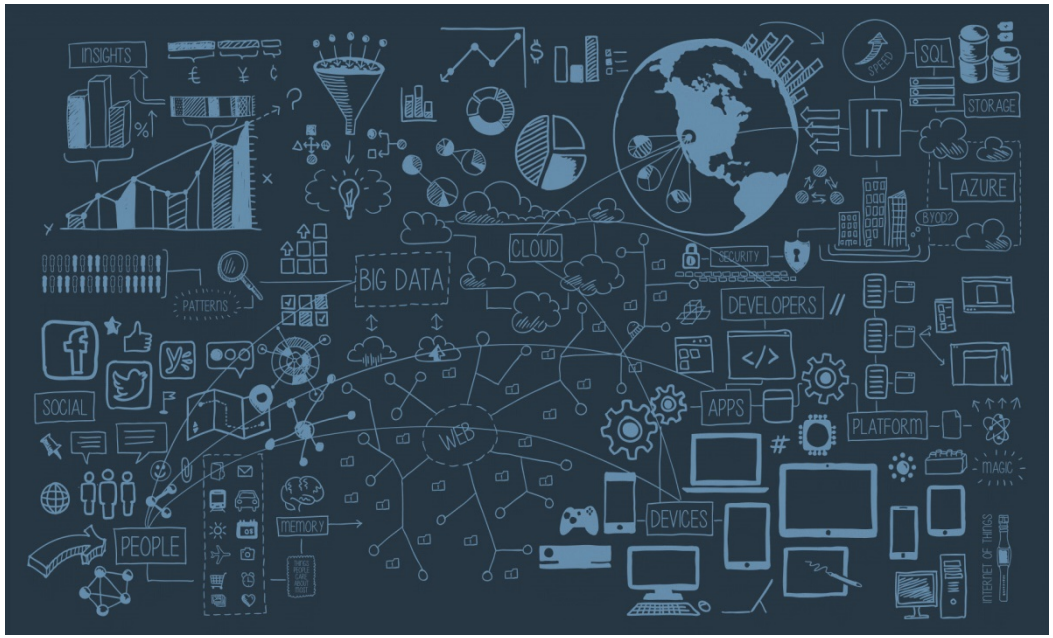
Arquitecturas para Aplicaciones de Big Data

Clase 6: Capa de Procesamiento y Análisis
Apache Flink

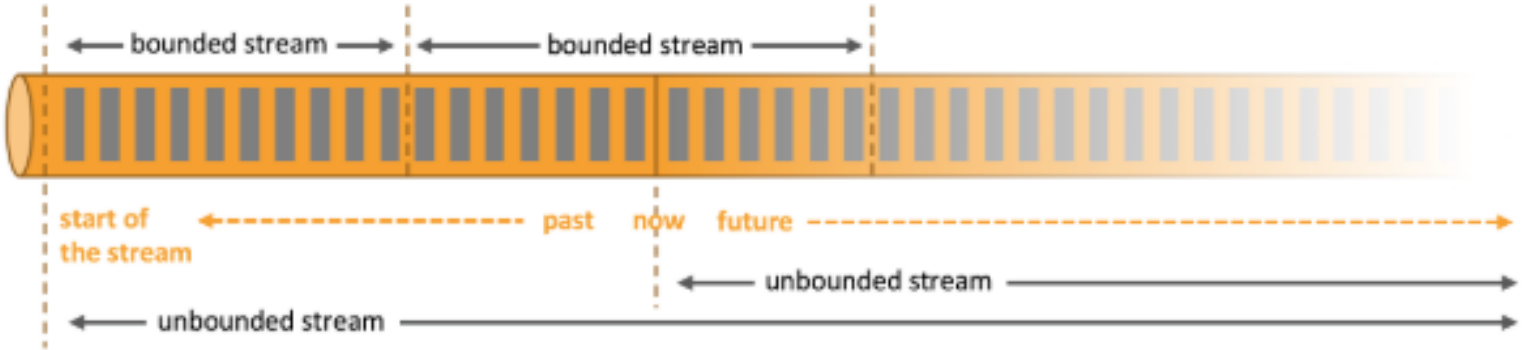


UNIVERSIDAD ORT Uruguay

¿Qué es Apache Flink?

- Es al mismo tiempo un framework y un motor de procesamiento distribuído para cálculos con estado sobre streams

- Los streams pueden ser “bounded” o “unbounded”, lo que habilita procesamiento en tiempo real o batch.



- bounded: tienen comienzo y fin. Pueden ser ordenados para ser procesados.

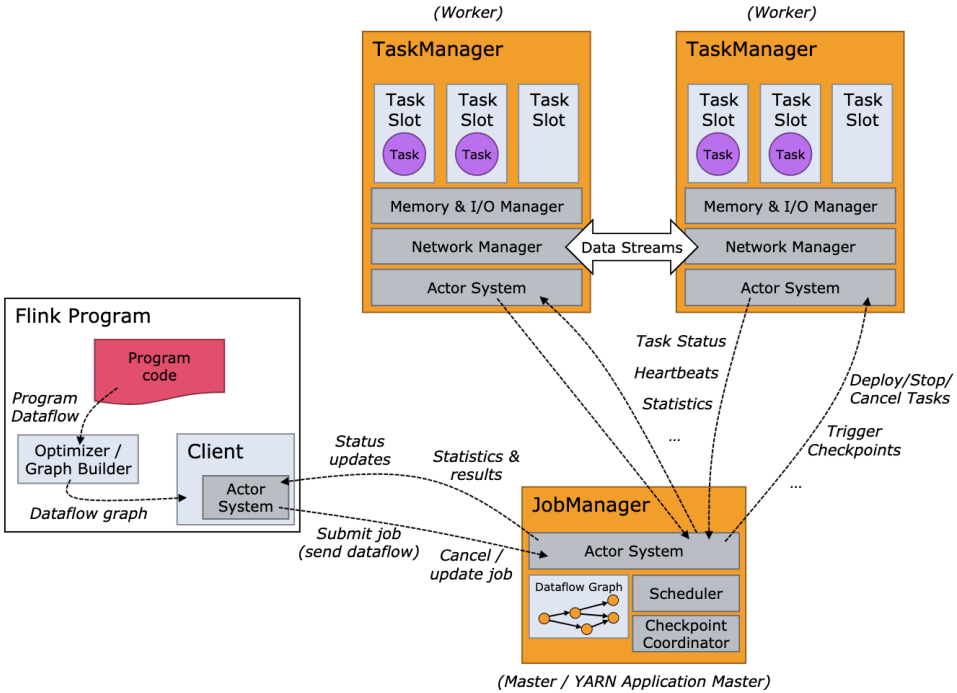
- Unbounded: tienen comienzo pero no fin, y deben ser procesados en forma continua.

¿Cómo funciona?

- Flink es un sistema distribuido, y requiere recursos de computación para ejecutar tareas.
- Puede integrarse con gestores de recursos (Hadoop yarn, Apache Mesos, Kubernetes) o ser configurado como un cluster por si mismo.
- Al desplegar una aplicación, Flink identifica los recursos necesarios basado en el nivel de paralelismo confiurado y los solicita al gestor de recursos.
- Está diseñado para ejecutar aplicaciones de streaming con estado, paralelizando las tareas y ejecutándolas en forma concurrente dentro del cluster.

¿CÓMO FUNCIONA?

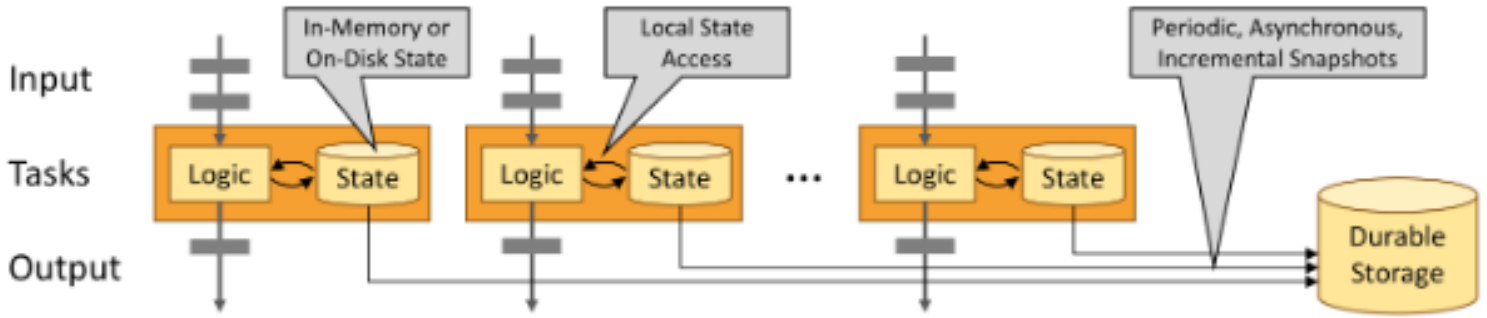
- JobManagers: Agendan tareas (tasks), coordinan checkpoints, recuperación de fallos, etc. Al menos debe haber 1
- TaskManagers: Son los “workers”. Ejecutan las tareas
- Cliente: preparan y envían las aplicaciones a ser ejecutadas. Se pueden desconectar o permanecer conectados para recibir updates.



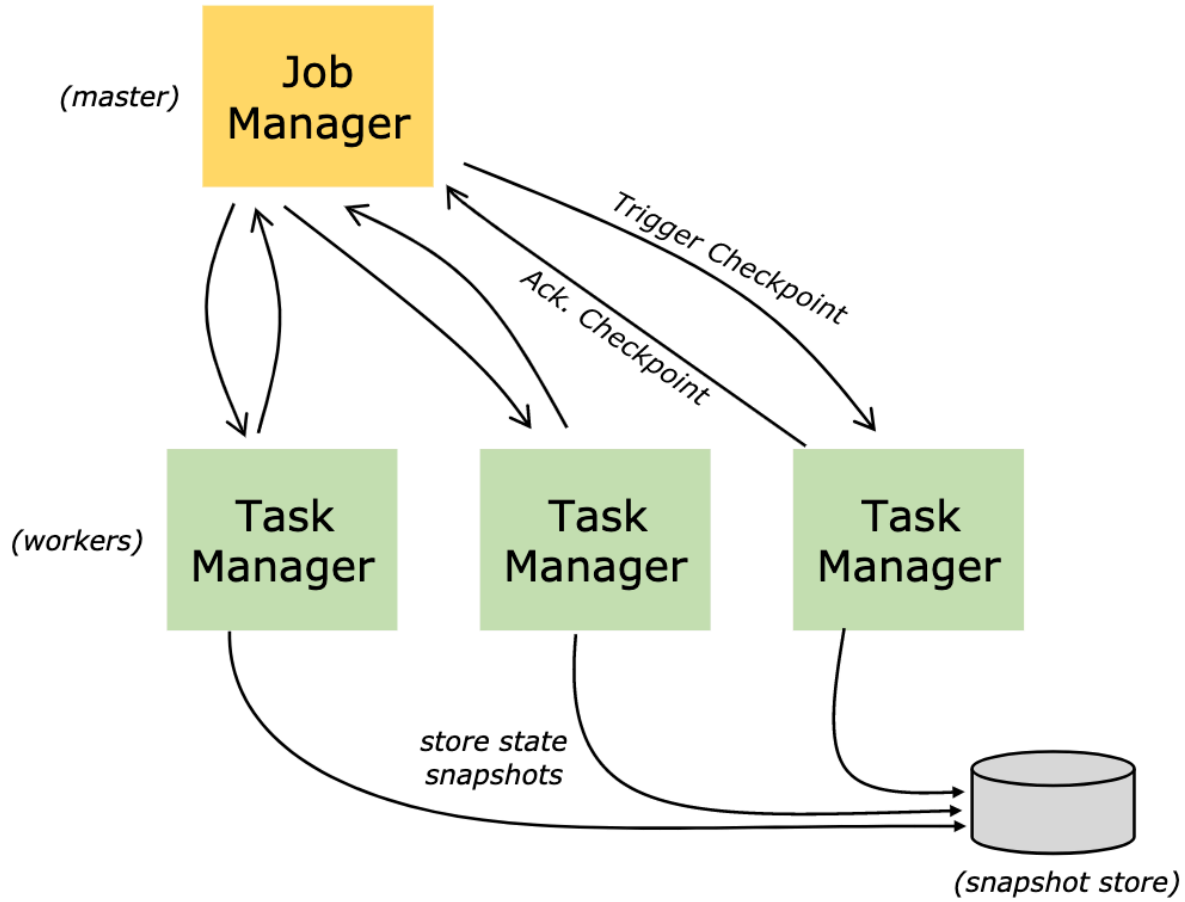
PERFORMANCE BASADA EN MEMORIA

- Está optimizado para que el estado de las aplicaciones sea accedido en forma local, preferentemente en memoria

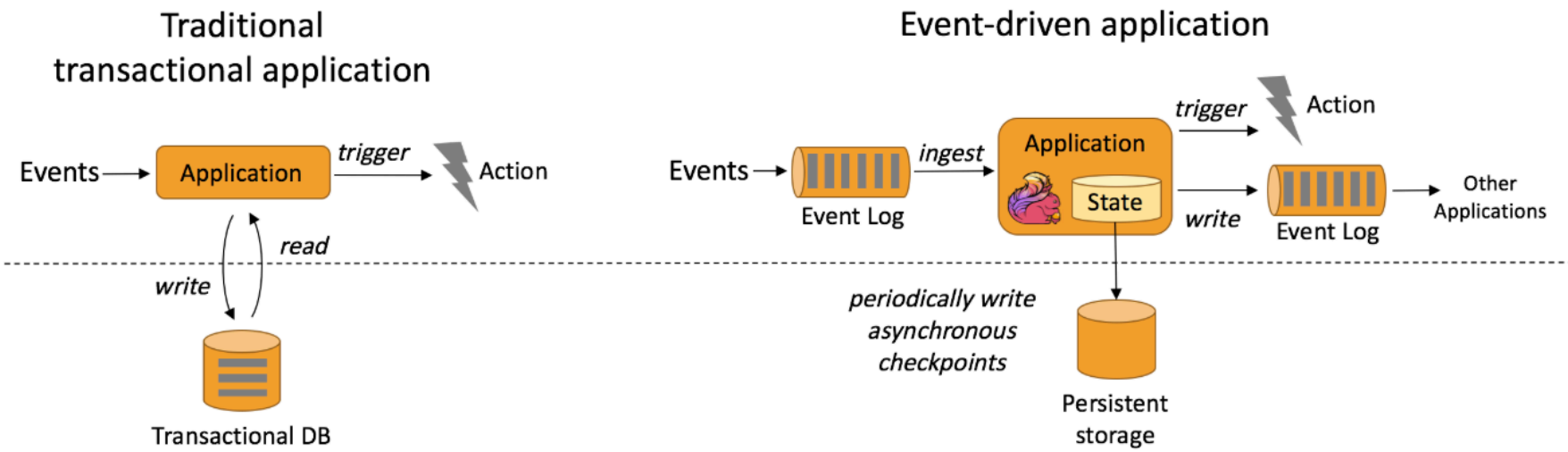
- Si el estado supera las capacidades de memoria, es posible utilizar mecanismos de almacenamiento de acceso eficiente o estructuras de datos en disco.



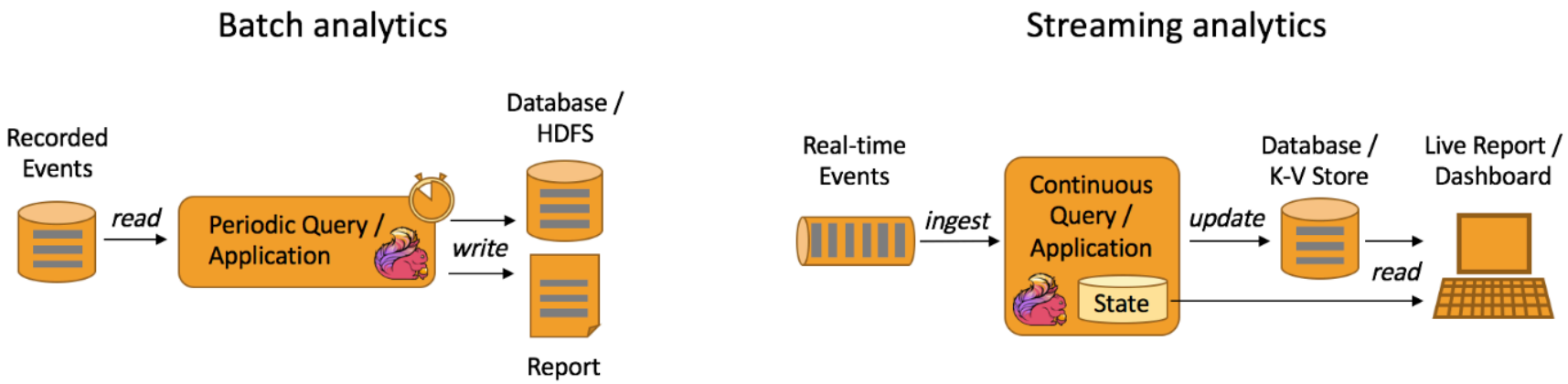
BACKENDS PARA GUARDAR ESTADO



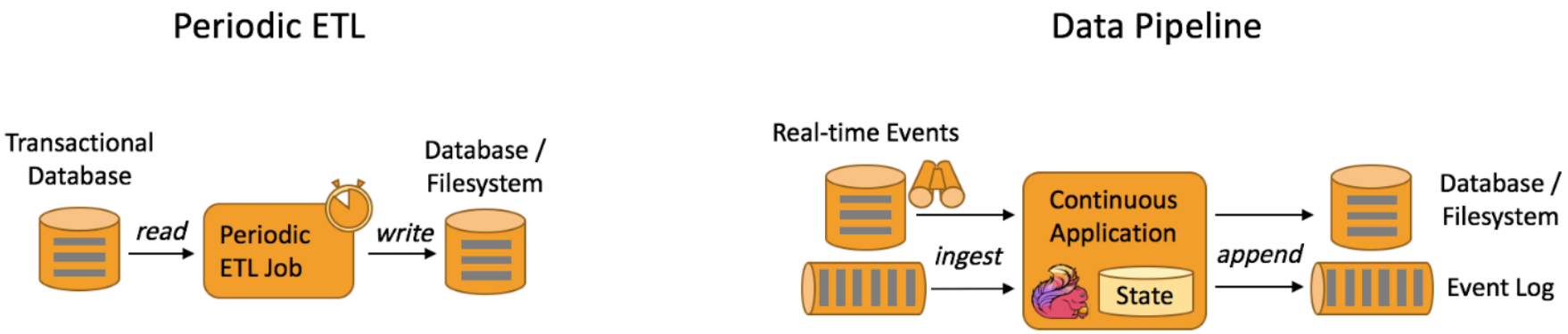
Casos de Uso: Event Drviven Apps



Casos de Uso: Data Analytics Apps



Casos de Uso: Data Pipline Apps



¿Quién lo Usa?


Alibaba.com


BetterCloud


bouygues
TELECOM


CapitalOne


COMCAST


D
TRB


ebay


ERICSSON


UBER


yelp

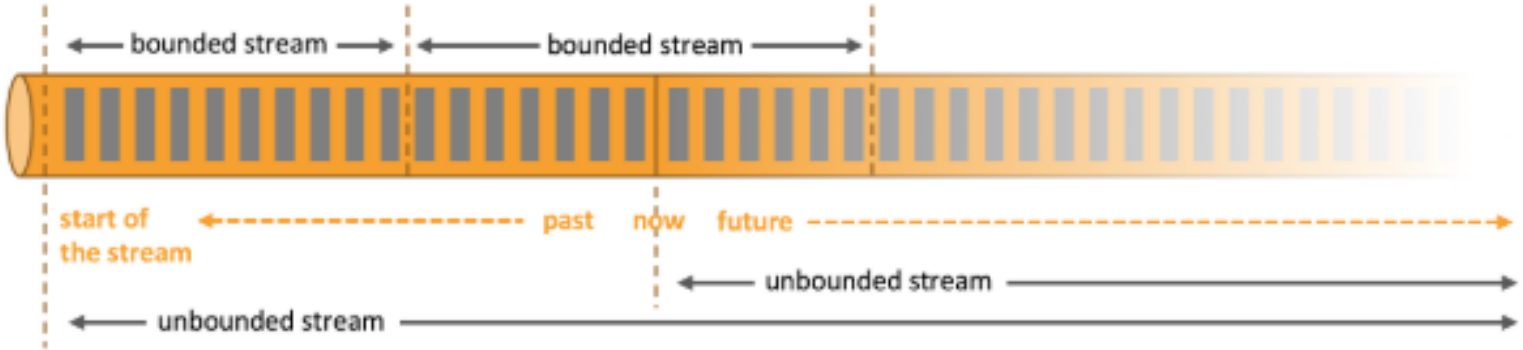

zalando

Conceptos Centrales en Flink: Streams

Streams

- Bounded vs. Unbounded
- Real-time vs. Recorded

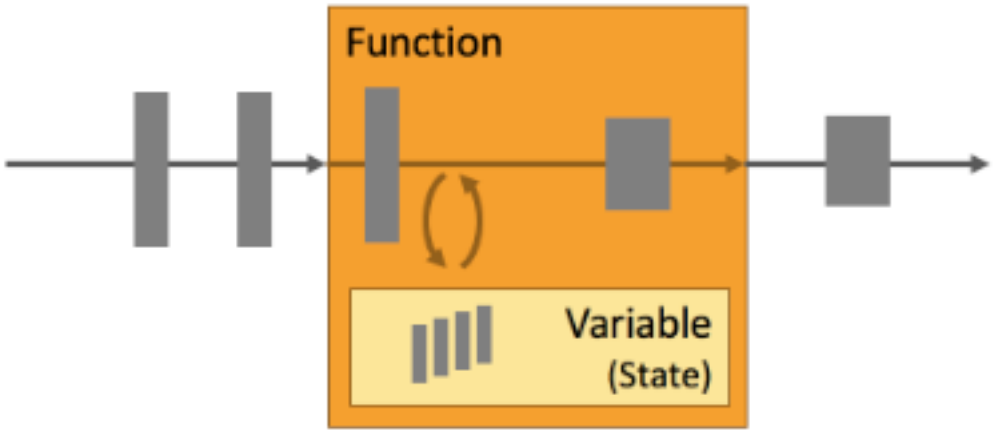
- Dependiendo de la naturaleza del stream, flink provee distintos mecanismos para procesarlos en forma eficiente



Conceptos Centrales en Flink: State

State:

- Primitivas para manejo de estado: valores atómicos, listas, mapas, etc.
- Backends para almacenamiento de estado “pluggables”
- Consistencia: algoritmos de checkpoint y recovery
- Estados muy grandes (varios TB) mediante algoritmo de checkpoint asíncrono.
- Escalabilidad: distribución del estado



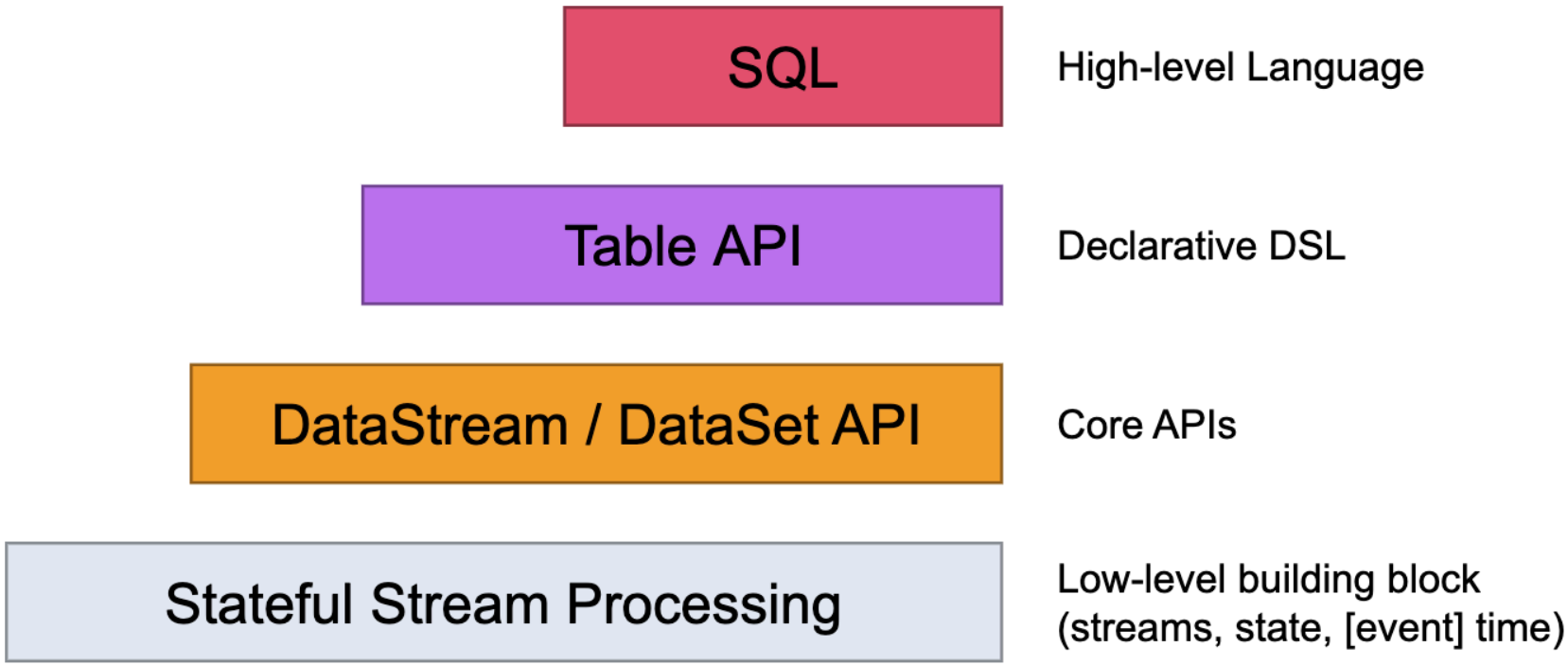
Conceptos Centrales en Flink: Time

- Time: la mayoría de las aplicaciones de streaming tienen semántica de tiempo inerente al tipo de eventos. Incluso muchos cálculos están basados en el tiempo, como la agregación en ventanas de tiempo, la identificación de sesiones, la detección de patrones, etc.
- Flink provee un conjunto de ”features” relativas al estado.

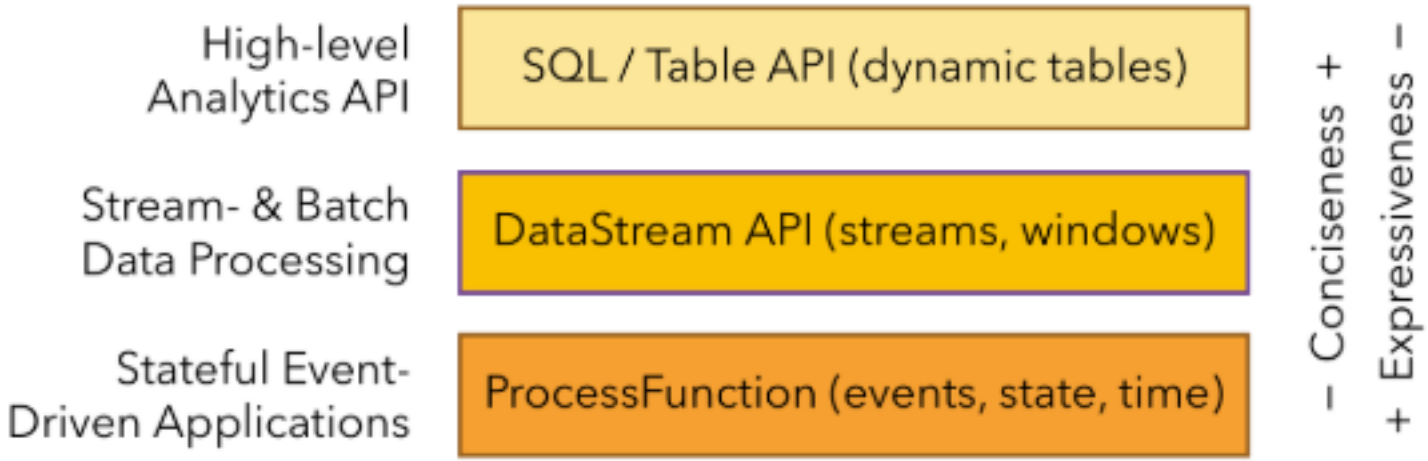
Flink is able to process streaming data based on different notions of *time*.

- *Processing time* refers to the system time of the machine (also known as “wall-clock time”) that is executing the respective operation.
- *Event time* refers to the processing of streaming data based on timestamps which are attached to each row. The timestamps can encode when an event happened.
- *Ingestion time* is the time that events enter Flink; internally, it is treated similarly to event time.

APIs EN CAPAS



APIs EN CAPAS



PROGRAMAS Y FLUJO DE DATOS

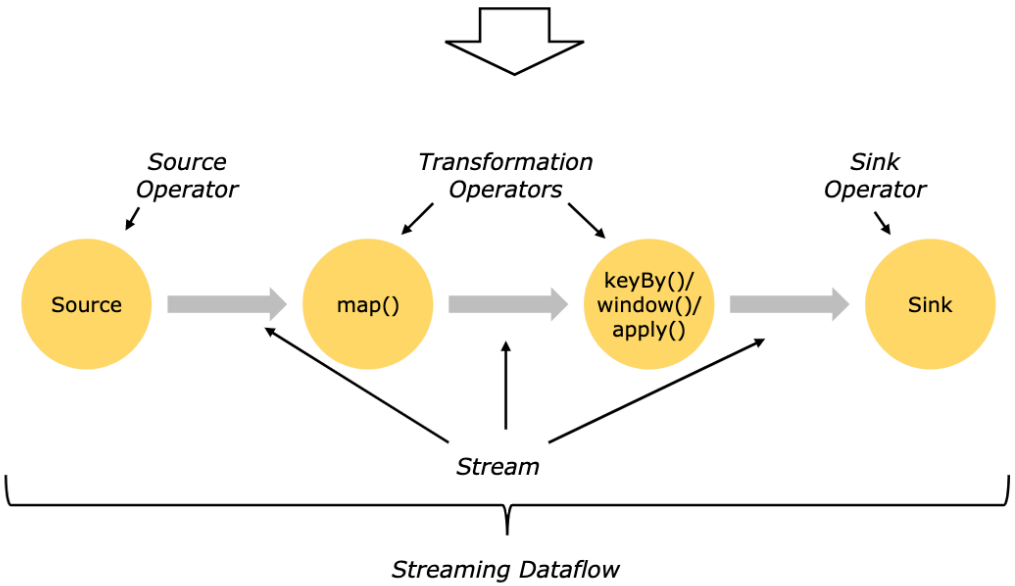
```
DataStream<String> lines = env.addSource(
                                new FlinkKafkaConsumer<>(…));

DataStream<Event> events = lines.map((line) -> parse(line));

DataStream<Statistics> stats = events
        .keyBy("id")
        .timeWindow(Time.seconds(10))
        .apply(new MyWindowAggregationFunction());

stats.addSink(new BucketingSink(path));
```

Source

Transformation

Transformation

Sink

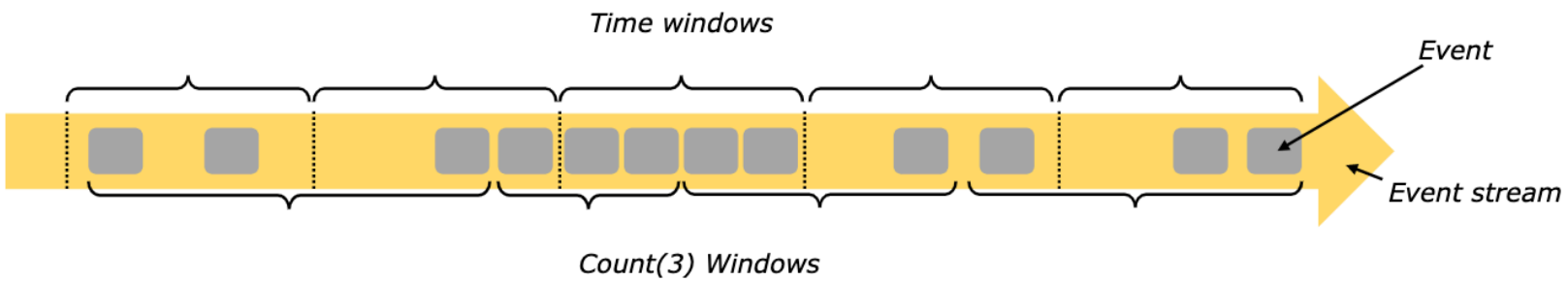


VENTANAS

Agregar eventos en streams es diferente a batch, porque no se tienen todos los datos. Flink propone el modelo de ventanas.

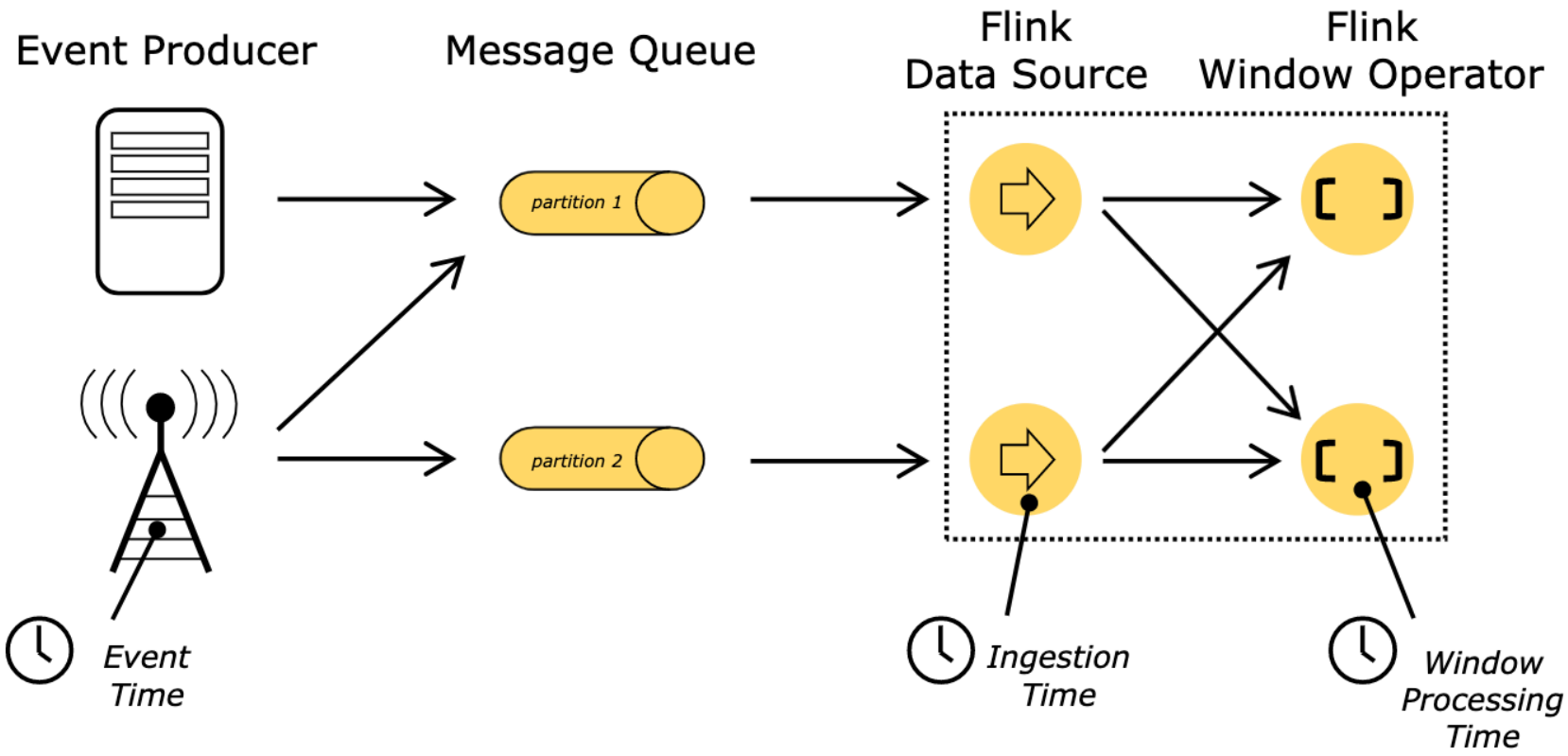
Pueden ser por tiempo (cada 30 segundos por ejemplo) o por datos (cada 100 elementos).

Distintos tipos de ventanas: Tumbling (sin solapamiento), sliding (con solapamiento, Session (delimitadas por períodos de inactividad).

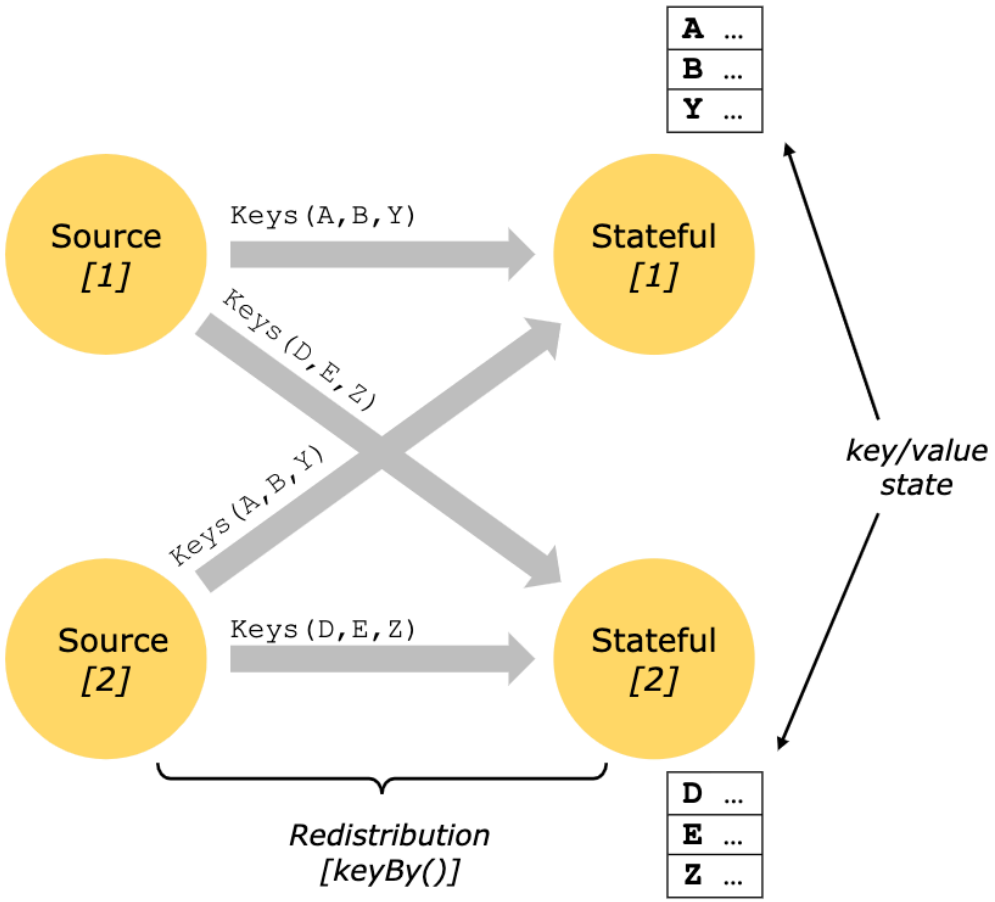


NOCIÓN DEL TIEMPO

Para hacer referencia al tiempo, por ejemplo para definer ventanas, Flink ofrece 3 posibilidades:



OPERACIONES CON ESTADO



1 – Process Functions

- El mecanismo de más bajo nivel, y por tanto, más expresivo.
- A nivel de código, procesan eventos individuales desde uno o más streams.
- Pueden modificar su estado y registrar “calbacks” para invocar funciones en el futuro.

```
public static class StartEndDuration
    extends KeyedProcessFunction<String, Tuple2<String, String>, Tuple2<String, Long>> {

  private ValueState<Long> startTime;

  @Override
  public void open(Configuration conf) {
    // obtain state handle
    startTime = getRuntimeContext()
      .getState(new ValueStateDescriptor<Long>("startTime", Long.class));
  }
```

```java
/** Called for each processed event. */
@Override
public void processElement(
    Tuple2<String, String> in,
    Context ctx,
    Collector<Tuple2<String, Long>> out) throws Exception {

  switch (in.f1) {
    case "START":
      // set the start time if we receive a start event.
      startTime.update(ctx.timestamp());
      // register a timer in four hours from the start event.
      ctx.timerService()
        .registerEventTimeTimer(ctx.timestamp() + 4 * 60 * 60 * 1000);
      break;
    case "END":
      // emit the duration between start and end event
      Long sTime = startTime.value();
      if (sTime != null) {
        out.collect(Tuple2.of(in.f0, ctx.timestamp() - sTime));
        // clear the state
        startTime.clear();
      }
    default:
      // do nothing
  }
}
```

```java
/** Called when a timer fires. */
@Override
public void onTimer(
    long timestamp,
    OnTimerContext ctx,
    Collector<Tuple2<String, Long>> out) {

  // Timeout interval exceeded. Cleaning up the state.
  startTime.clear();
}
```

2 – DataStream API

- Ofrece primitivas para operaciones de procesamiento común, como windowing, consultas a almacenamientos externos o transformaciones de eventos.

```
// a stream of website clicks
DataStream<Click> clicks = ...

DataStream<Tuple2<String, Long>> result = clicks
  // project clicks to userId and add a 1 for counting
  .map(
    // define function by implementing the MapFunction interface.
    new MapFunction<Click, Tuple2<String, Long>>() {
      @Override
      public Tuple2<String, Long> map(Click click) {
        return Tuple2.of(click.userId, 1L);
      }
    })
  // key by userId (field 0)
  .keyBy(0)
  // define session window with 30 minute gap
  .window(EventTimeSessionWindows.withGap(Time.minutes(30L)))
  // count clicks per session. Define function as lambda function.
  .reduce((a, b) -> Tuple2.of(a.f0, a.f1 + b.f1));
```

2 – SQL & TABLE API

- Apis para uso relacional sobre streams, sin importar el tipo (modelo consistente).

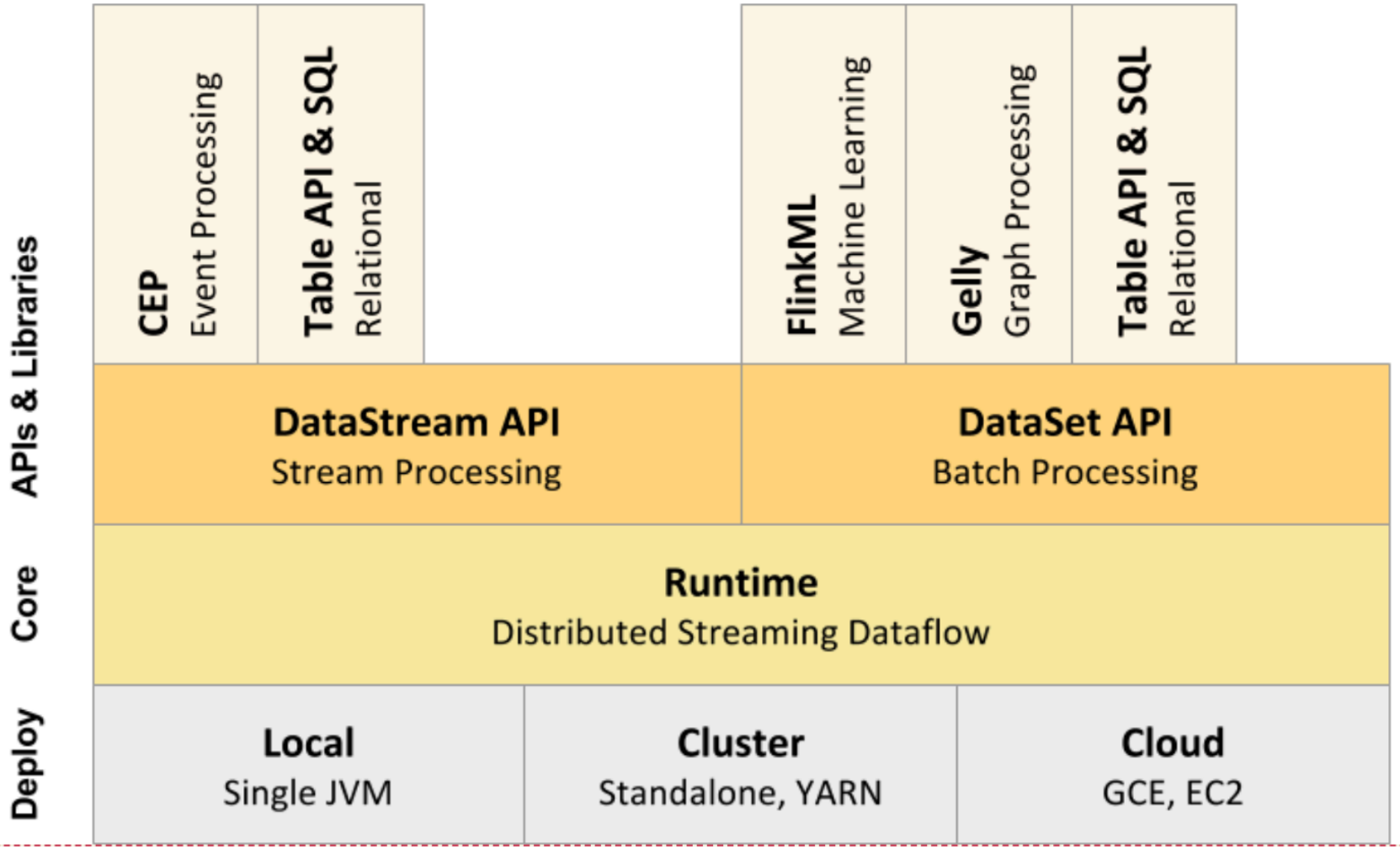
```
SELECT userId, COUNT(*)
FROM clicks
GROUP BY SESSION(clicktime, INTERVAL '30' MINUTE), userId
```

LIBRERÍAS

Flink provee librerías con implementaciones de casos de uso de procesamientos de datos comunes:

- Complex Event Processing (CEP): Provee detección de patrones integrada al DataStream API.
- DataSet API: Core API para procesamiento batch. Incluye primitivas como map, reduce, join group e iterate.
- Gelly: Procesamiento y análisis de grafos.

STACK



Demo 1

- Process Functions
- Mismo caso que con Storm (contar palabras)
- Cluster local, con un solo nodo
- Leemos palabras desde socket y contamos

- Enviar job a la instalación local

- Precisamos una nueva máquina virutal (Big Data 3) desde:

https://drive.google.com/open?id=1rK2rJF98UhEIc0eyUkHiPnSAXTqejU9p

Demo 2

- Process Functions + Datastream API
- Mismo caso que con Storm (contar palabras) con ventanas de tiempo
- Cluster local, con un solo nodo
- Leemos palabras desde socket y contamos (ventana de 5 segundos)

DEMO 3

- SQL API