Módulo Profesional: Big Data Aplicado

Apache Flink

¿Qué es Apache Flink?

¿Qué es Apache Flink?

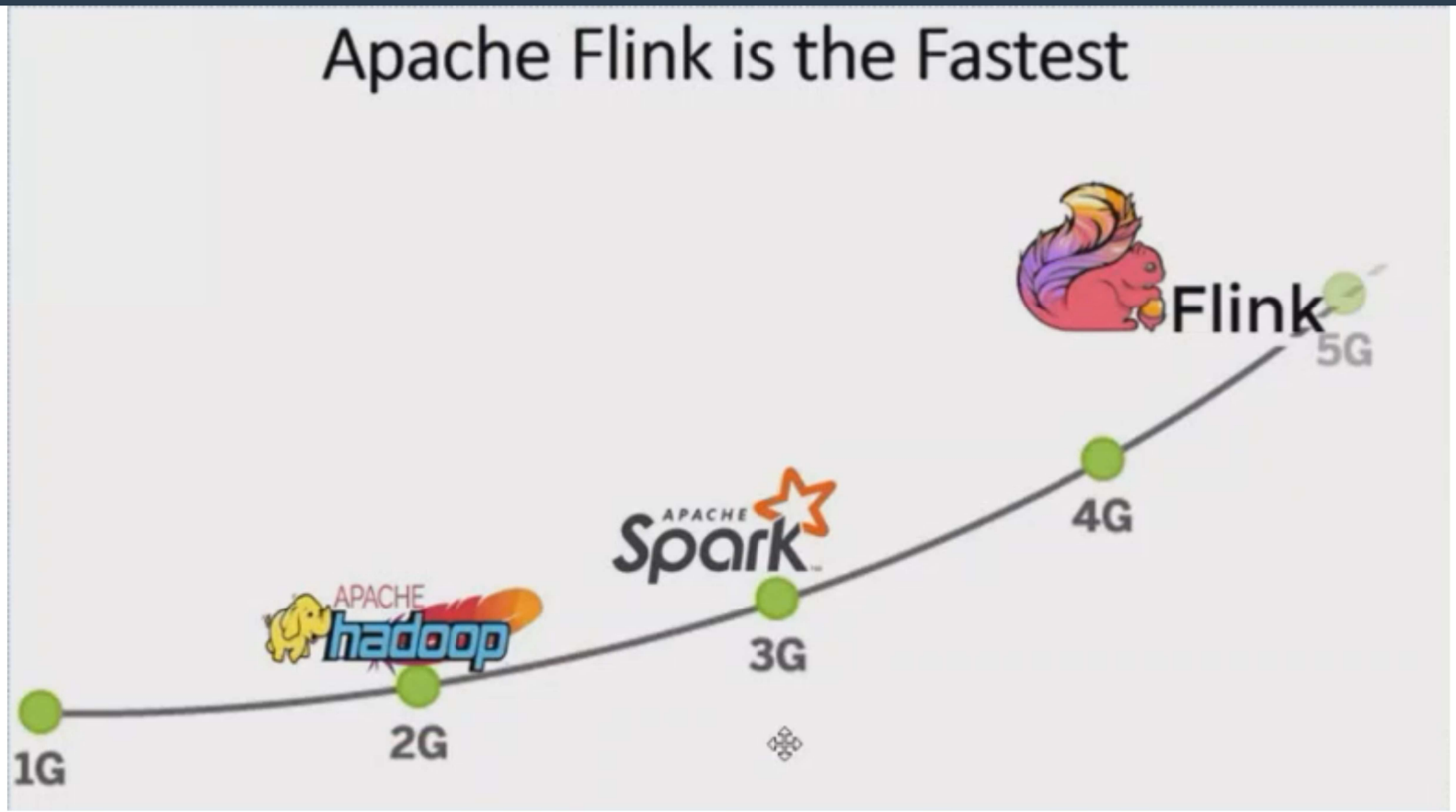
Un framework y motor de procesamiento distribuido para cálculos con estado sobre flujos de datos acotados e inagotables, en tiempo real o por lotes.

- **Con estado (Stateful)**: Las funciones mantienen un estado embebido y tolerante a fallos, accesible localmente como una variable.

- **Virtual**: Las funciones no reservan recursos; las funciones inactivas no consumen CPU ni memoria.

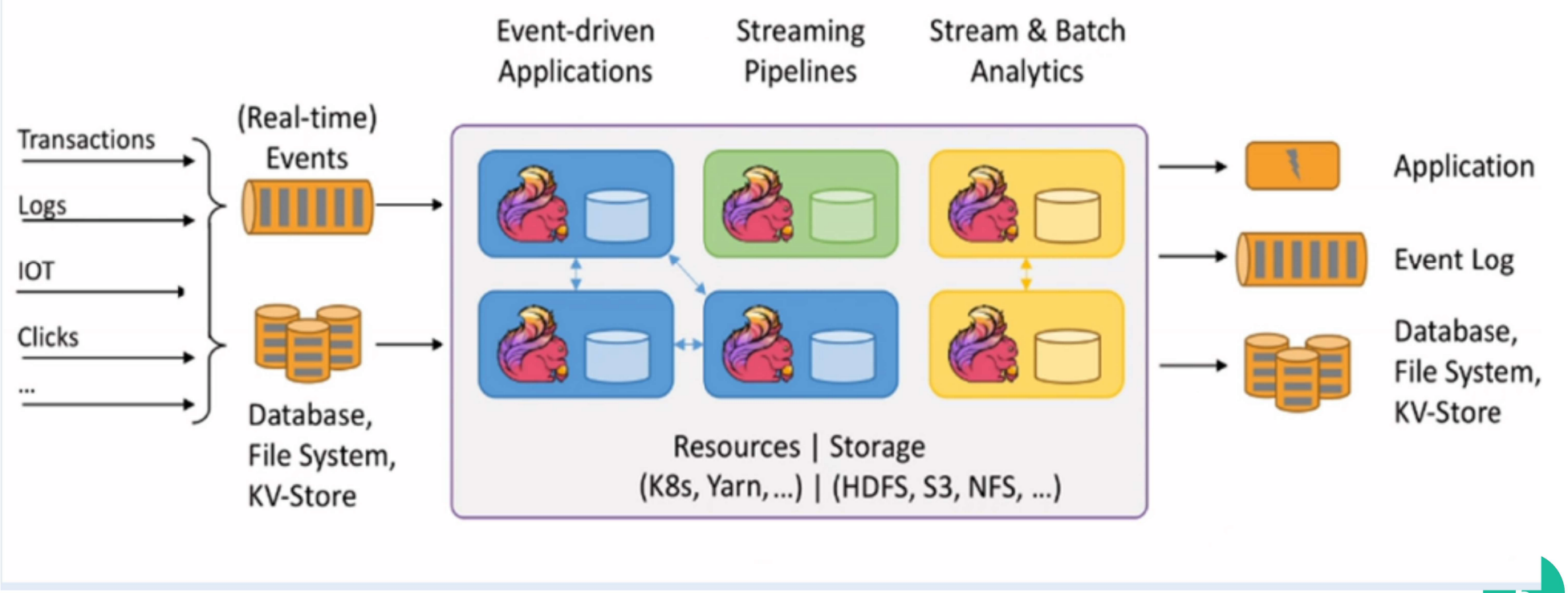
- **Garantiza semántica de exactamente una vez (Exactly-Once)**.

¿Qué es Apache Flink?

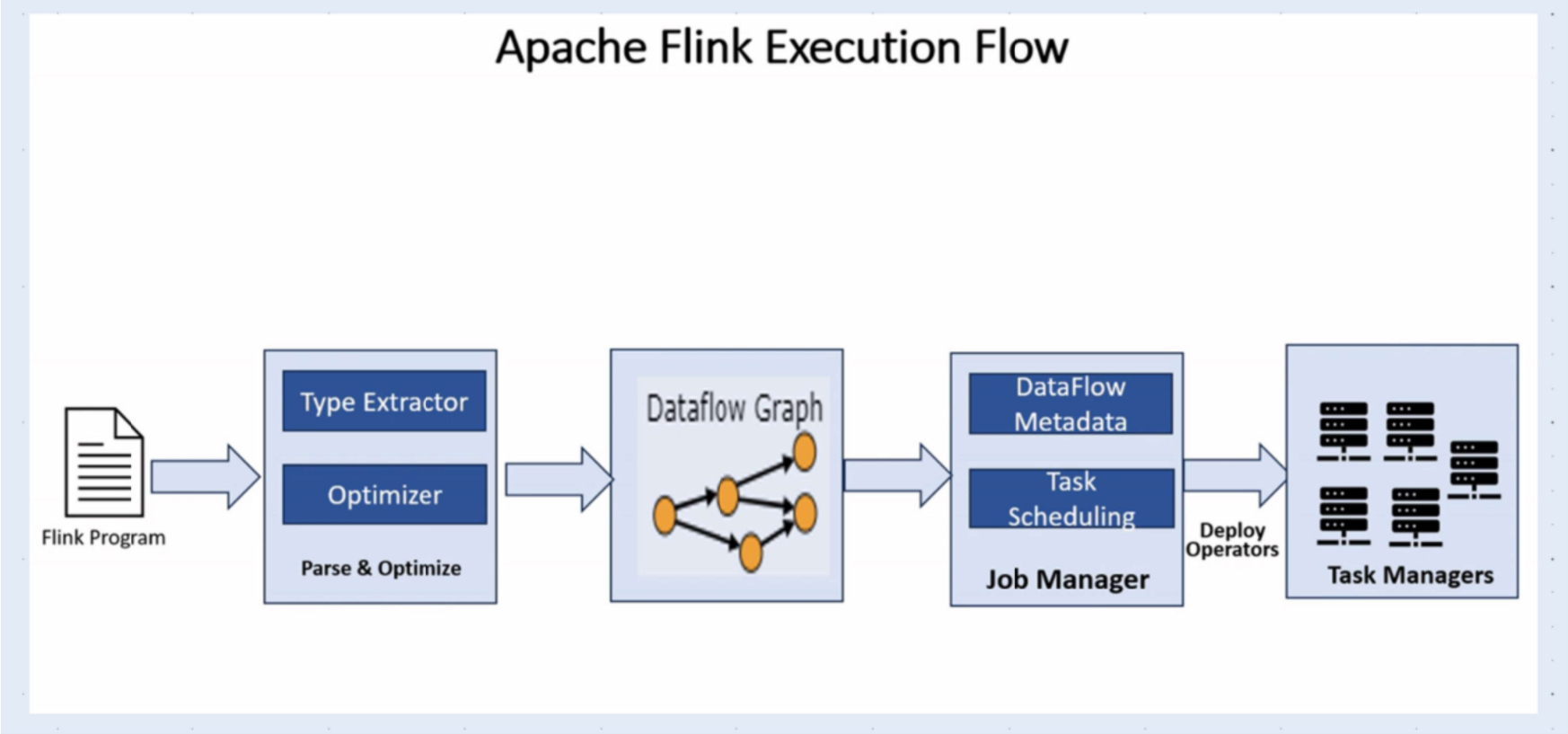


¿Qué es Apache Flink?

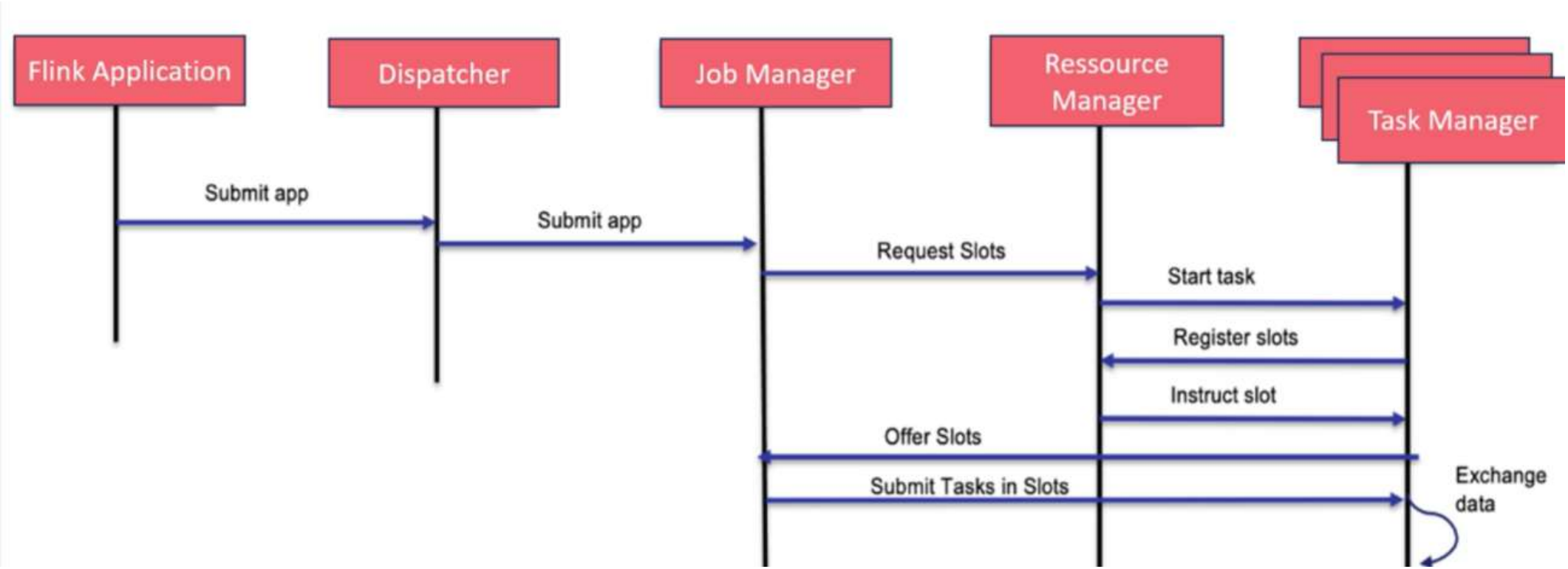
Es una solución todo en uno para procesamiento por lotes, análisis de grafos, operaciones sobre tablas e incluso aprendizaje automático.



Flujo de ejecución de Apache Flink



Conceptos de Apache Flink



Flink Application

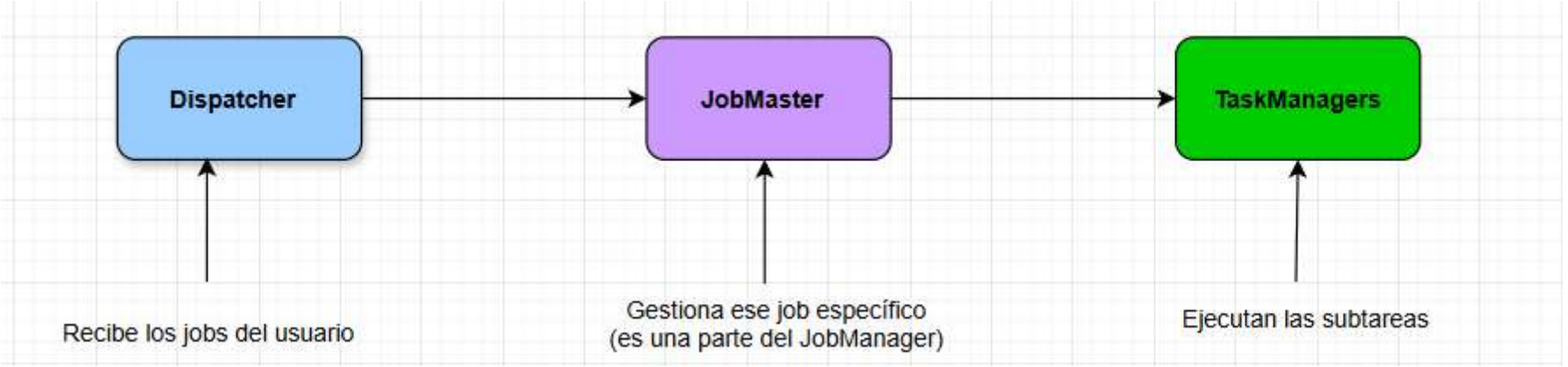
Una Flink Application es un programa que defines usando las APIs de Flink (como DataStream API o Table API) para procesar datos en tiempo real o por lotes.

Contexto: Cualquier programa que use Flink para procesar flujos o lotes. Por ejemplo, un job que consume de Kafka y escribe en S3.

Dispatcher

Es un servicio que vive dentro del proceso del JobManager principal, cuya tarea es:

- Recibir trabajos (jobs) desde clientes Flink (por ejemplo, cuando haces flink run).
- Validar y registrar el job.
- Delegar la ejecución del job a un JobMaster, que es una instancia especializada del JobManager enfocada en ese job particular.



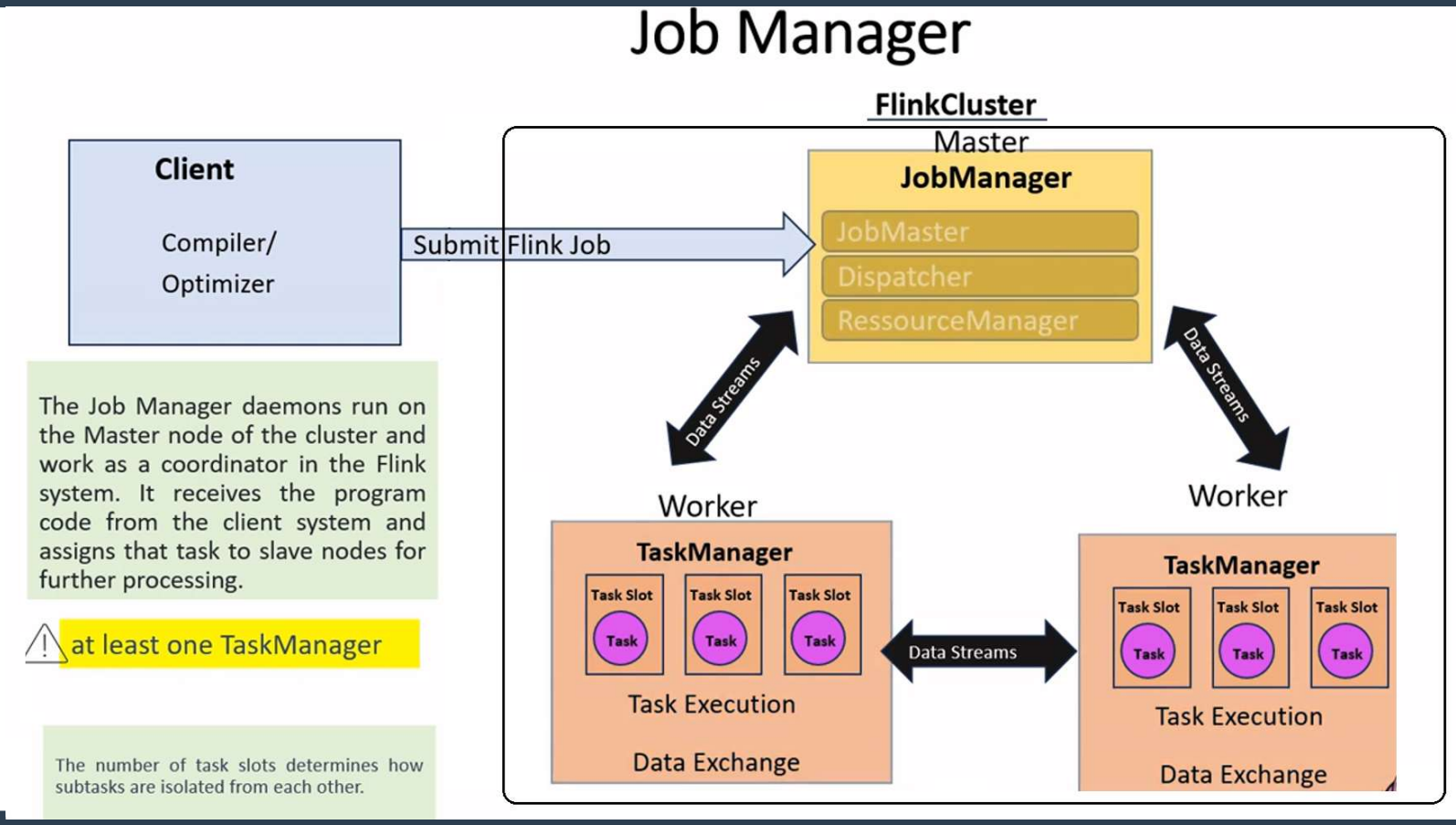
Job Manager

Es un proceso maestro que:

- Coordina la ejecución del trabajo.
- Gestiona los recursos del clúster.
- Supervisa el estado de las tareas.
- Maneja la recuperación ante fallos.

El Job Manager en Flink es el cerebro del clúster, responsable de planificar, coordinar y monitorear la ejecución de los trabajos distribuidos.

Job Manager


Job Manager
FlinkCluster
Master
Client
Compiler/
Optimizer
Submit Flink Job
JobManager
JobMaster
Dispatcher
RessourceManager
Data Streams
Data Streams
Worker
Worker
TaskManager
Task Slot
Task
Task Execution
Data Exchange
Data Streams
TaskManager
Task Slot
Task
Task Execution
Data Exchange
The Job Manager daemons run on the Master node of the cluster and work as a coordinator in the Flink system. It receives the program code from the client system and assigns that task to slave nodes for further processing.
at least one TaskManager
The number of task slots determines how subtasks are isolated from each other.

Recursos Manager

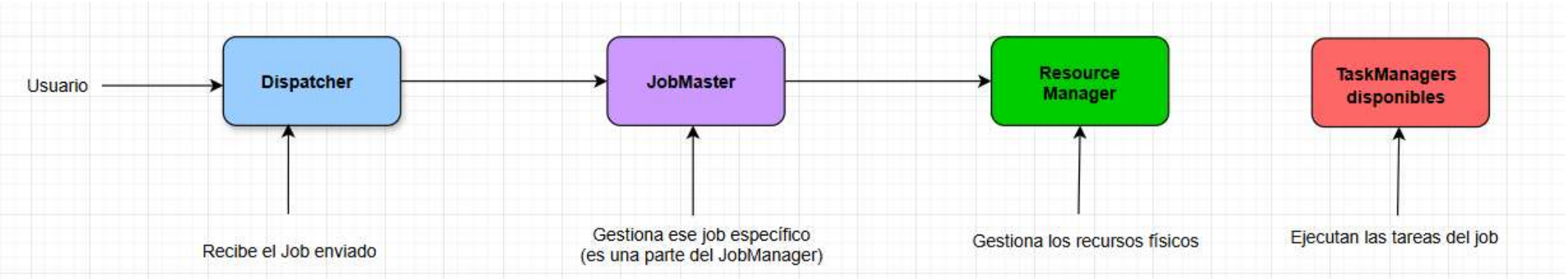
El Resource Manager es el componente que gestiona los recursos físicos del clúster (CPU, memoria, contenedores, máquinas virtuales, etc.) para poder ejecutar las tareas (tasks) de los trabajos.

El Resource Manager se encarga de pedir, asignar y liberar recursos (TaskManagers) para que los jobs de Flink se puedan ejecutar de manera eficiente.

Relación con otros componentes:

- El Dispatcher manda jobs.
- El JobMaster organiza la ejecución de un job específico.
- El Resource Manager provee TaskManagers para que el JobMaster pueda ejecutar las tareas.

Flujo


Usuario
Dispatcher
JobMaster
Resource Manager
TaskManagers disponibles
Recibe el Job enviado
Gestiona ese job específico
(es una parte del JobManager)
Gestiona los recursos físicos
Ejecutan las tareas del job

Task Manager

El Task Manager es el trabajador (worker) en el clúster de Flink.

Es el proceso que ejecuta las tareas físicas de un trabajo Flink: las operaciones concretas como leer datos, transformarlos, hacer agregaciones, escribir resultados, etc.

Cada Task Manager ejecuta una parte del job (una o varias subtareas).

Funciones principales:
- Ejecutar subtareas (subtasks) del job que recibe desde el JobMaster.
- Usar slots: Cada Task Manager tiene varios slots (ranuras) disponibles. Cada slot puede ejecutar una o más subtareas.
- Administrar memoria para las operaciones de procesamiento.
- Manejar el tráfico de red: Transfiere datos de una subtarea a otra, incluso entre diferentes nodos.
- Reportar su estado (salud, métricas, etc.) al Resource Manager y al JobMaster.

Task Manager

Task Manager
El proceso que corre en una máquina. Puede ser una JVM en YARN, un pod en Kubernetes, o un proceso local.

Slot
Una "ranura" donde puede ejecutarse una o varias subtareas. Cada Task Manager tiene varios slots.

Subtarea
La unidad básica de ejecución, parte del job, que corre en un slot.

Ejecución paso a paso

1.- El usuario lanza un job (por CLI, API, etc.).

2.- El Dispatcher recibe el job (se encarga d recibir trabajos).

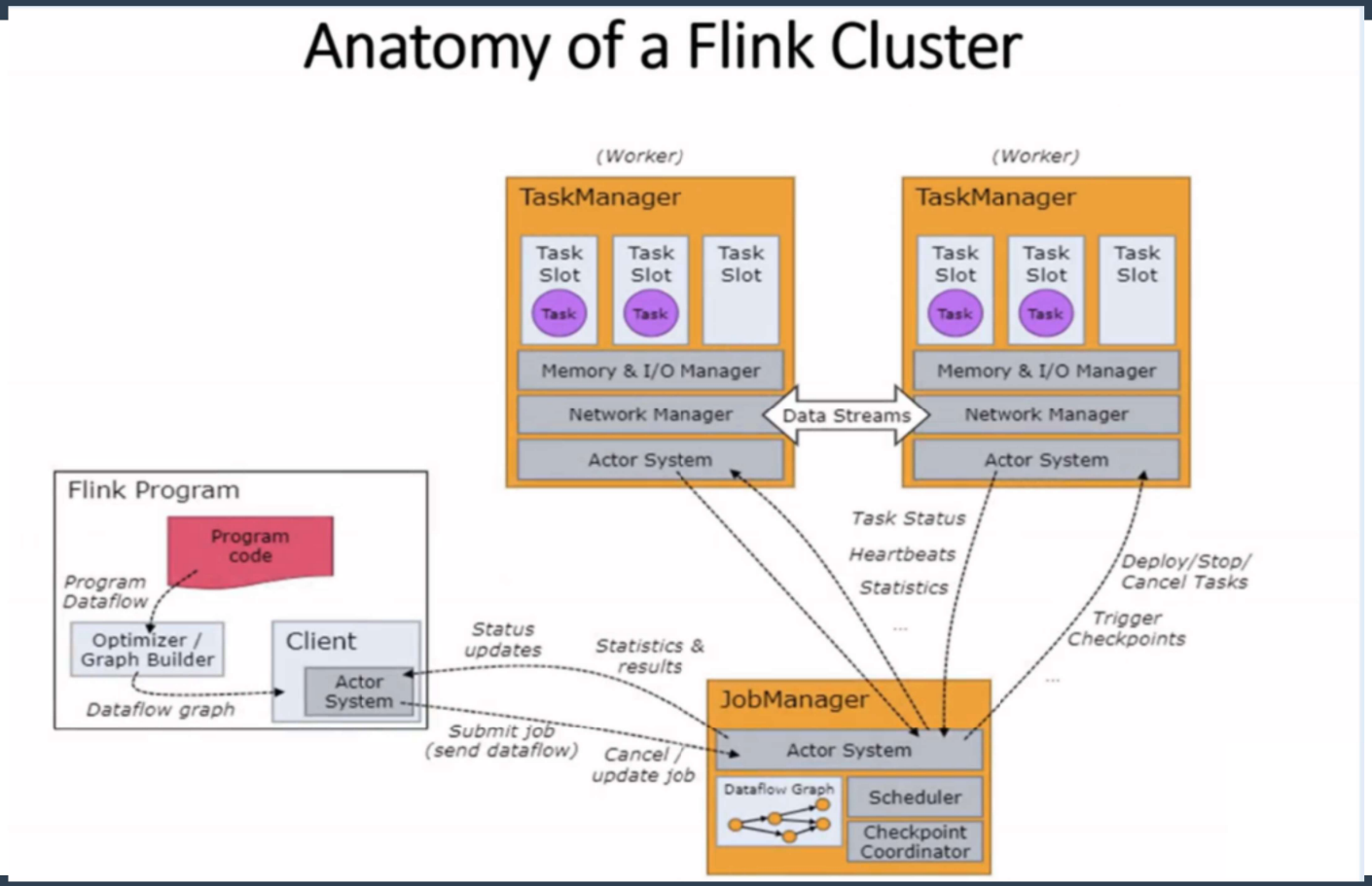
3.- El Dispatcher registra el job JobMaster (que maneja ese job).

4.- El JobMaster necesita recursos, entonces le pide al Resource Manager que le consiga slots disponibles. (Planifica el trabajo)

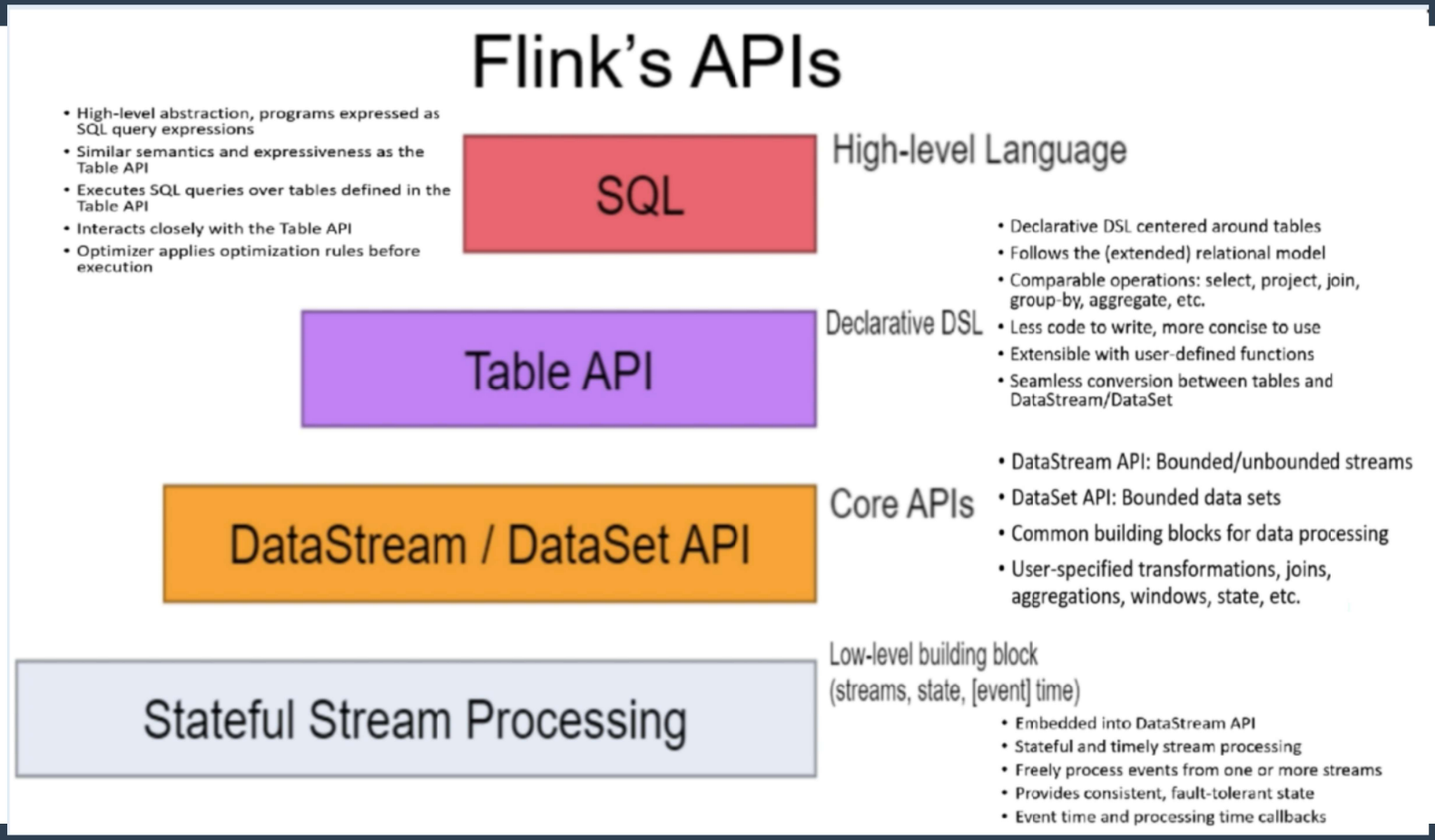
5.- El Resource Manager (asigna recursos):
- Encuentra TaskManagers ya disponibles o
- Pide nuevos TaskManagers al sistema subyacente (YARN, Kubernetes, máquinas físicas).

6.-Los TaskManagers se registran y empiezan a ejecutar las subtareas del job. (Ejecutan tareas)

Clúster de Flink



APIs Flink


Flink's APIs
• High-level abstraction, programs expressed as SQL query expressions
• Similar semantics and expressiveness as the Table API
• Executes SQL queries over tables defined in the Table API
• Interacts closely with the Table API
• Optimizer applies optimization rules before execution
SQL
High-level Language
Table API
Declarative DSL
• Declarative DSL centered around tables
• Follows the (extended) relational model
• Comparable operations: select, project, join, group-by, aggregate, etc.
• Less code to write, more concise to use
• Extensible with user-defined functions
• Seamless conversion between tables and DataStream/DataSet
DataStream / DataSet API
Core APIs
• DataStream API: Bounded/unbounded streams
• DataSet API: Bounded data sets
• Common building blocks for data processing
• User-specified transformations, joins, aggregations, windows, state, etc.
Stateful Stream Processing
Low-level building block
(streams, state, [event] time)
• Embedded into DataStream API
• Stateful and timely stream processing
• Freely process events from one or more streams
• Provides consistent, fault-tolerant state
• Event time and processing time callbacks

Cuestionario

REALIZAR CUESTIONARIO CON PREGUNTAS DE FLINK

Spark vs Flink

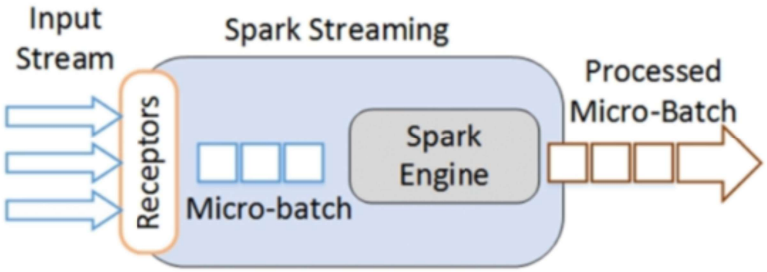
	Apache Spark	Apache Flink
Latencia	Alta	Baja
Arquitectura de procesamiento	Micro-batch	True Streaming
Tolerancia a fallos	RDD linaje, Data Replication	Snapshots distribuidos y Checkpoints
State Management	Basic State Management	Stateful Computación avanzada
Semántica	Micro-batch, windowing	Event-Driven, Windowing

Spark vs Flink

Processing Architecture Comparison

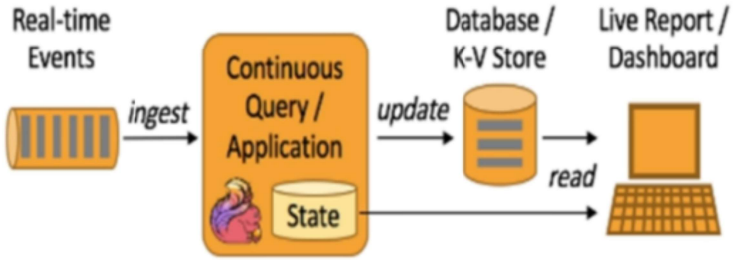
Apache Spark: **Micro-batch Processing**

- Data is divided into small batches for processing.
- Suitable for processing data in small batches rather than real-time.



Apache Flink: **True Streaming Processing**

- Enables continuous, real-time data processing without batching.
- Ideal for time-sensitive applications that require immediate processing of data as it arrives.



Spark vs Flink. Ejemplos de uso

Spark:

- Procesamiento de logs históricos.
- Generación de modelos de machine learning offline.
- ETLs batch pesadas.

Flink:

- Detección de fraudes en tarjetas de crédito en tiempo real.
- Monitoreo de sensores de IoT con latencia ultra baja.
- Procesamiento continuo de eventos de videojuegos online.

Spark = "Batch que aprendió a hacer Streaming."

Flink = "Streaming puro que también puede hacer Batch."