSQL

Exactly once, comment ?

- Idempotence (on ne peut pas écrire 2 fois le même message dans le log)
  - Utilisation d'un coordinateur
  - Unicité grâce à des champs dans la requête de production
    - ✓ pid
    - ✓ sequence_number
    - ✓ generation

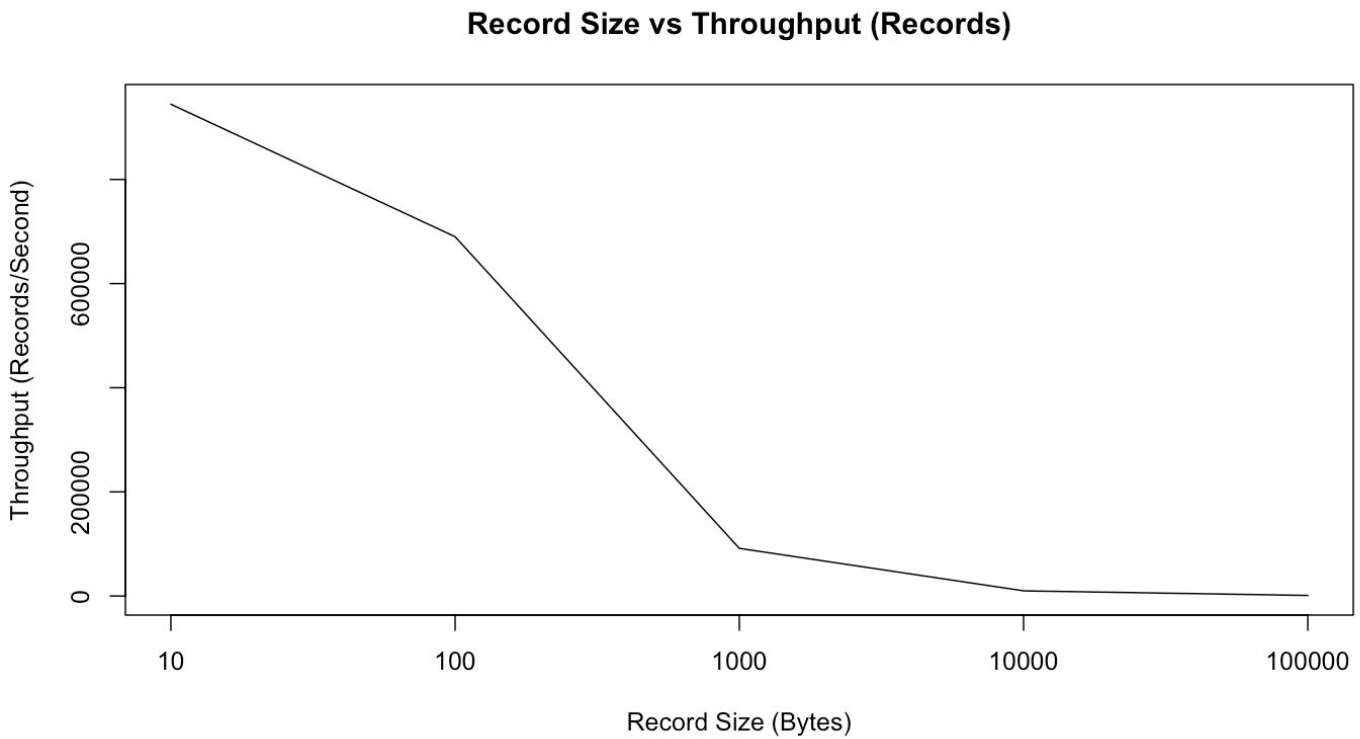
- Support des transactions
  - Ecriture dans plusieurs topics
  - Peut être utilisé avec les offsets pour garantir le traitement

- Choix du mode de lecture
  - read_commited
  - read_uncommited

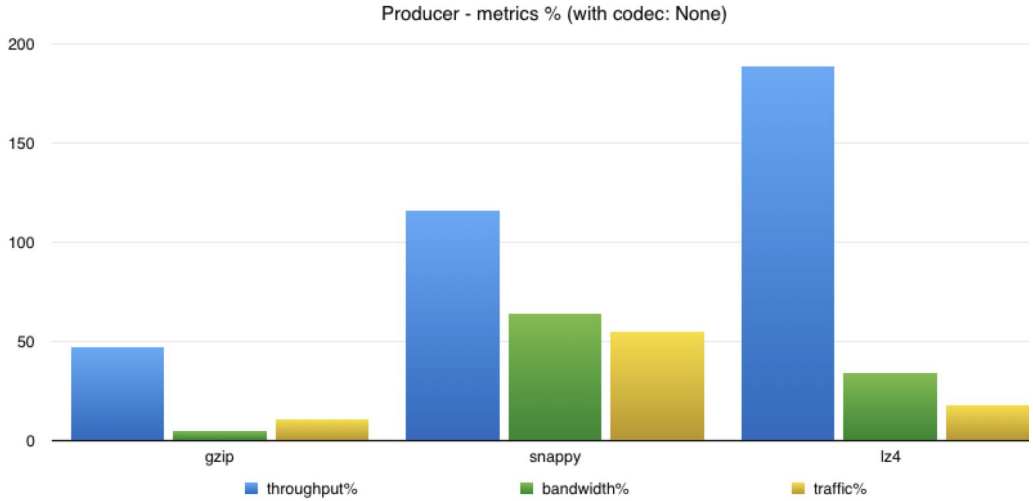
Plateforme Confluent

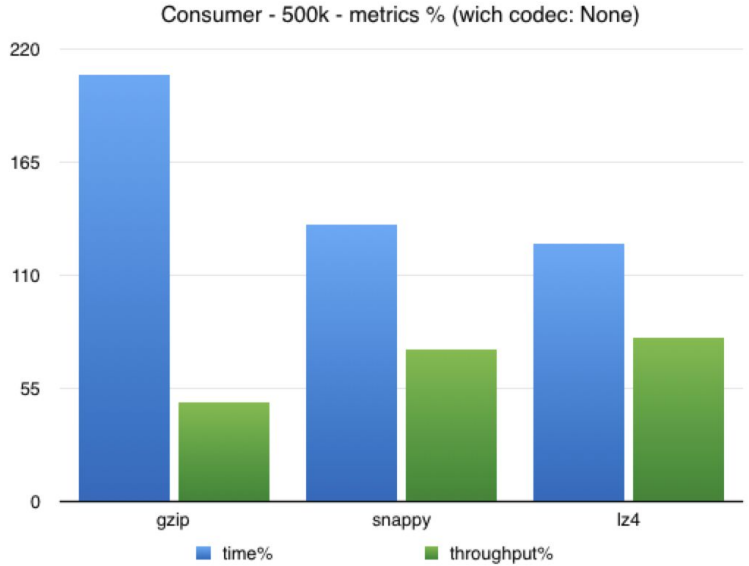
- Schema Registry
- REST Proxy
- Connector REST interface
- Additional connectors (€)
- Kafka replicator (€)
- Confluent control center (€)
- Support (€)
- ...

Performances



Compression





[Benchmark détaillé]

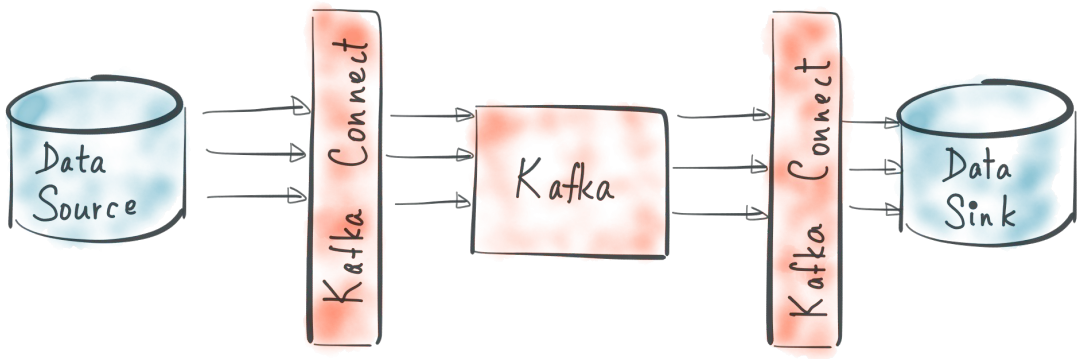
Kafka Connect

Point d'avancement

Consommation de tweets

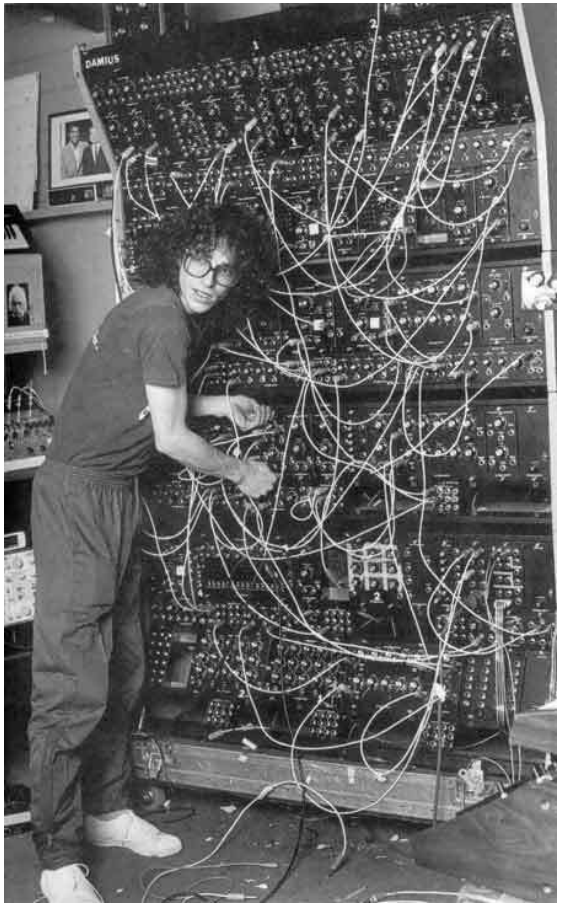
Grand angle


KAFKA CONNECT

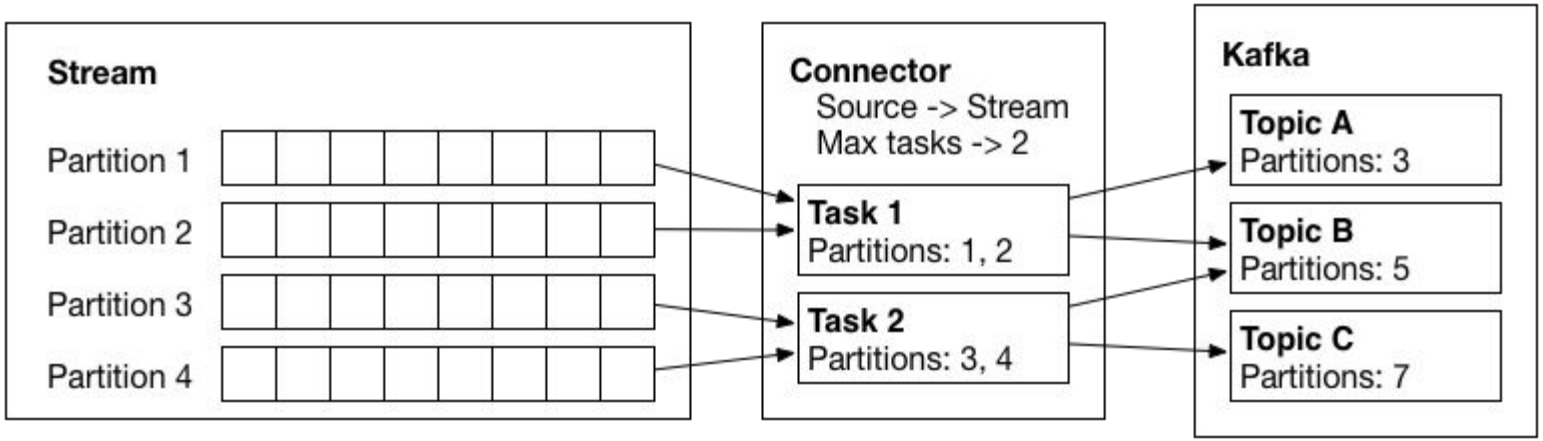

Data Source
Kafka Connect
Kafka
Kafka Connect
Data Sink

C’est quoi ?

- Intégration avec systèmes externes
  - Connecteurs communautaires
  - Connecteurs personnalisables
- Types de fonctionnement
  - Distribué
  - Standalone
- Offset géré de façon automatique
- Distribué et scalable “by design”



Connecteurs

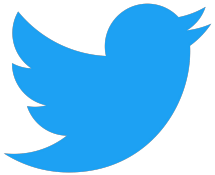


Build your own !

- SourceConnector / SinkConnector
  - Configuration de vos tâches
- SourceTask / SinkTask
  - Exemple de lecture d'un fichier

```java
@Override
public List<SourceRecord> poll() throws InterruptedException {
  try {
    ArrayList<SourceRecord> records = new ArrayList<>();
    while (streamValid(stream) && records.isEmpty()) {
      LineAndOffset line = readToNextLine(stream);
      if (line != null) {
        Map sourcePartition = Collections.singletonMap("filename", filename);
        Map sourceOffset = Collections.singletonMap("position", streamOffset);
        records.add(new SourceRecord(sourcePartition, sourceOffset, topic, Schema.STRING_SCHEMA, line));
      } else {
        Thread.sleep(1);
      }
    }
    return records;
  } catch (IOException e) {
    // Underlying stream was killed, probably as a result of calling stop. Allow to return
    // null, and driving thread will handle any shutdown if necessary.
  }
  return null;
}
```

Connecteurs



Liste complète

Démo Kafka Connect
Vérification consumer JAVA

https://github.com/ImFlog/music-with-kafka

Consommation Kotlin

```kotlin
val configuration = kafkaConfiguration()
val consumer: KafkaConsumer<String, String> = KafkaConsumer(configuration)
val topic = "twitter_json"
consumer.subscribe(setOf(topic))
while (running) {
  val records: ConsumerRecords<String, String> = consumer.poll(100)
  records.forEach { record →
      val message = record.value()
      println(message)
  }
}
```

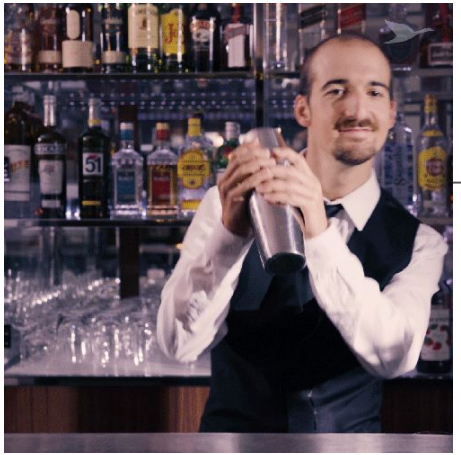
Production Kotlin

```kotlin
val configuration = kafkaConfiguration()
val producer: KafkaProducer<String, String> = KafkaProducer(configuration)
val topic = "twitter_json"
producer.send(new ProducerRecord<>(topic, "my_key", "my value"))
```

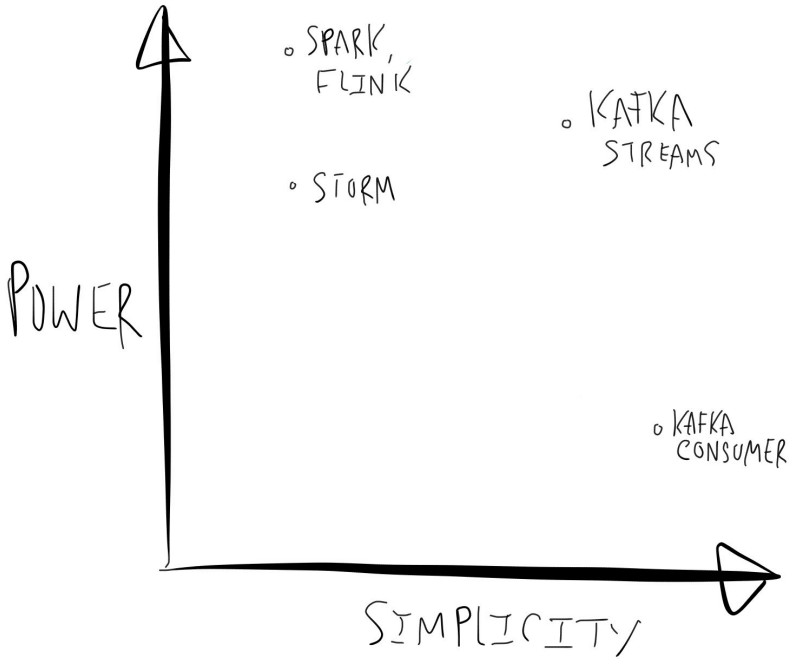
Kafka Streams

Point d’avancement

Traitements intermédiaires



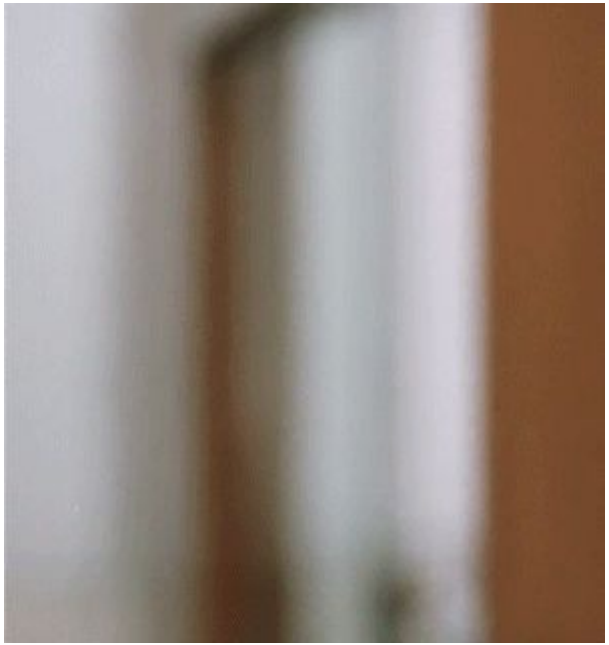
The goal



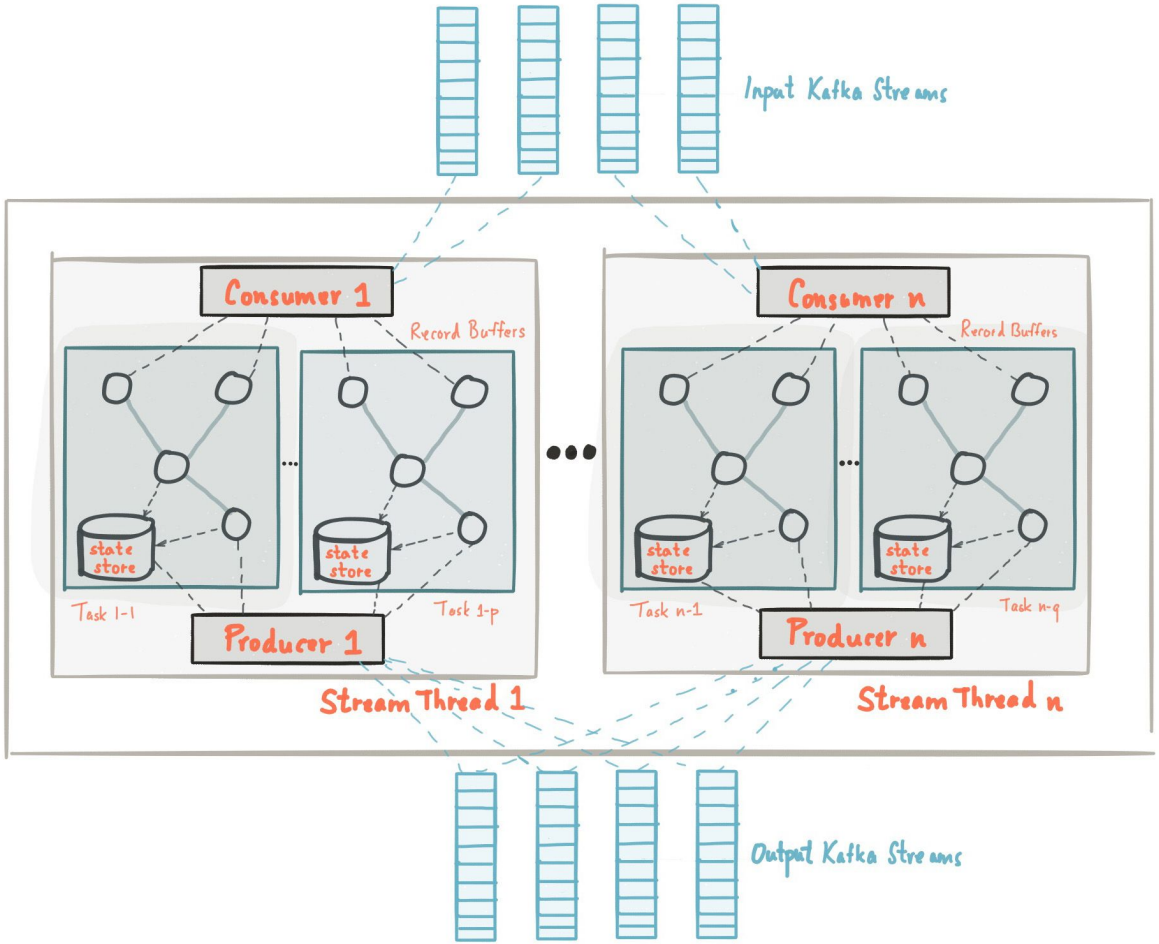
Source www.confluent.io

Kafka streams

- Disponible depuis la version 0.10.0
- Une librairie simple et légère
- Event-at-a-time processing
- Latence de l'ordre de la milliseconde
- Transformations "Stateless" & "Stateful"
- Garantie "Exactly once" !



Anatomie



Source: Kafka documentation

KStream

- Une abstraction d'un flux d'événements
  - Chaque "record" est une "insertion"
  - Pas de remplacement pour un "record" avec la même clé

- **Exemple**
  1. ("Devoxx", 1)
  2. ("Devoxx", 3)
  3. Sum("Devoxx") = ?
  4. Sum("Devoxx") = 4

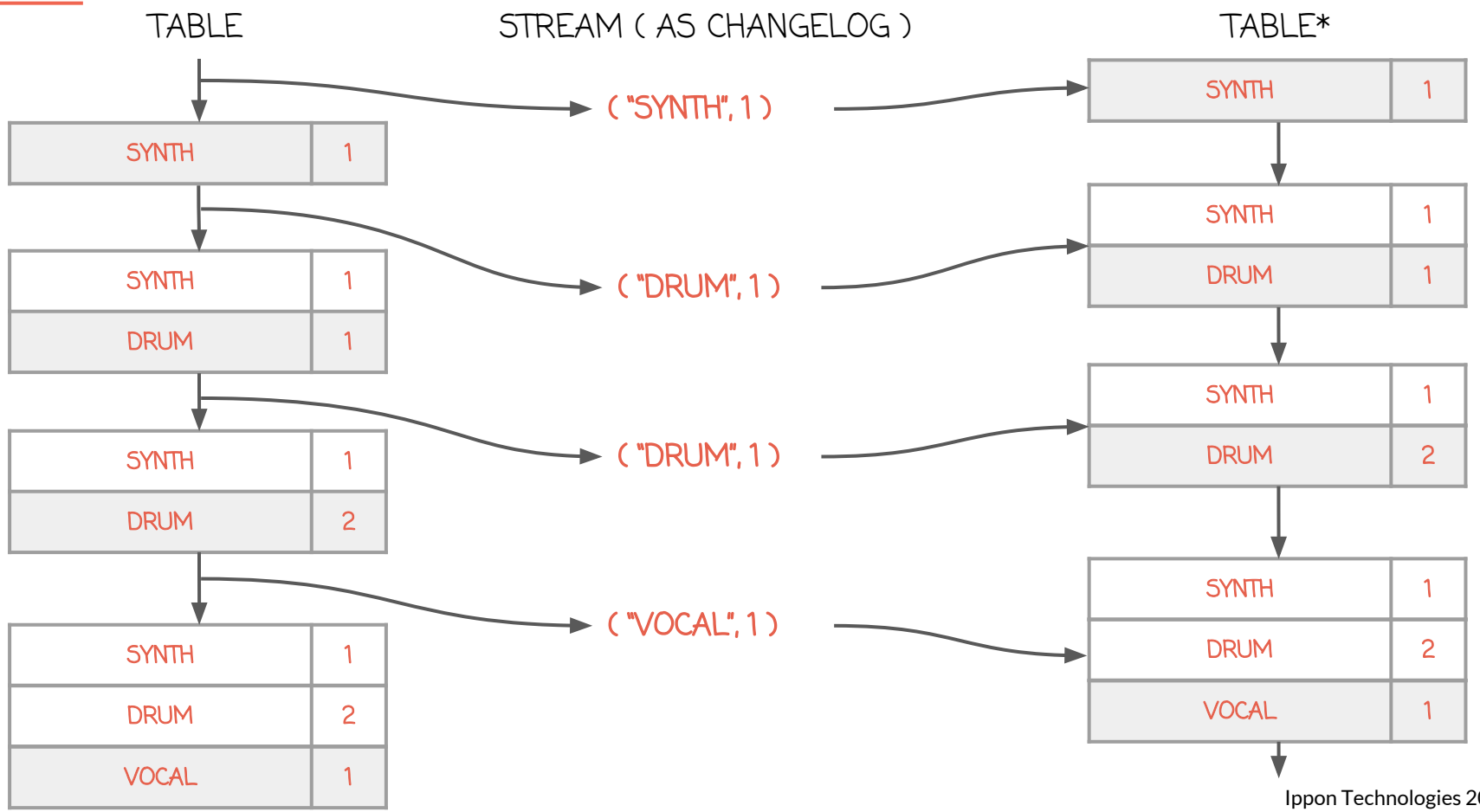
(Global) KTable

- KTable -> Abstraction d'un "changelog"
  - Chaque "record" est un "upsert"
  - Basé sur les clés

- Global KTable
  - Répliqué sur chaque consommateur

- **Exemple**
  1. ("Devoxx", 1)
  2. ("Devoxx", 3)
  3. Sum("Devoxx") = ?
  4. Sum("Devoxx") = 3

De la *KTable* au *KStream*



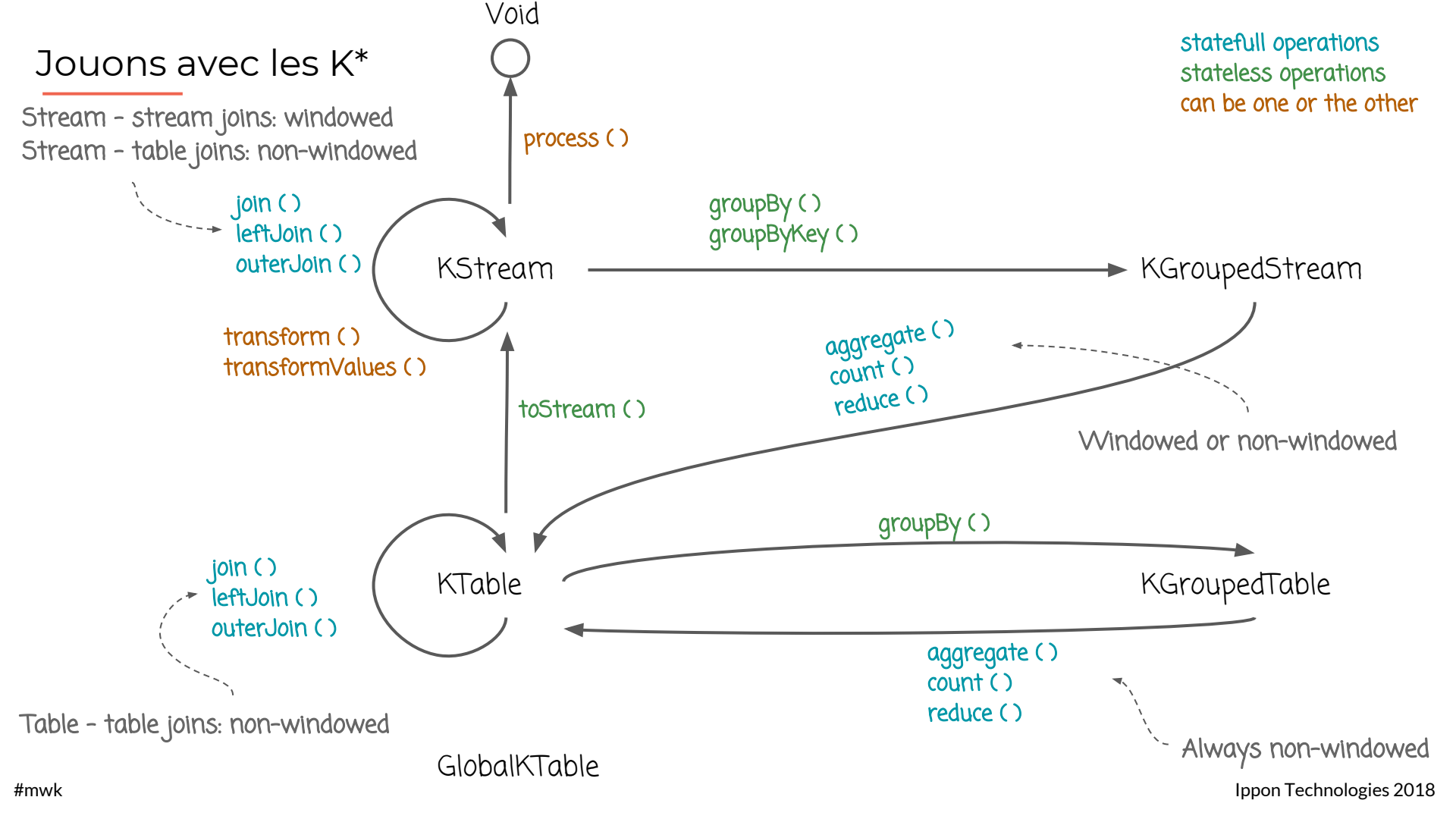
Pour les développeurs

- Java et Scala
- Une dépendance kafka-stream
- Une API haut niveau
  - Similaire aux lambdas Java 8
  - Map, filter, join, ...
- Une API bas niveau
  - Création de processeurs custom
  - Possibilité d'agir directement sur le stockage d'état

API haut niveau

- “Stateless”
  - map
  - groupBy
  - filter
- “Stateful”
  - Windowing
  - Joins
  - Aggregations
- “Terminal”
  - print
  - to

Jouons avec les K*



API bas niveau

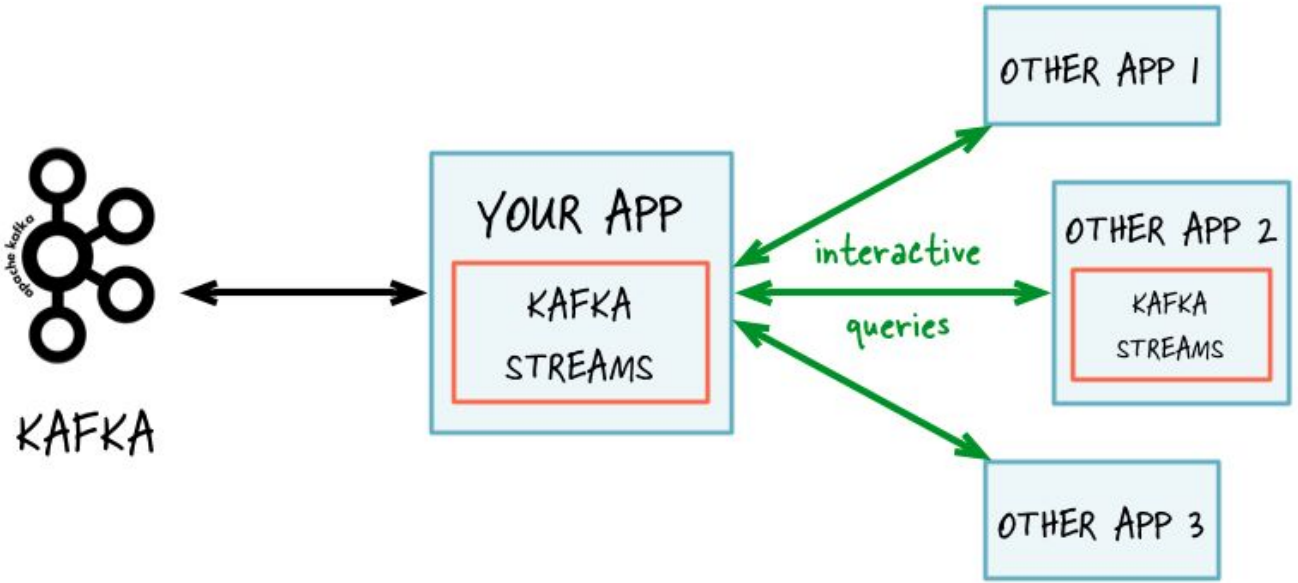
```java
public interface Processor<K, V> {

    void init(ProcessorContext context);

    void process(K key, V value);

    @Deprecated
    void punctuate(long timestamp);

    void close();
}
```

La magie du “state store”

- Stocker & requêter les données
- Automatique avec les opérations “stateful”
- Utilise “RocksDB” par défaut
- Etat global réparti entre les applications
- Résistant à la panne
    - L’état peut être reconstruit grâce à un topic spécifique
    - Compaction pour réduire l’empreinte

Interactive queries

1. Capture business events in Kafka
2. Process the events with Kafka Streams
3. With interactive queries, other apps can directly query the latest results



Démo Kafka Streams

https://github.com/ImFlog/music-with-kafka

La musique !

Point d’avancement

Front pour la musique ♡

Party time ! #mwk

https://github.com/ImFlog/music-with-kafka

Music With Kafka

drum

heavy_bass

lead_bass

line_bass

melody

pad

synth

vocal

Votre tweet doit contenir #mwk et un de ces mots :

*Ex : '**Donnez moi de la heavy_bass #mwk***

KSQL

Point d’avancement

Analyses

Késako ?

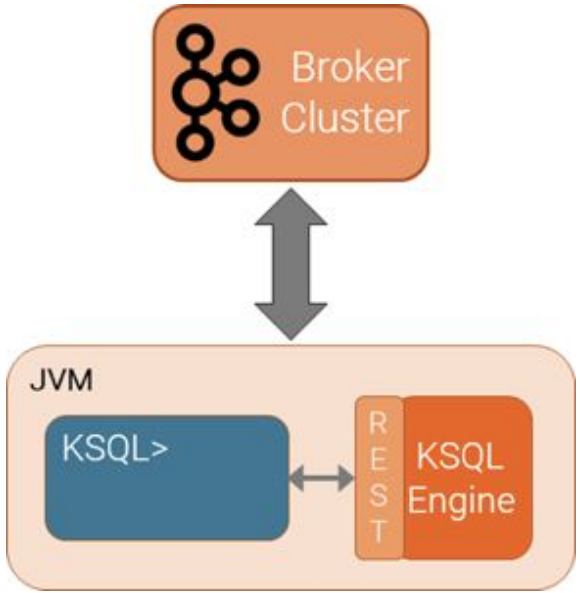
KSQL is an open source streaming SQL engine that implements continuous, interactive queries against Apache Kafka™.

- Basé sur SQL => Simple et intuitif
- Déploiement simplifié => Pas de jars ni binaires
- Très facile de transformer des données
- Basé sur Kafka et Kafka Streams (distribué, scalable, fiable et temps réel)

Standalone mode

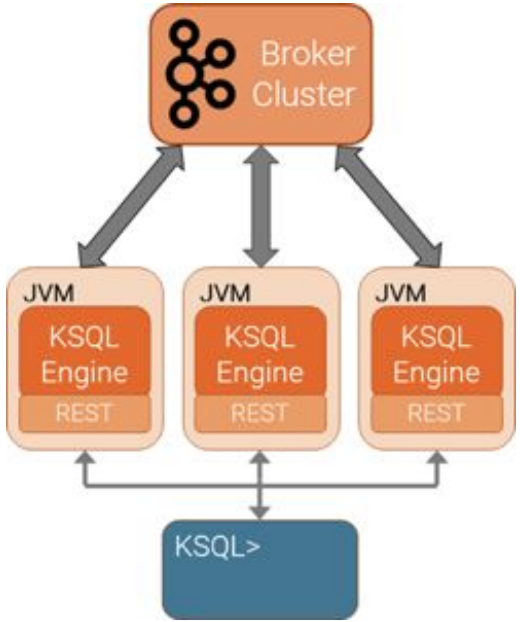
- 1 Client / 1 Serveur
  - Même machine
  - Même JVM
  - Démarré ensemble

- Meilleure solution pour développer et tester



Client-server mode

- n Clients / n Serveurs
  - Différentes machines (VM, Conteneurs, ...)
  - Connections via HTTP
  - Plusieurs pools de connections



Syntaxe

[CREATE | DROP] [STREAM | TABLE] stream_name

SELECT ⇔ continuously print selected stream|table columns

PRINT ⇔ print kafka topic

WHERE, HAVING ⇔ select columns

GROUP BY ⇔ group

AGGREGATIONS (COUNT, MAX, MIN, TOPK ...)

DESCRIBE

Et bien plus !

Encore en développement !

- GA dans la confluent platform Community
- Projet très actif !
- Mais avec des fonctionnalités manquantes
  - CASE support : https://github.com/confluentinc/ksql/issues/620
  - GROUP BY requis pour chaque AGG
  - Pas de ORDER BY
  - Table / Table join
- Pas si simple à prendre en main
- Très utile pour des cas simples, mais bien moins puissant que Kafka Streams.

Démo KSQL

https://github.com/ImFlog/music-with-kafka

En production !

Configuration de base

- Java 8 - 9
  - G1GC recommandé
  - 6-8 G heap
- RAM
  - Pour la JVM
  - Pour le cache linux
- OS tuning
  - Tuning TCP
  - Limite de descripteurs de fichiers > 100k
- Cluster Zookeeper **distinct**

Sizing

	Taille cluster	Memoire	CPU	Storage
Kafka Brokers	3+	24G+ (si petit) 64G+ (si large)	12 CPU+, Hyper threading	6+ x 1TB dedicated disks (or RAID10 / JBOD)
Zookeeper	3 (si petit) 5 (si large)	8G+ (si petit) 24G+ (si large)	2+ core	SSD for Transaction logs

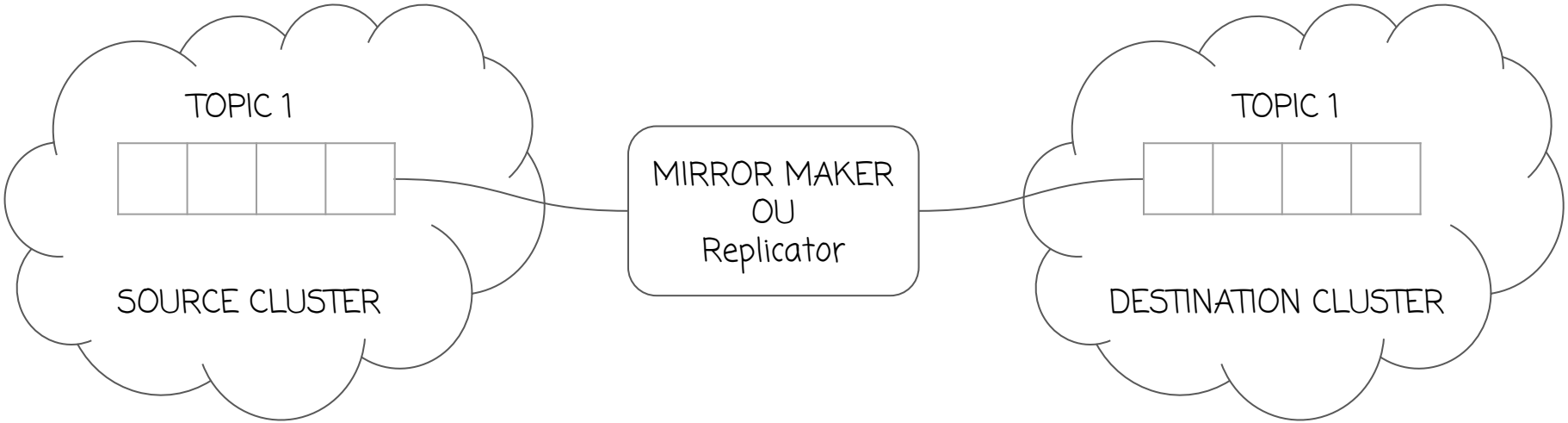
Topics avec plus de partitions :

- Plus grand débit
- Nécessite plus de descripteurs de fichiers
- Peut augmenter l'indisponibilité en cas de rebalance
- Peut augmenter la latence "End to End"
- Peut nécessiter plus de mémoire côté client

Broker settings

- num.partitions
- default.replication.factor
- log.retention.ms
- min.insync.replicas
- unclean.leader.election.enable
- num.[io|network].threads
- security (SSL / SASL)
- And many more

Cross Datacenter



https://fr.slideshare.net/gwenshap/multidatacenter-kafka-strata-san-jose-2017

Monitoring

Monitor Kafka Like a Pro

Conference

Architecture, Performance et Securité — Intermédiaire

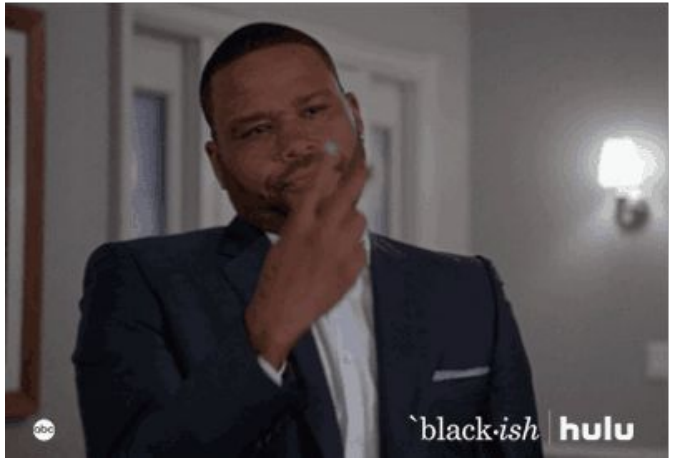
Thursday 12:55 - 13:40

Every business has a central nervous system through which all information flows and around which all decisions are made. Sometimes this system is ad-hoc and non-standard, resulting in an architecture that is difficult to reason about and even harder to keep running.

Kafka operators need to provide guarantees to the business that Kafka is working properly and delivering data in real time, and they need to identify and triage problems so they can solve them before end users notice them. This elevates the importance of Kafka monitoring from a nice-to-have to an operational necessity.

In this talk, Kafka operations experts Xavier Léauté and Gwen Shapira share their best practices for monitoring Kafka and the streams of events flowing through it. How to detect duplicates, catch buggy clients, and triage performance issues — in short, how to keep the business's central nervous system healthy and humming along, all like a Kafka pro.

Kafka | monitoring | Performance tuning



Outils

- Monitoring:
  - Confluent Control Center (€)
  - SPM (€)
  - Burrow
  - Kafka Topic UI
  - JMX

- Management
  - Kafka Manager
  - Kafkat

En bref

Recap

Consommation de tweets

Traitements intermédiaires

Analyses

Front pour la musique ♡

Retour


Simplicité


Polyglot


Mise en place


Rendu musical


Hype

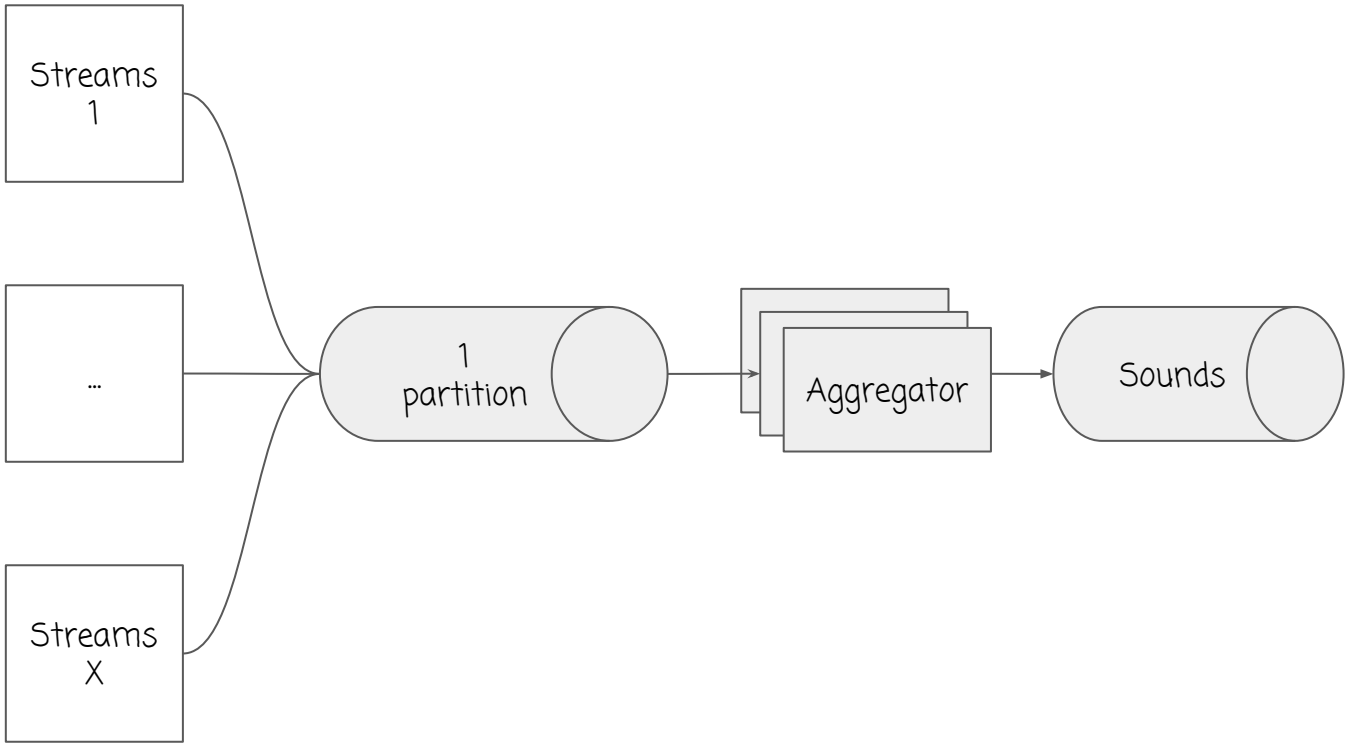

Cas d’utilisation un peu alambiqué


/!\ Mise en place à l’échelle


Overkill pour notre cas


Lien fort avec twitter

Potentiel passage à l'échelle



10 commandements pour aimer Kafka

1. Avec Zookeeper gentil tu seras
2. Le “Exactly Once” à bon escient tu utiliseras
3. De gros messages tu n’enverras pas
4. Tuner tes producer & consumer tu devras
5. Tes topics tu dimensionneras
6. Compresser si besoin tu feras
7. Compacter tu considéreras
8. Surréagir tu éviteras
9. Le streaming tu utiliseras
10. Une solution miracle ce n’est pas

Ressources

- https://github.com/ImFlog/music-with-kafka
- https://docs.confluent.io/
- http://kafka.apache.org
- https://github.com/confluentinc/ksql
- http://www.madeon.fr/adventuremachine/
- http://blog.yaorenjie.com/2017/01/03/Kafka-0-10-Compression-Benchmark/
- https://fr.slideshare.net/HadoopSummit/apache-kafka-best-practices
- https://www.confluent.io/blog/how-to-choose-the-number-of-topicspartitions-in-a-kafka-cluster
- https://www.confluent.io/blog/using-ksql-to-analyse-query-and-transform-data-in-kafka

PARIS
BORDEAUX
NANTES
LYON
MARRAKECH
WASHINGTON DC
NEW-YORK
RICHMOND
MELBOURNE

contact@ippon.fr
www.ippon.fr - www.ippon-hosting.com - www.ippon-digital.fr
@ippontech
-
01 46 12 48 48