

EDIT.

REAL TIME STREAMING

junho 2025

SOBRE MIM

- Apaixonado por dados, tecnologia e por encontrar soluções que realmente fazem a diferença.
- Passei por vários processos de ETL, criação de dashboards, automações e bases de dados em ferramentas que ajudam grandes empresas (como bancos e energéticas) a tomarem decisões melhores e mais rápidas.
- Curiosidades?



SOBRE VOCÊS



Gabriel Dantas



Ruben Sales



**Ricardo Espírito
Santo**



Tiago Canarias



Nuno Fonseca



Daniel Celac



Rodolfo Fernades

AGENDA

18/06

19h-23h

- Introdução ao Processamento de Dados em Streaming;
- Arquitetura Pub/Sub e Brokers de Eventos;
- Apache Kafka — Fundamentos;
- Ferramentas de Ingestão com Kafka (CLI);
- Exercício Prático com Kafka (CLI).

23/05

19h-23h

- Introdução ao Apache Flink;
- SQL Streaming em Flink
- Event-Time e Dados Fora de Ordem;
- Flink com Kafka: Pipeline Completa;
- Exercício Prático com Flink SQL + Kafka

DATA STREAMING

Ou “transmissão de dados” engloba a captura, tratamento e análise **contínua** dos dados à medida que são gerados.



BATCH

Frequência Diária, Horária, Semanal

Latência Alta, minutos ou horas

Grande volume de dados tratados **em lotes**

Custo menos elevado

Complexidade menor, mais fácil de implementar

STREAMING

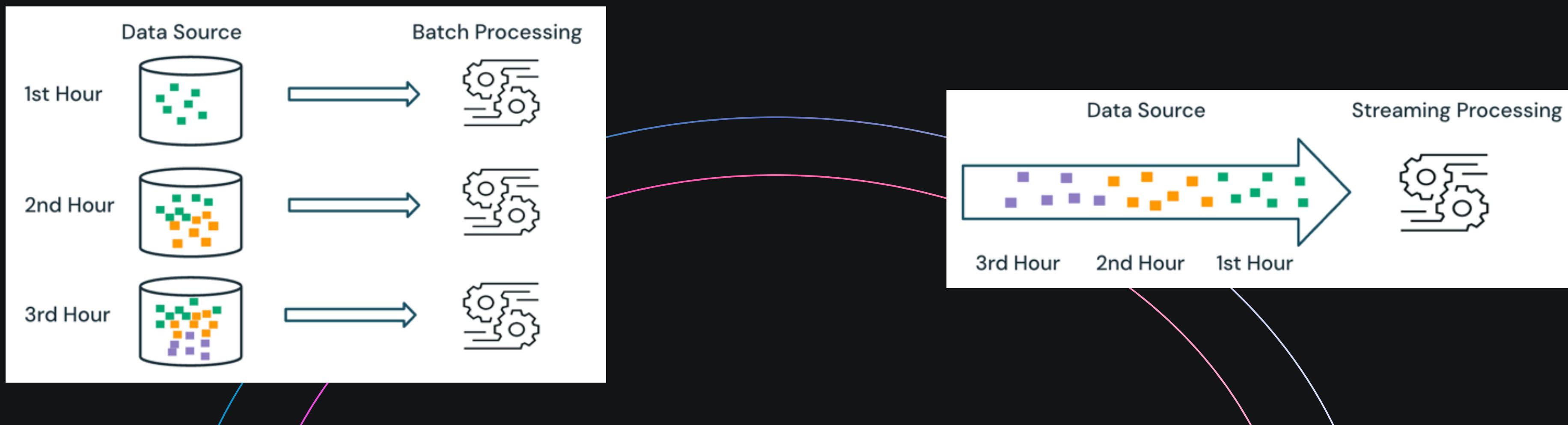
Frequência Contínua

Latência Baixa, segundos ou milisegundos

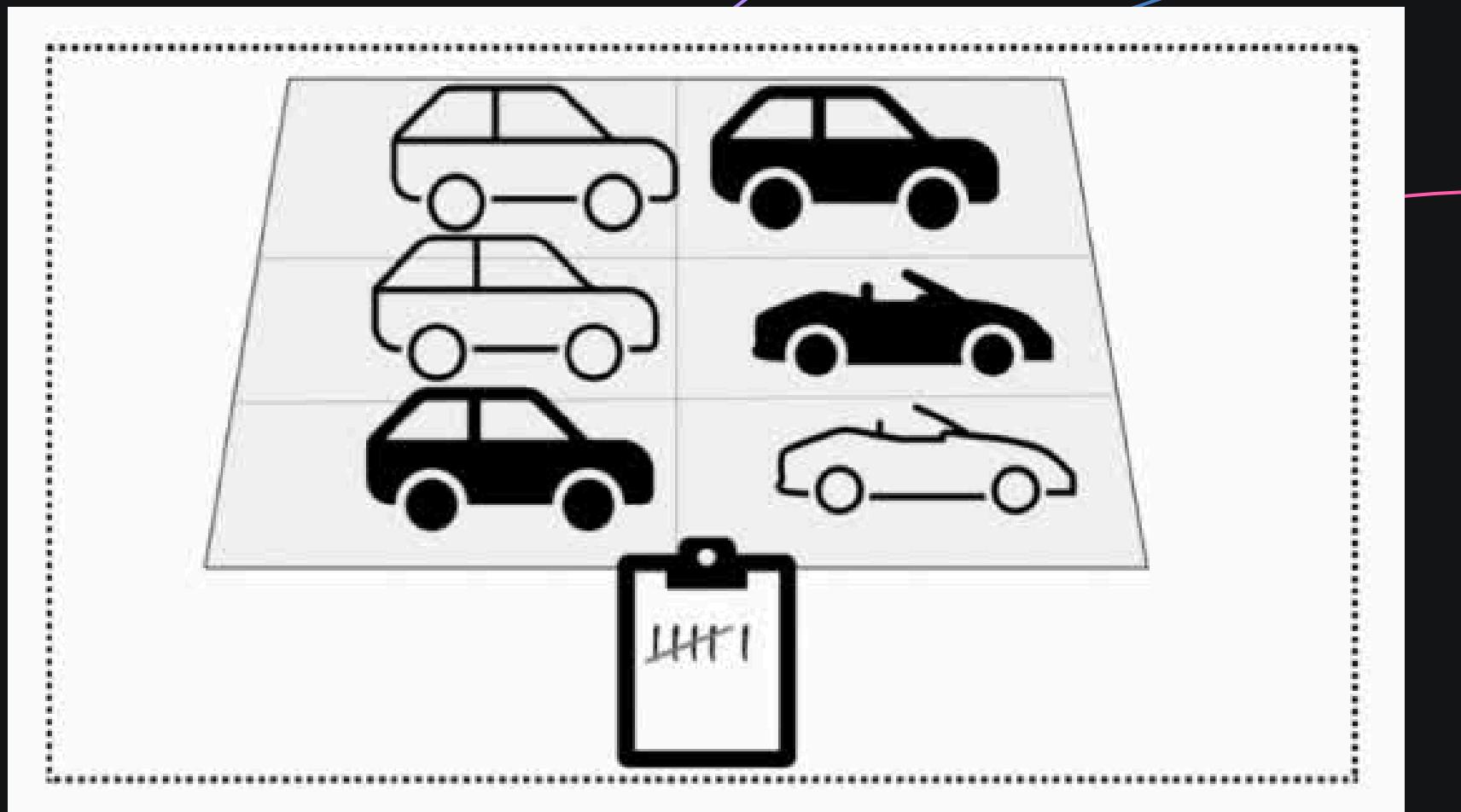
Grande volume de dados tratados on-time

Custo mais elevado

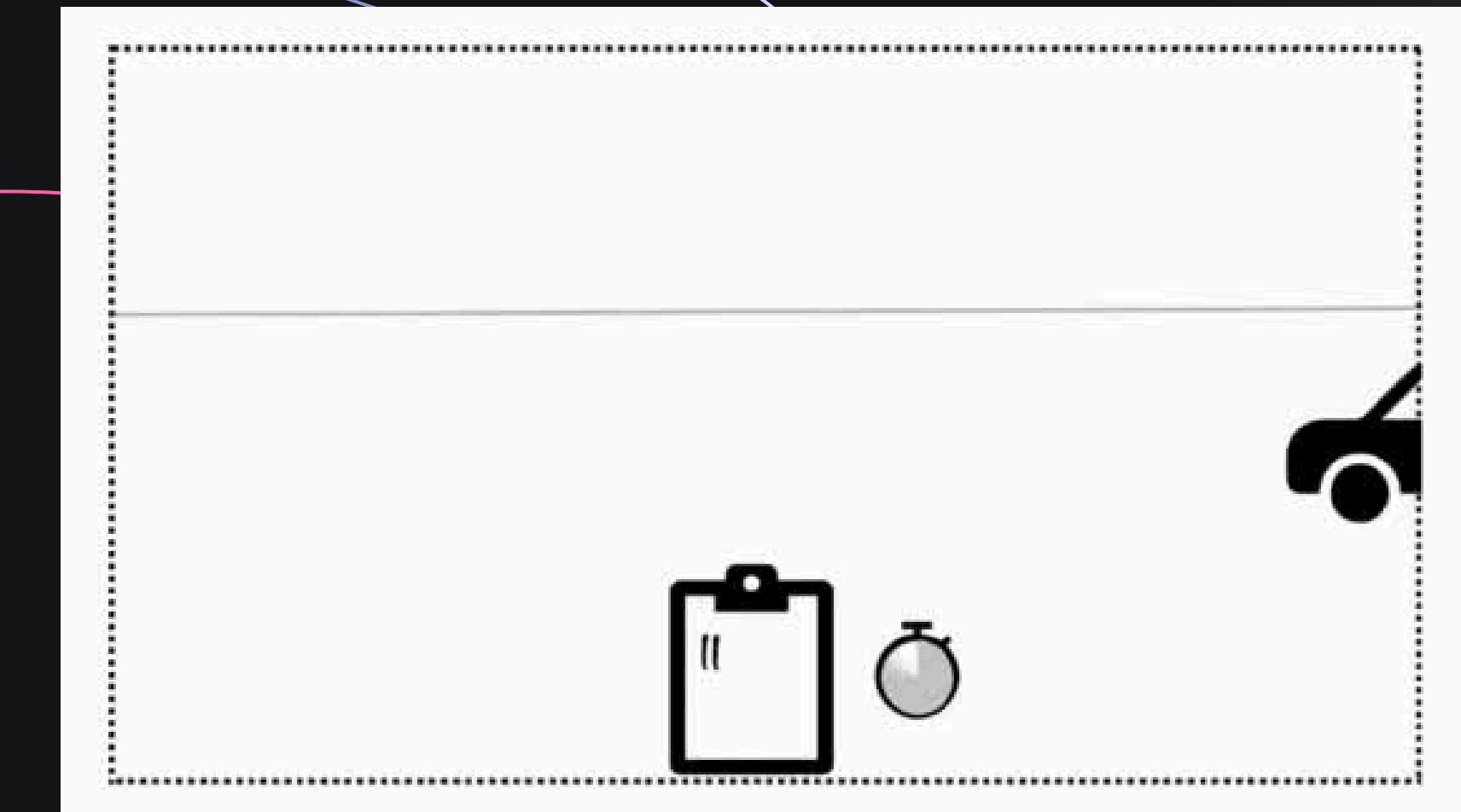
Complexidade maior, arquiteturas mais resilientes e complexas



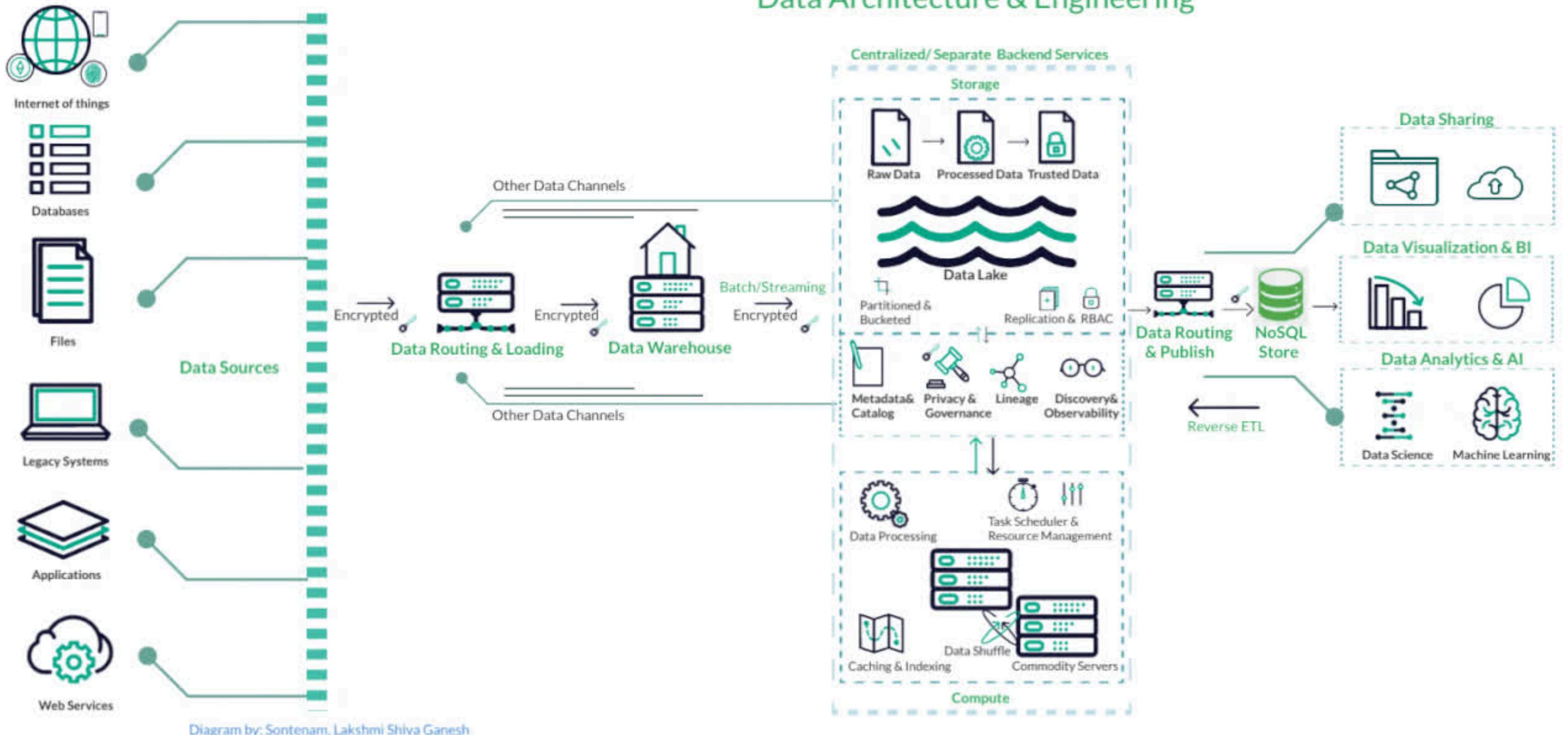
BATCH



STREAMING



Data Architecture & Engineering



REAL USE CASES

- Deteção de fraude;
- IoT;
- Sistemas de recomendações online;
- Monitorizar aplicações/logs;
- Supply Chain e Logistica;
- Migração de base de dados;
- ...

TECNOLOGIAS

KAFKA

Plataforma open-source de streaming de eventos utilizada por empresas para publicar, armazenar e processar dados em tempo real

Kafka Streams;
Integração com Flink, Spark,
Hadoop;
Alta performance



FLINK

Framework especializada para processamentos de dados continuos, foca-se mais em **aplicações de alta disponibilidade**



SPARK

Framework desenhada tanto para processamento em lote (batch) ou streaming, com menor latência.
Arquitetura micro-batches

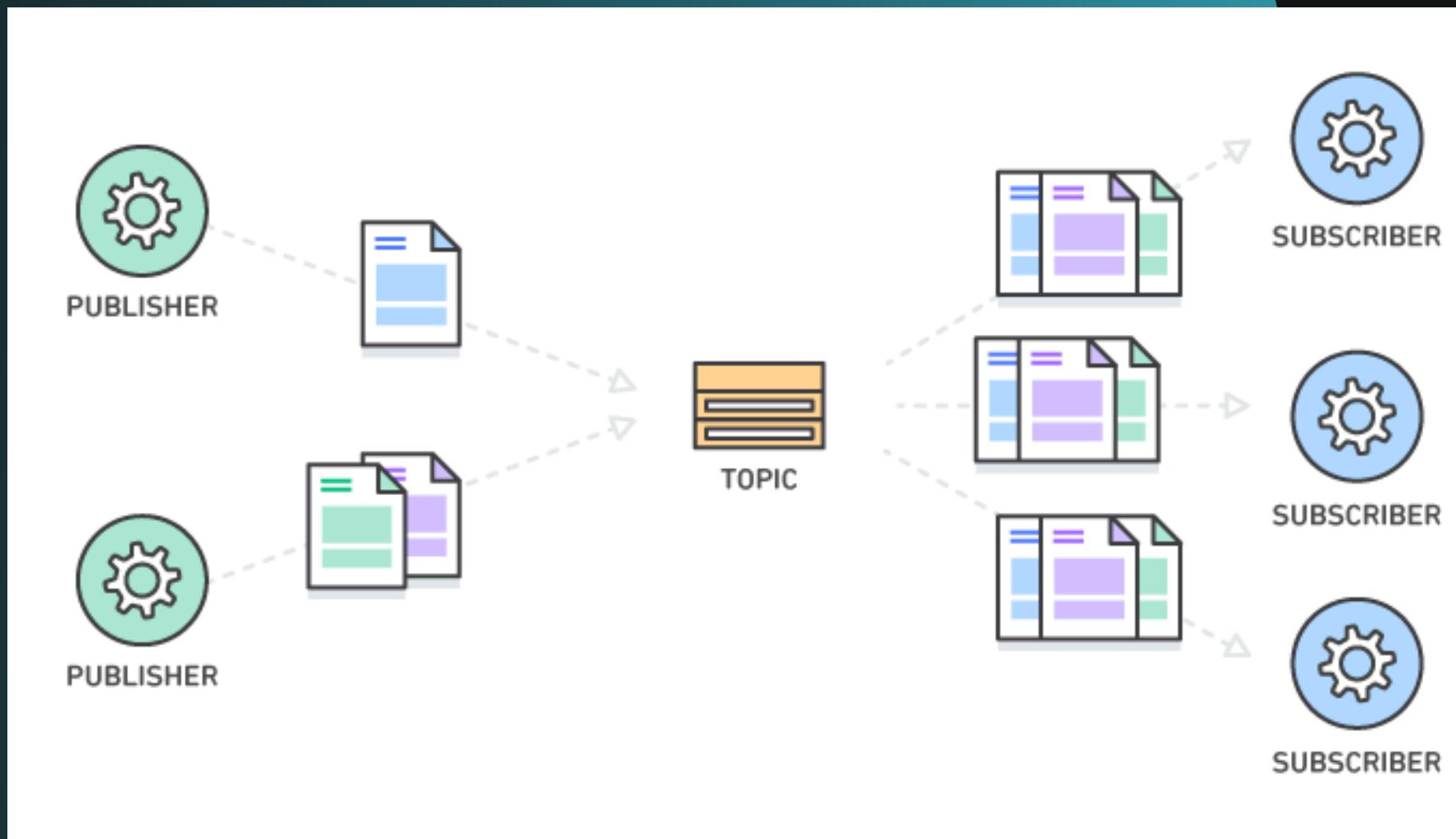


AMAZON KINESIS

Conjunto de serviços, fornecidos pela Amazon Web Services, utilizados para processar e analisar dados em tempo real



ARQUITETURA PUB/SUB



1

Publicadores

- Quem produz as mensagens que serão direcionadas a um assinante.
- Não tem conhecimento dos assinantes.

2

Tópicos

- As mensagens derivadas de publicadores são enviadas para um tópico específico onde armazena a informação.

3

Assinantes

- Recebem as mensagens dependendo dos tópicos que estiverem inscritos.
- As mensagens podem ser processadas de maneiras independentes.

4

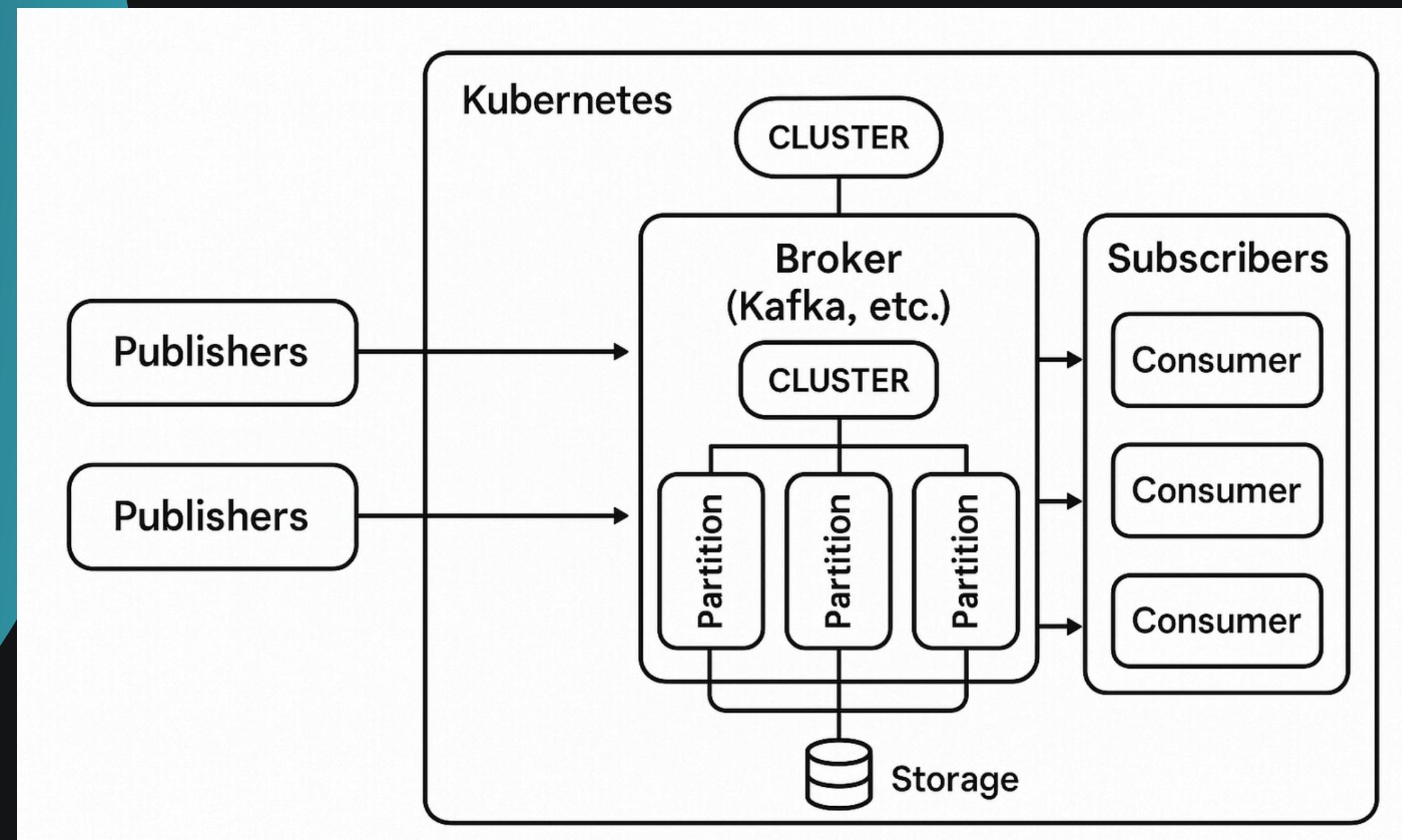
Broker

- Sistema/Framework intermediário responsável pela classificação dos tópicos, distribuição de mensagens e garantia de entrega,
- Organiza, Filtra, Entrega, Garante

ARQUITETURA PUB/SUB

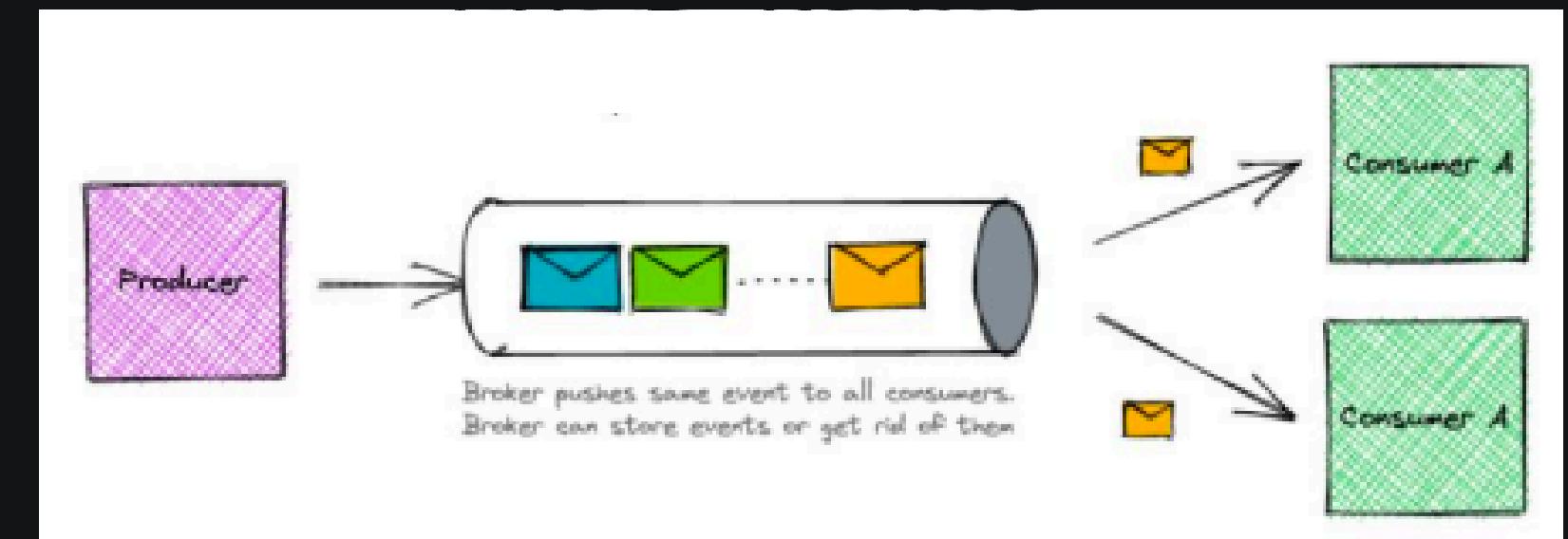
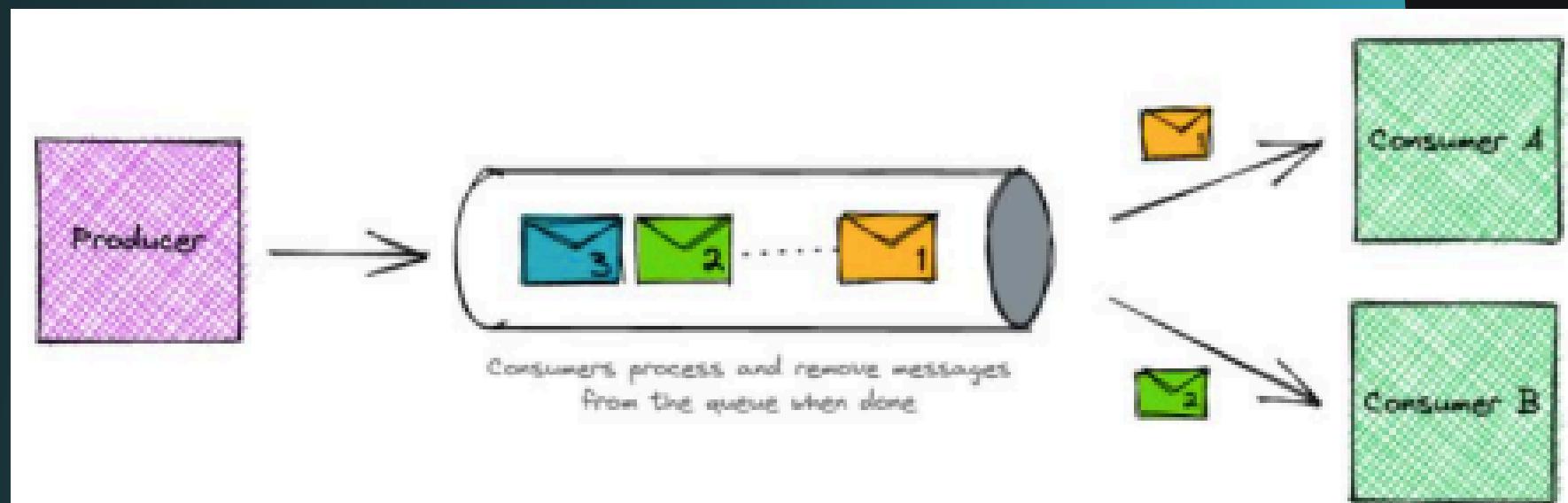
PRINCIPAIS CARACTERÍSTICAS

1. Desacoplamento
2. Comunicação assíncrona
3. 1 para N
4. Arquitetura escalável (cluster, kubernetes)
5. Resiliência/Tolerância a falhas



ARQUITETURA PUB/SUB

FILA DE MENSAGENS vs BROKER DE EVENTOS



RabbitMQ



amazon
SQS

kafka



amazon
KINESIS

APACHE kafka

INTRODUÇÃO

1. Apache Kafka — Fundamentos

- a. cluster, brokers, tópicos e partições;
- b. offsets;
- c. Garantias: at-most-once, at-least-once, exactly-once;

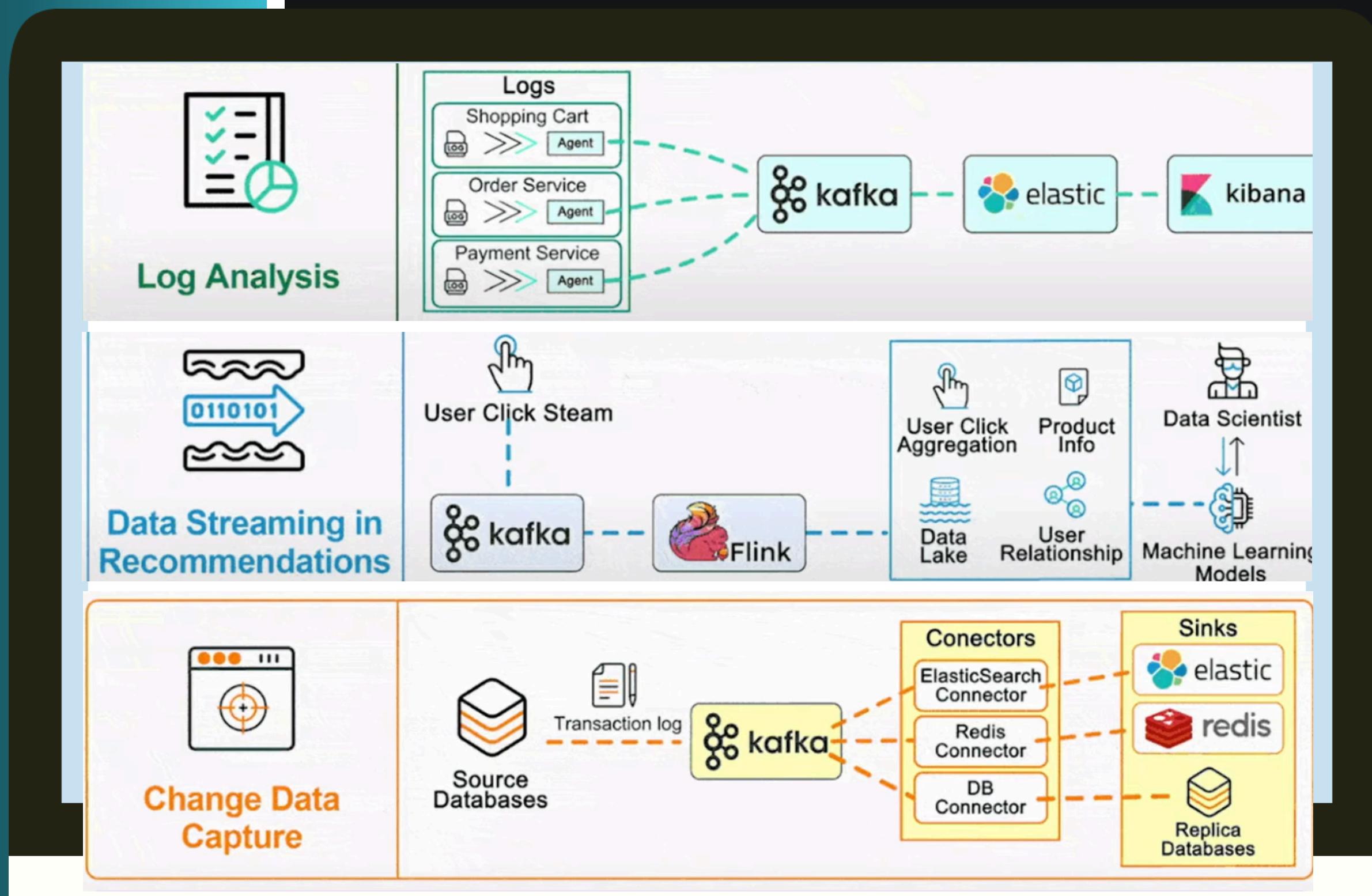
2. Ferramentas de Ingestão com Kafka (CLI)

- a. Criação de tópicos: kafka-topics.sh;
- b. Produção e consumo: kafka-console-producer.sh, kafka-console-consumer.sh;
- c. Monitorização de consumo: kafka-consumer-groups.sh;

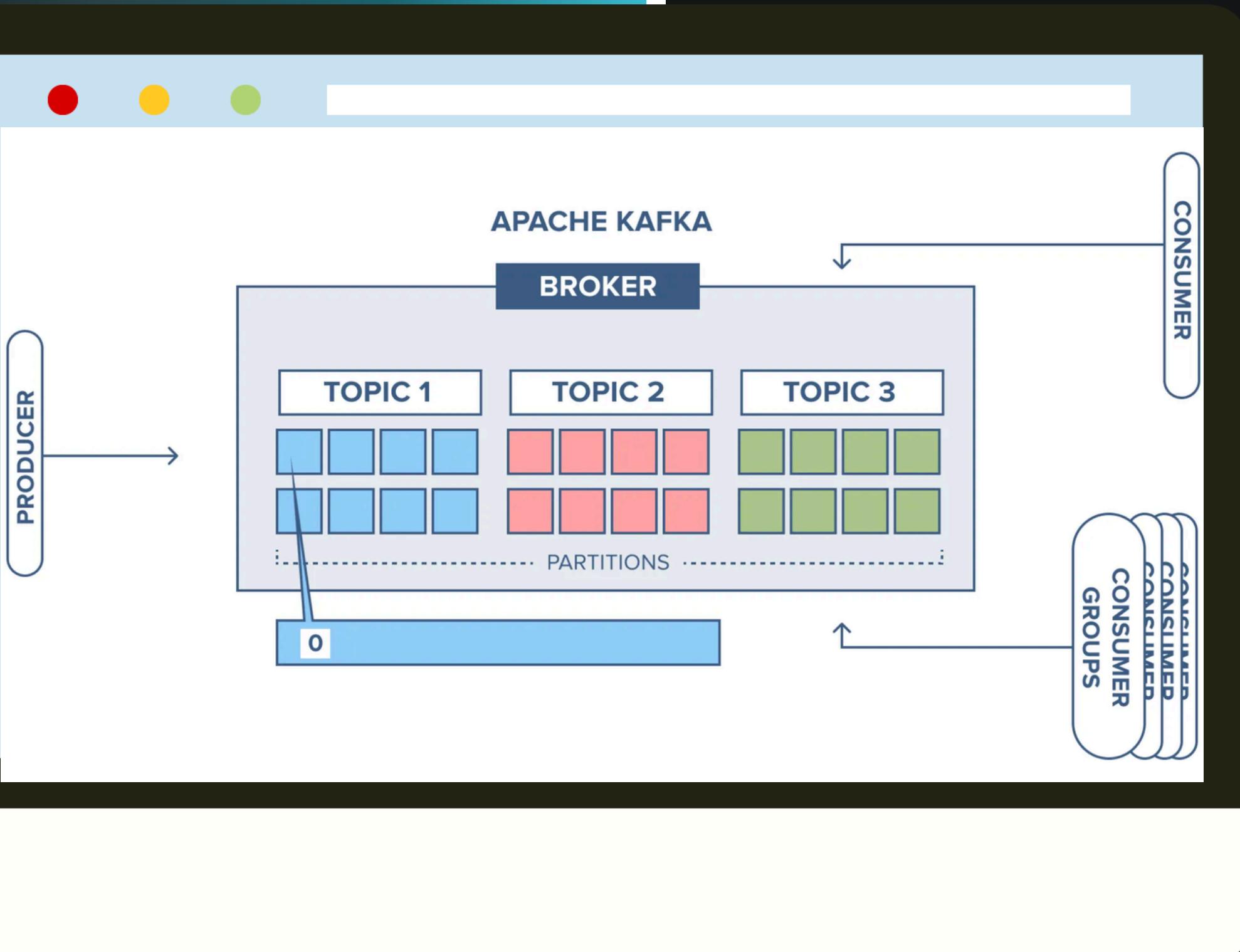
3. Exercício Prático com Kafka (CLI)

APACHE kafka

KAFKA USE CASES



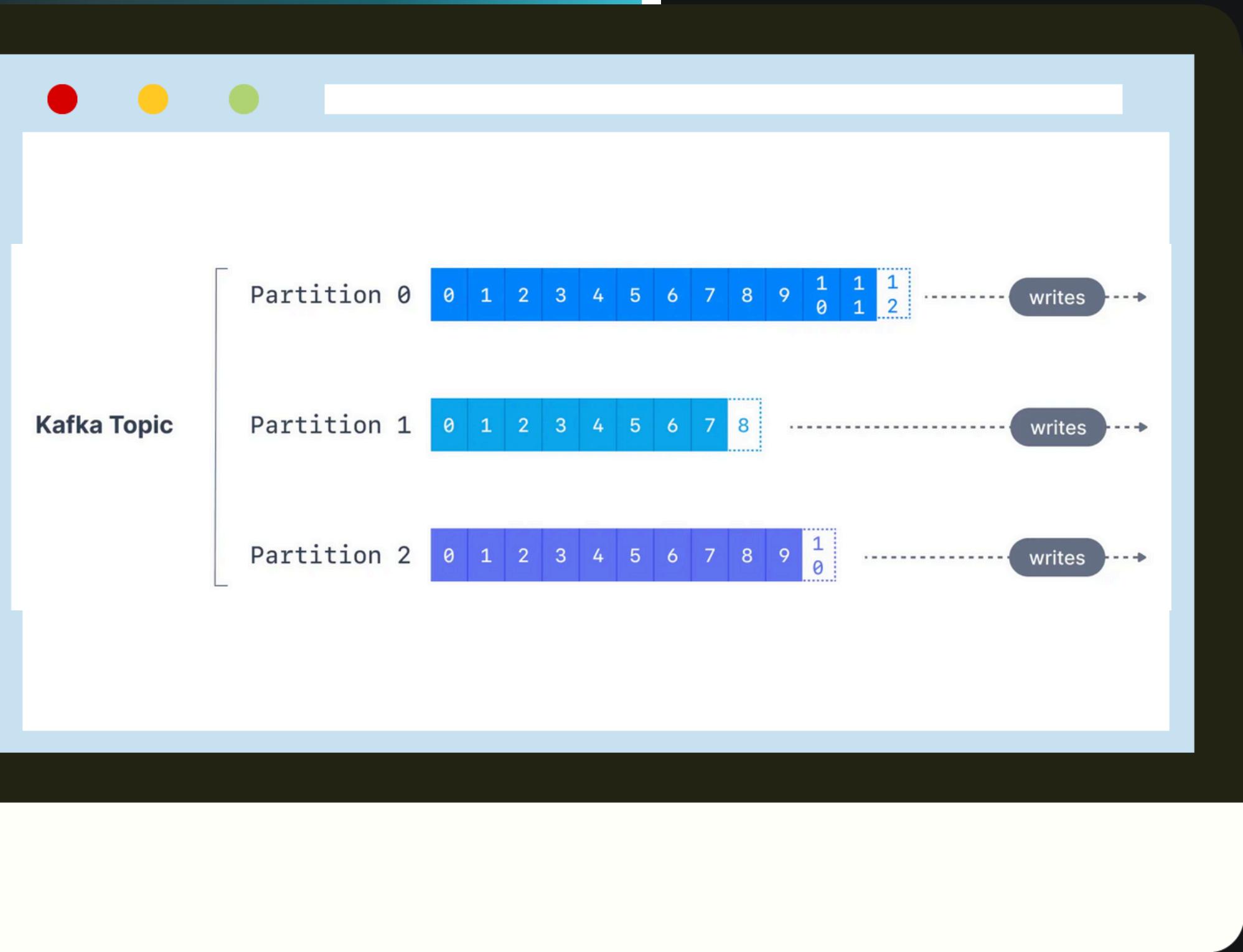
APACHE FUNDAMENTOS



- **TOPIC**
- **PARTITIONS**
- **BROKER**
- **PRODUCER**
- **CONSUMER**
- **OFFSET**

APACHE

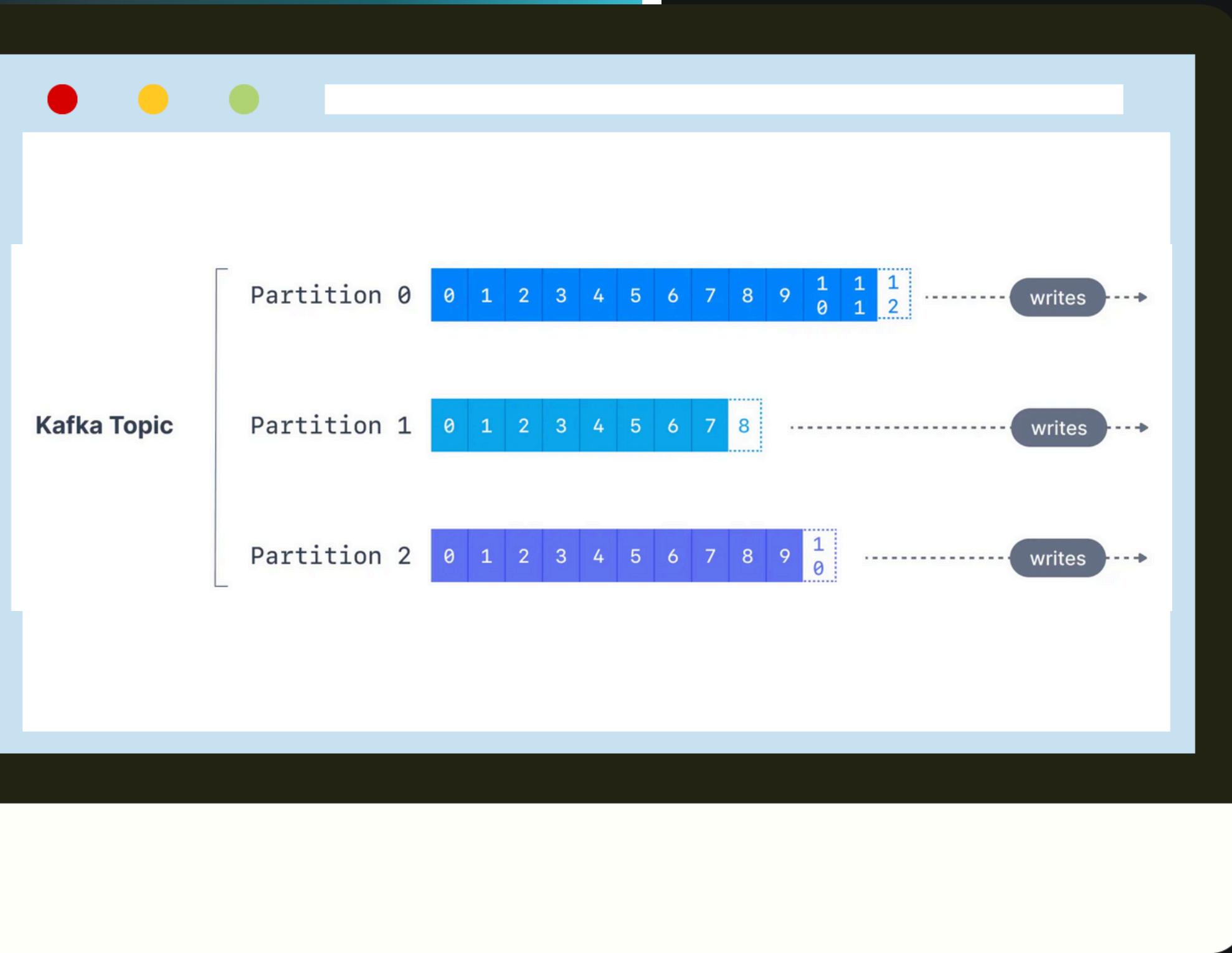
FUNDAMENTOS



- **TOPIC**
- **PARTITIONS**
- **OFFSET**

APACHE

FUNDAMENTOS



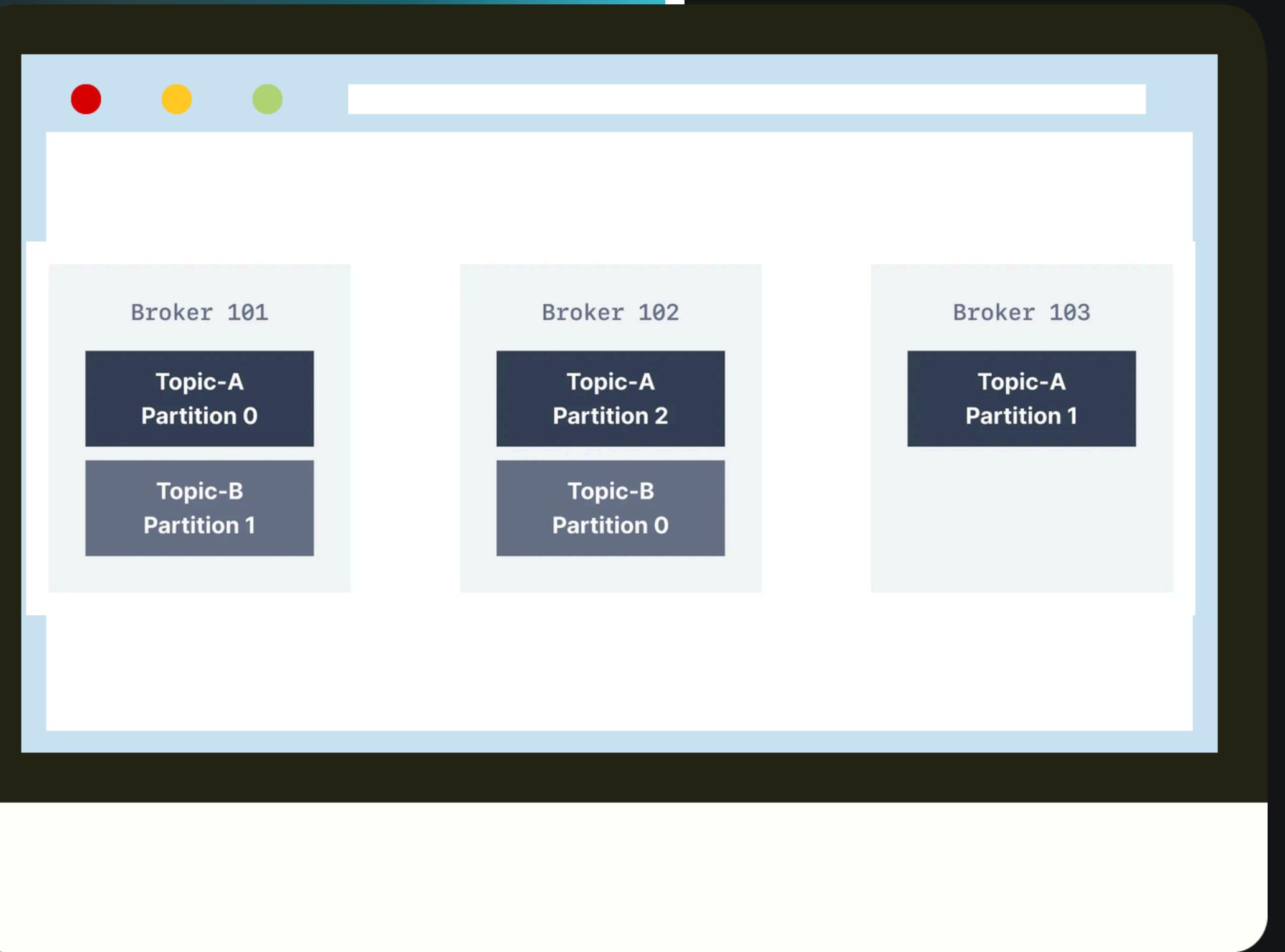
• TOPIC

<message type>. <dataset name>. <data name>

- **logging** (para data logs)
`logging.app1.errors`
- **queuing**
`queuing.email.send`
- **tracking** (monotrizar eventos, clicks, views, etc)
`tracking.website.clicks`
- **etl/db** (para ETL ou monitorizar base de dados)
- **streaming**
- **push** (dados de batch para streaming)
- **user** (para projetos pessoais ou testes)

APACHE

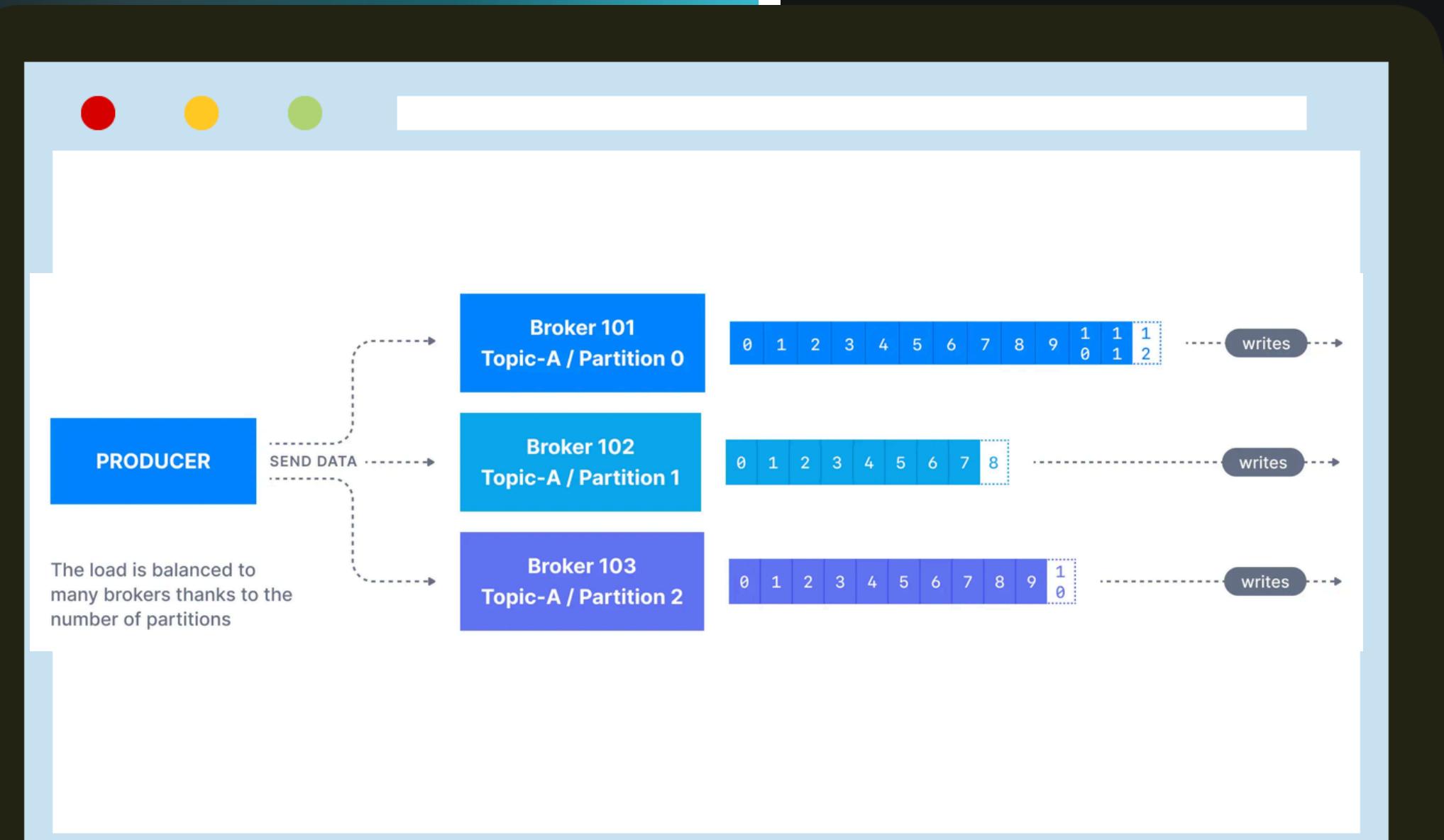
FUNDAMENTOS



- **BROKER**

APACHE

FUNDAMENTOS



- **PRODUCER**

APACHE kafka

FUNDAMENTOS

ENTREGA	DESCRÍÇÃO	CARACTERÍSTICAS
At least once	<ul style="list-style-type: none">• Cada mensagem é processada contudo pode ser processada mais do que uma vez.• Offset é atribuído antes do processamento;	Possível duplicação
At most once	<ul style="list-style-type: none">• Cada mensagem pode ou não ser processada. Quando processada, apenas é processada uma vez;• Offset é atribuído após o processamento;	Possível perca de dados
Exactly once	<ul style="list-style-type: none">• Cada mensagem é processada uma única vez.	Sem duplicação e sem perca de dados

APACHE

FERRAMENTAS DE INGESTÃO CLI



```
kafka-topics.sh --bootstrap-server localhost:9092 --topic topic1 --create --  
partitions 3 --replication-factor 1
```

CRIAÇÃO DE TÓPICOS

kafka-topics.sh

- Ferramenta que permite criar tópicos (`topic1`)
- Comandos:
 - `--topic`;
 - `--partitions`;
 - `--replication-factor` ;

```
Kafka-console-producer.sh --bootstrap-server localhost:9092 --topic my-topic
```

```
kafka-console-producer.sh --bootstrap-server localhost:9092 --topic my-topic < example.txt
```

PRODUÇÃO DE MENSAGENS

kafka-console-producer.sh

- Ferramenta que lê dados e envia para um tópico específico do KAFKA;
- Podem enviar mensagens ou arquivos (ex: txt, csv, etc)



```
kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic my-topic
```

```
Kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic my_topic --from-beginning
```

CONSUMO DE MENSAGENS

kafka-console-consumer.sh

- Ferramenta que lê as mensagens dos tópicos e mostra no terminal;
- Comandos:
 - **--from beginning**;
 - **--property**;
 - **--property print.key=true and --property print.value=true** ;
 - **--consumer-property group.id=**;

APACHE

FERRAMENTAS DE INGESTÃO CLI



```
kafka-consumer-groups.sh \ --bootstrap-server localhost:9092 \ --list
```

```
kafka-consumer-groups.sh \ --bootstrap-server localhost:9092 \ --describe \ --group nome-do-grupo
```

MONITORIZAÇÃO DE CONSUMO

kafka-consumer-groups.sh

- Ferramenta que monitoriza e gere o consumo de mensagens por grupos de consumidores.
- Permite:
 - Ver quais consumidores estão ativos.
 - Acompanhar offsets e lags.
 - Diagnosticar atrasos no consumo.
- Comandos:
 - **-list;**
 - **-describe;**

APACHE kafka

FERRAMENTAS DE INGESTÃO CLI



```
# 1. Criar um tópico Kafka
kafka-topics.sh \ --bootstrap-server localhost:9092 \
--create \ --topic exemplo-topico \ --partitions 2 \ --replication-factor 1

# 2. Produzir mensagens manualmente
kafka-console-producer.sh \ --bootstrap-server localhost:9092 \
--topic exemplo-topico

# 3. Consumir mensagens (do início)
kafka-console-consumer.sh \ --bootstrap-server localhost:9092 \
--topic exemplo-topico \
--from-beginning \
--consumer-property group.id=grupo-exemplo \
--property print.key=true \
--property print.value=true

# 4. Monitorizar o grupo de consumo
kafka-consumer-groups.sh \ --bootstrap-server localhost:9092 \
--describe \ --group grupo-exemplo
```

APACHE kafka

HANDS-ON

PRÉ REQUISITOS

1. Java 11 instalado e configurado (`java -version`)
2. Kafka 3.x ou superior instalado
3. Ambiente Kafka iniciado com o modo KRaft (sem Zookeeper)
4. `bootstrap-server` configurado corretamente (ex: `localhost:9092`)

OBJETIVOS

1. Criar tópicos via CLI
2. Produzir mensagens manualmente com estrutura de dados
3. Entender consumo em grupo
4. Observar partícões e offsets
5. Raciocinar sobre garantias e leitura paralela

APACHE Flink



INTRODUÇÃO

1. Apache Flink – Fundamentos

- a. Arquitetura do Flink;
- b. Conceitos: DataStream API, SQL, Event-time, State;
- c. Como o Flink se integra com Kafka;

2. SQL Streaming em Flink

- a. CREATE TABLE com connector Kafka;
- b. SELECT, WHERE, INSERT INTO;
- c. Formatos: JSON, CSV, etc.

3. Event-Time e Dados Fora de Ordem;

4. Exercícios Práticos com Flink SQL + Kafka;

APACHE Flink

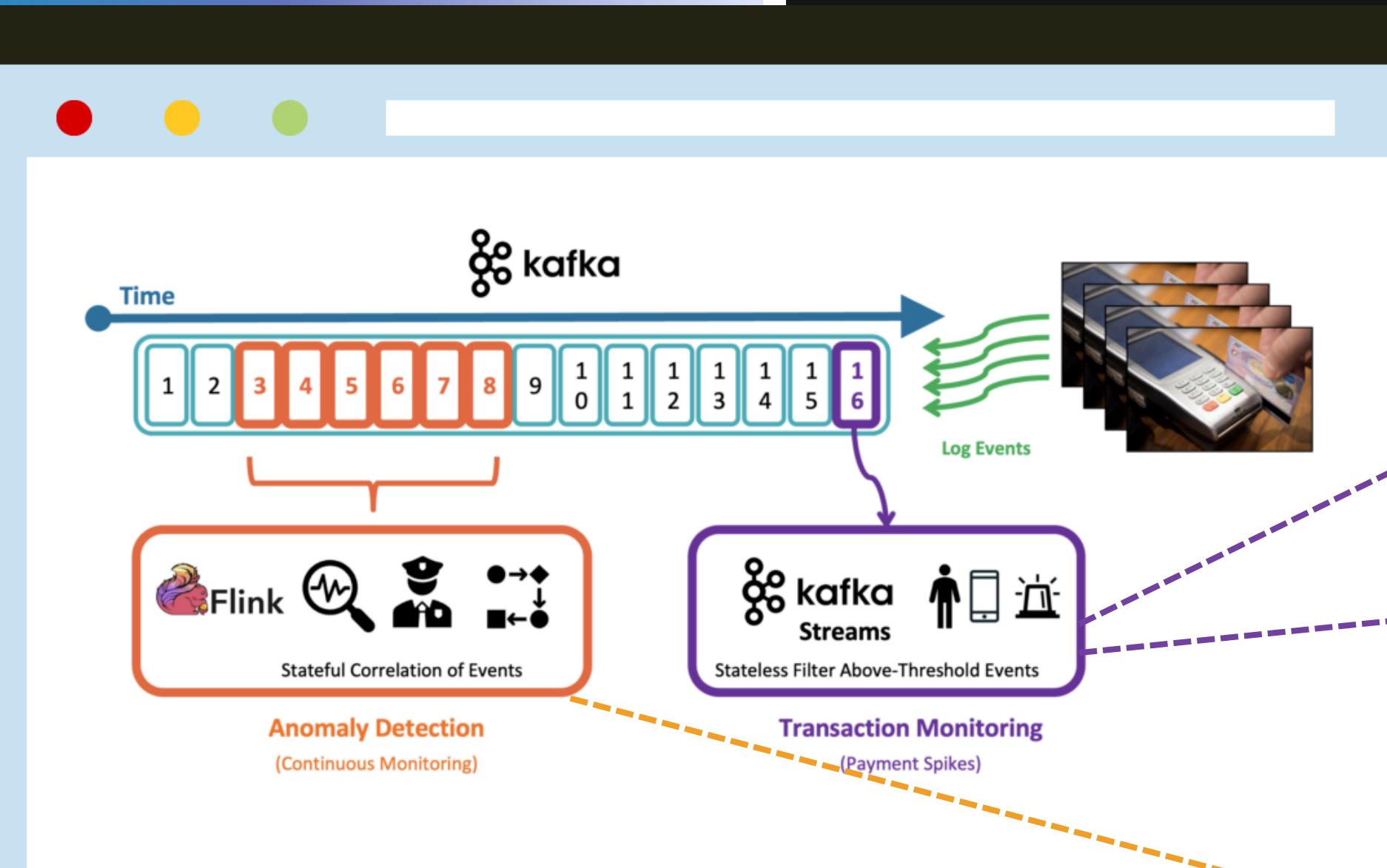
FLINK USE CASES



- E-commerce / Media
ALIBABA
- Tecnologia / DevOps
Netflix & Uber
- Redes Sociais/Streaming
Twitter
- Processos ETL
Airbnb & Booking

APACHE Flink

Use Case



APACHE Flink

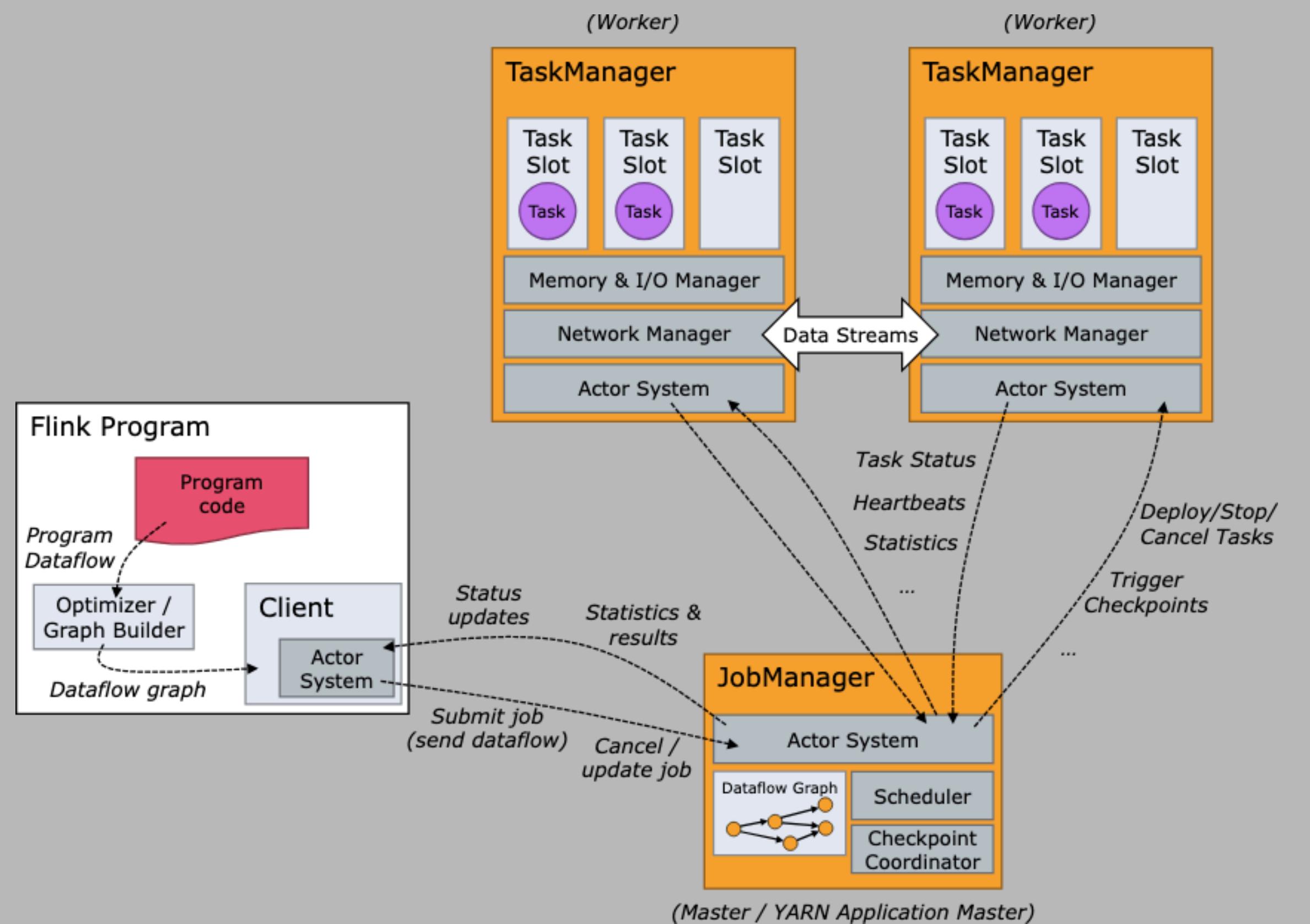
ARQUITETURA



- Processa dados infinitos (streaming) e limitados (em lote);
- Alta escalabilidade;
- Integra-se com outros recursos, por exemplo Hadoop YARN e Kubernetes;
- Processos JobManager e TaskManager mediado por um Client;
- Checkpoints/Savepoints;

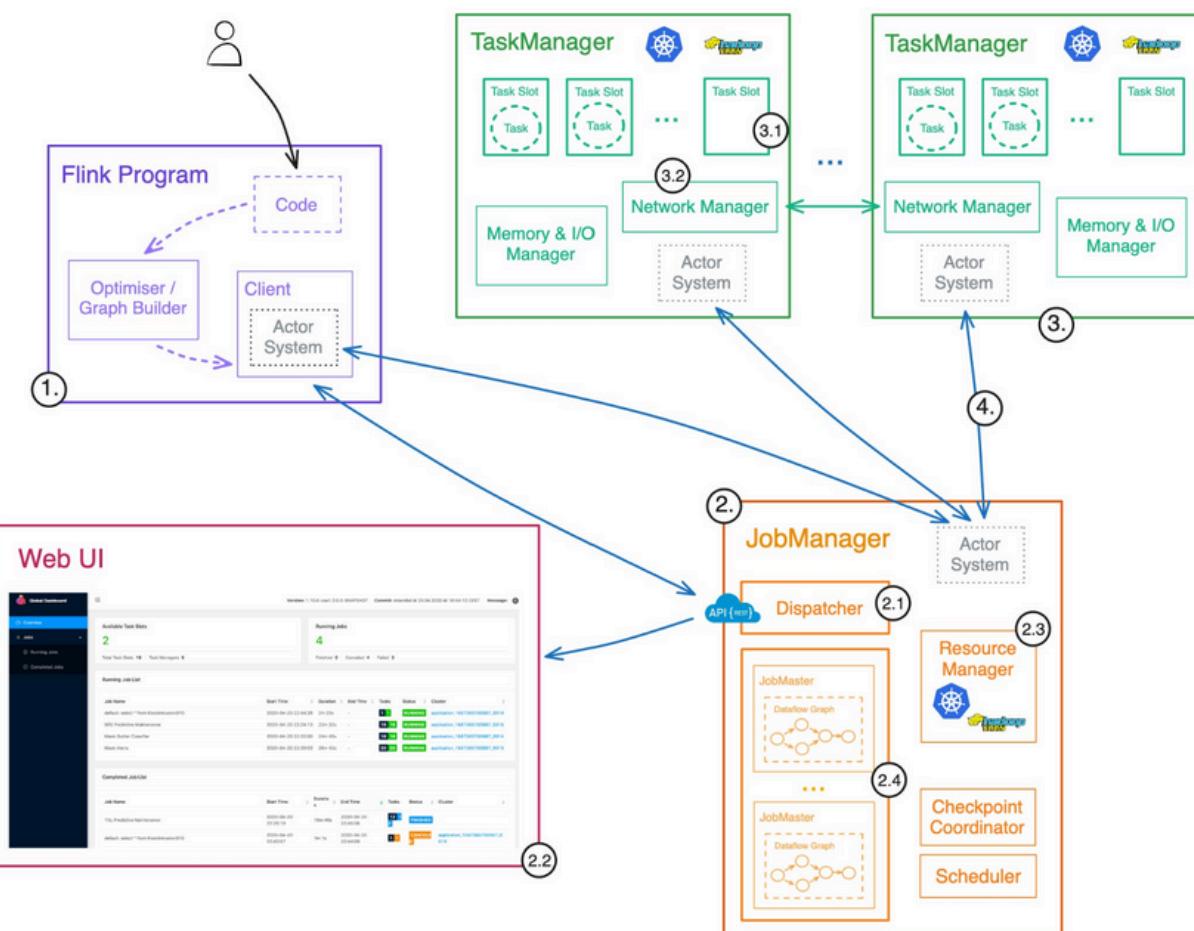
APACHE Flink

ARQUITETURA



APACHE Flink

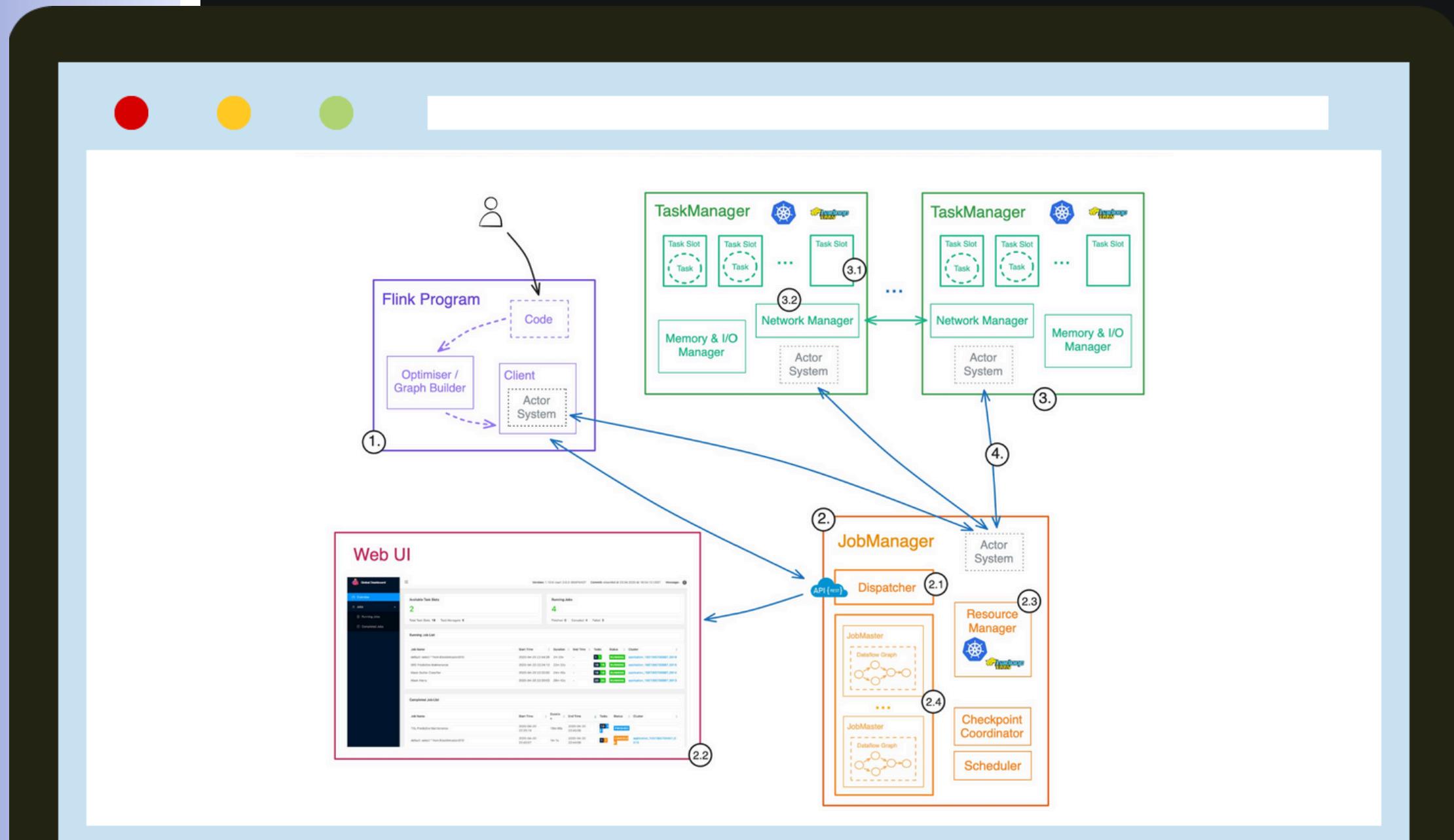
ARQUITETURA



1. O Client envia o processo para o JobManager;
2. O JobManager coordena;
3. Os TaskManagers executam;
4. Os CheckPoints salvam o progresso;

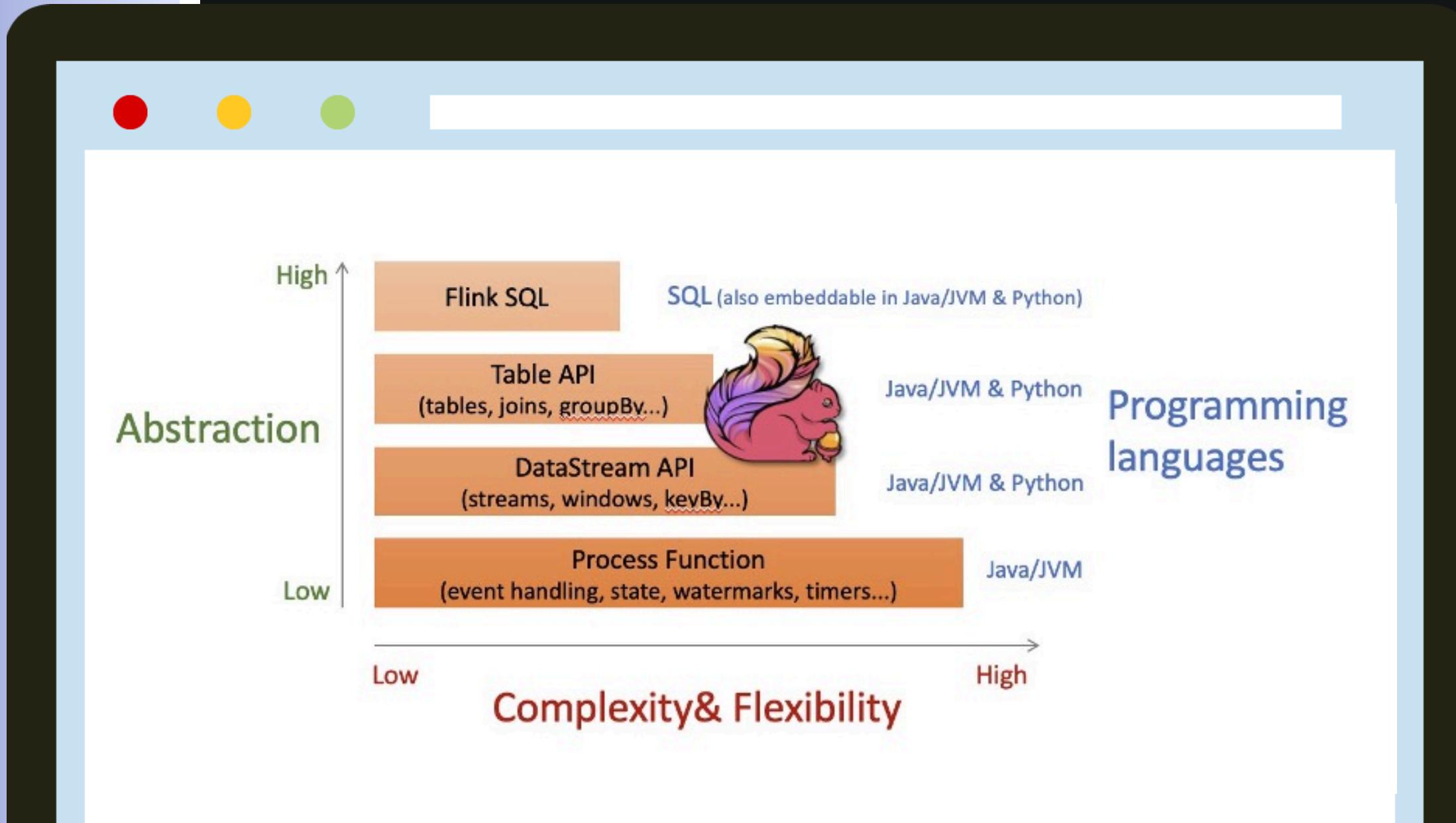
APACHE Flink

HIERARQUIA DE APIs



APACHE Flink

HIERARQUIA DE APIs



APACHE Flink

HIERARQUIA DE APIs



```
SELECT userId, COUNT(*) FROM clicks GROUP BY userId;
```

SQL

- Providencia o **nível mais alto de abstração** (menos complexo);
- Usa linguagem **SQL padrão** (com extensões para stream processing);
- Ideal para utilizadores sem background de programação Java/Scala.

APACHE Flink

HIERARQUIA DE APIs



```
table.groupBy($"userId").select($"userId", $"clicks".count());
```

Table API

- API declarativa baseada em tabelas;
- Usa linguagem Java/Scala;
- Semelhante a tabelas de bases de dados relacionais;
- Extensível com funções personalizadas;
- Integração fácil com DataStream API;

APACHE Flink

HIERARQUIA DE APIs



```
stream.keyBy(event -> event.userId)
    .window(TumblingEventTimeWindows.of(Time.minutes(1)))
    .sum("clicks");
```

DataStream API

- API imperativa baseada em tabelas;
- Os dados representam classes/objetos (linguagem Python, Java, Scala);
- Permite aplicar map(), filter(), window(), keyBy(), etc;
- Ideal para lógica de fluxo de dados contínuo;
- Integração com Process Function;

APACHE Flink

HIERARQUIA DE APIs



```
stream
    .keyBy(event -> event.userId)
    .process(new KeyedProcessFunction<String, Evento, String>() {

        private ValueState<Long> timerState;

        @Override
        public void open(Configuration parameters) {
            ValueStateDescriptor<Long> descriptor =
                new ValueStateDescriptor<>("timer", Long.class);
            timerState = getRuntimeContext().getState(descriptor);
        }

        @Override
        public void processElement(Evento event, Context ctx, Collector<String> out) throws Exception {
            long triggerTime = ctx.timerService().currentProcessingTime() + 10_000;
            ctx.timerService().registerProcessingTimeTimer(triggerTime);

            timerState.update(triggerTime);
        }

        @Override
        public void onTimer(long timestamp, OnTimerContext ctx, Collector<String> out) throws Exception {
            out.collect("⚠ Utilizador " + ctx.getCurrentKey() + " inativo há 10 segundos.");
            timerState.clear();
        }
    });
});
```

Process Function API

- Nível mais baixo dentro da DataStream API;
- Tratamento evento a evento;
- Boa para:
 - lógica temporal personalizada;
 - estados que expiram;
 - joins complexos;
 - comportamento adaptado ao watermark e event time.

APACHE Flink

HIERARQUIA DE APIs



Abstração	O que o programador faz	Quem trata dos detalhes
SQL (mais alto nível)	Escreve uma consulta: <code>SELECT * FROM vendas</code>	Flink gera operadores, paralelismo, estado, etc.
Table API	Usa código tipo SQL em Java/Scala	Flink ainda trata da maior parte da lógica
DataStream API (baixo nível)	Define o fluxo passo a passo: <code>map()</code> , <code>filter()</code> , etc.	Programador define o pipeline com mais controlo
Process Function (baixo nível)	Controla estado, timers, eventos manualmente	

APACHE Flink

Conceitos



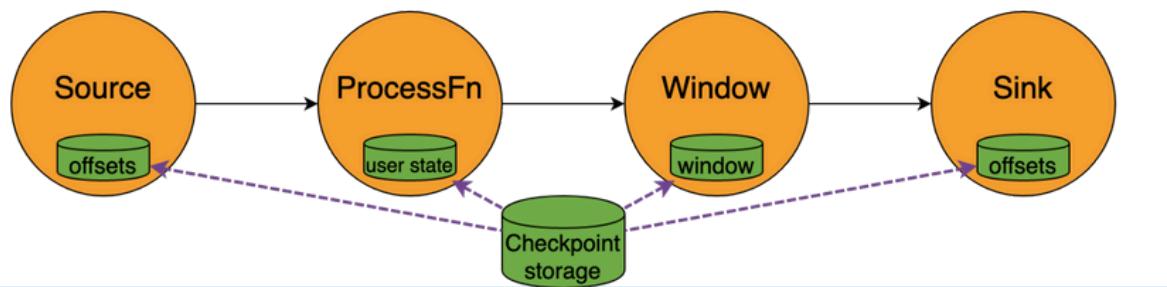
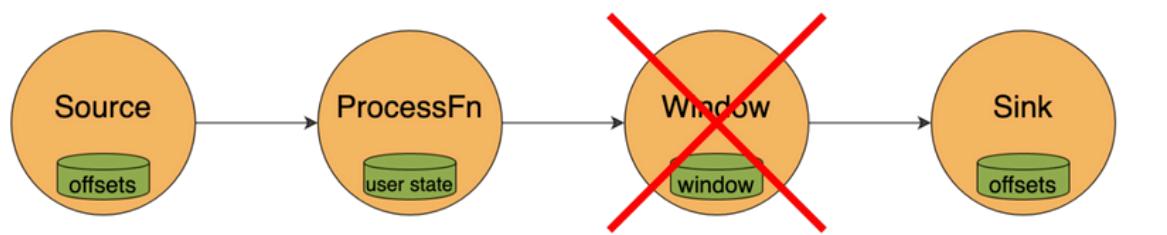
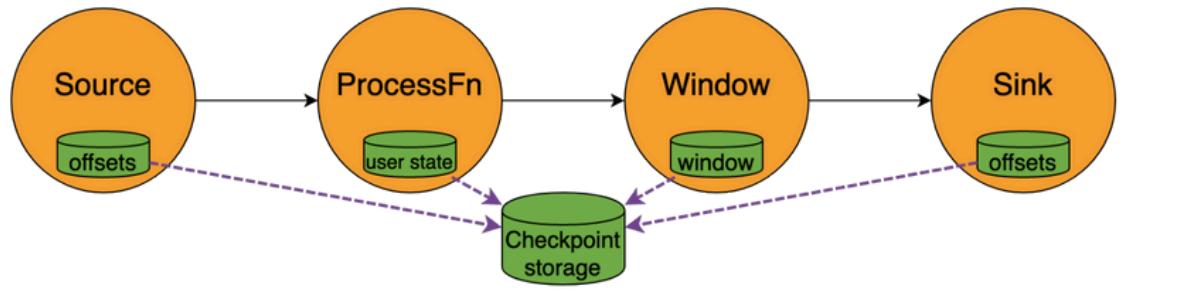
- Checkpoints & Savepoints;
- Events;
- Event-Time & Processing Time;
- Watermarks;
- State;
- Windows;

APACHE Flink

Conceitos

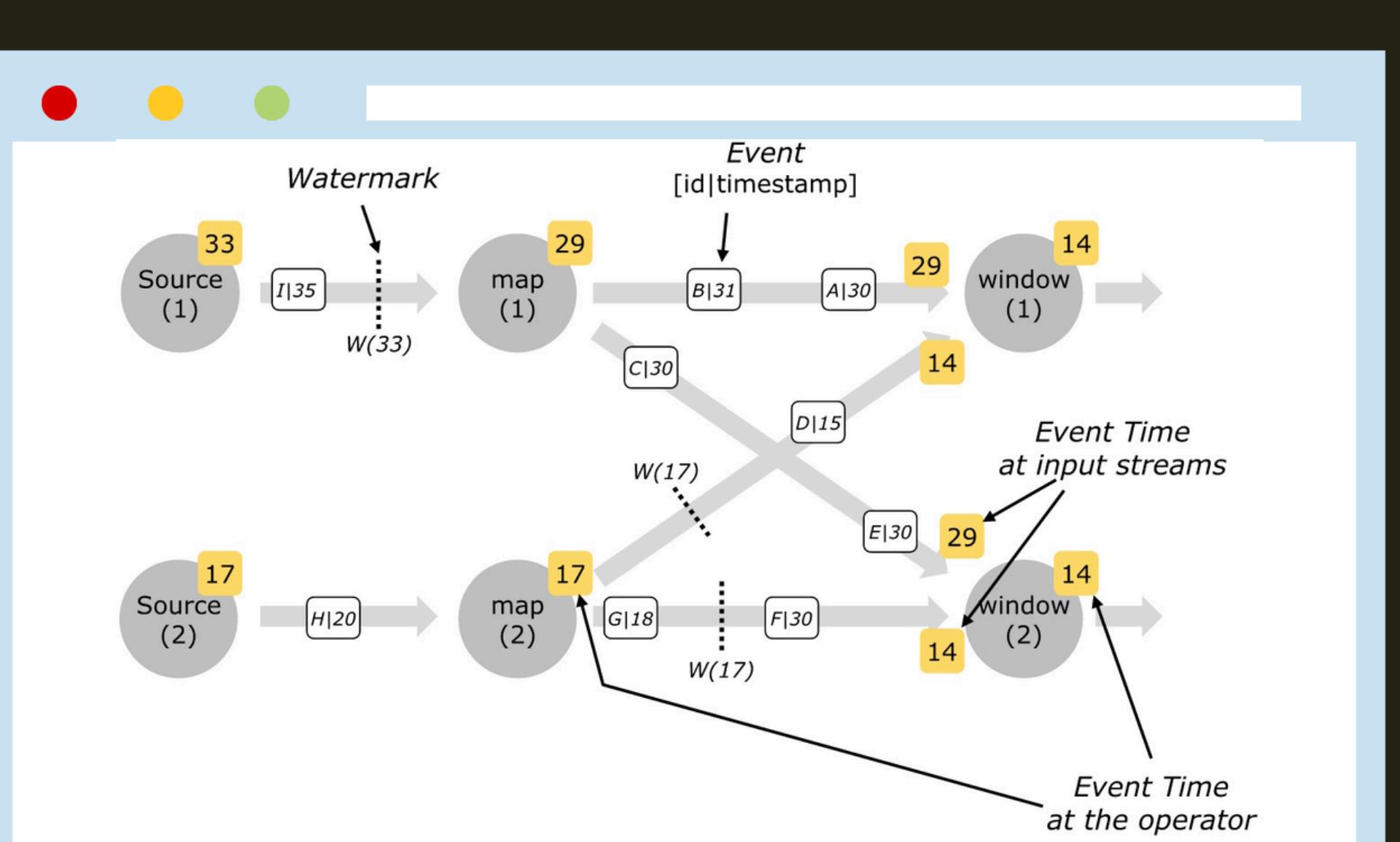


- Checkpoints & Savepoints;



APACHE Flink

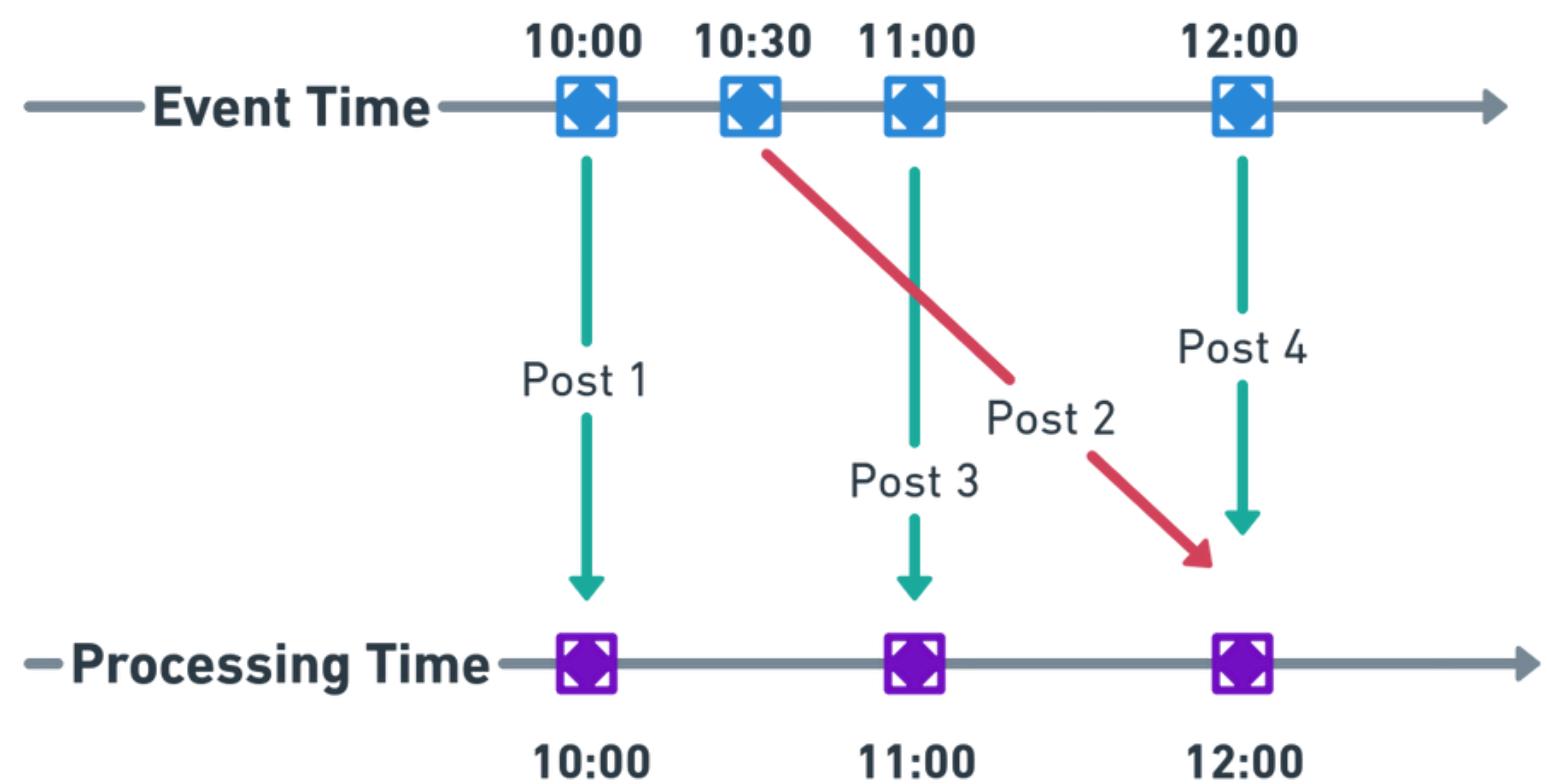
Conceitos



- Events;
- Event-Time & Processing Time;
- Watermarks;

APACHE Flink

Conceitos

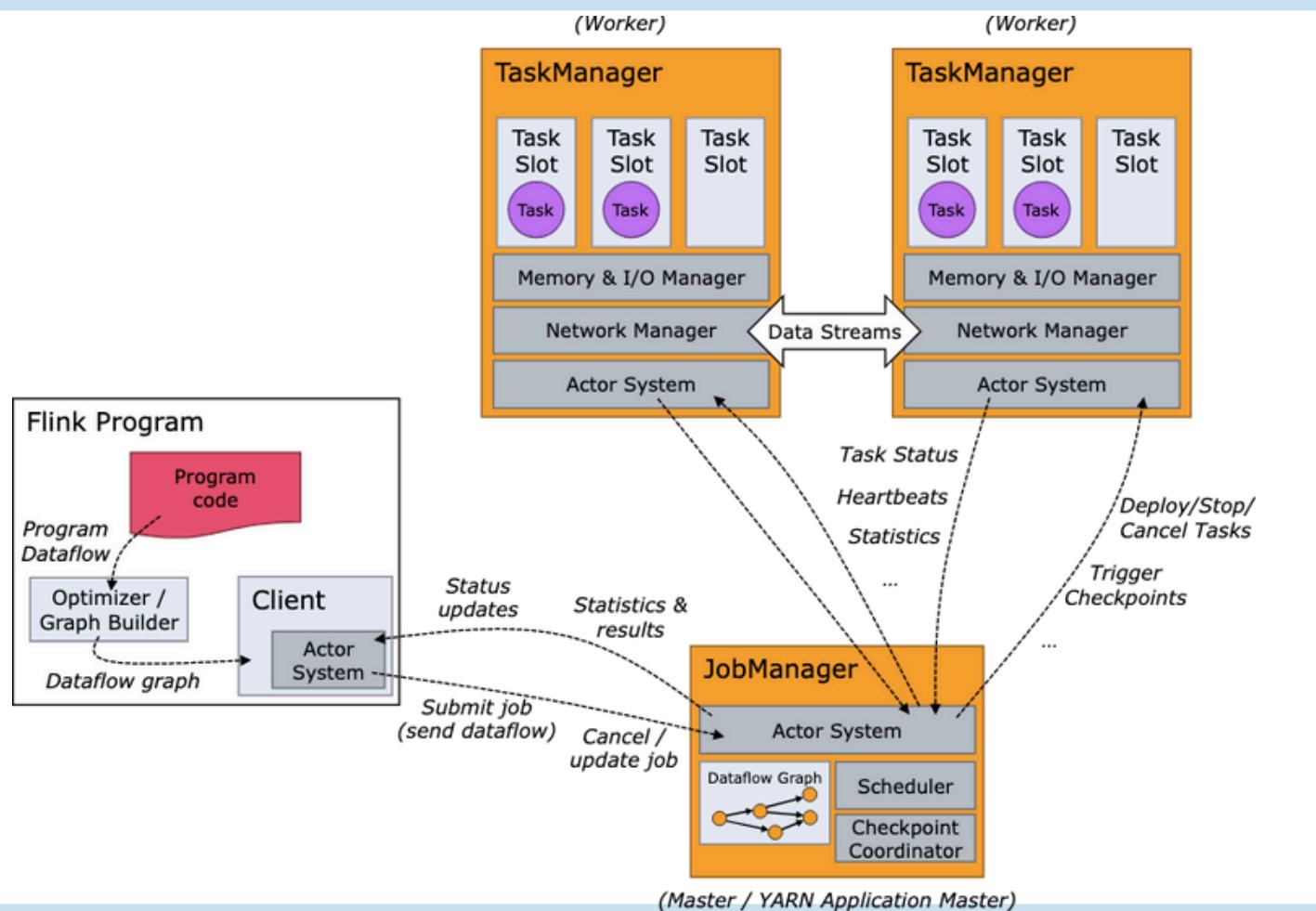


- Events;
- Event-Time & Processing Time;
- Watermarks;

```
env.setStreamTimeCharacteristic(TimeCharacteristic.ProcessingTime);
stream
    .timeWindow(Time.minutes(1))
    .sum("value");
```

APACHE Flink

Conceitos



- **State:**
- Keyed State
- `ValueState<T>;`
 - `ListState<T>;`
 - `MapState<K,V>;`
 - `ReducingState<T> e AggregatingState;`
- Operator State

APACHE Flink

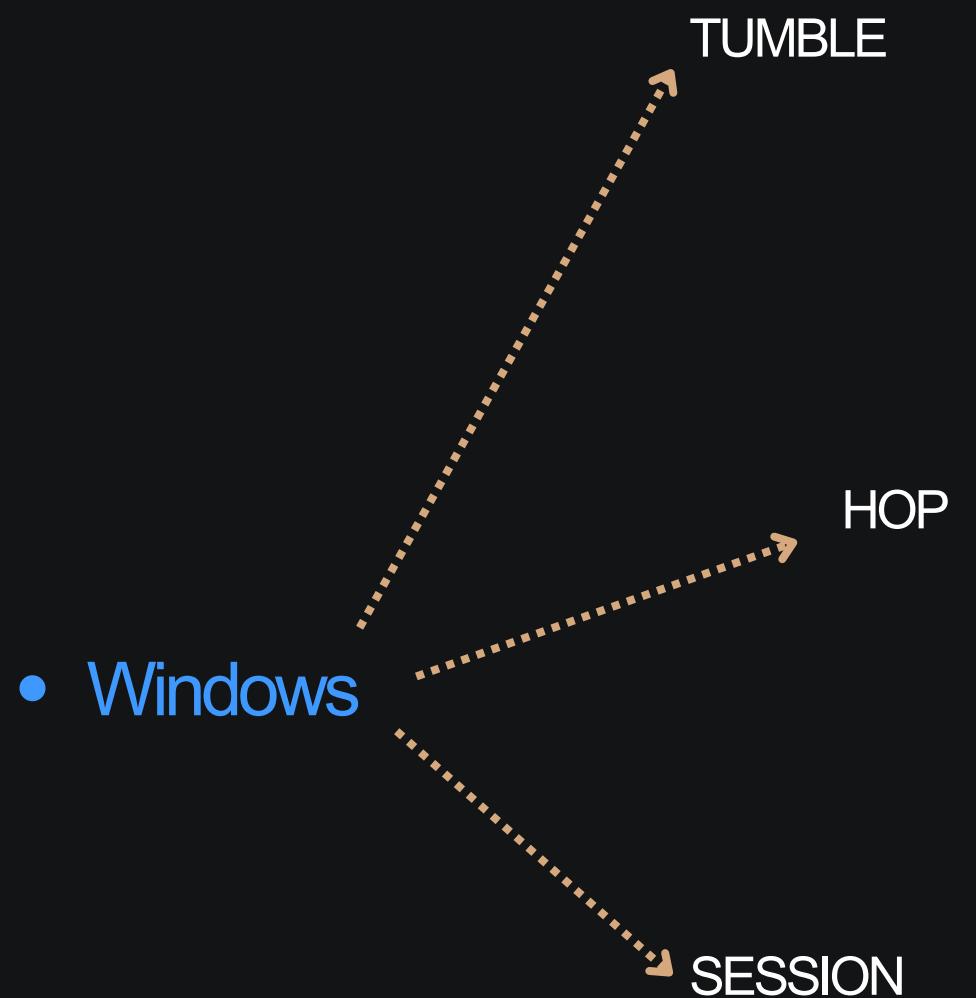
Conceitos



```
TUMBLE(table, DESCRIPTOR(event_time), INTERVAL '5' MINUTES)
```

```
HOP(table, DESCRIPTOR(event_time), INTERVAL '5' MINUTES, INTERVAL '10' MINUTES)
```

```
SESSION(table, DESCRIPTOR(event_time), INTERVAL '30' SECONDS)
```



APACHE Flink

Integração Kafka + Flink



- Flink integra com Kafka através de Kafka Source e Kafka Sink;
- Usa checkpoints para manter consistência de leitura e escrita;
- Permite event time, paralelismo, janelas, joins, estado;
- Garante exactly-once, mesmo com falhas;
- Muito usado para pipelines de dados em tempo real.

APACHE Flink

SQL Streaming em Flink



```
CREATE TABLE transacoes (
    user_id STRING,
    valor DOUBLE,
    ts TIMESTAMP(3),
    WATERMARK FOR ts AS ts - INTERVAL '5' SECOND
) WITH (
    'connector' = 'kafka',
    'topic' = 'pagamentos',
    'properties.bootstrap.servers' = 'localhost:9092',
    'scan.startup.mode' = 'earliest-offset',
    'format' = 'json'
);
```

- Esta tabela representa um tópico Kafka chamado pagamentos;
- Cada linha tem um utilizador, um valor e um timestamp;
- Esse timestamp representa o momento real do evento (event time);
- Vamos aceitar atrasos até 5 segundos usando watermarks;
- Os dados vêm em JSON.;

APACHE Flink

SQL Streaming em Flink



```
INSERT INTO alertas
SELECT user_id, SUM(valor) AS total
FROM transacoes
WHERE valor > 1000
GROUP BY user_id;
```

APACHE Flink

SQL Streaming em Flink



```
'format' = 'json'
```

```
'format' = 'csv',  
'csv.field-delimiter' = ';'
```

APACHE Flink

HANDS-ON



PRÉ REQUISITOS

1. Java 11 instalado e configurado (`java -version`)
2. Ambiente Kafka iniciado com o modo KRaft (sem Zookeeper)
3. Descarregar e iniciar Flink SQL;

OBJETIVOS

1. Criar tabelas em FLINK usando o conector KAFKA;
2. Ingestão de dados via JSON e CSV;
3. Aprofundar comandos SQL (Insert into, select, where, etc);
4. Ingestão de dados para KAFKA via FLINK;

CONTATO



<https://github.com/Rodmnj/KAFKA>



<https://www.linkedin.com/in/rodrigoalexmarques/>



rodalexmqrs@gmail.com



+351 934695932



Lisboa, Portugal

OBRIGADO