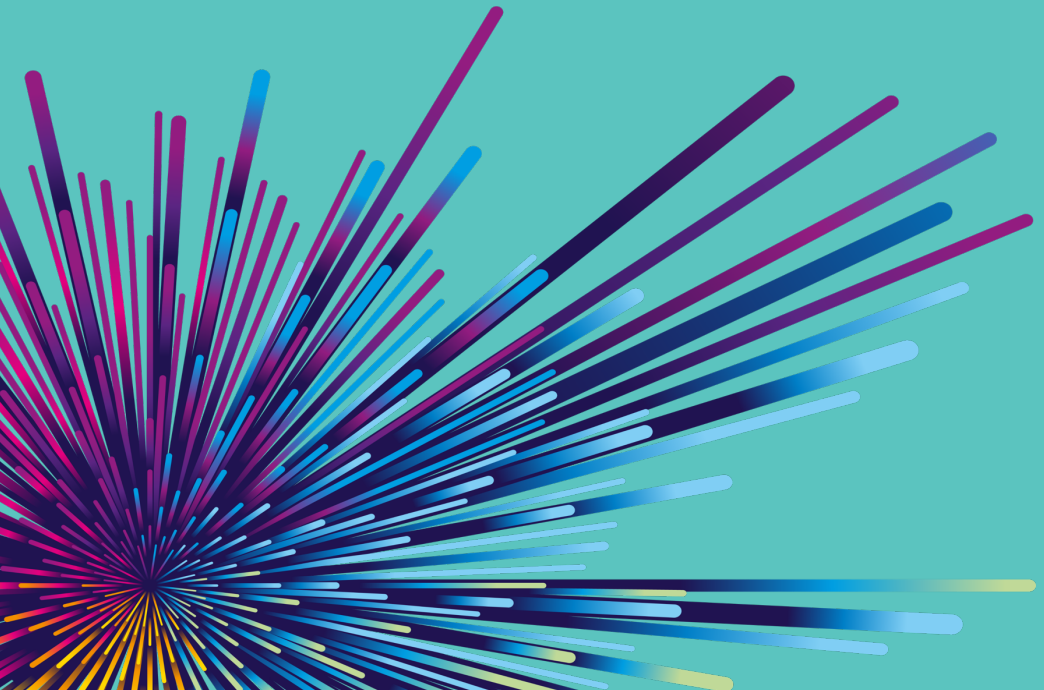
La vie est un stream d'évènements

Emmanuel, Péru

CTO

Ineat Group



Éditer le profil

Emmanuel Péru

@EmmanuelPeru

Papa de princesses, leader d'une équipe de pépites, curieux de tout 👯💻📱
🍺🥃🎥🍰🔨 coanimateur du compte @ineatlab CTO @ineatgroup

Billy-Berclau, France emmanuelperu.fr Naissance le 25 juillet 1984
A rejoint Twitter en avril 2012

#Euratech10

euratechnologies
EXCELLENCE & INNOVATION

MEL MÉTROPOLE EUROPÉENNE DE LILLE

Région Hauts-de-France

ville de lille

Stream de la prez

- Stream me I’m famous
- Stacks techniques
  - Cloud streaming
  - Home streaming
- Google DataFlow focus
- Apache Kafka Focus

Stream

« […] *technique consistant à **transmettre** des données à un utilisateur et à les lui rendre **disponible quasiment aussitôt** (c'est-à-dire pratiquement en **direct**) pour ainsi lui **éviter l'attente** du téléchargement et regarder l'ensemble des données (rassemblées dans un fichier). »*

- *Data Source (unbounded)*
- *Data Sink*
- *Event streaming*
- *Data streaming*
- *Stream processing*

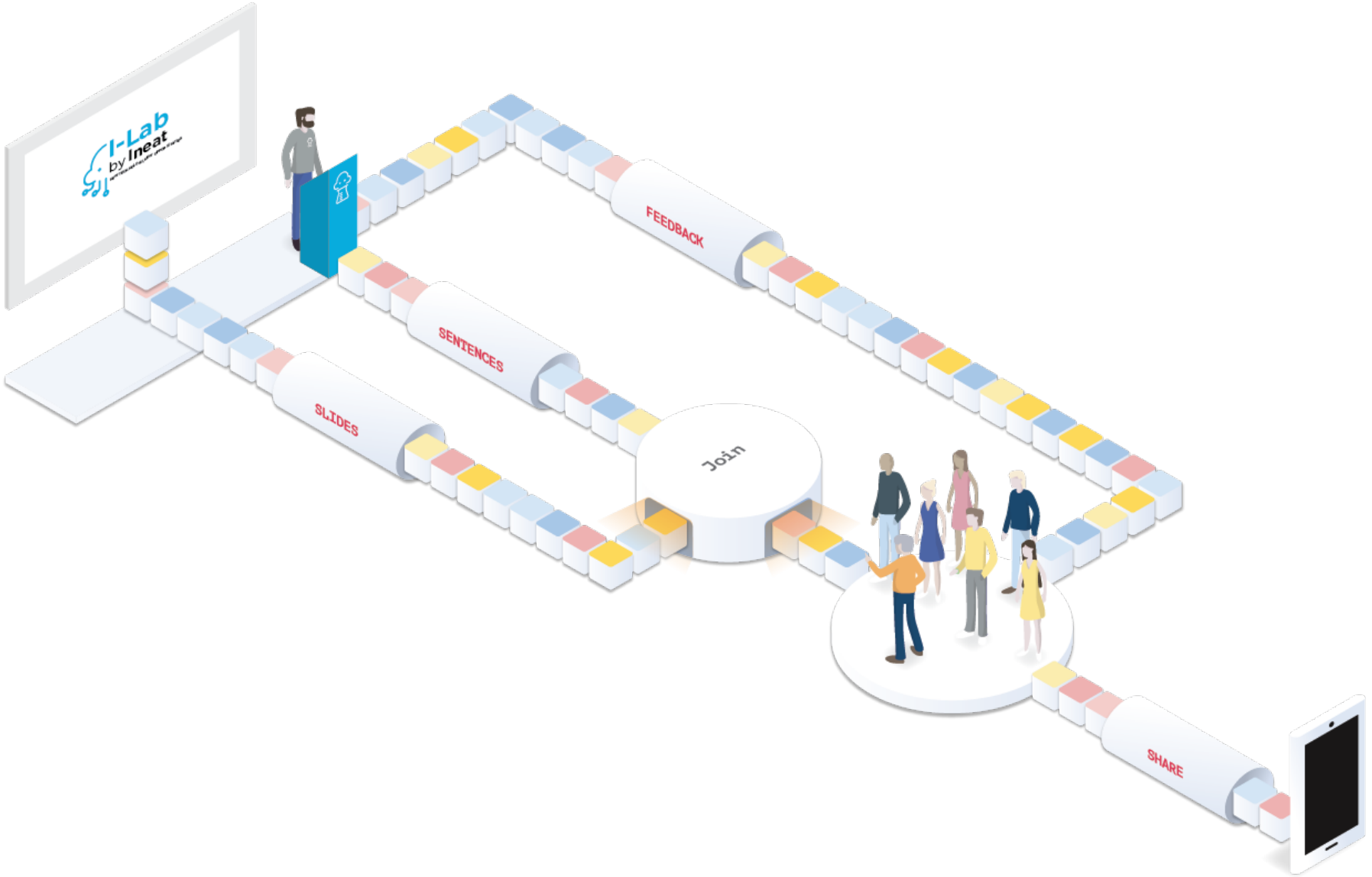
Sortir d’une logique de batch

Alex Kunchevsky

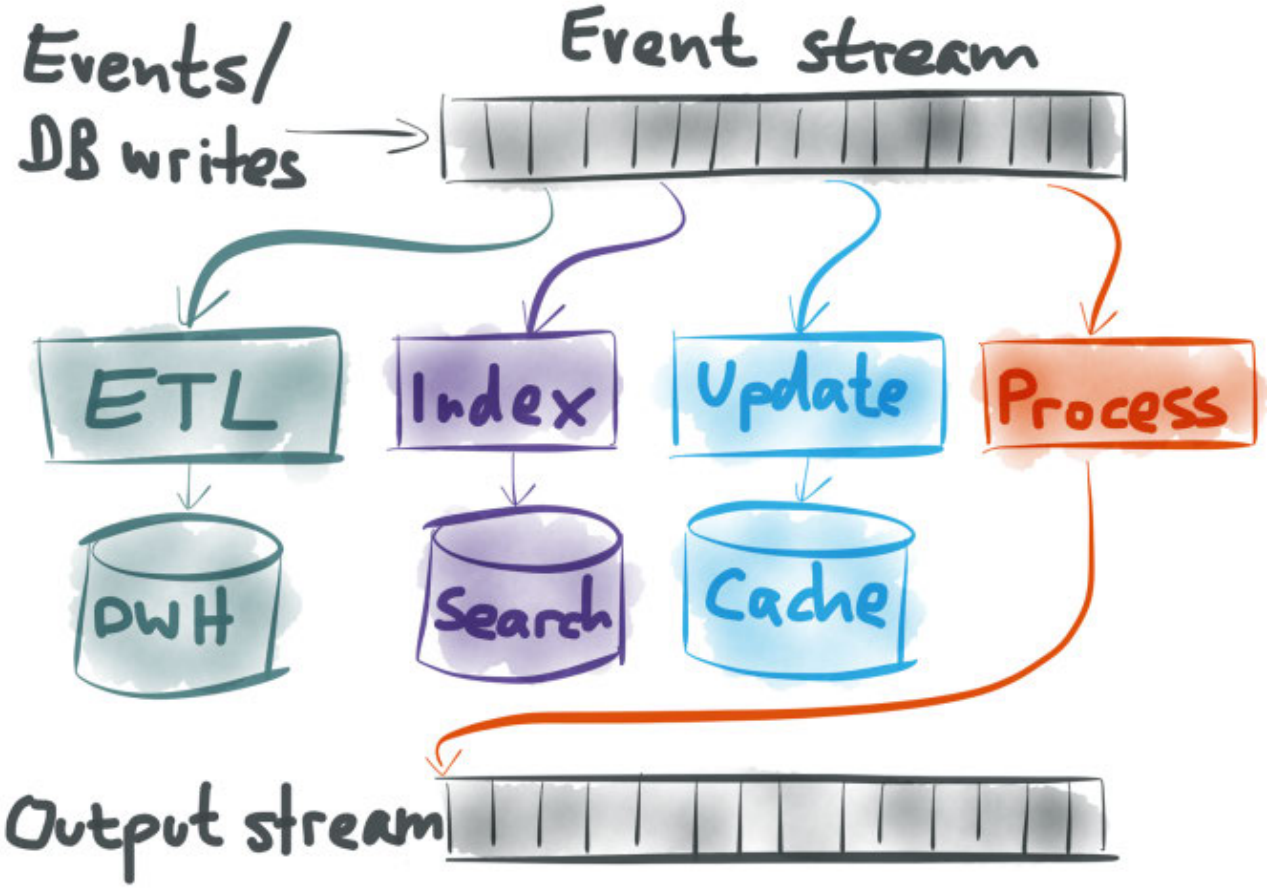
~~VS~~

David Papanikolau

La vie est une suite d’évènements



Big Picture Technique

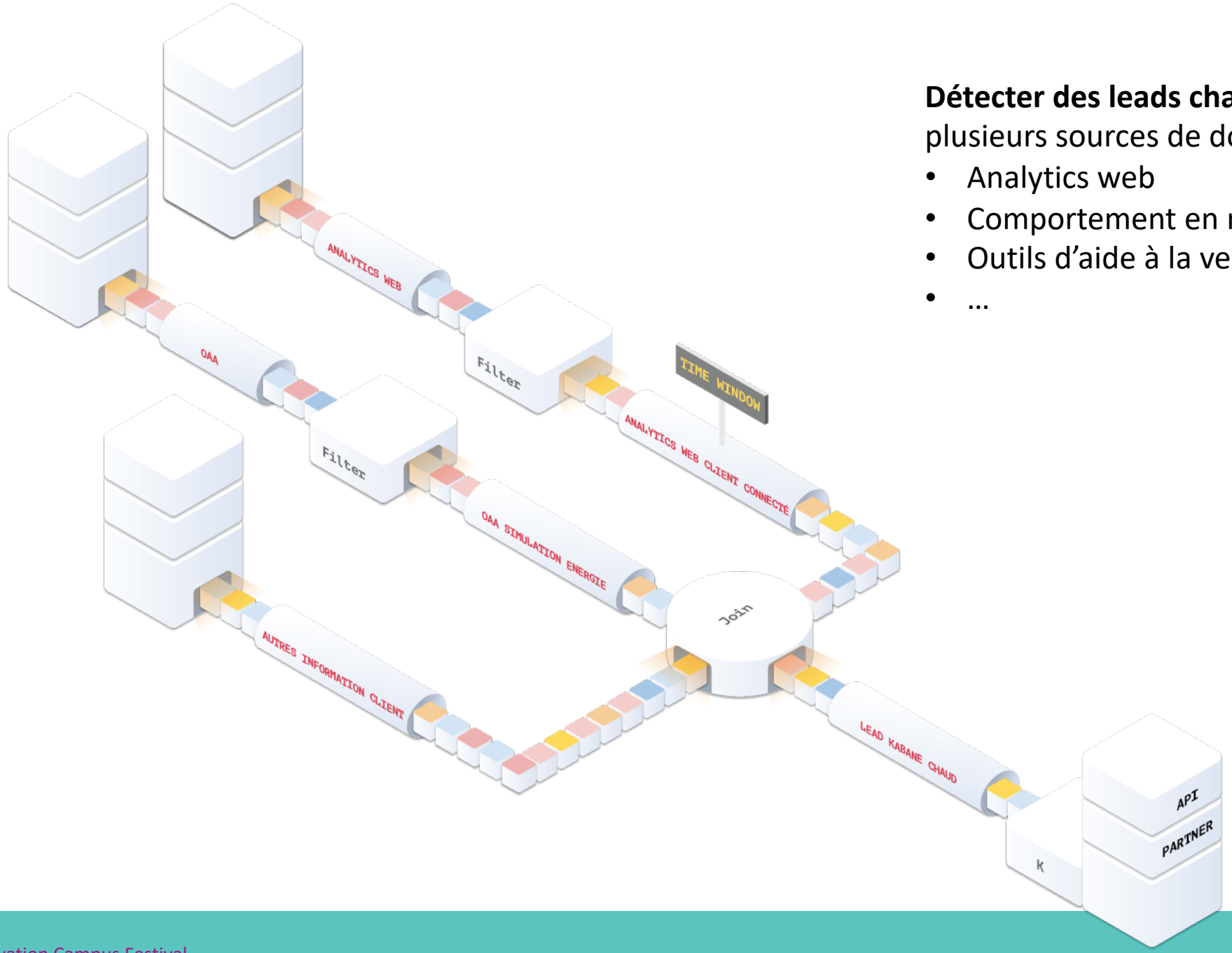


Source : blog confluent.io

Usecases

- Utiles quand vos données n’ont pas de début et surtout **pas de fin**

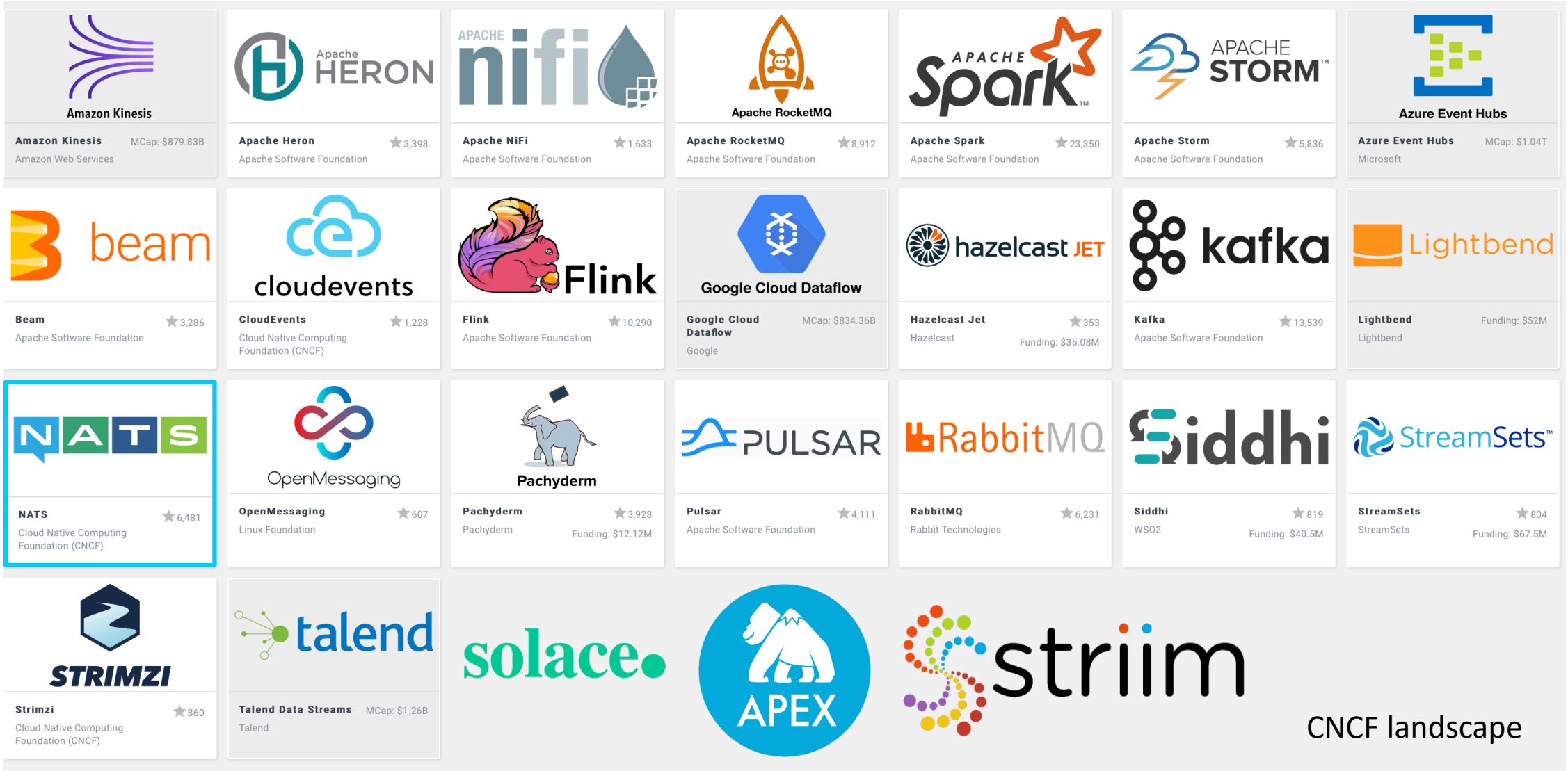
- Exemples :
  - Suivi temps réel de **consommation énergétique** (IOT)
  - **Suivi** de fabrication **industrielle** et logistique
  - **Détection de fraudes** sur des sites e-commerce
  - Analyse de transactions **financières** world wide.
  - Analyse du **trafic global** d’une région
  - …



Détecter des leads chauds en fonction de plusieurs sources de données :

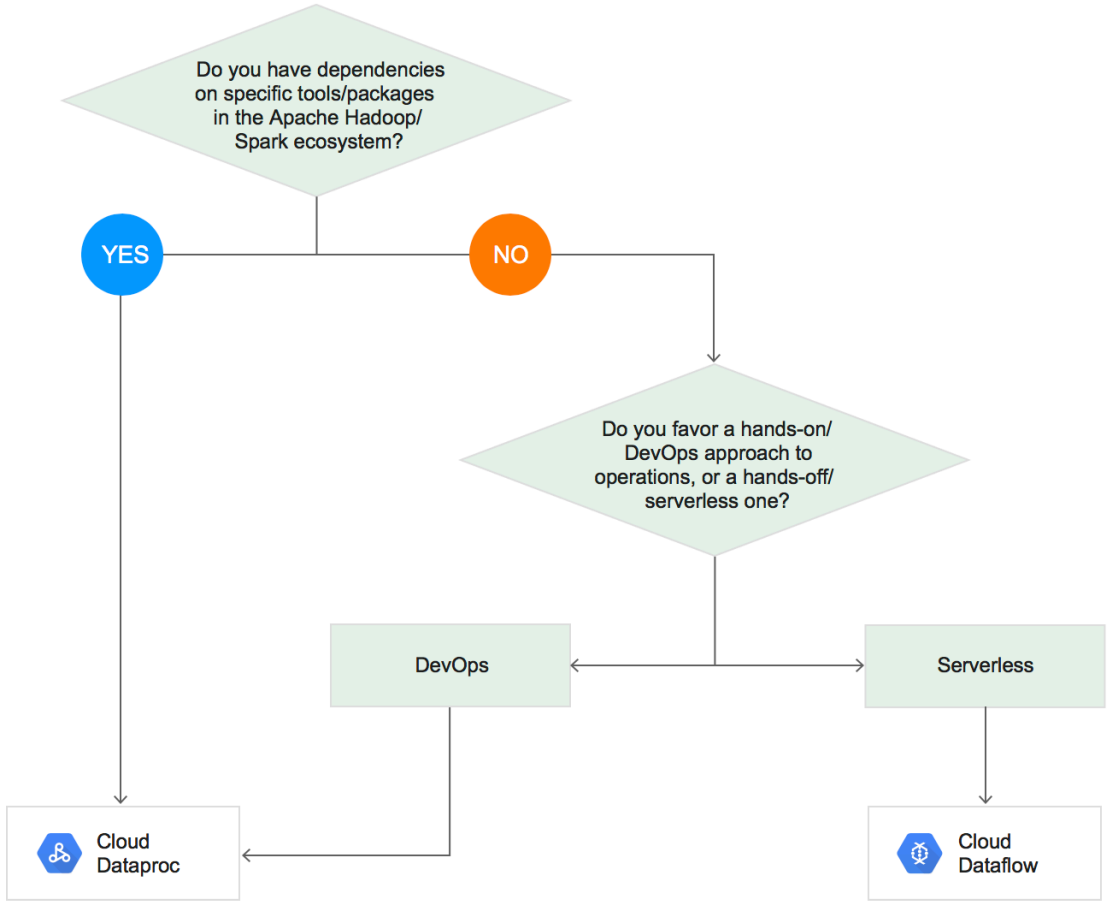
- Analytics web
- Comportement en magasin
- Outils d'aide à la vente
- …

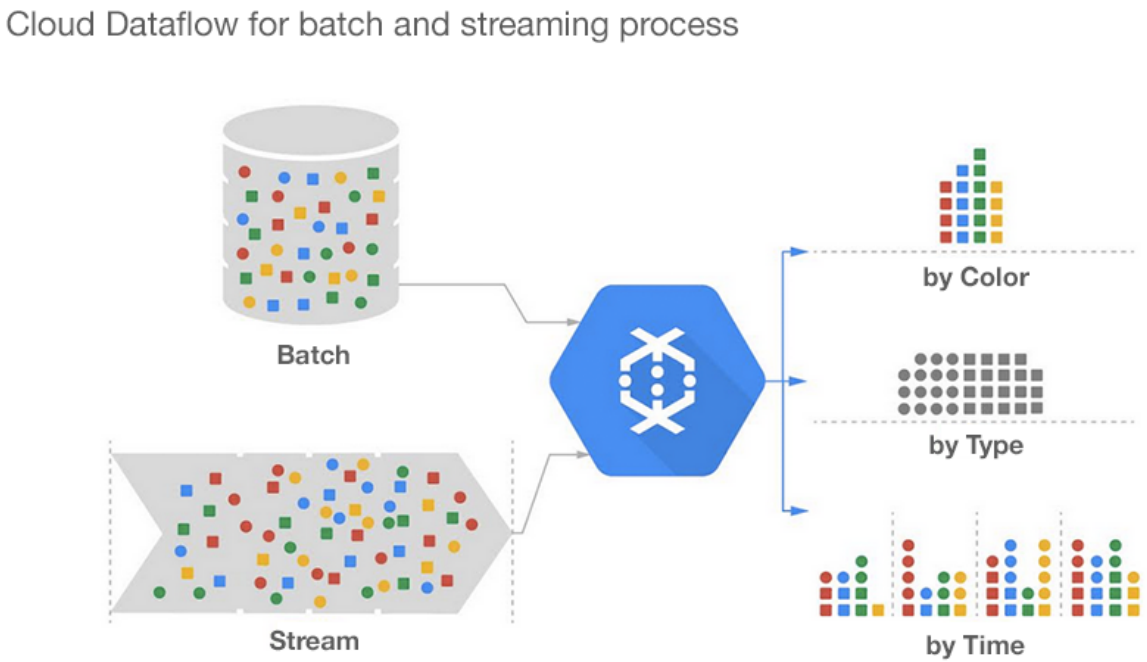
Streaming dans le cloud ou on premise



CNCF landscape

Google Cloud Dataflow





Source : Google Dataflow documentation

Apache Beam

Google publie une implémentation du SDK en 2014.

- SDK **Java** et Python
- Langage unifié pour batch et streaming
- Portable : multi moteur
  - Apex, Flink, Spark, **Dataflow**, Samza, Gearpump
- Extensible
  - Librairies de transformation
  - Input/Output

Apache Beam

Déjà pas mal de possibilités

Language	File-based	Messaging	Database
Java	Beam Java supports Apache HDFS, Amazon S3, Google Cloud Storage, and local filesystems. FileIO (general-purpose reading, writing, and matching of files) AvroIO TextIO TFRecordIO XmlIO TikaIO ParquetIO	Amazon Kinesis AMQP Apache Kafka Google Cloud Pub/Sub JMS MQTT RabbitMqIO SqsIO	Apache Cassandra Apache Hadoop Input/Output Format Apache HBase Apache Hive (HCatalog) Apache Kudu Apache Solr Elasticsearch (v2.x, v5.x, v6.x) Google BigQuery Google Cloud Bigtable Google Cloud Datastore Google Cloud Spanner JDBC MongoDB Redis

Quelques lignes de code

```
static class ExtractWordsFn extends DoFn<String, String> {
  private final Counter emptyLines = Metrics.counter(ExtractWordsFn.class, "emptyLines");
  private final Distribution lineLenDist =
      Metrics.distribution(ExtractWordsFn.class, "lineLenDistro");

  @ProcessElement
  public void processElement(@Element String element, OutputReceiver<String> receiver) {
    lineLenDist.update(element.length());
    if (element.trim().isEmpty()) {
      emptyLines.inc();
    }

    // Split the line into words.
    String[] words = element.split(ExampleUtils.TOKENIZER_PATTERN, -1);

    // Output each word encountered into the output PCollection.
    for (String word : words) {
      if (!word.isEmpty()) {
        receiver.output(word);
      }
    }
  }
}
```

Ecriture d'une fonction parallélisable prenant un Element

Quelques lignes de code

```
/**
 * A PTransform that converts a PCollection containing lines of text into a PCollection of
 * formatted word counts.
 *
 * <p>Concept #3: This is a custom composite transform that bundles two transforms (ParDo and
 * Count) as a reusable PTransform subclass. Using composite transforms allows for easy reuse,
 * modular testing, and an improved monitoring experience.
 */
public static class CountWords
    extends PTransform<PCollection<String>, PCollection<KV<String, Long>>> {
  @Override
  public PCollection<KV<String, Long>> expand(PCollection<String> lines) {

    // Convert lines of text into individual words.
    PCollection<String> words = lines.apply(ParDo.of(new ExtractWordsFn()));

    // Count the number of times each word occurs.
    PCollection<KV<String, Long>> wordCounts = words.apply(Count.perElement());

    return wordCounts;
  }
}
```

Ecriture d'une fonction prenant une Collection

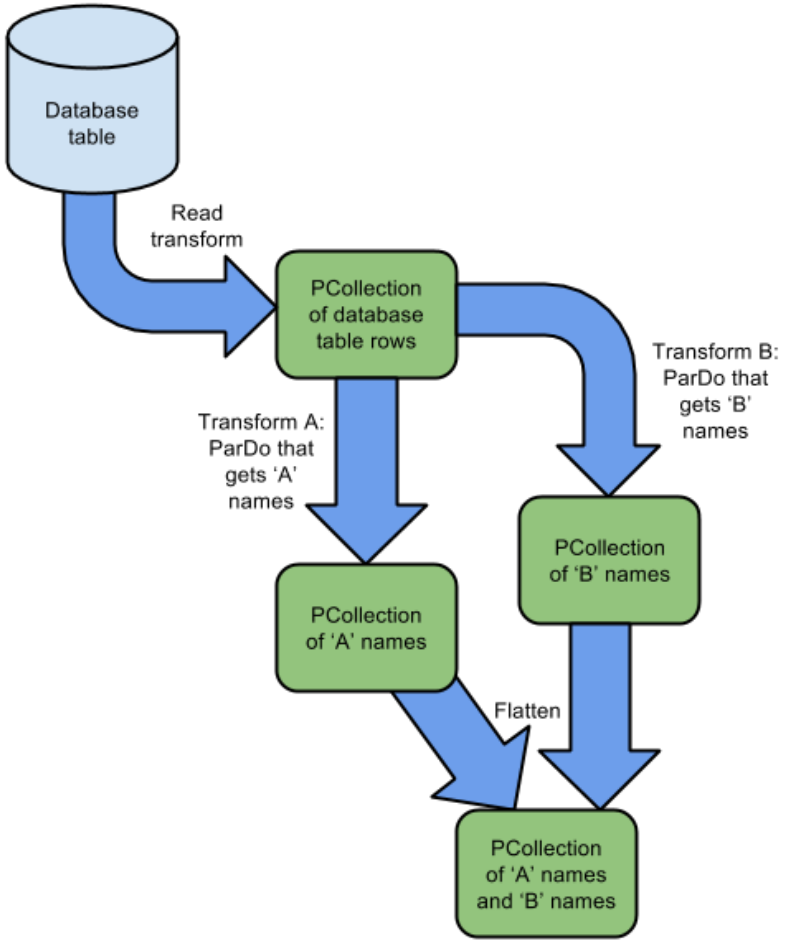
Quelques lignes de code

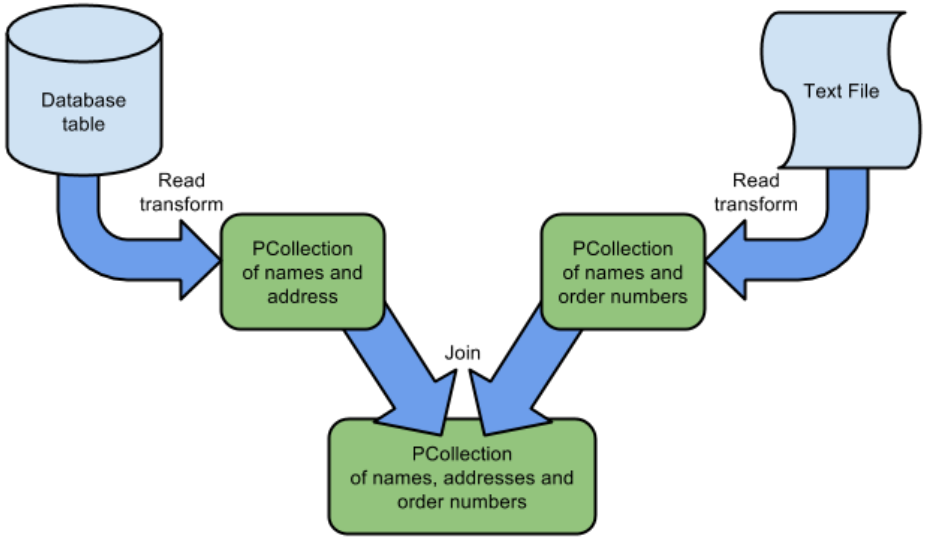
```
  static void runWordCount(WordCountOptions options) {
    Pipeline p = Pipeline.create(options);

    // Concepts #2 and #3: Our pipeline applies the composite CountWords transform, and passes the
    // static FormatAsTextFn() to the ParDo transform.
    p.apply("ReadLines", TextIO.read().from(options.getInputFile()))
        .apply(new CountWords())
        .apply(MapElements.via(new FormatAsTextFn()))
        .apply("WriteCounts", TextIO.write().to(options.getOutput()));

    p.run().waitUntilFinish();
  }

  public static void main(String[] args) {
    WordCountOptions options =
        PipelineOptionsFactory.fromArgs(args).withValidation().as(WordCountOptions.class);

    runWordCount(options);
  }
}
```

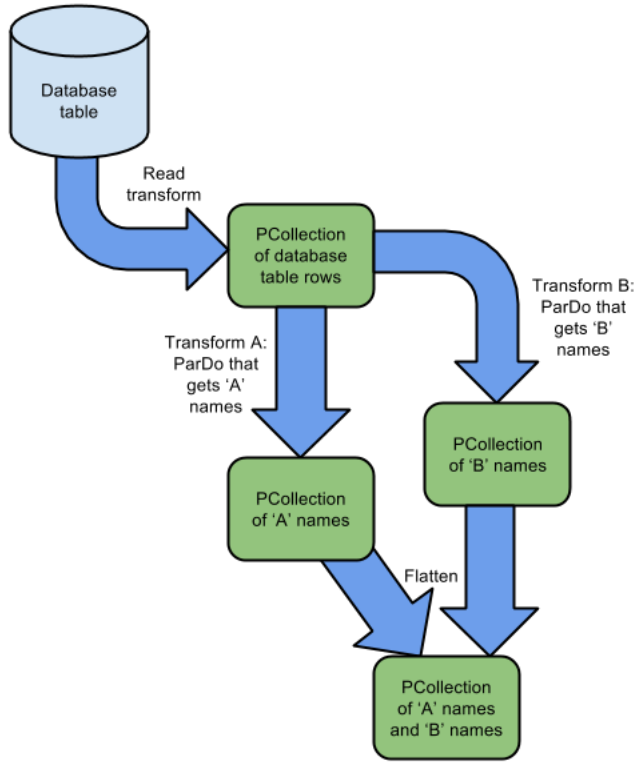
Ecriture du pipeline :

- InputIO
- Transformation
- OutputIO

In/Out/Transformations/ Side Input



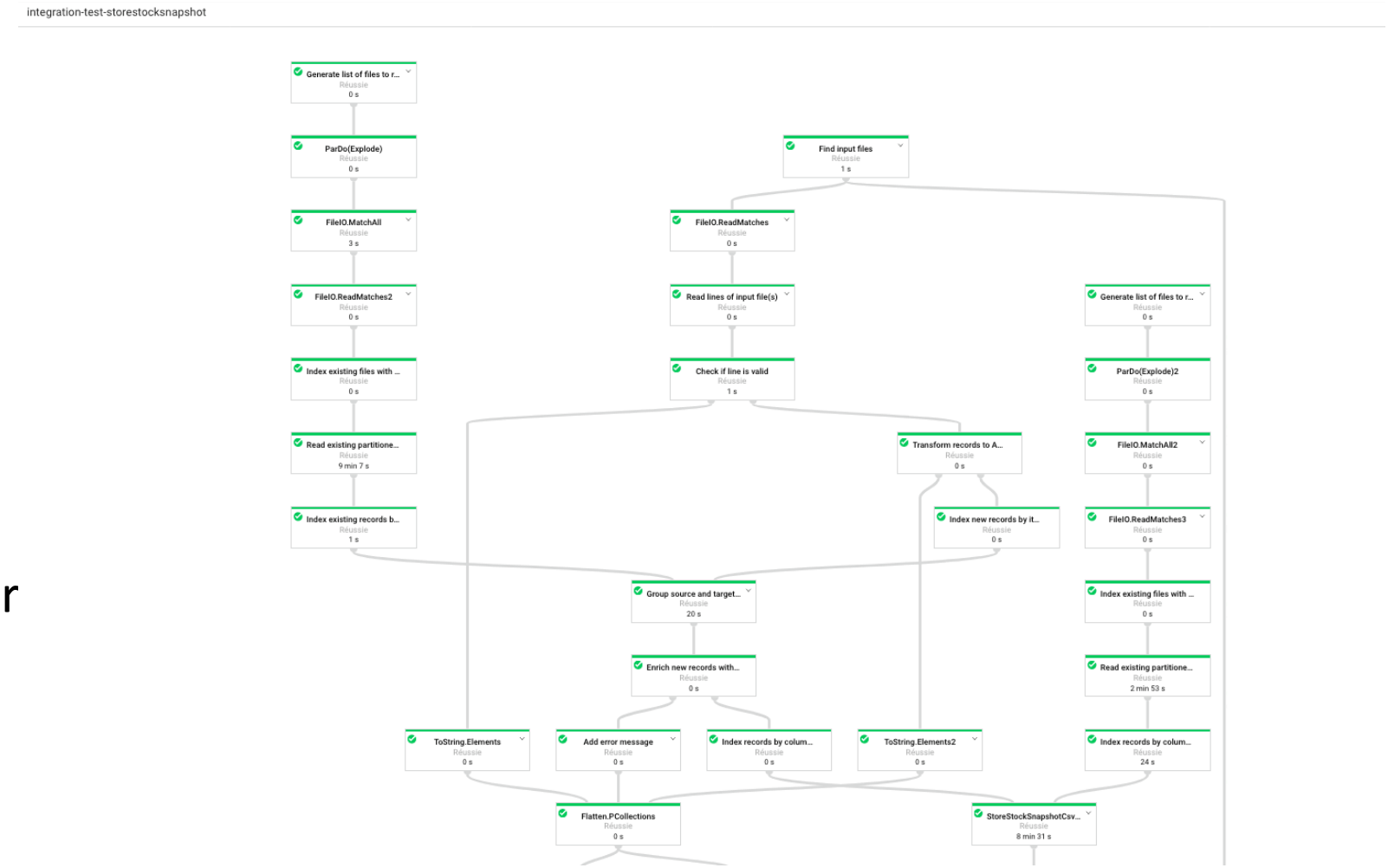




Source : Google Dataflow Documentation

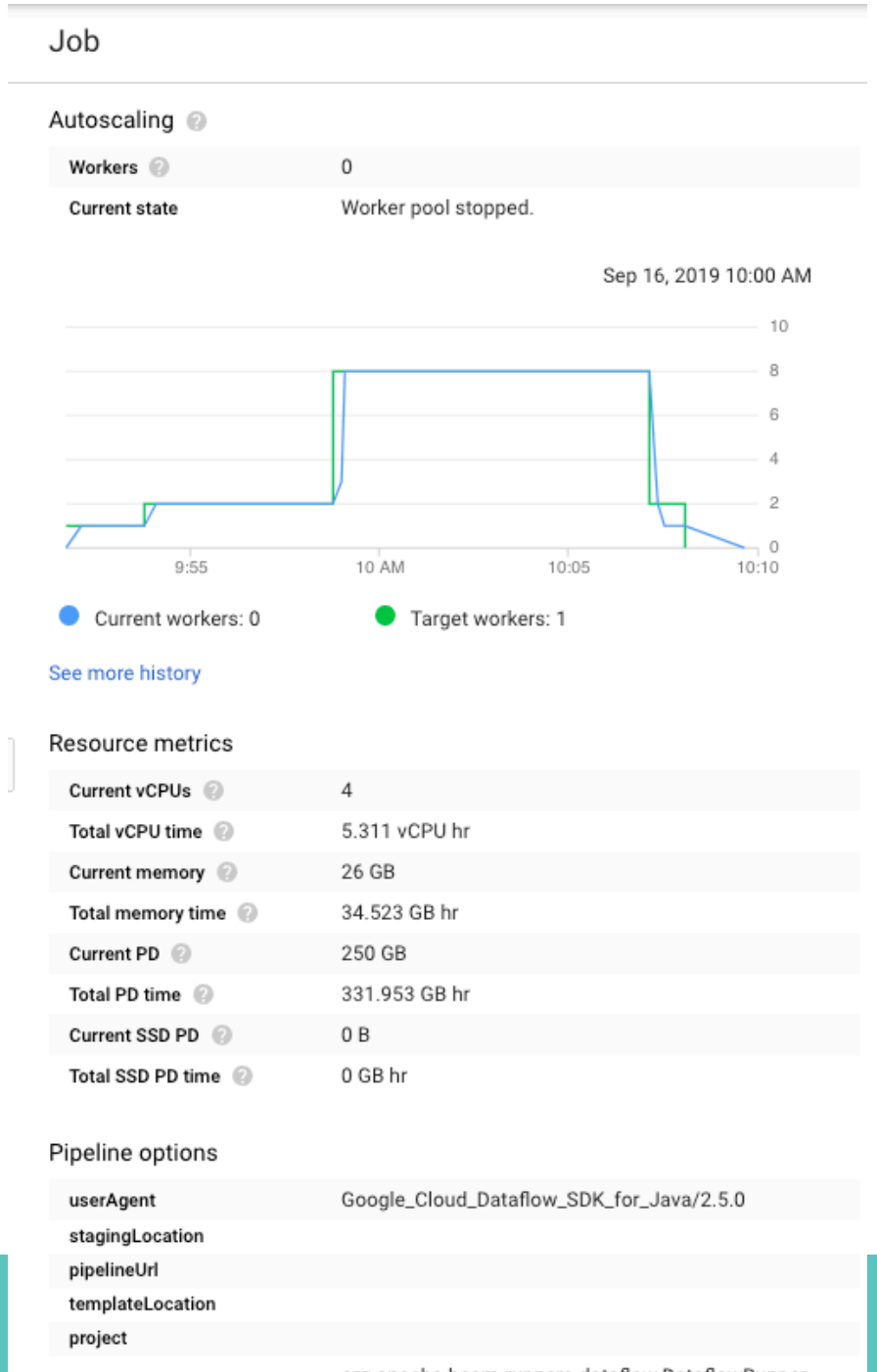
Google Dataflow en 6 étapes

1. Créer votre pipeline
2. Run en local
3. Upload sur la plateforme
    1. Gcloud console
    2. Git repository
4. Lancer et visualiser l’arbre
5. Surveiller ses machines !!!
6. Monitorer dans StackDriver



Autoscaling

- Capacity planning
- Alerte de facturation
  - 1 bug peut être couteux
- Garde fou :
  - Sur les durées de batches
  - Sur le nombre de machines
- Capitaliser sur Stackdriver
- Toujours **tester** !



Apache Kafka

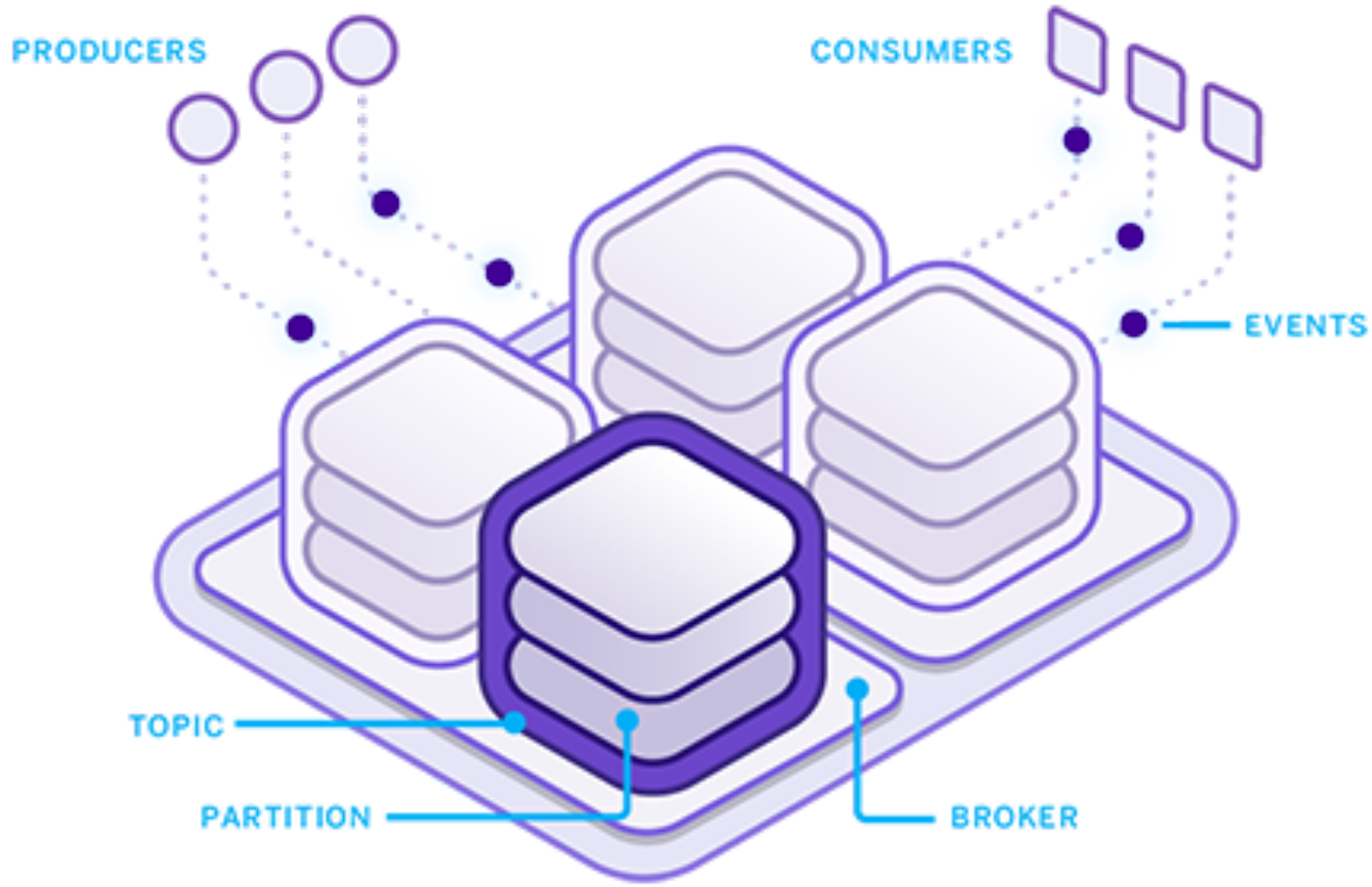
- Développé chez **Linkedin** et rendu **opensource** vers 2011 à la base pour un mécanisme de pub/sub de messagerie.

- Le cœur de Kafka est écrit en **Java/Scala** et l'ensemble de l'écosystème est majoritairement en Java comme d'autres technos Apache du monde du streaming et du big data.

- Passage dans l'incubateur d'**Apache** en Octobre 2012.

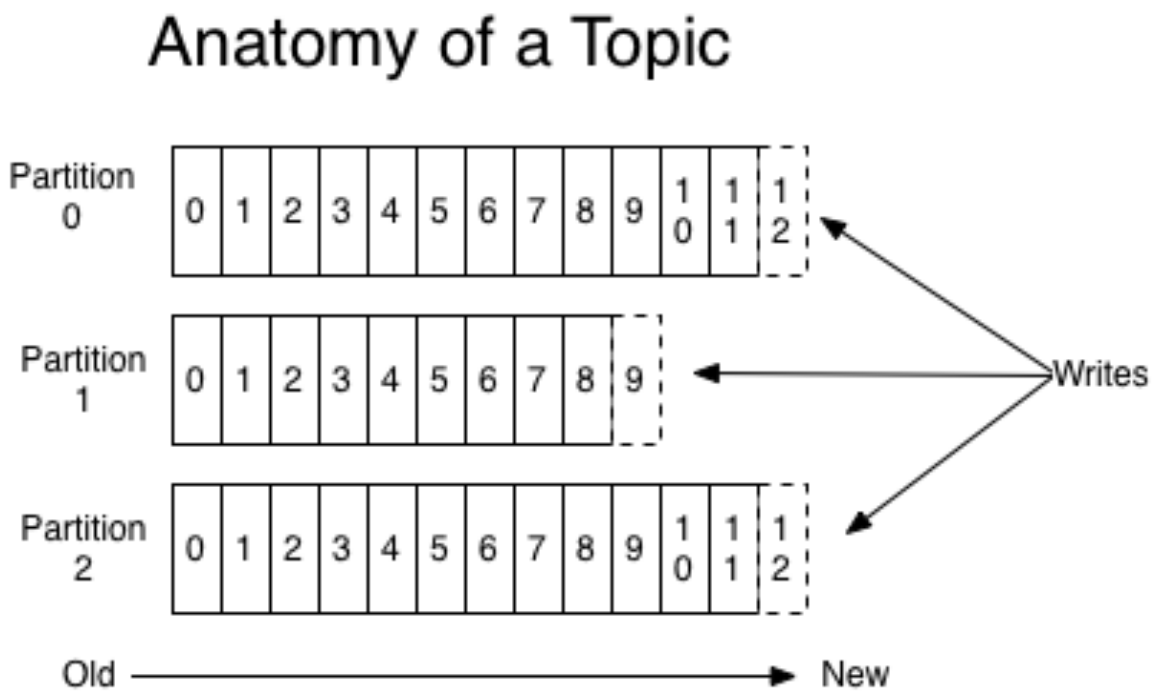
- En 2014, ses créateurs Jun Rao, Jay Kreps, and **Neha Narkhede (CEO de l'année 2018)** créent Confluent.

Terminologie



Source : HerokuApp

Terminologie



Source : Apache Kafka documentation

Un topic est une « **catégorie** » de données.

On y écrit généralement des données de **même nature** ou des données pour lesquelles **l'ordre** a une importance.

Kafka choisi la partition dans laquelle il va écrire les données avec une fonction, **overridable**.

Une **partition** est une unité de **parallélisme** : 2 consommateurs ne peuvent lire les messages d'une même partition.

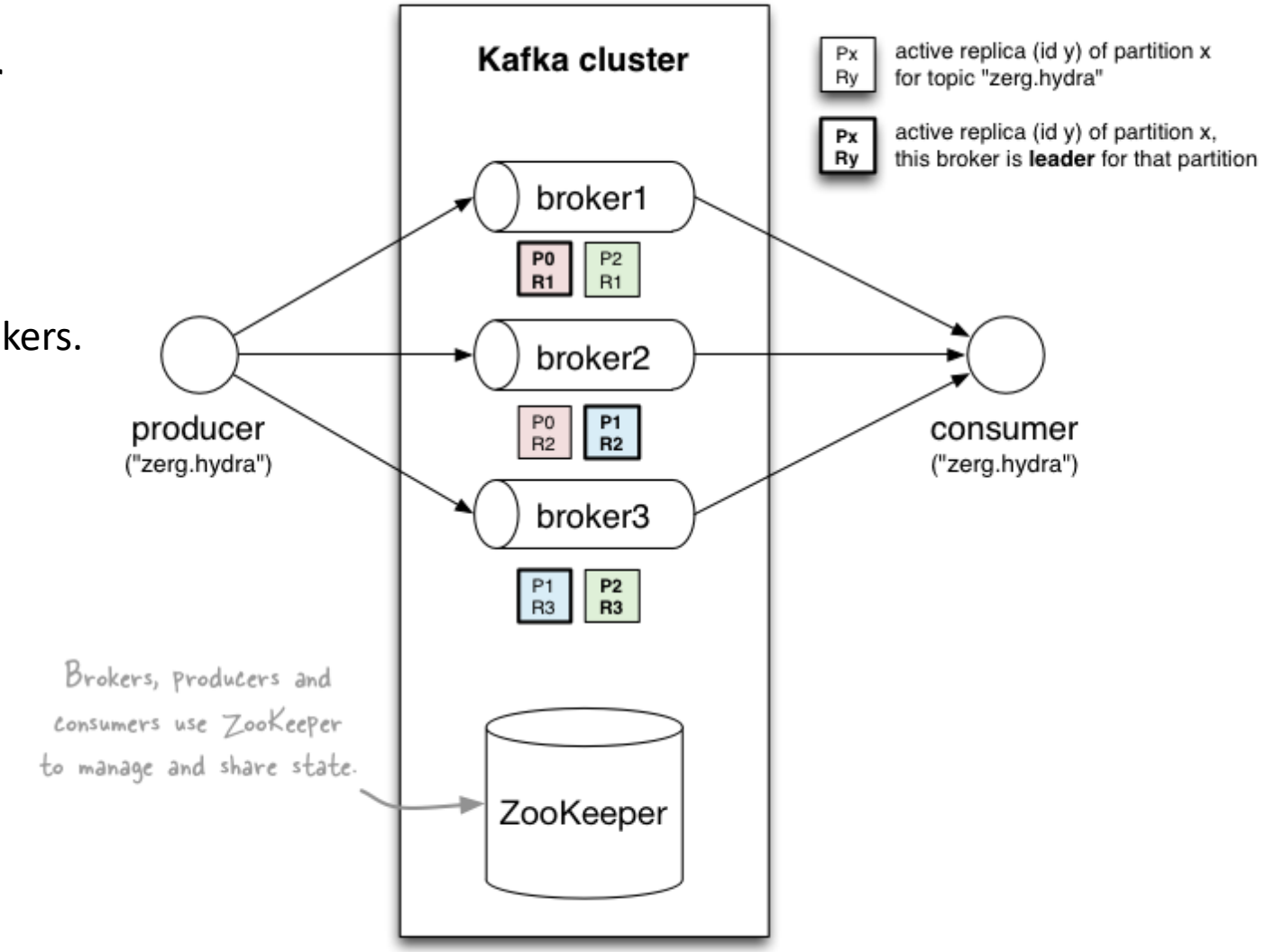
Un consommateur peut lire **plusieurs** partitions (consumer rebalance à tester en démo).

Terminologie

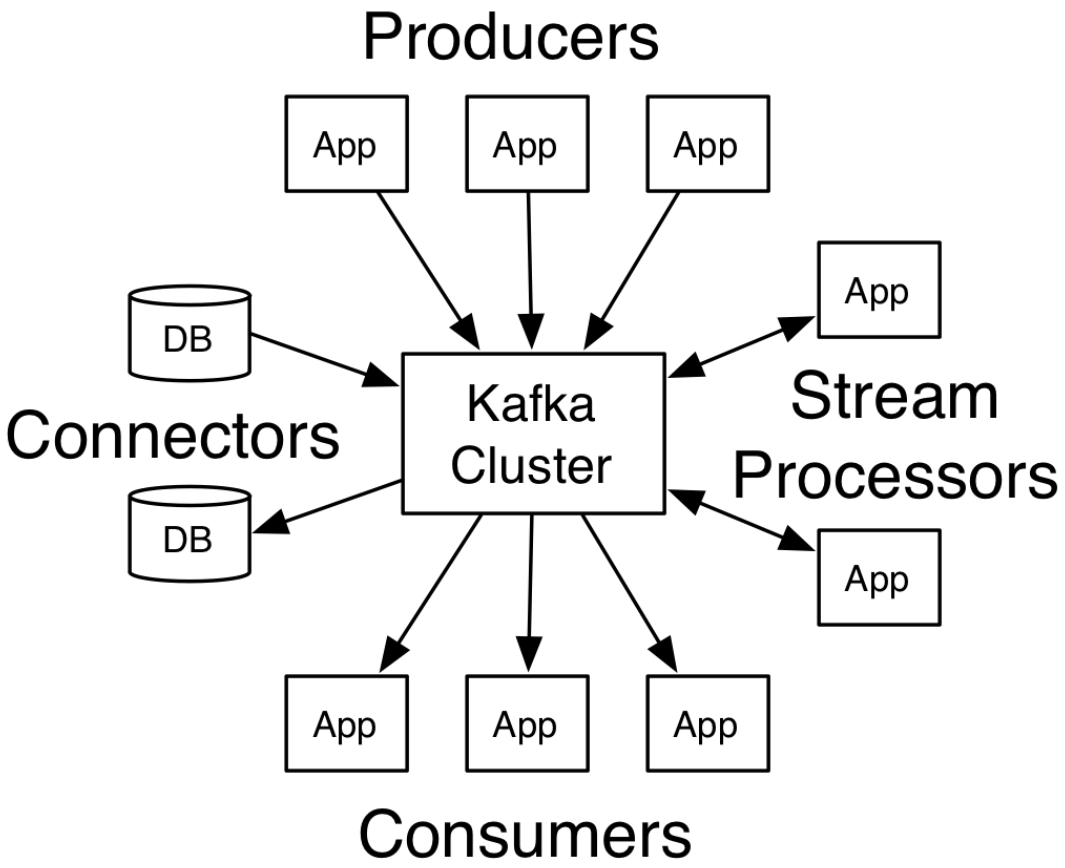
Un cluster Kafka composé de :

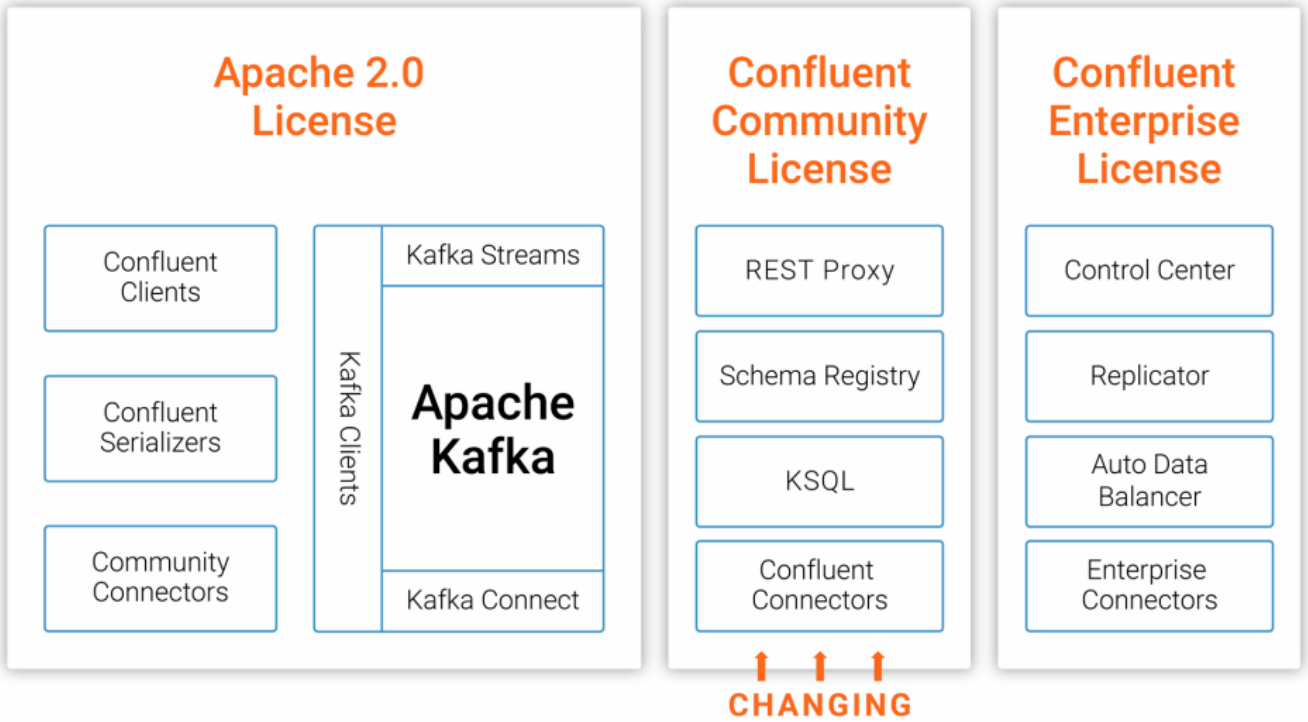
- X nœud de données appelés **Broker**
  - **Controller**
  - **Partition Leader** & **Follower**
- Un cluster **Zookeeper** contenant

les metadatas des partitions et des brokers.



Source : Apache Kafka documentation

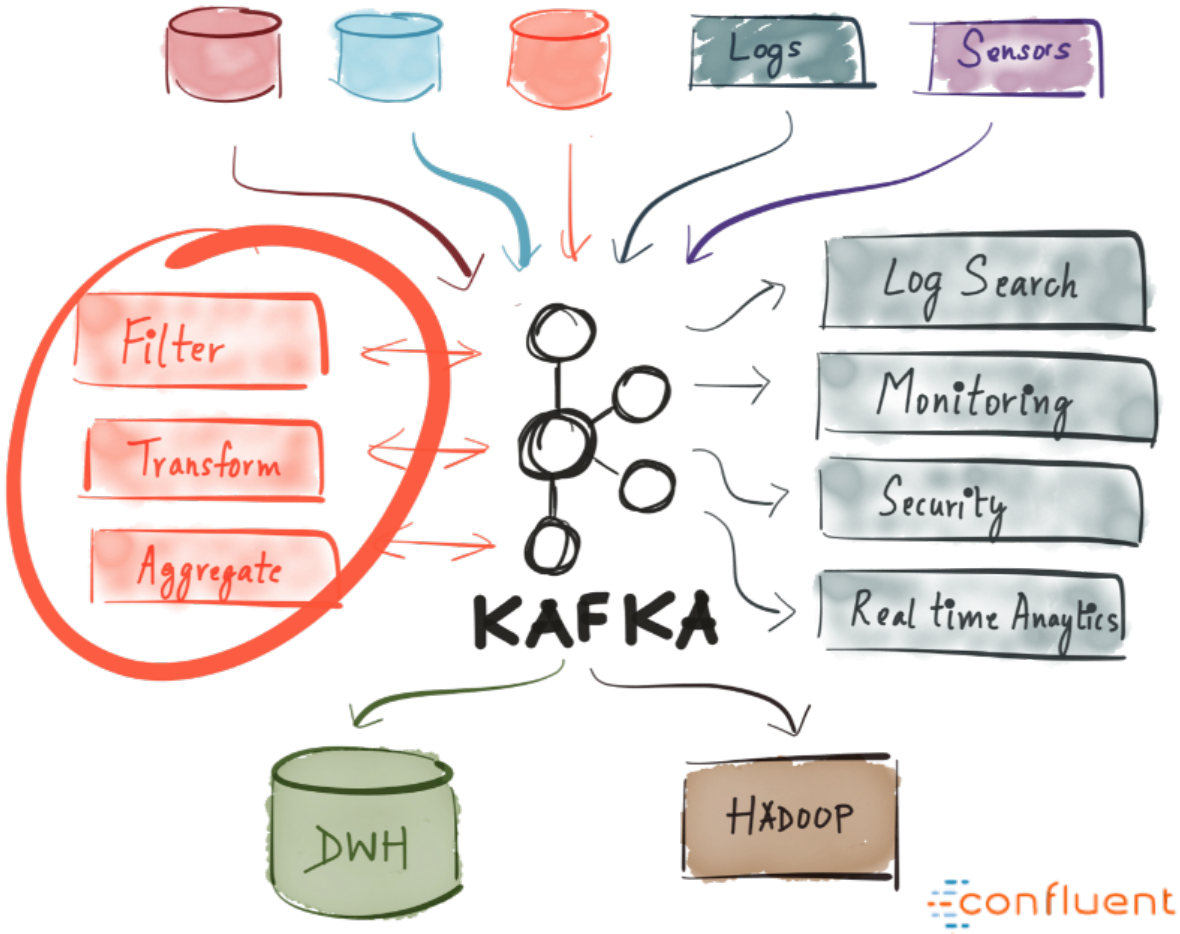




Source : Confluent.io

KStream

- **Elastic**, highly scalable, fault-tolerant
- **Deploy** to containers, VMs, bare metal, cloud
- **Equally** viable for small, medium, & large use cases
- **Fully integrated** with Kafka security
- Write **standard Java and Scala** applications
- **Exactly-once** processing semantics
- No **separate processing** cluster required



Stream vs Table

« *If you enable log compaction, there is no time-based expiry of data.* »
Martin Kleppmann *in* Making Sense of Stream Processing

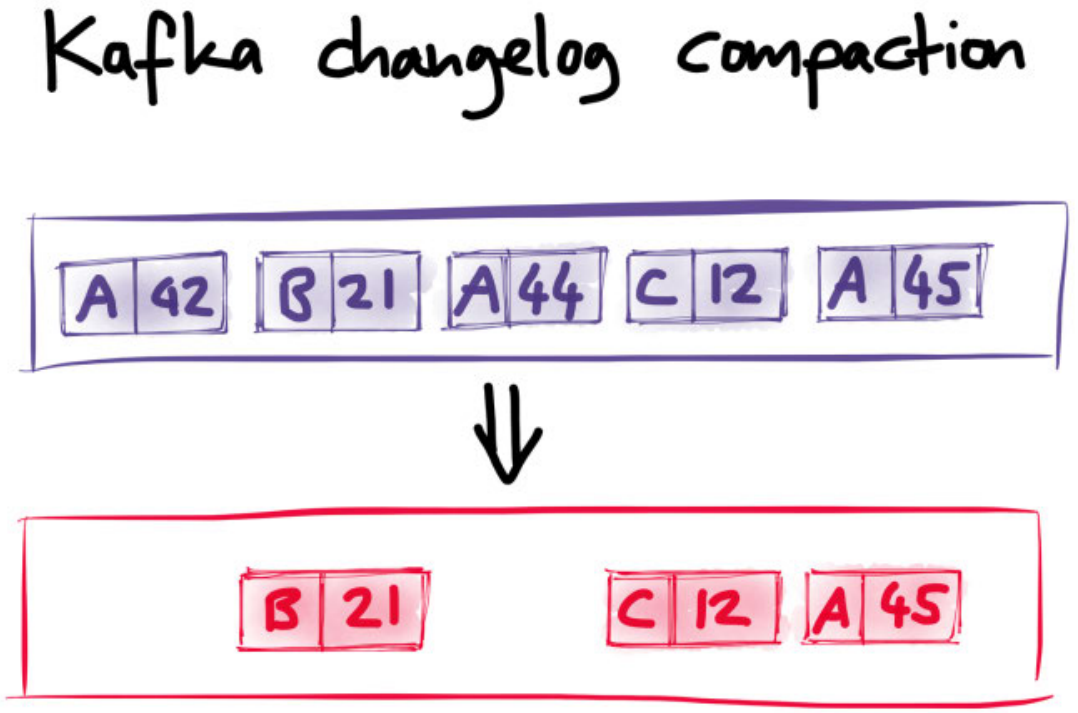
- **KStream** est un flux de données temps réel.

- **KTable** est une représentation de la dernière valeur d'une clé.

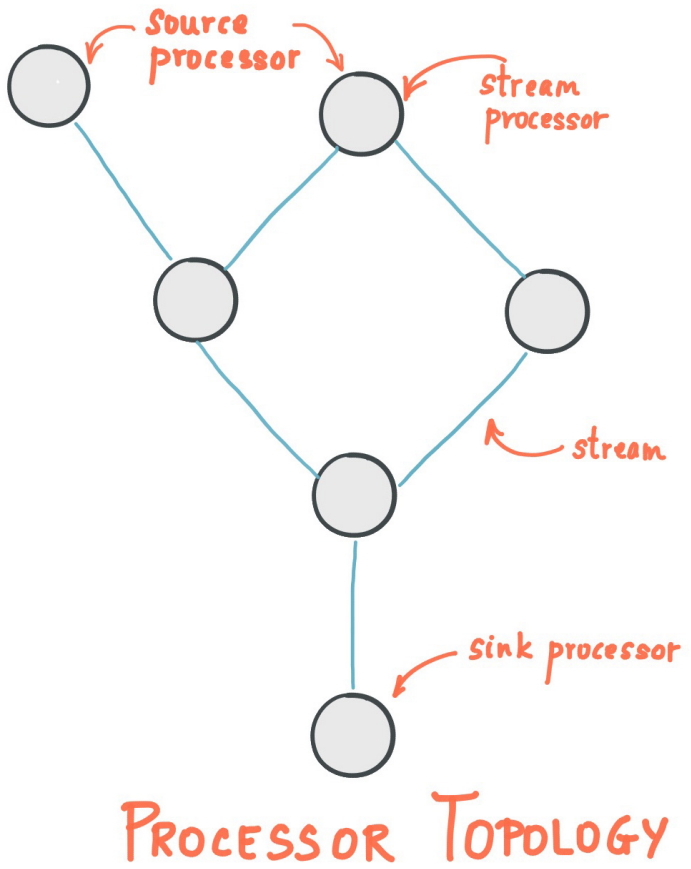
- Induit la notion de **log compaction**.
- Induit la notion de state store local

- **KTable** a une représentation en mémoire des Données : **RocksDB : localstatestore.**

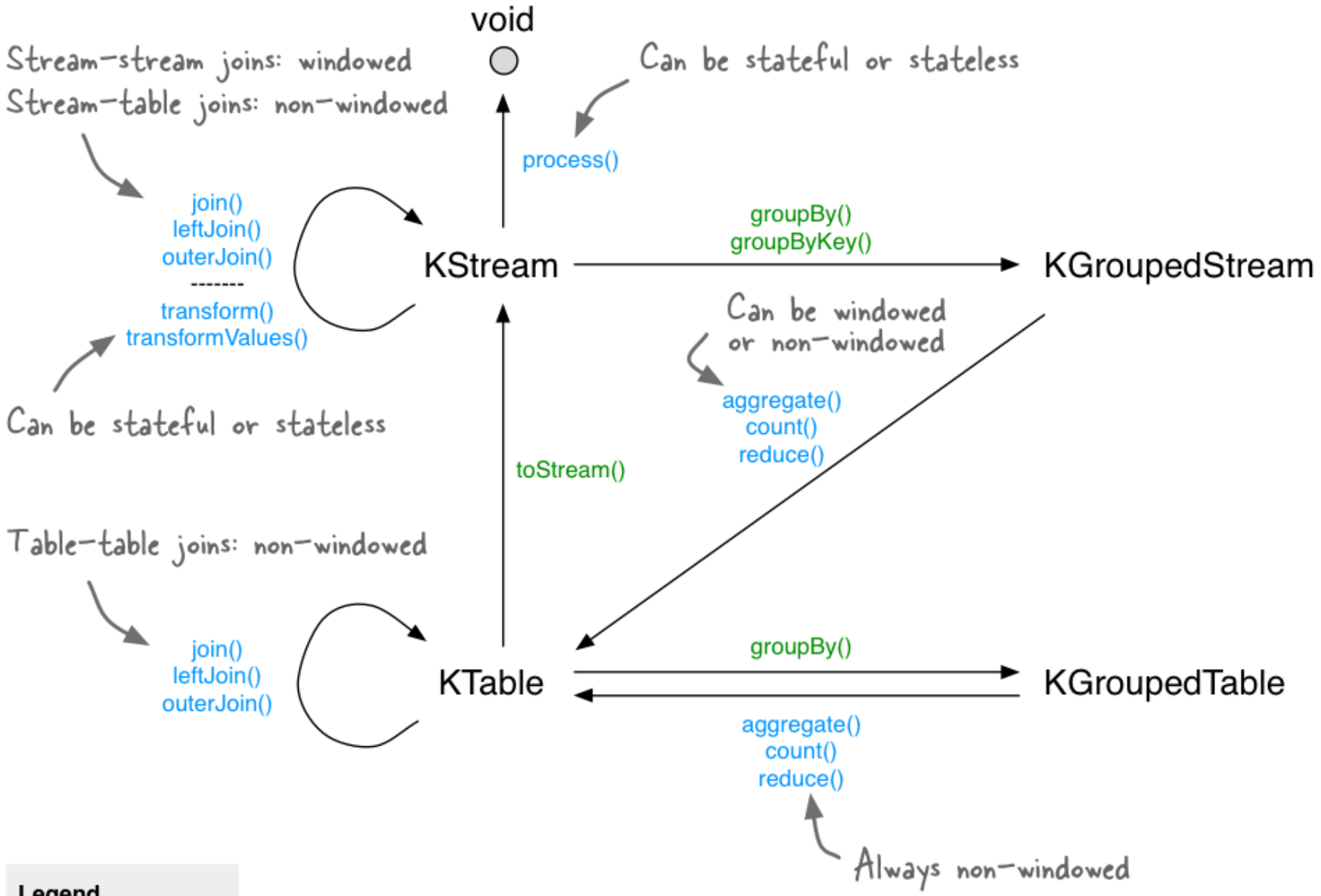
- **-> différentes implémentations**



Join operands	Type	(INNER) JOIN	LEFT JOIN	OUTER JOIN
KStream-to-KStream	Windowed	Supported	Supported	Supported
KTable-to-KTable	Non-windowed	Supported	Supported	Supported
KStream-to-KTable	Non-windowed	Supported	Supported	Not Supported
KStream-to-GlobalKTable	Non-windowed	Supported	Supported	Not Supported
KTable-to-GlobalKTable	N/A	Not Supported	Not Supported	Not Supported



- **Stateful or Stateless**



Source : Confluent documentation

Quelques lignes de code

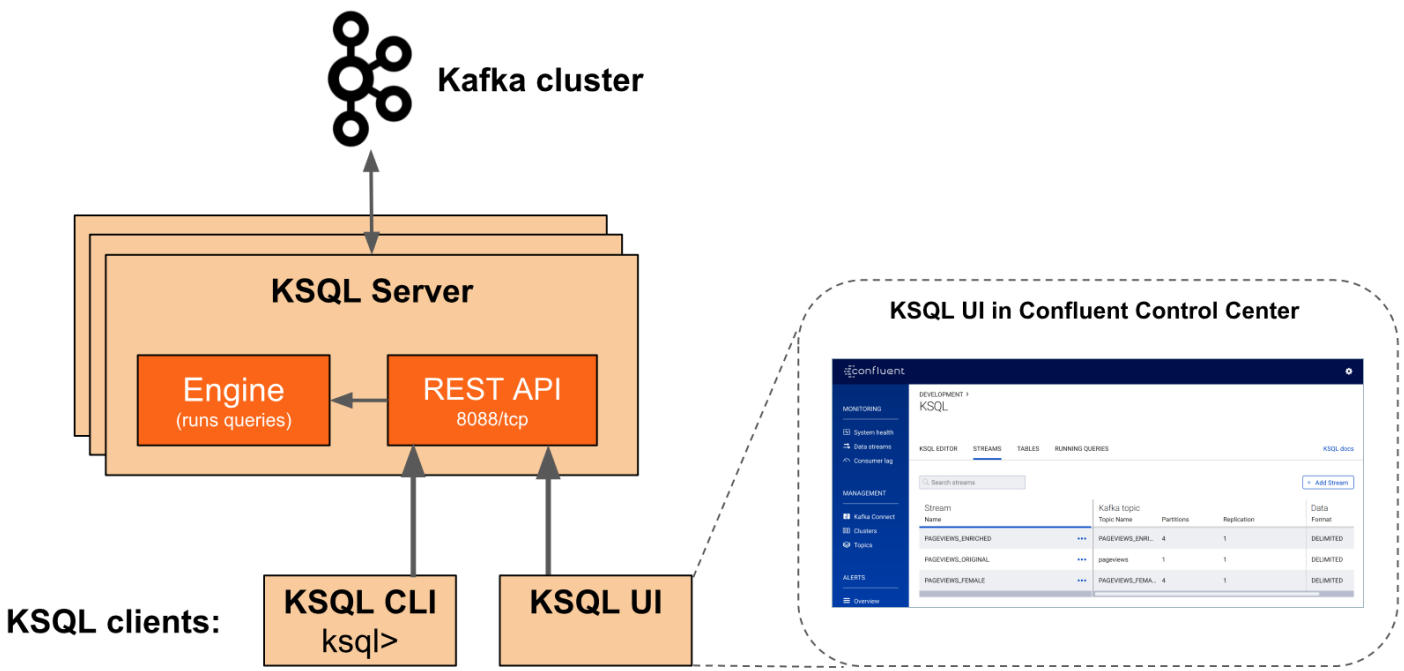
```
15  public class WordCountApplication {
16
17      public static void main(final String[] args) throws Exception {
18          Properties props = new Properties();
19          props.put(StreamsConfig.APPLICATION_ID_CONFIG, "wordcount-application");
20          props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka-broker1:9092");
21          props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass());
22          props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass());
23
24          StreamsBuilder builder = new StreamsBuilder();
25          KStream<String, String> textLines = builder.stream("TextLinesTopic");
26          KTable<String, Long> wordCounts = textLines
27              .flatMapValues(textLine -> Arrays.asList(textLine.toLowerCase().split("\\W+")))
28              .groupBy((key, word) -> word)
29              .count(Materialized.<String, Long, KeyValueStore<Bytes, byte[]>>as("counts-store"));
30          wordCounts.toStream().to("WordsWithCountsTopic", Produced.with(Serdes.String(), Serdes.Long
31
32          KafkaStreams streams = new KafkaStreams(builder.build(), props);
33          streams.start();
34      }
35
36  }
```

Source : Confluent Github

KSQL

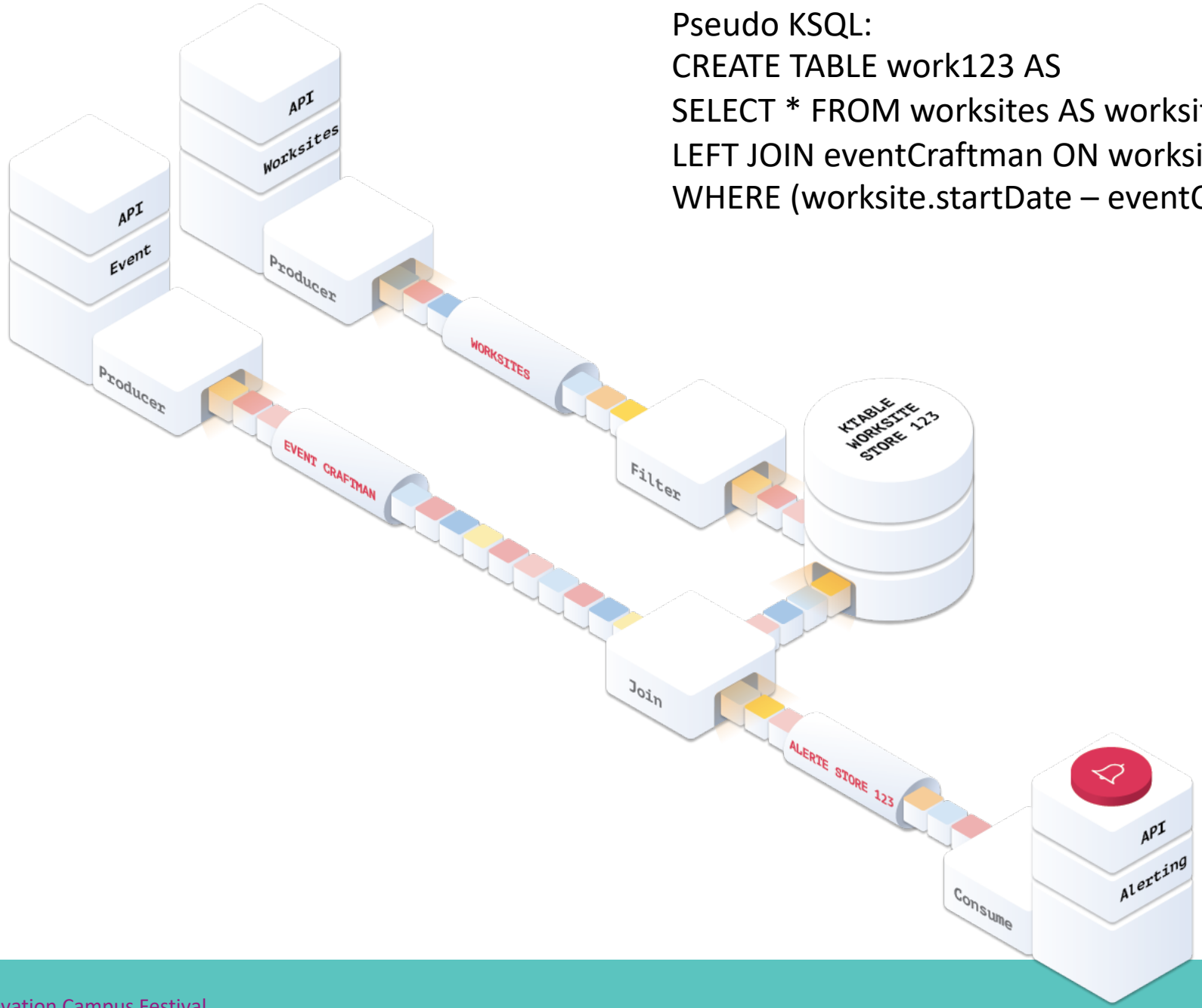
- *KSQL is the streaming SQL engine for Apache Kafka®.*

- Cluster indépendant du cluster.
- Les requêtes KSQL sont transformées en application KStream.

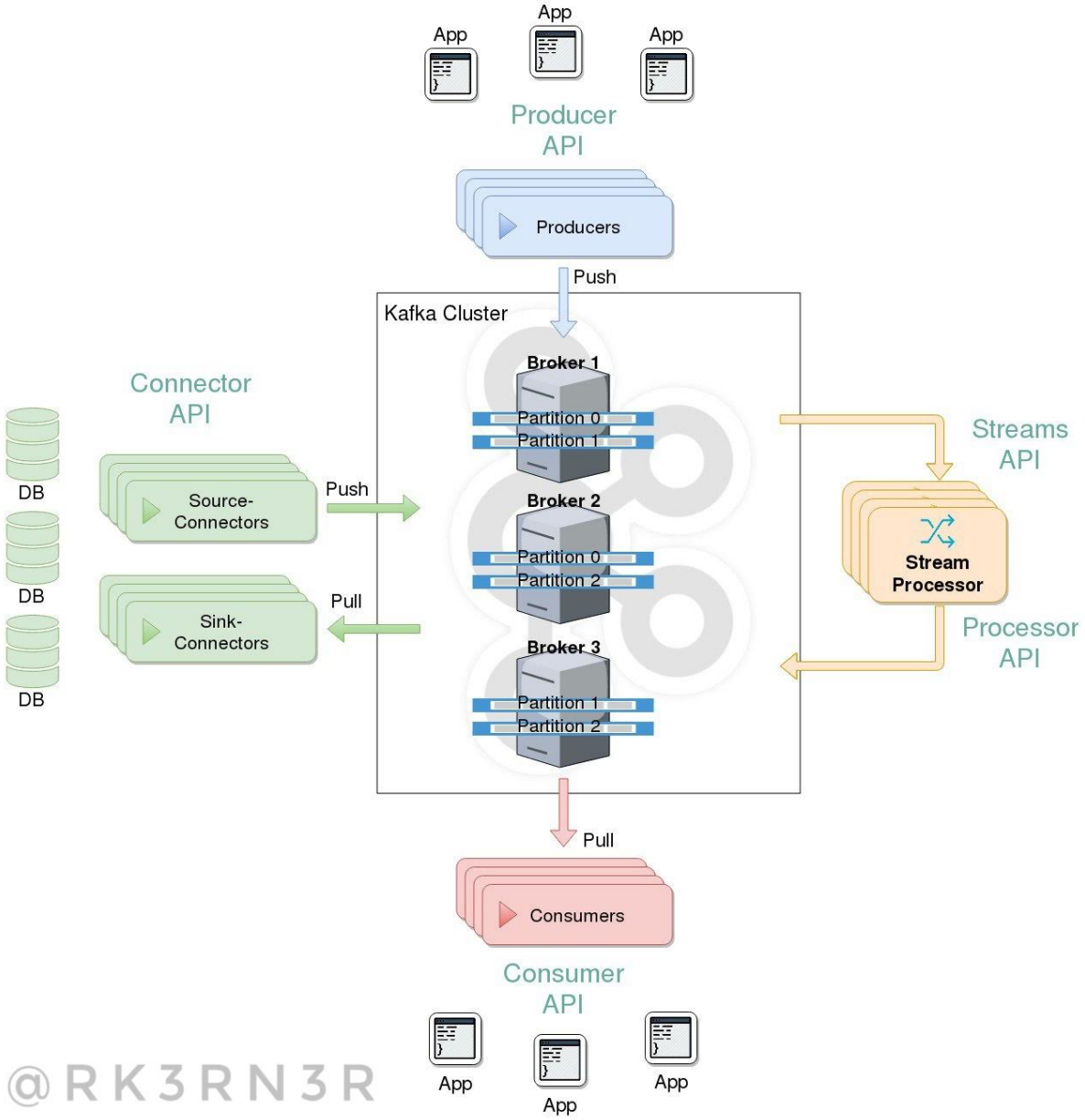


Source : Confluent documentation

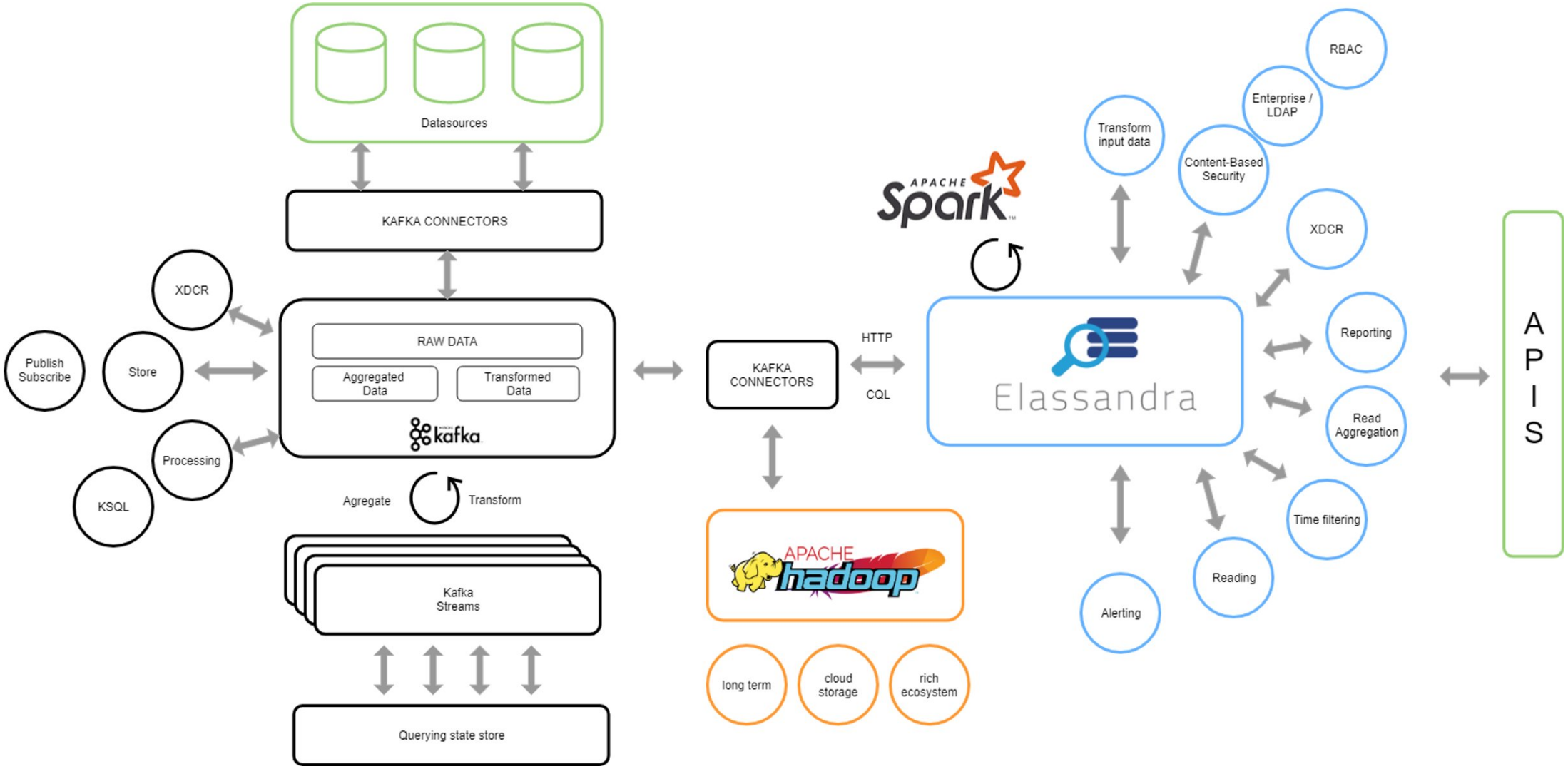
Pseudo KSQL:
CREATE TABLE work123 AS
SELECT * FROM worksites AS worksite
LEFT JOIN eventCraftman ON worksite.id = eventCraftman.worksite.id
WHERE (worksite.startDate – eventCraftMan.starDate) > ?;



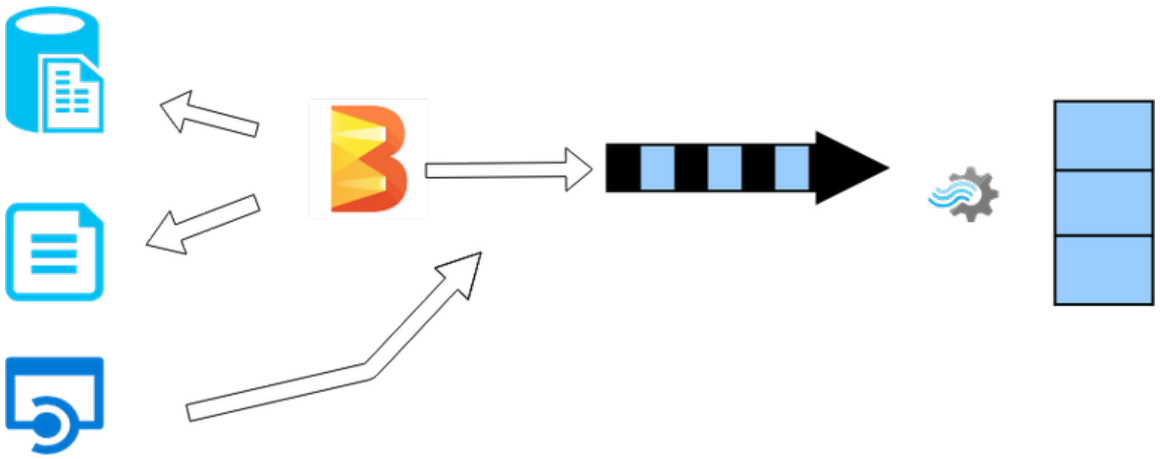
Kstream, Ktable, operations



Source, Stream, Process and Sink



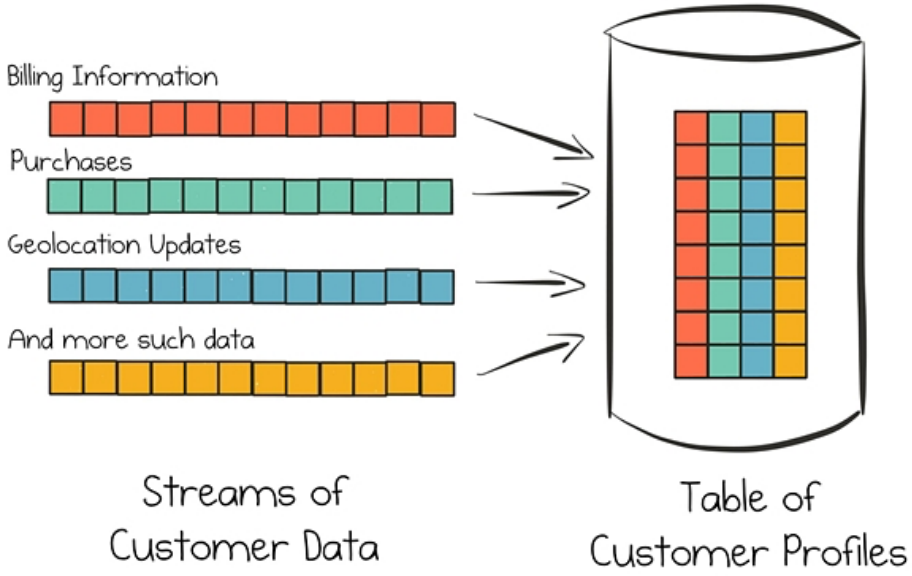
Agregate and process



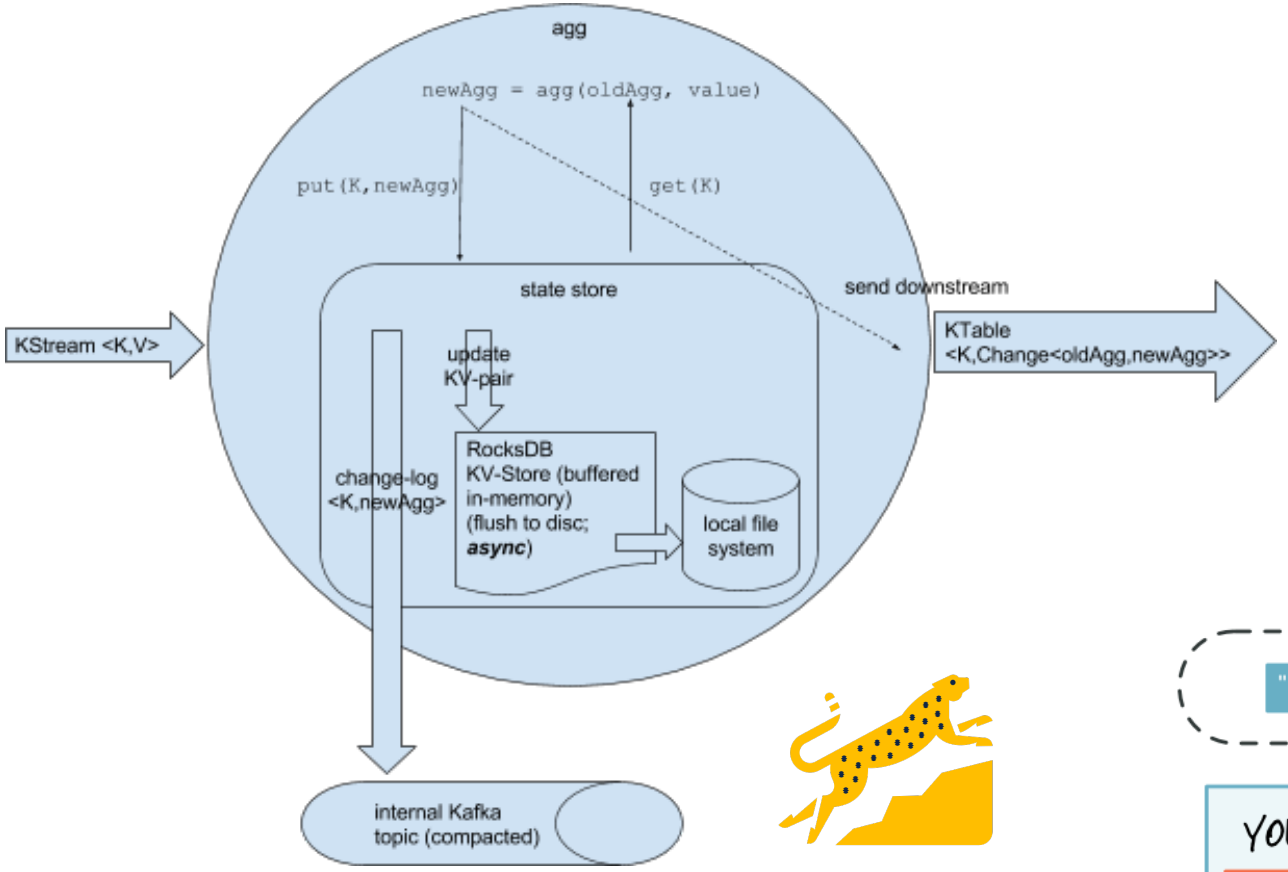
Une source de données brutes =

- Un topic de données brutes
- + Un ou plusieurs traitement
- + Une **Ktable** de données

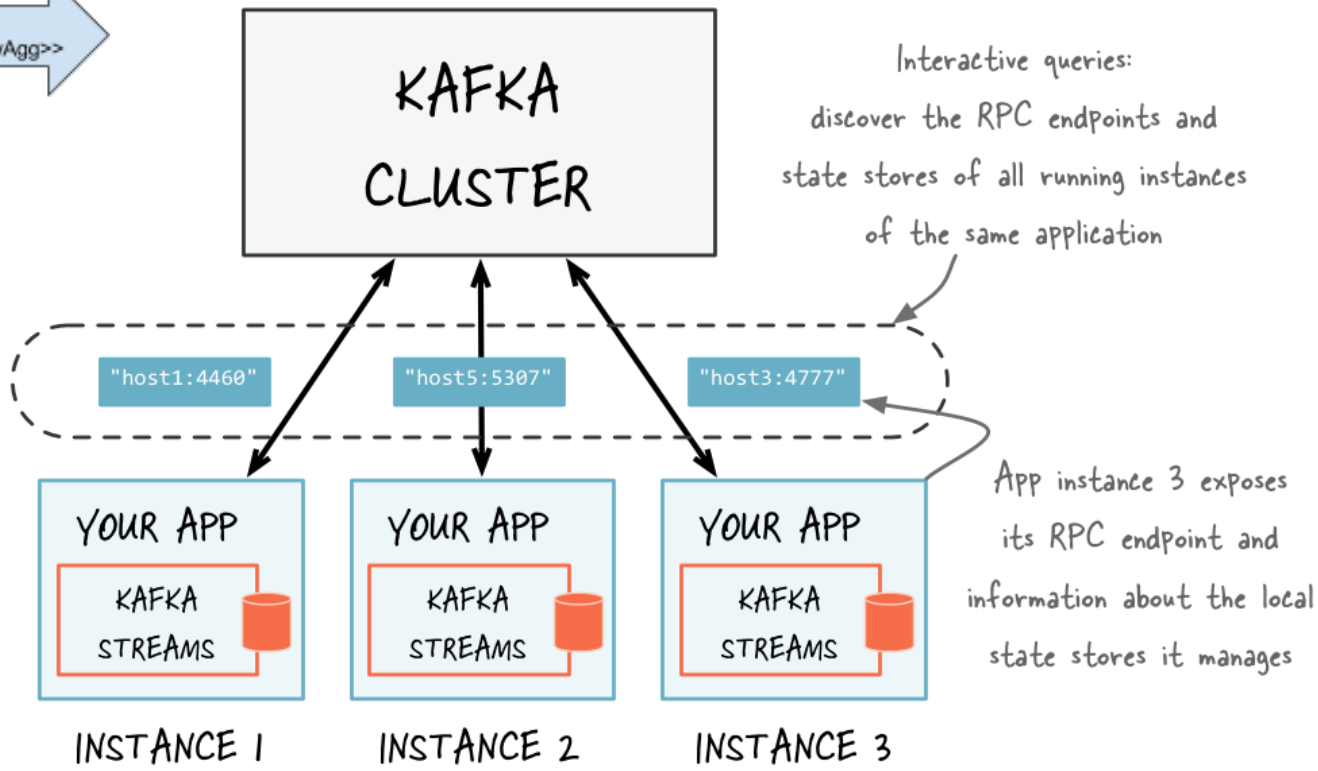
- **Disposer** des états intermédiaires
- Avoir une ou plusieurs **vues agrégées** des données
- Une ou plusieurs applications dans une ou plusieurs versions



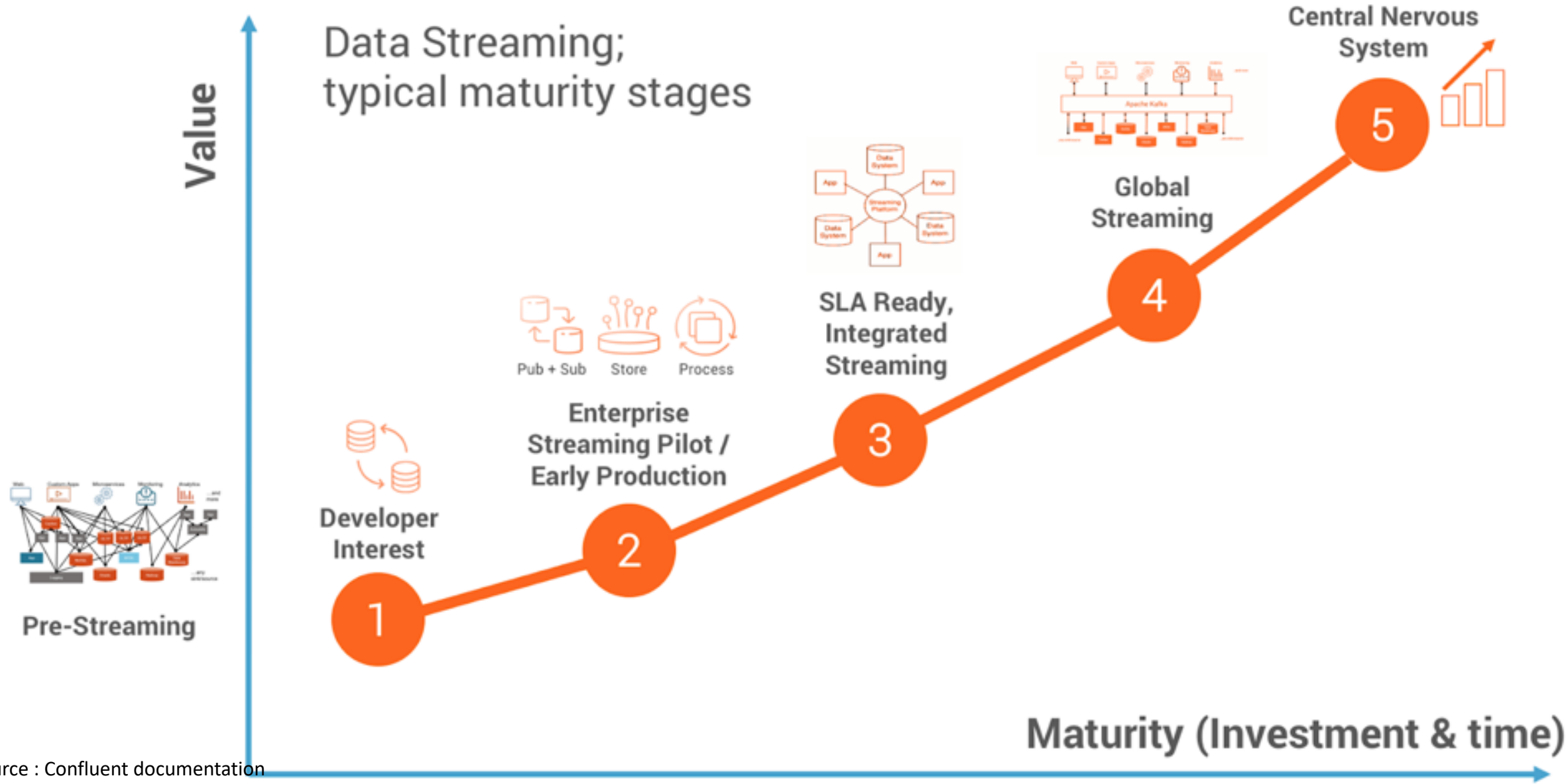
Source : Confluent documentation



RocksDB ou ?



Source : Confluent documentation



Source : Confluent documentation

Synthèse

- Le data streaming permet de traiter l'information en temps réel
  - Temps réel != latence
- Lisse la charge de travail
- Attention aux volumétries gérées, pensez scalabilité
- Pensez aux rollbacks et à l'observabilité

Merci !