DATACOUCH
CREATING INFINITE POSSIBILITIES

DSAI
Data Science and AI Academy

Piramal
Finance

Flink Training for Real-Time Data Engineering

The Shift to Real-Time: From Batch to Streaming

About Instructor

- About Instructor ... *<text size should be 16 and style should be Trebuchet MS>*

NAME

Datacouch Instructor

AGENDA

- Introduction to Apache Flink
- Overview of Flink's architecture

What is Apache Flink?

Apache Flink is an open-source, distributed engine for stateful processing over unbounded (streams) and bounded (batches) data sets.

- Runs continuously with minimal downtime, processing data as ingested.
- Designed for **low latency**, **in-memory computations**, and **high availability**.
- Removes single points of failure and **scales horizontally**.
- Provides **exactly-once state management**.
- Supports **event-time processing**, handling out-of-order and late data.
- **Streaming-first** framework with a unified API for stream and batch processing.

Why Use Apache Flink?

Versatile framework for both streaming and batch applications.

Event-driven applications:

- Ingest events from one or more streams.
- Perform computations, update state, or trigger external actions.
- Supports **stateful processing**, enabling logic that depends on event history.

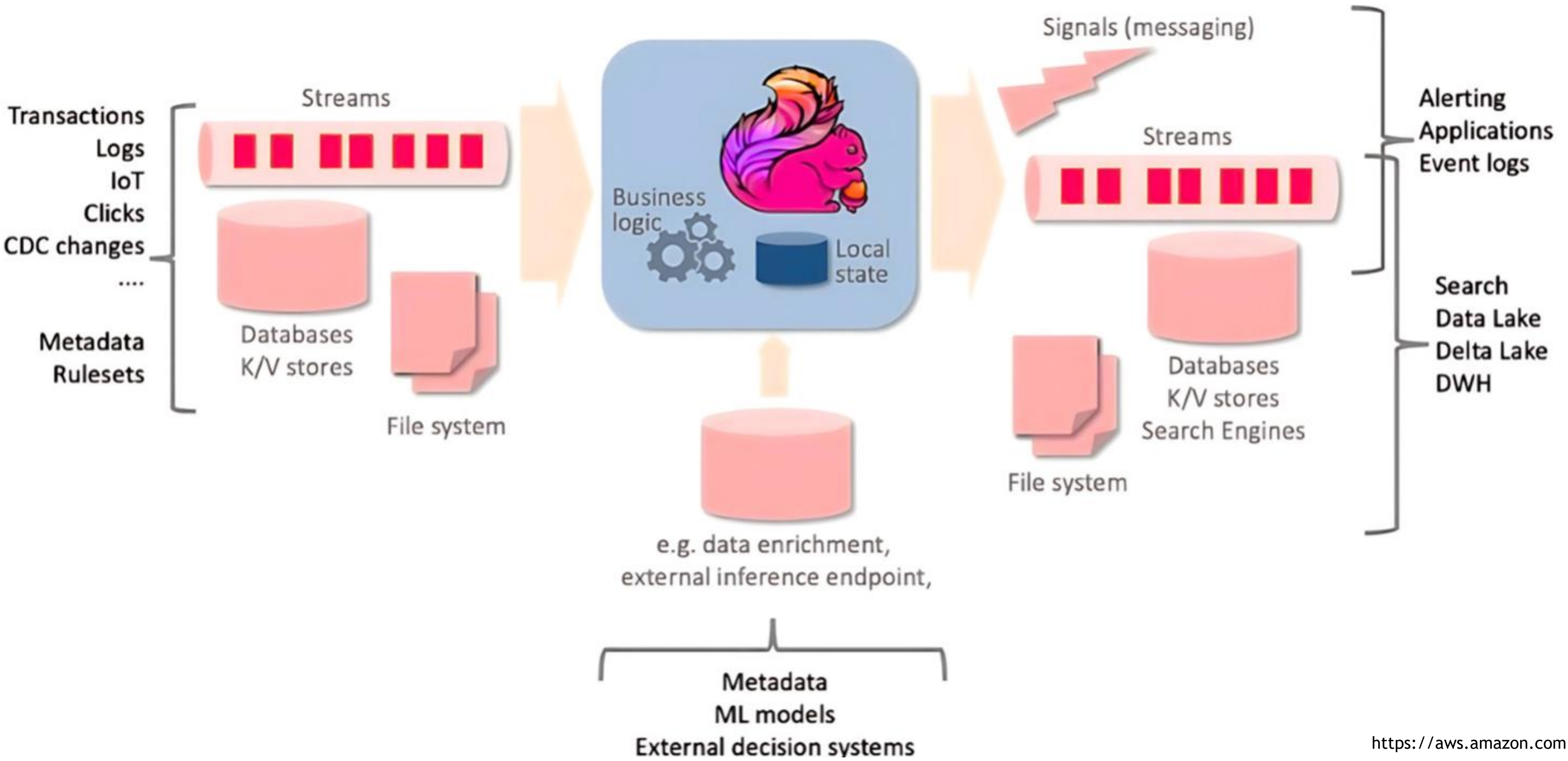
Data analytics applications:

- Extract insights continuously instead of re-running queries on finite datasets.
- Enable **real-time streaming queries** that emit and update results continuously.

Data pipeline applications:

- Transform and enrich data while moving it between storages.
- Replace periodic batch ETL with **continuous, low-latency data movement**.

Why Use Apache Flink?

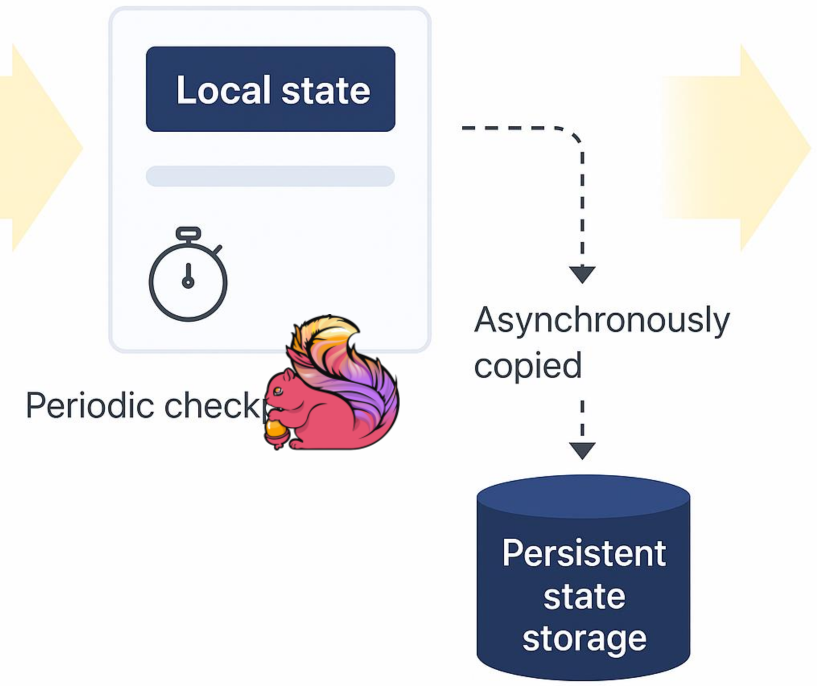


https://aws.amazon.com/what-is/apache-flink/

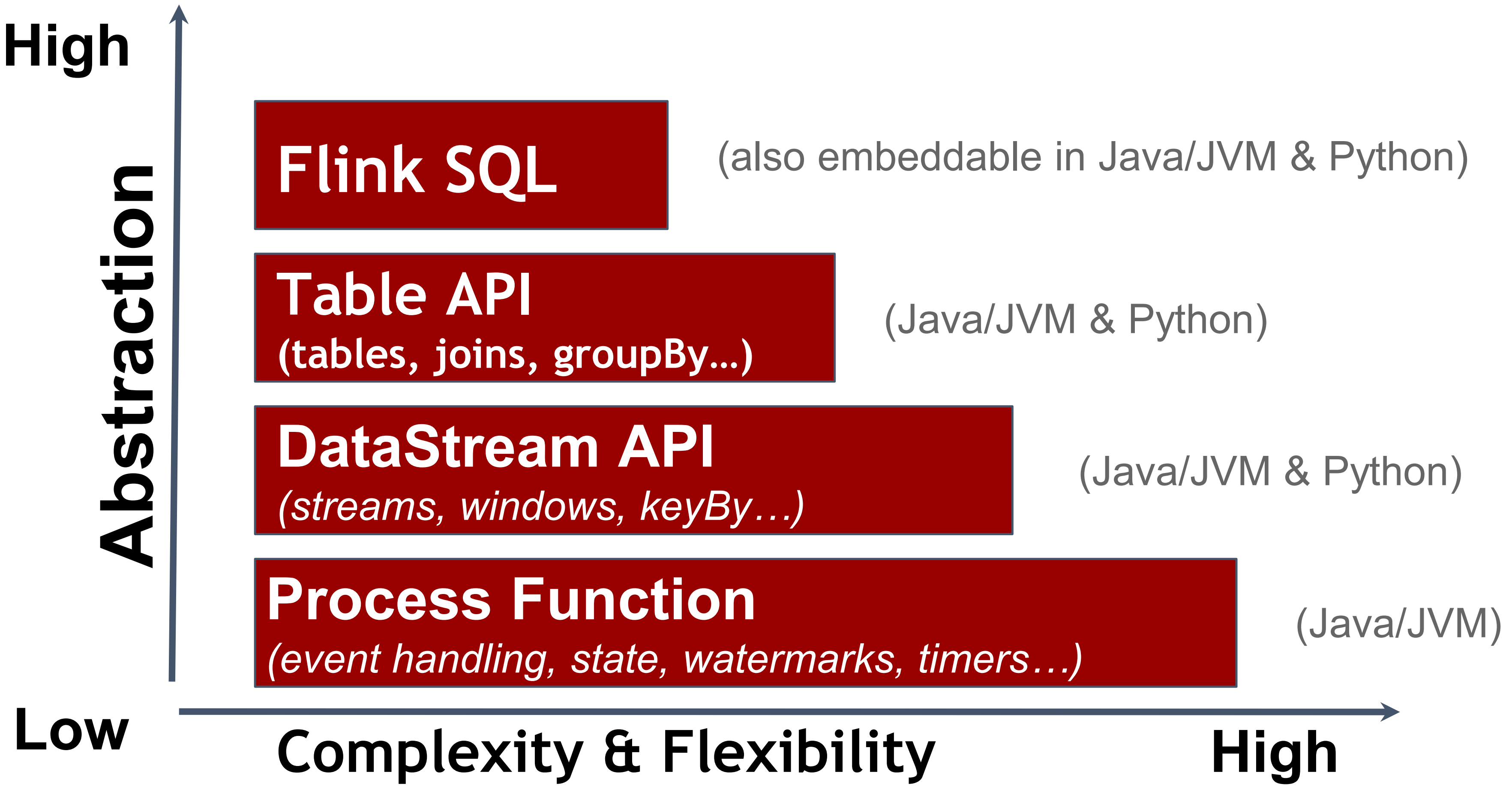
How does Apache Flink work?

1. Dataflow Model

- A Flink application is built as a **dataflow graph.**
    - **Sources:** where data comes in (files, message queues like Kafka, databases, search engines).
    - **Transformations:** operations applied on data streams (e.g., filtering, aggregations, pattern detection).
    - **Sinks:** where results are written (datastores, dashboards, files, etc.).
- The graph is **acyclic** but can be highly complex.
- Processing happens in **real time** with **high throughput** and **low latency.**



Apache Flink: Multiple Abstractions & Languages



High

Abstraction

Low

Flink SQL (also embeddable in Java/JVM & Python)

Table API (tables, joins, groupBy...) (Java/JVM & Python)

DataStream API *(streams, windows, keyBy…)* (Java/JVM & Python)

Process Function *(event handling, state, watermarks, timers…)* (Java/JVM)

Complexity & Flexibility

High

Why Flink Excels at Stateful Stream Processing

True Stream-first Engine – Batch is treated as bounded streams.

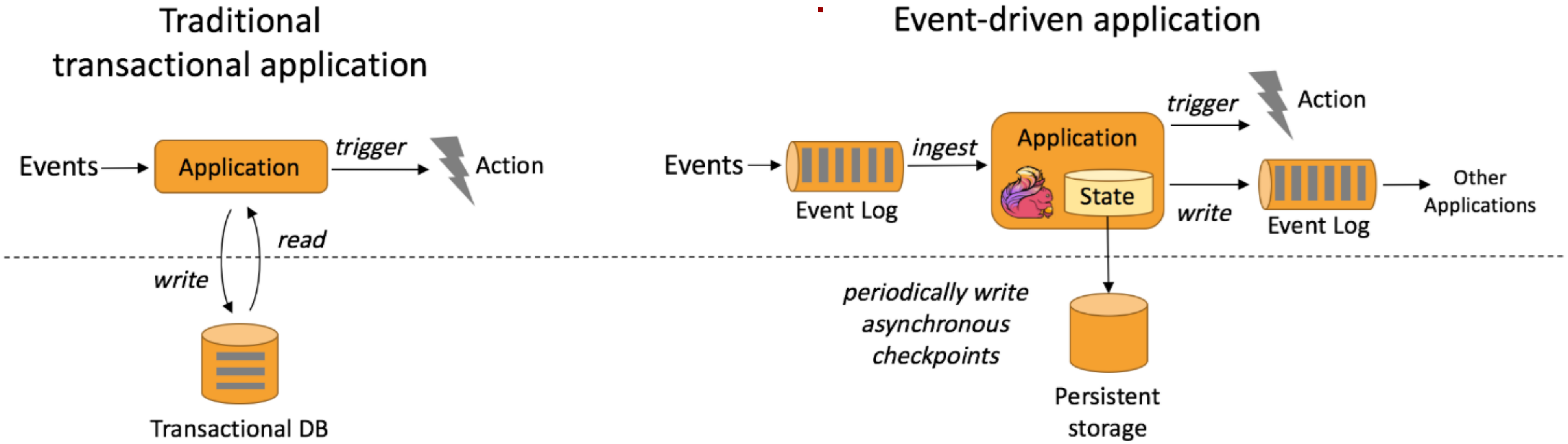
Stateful Processing – Applications keep **local state** for fast access.

Exactly-once Consistency – Through checkpoints & savepoints.

Low Latency + High Throughput – Designed for real-time workloads.

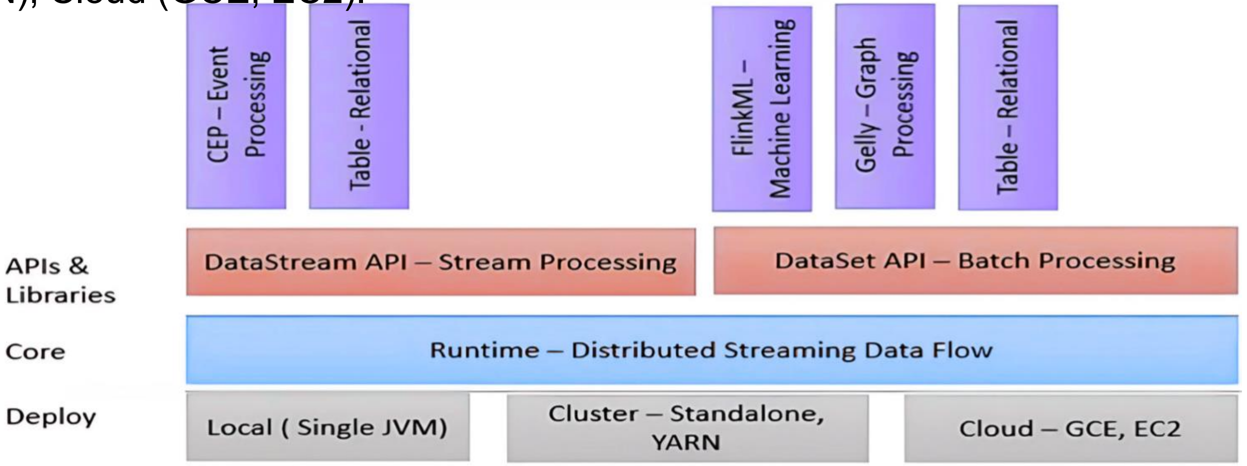
Scalability & Fault Tolerance – Distributed, resilient to failures.

Flexible Deployments – Local, cluster (YARN, Kubernetes), cloud.



Flink's architecture

- **APIs & Libraries**: CEP (events), Table/SQL, FlinkML (ML), Gelly (graphs).
- **Core APIs**:
  - **DataStream API** – Stream processing (unbounded).
  - **DataSet API** – Batch processing (bounded).
- **Runtime**: Distributed streaming data flow engine with parallelism & fault tolerance.
- **Deployment**: Local (JVM), Cluster (Standalone/YARN), Cloud (GCE, EC2).


CEP – Event Processing
Table - Relational
FlinkML – Machine Learning
Gelly – Graph Processing
Table – Relational
APIs & Libraries
DataStream API – Stream Processing
DataSet API – Batch Processing
Core
Runtime – Distributed Streaming Data Flow
Deploy
Local (Single JVM)
Cluster – Standalone, YARN
Cloud – GCE, EC2

Jobs, Tasks, and Operators

Jobs

- Represent a **user-submitted application**.
- Defined as a **JobGraph** (logical graph).
- Built using **sources** → **transformations** → **sinks**.

Operators

- Fundamental operations: **map, filter, keyBy, reduce, etc.**
- Connected into a **directed acyclic graph (DAG)**.
- Define the **flow of data** in the job.

Tasks

- **Parallel units of execution** derived from operators.
- Each processes a **data partition independently**.
- Mapped to **available cluster resources**.

Apache Flink Cluster Model

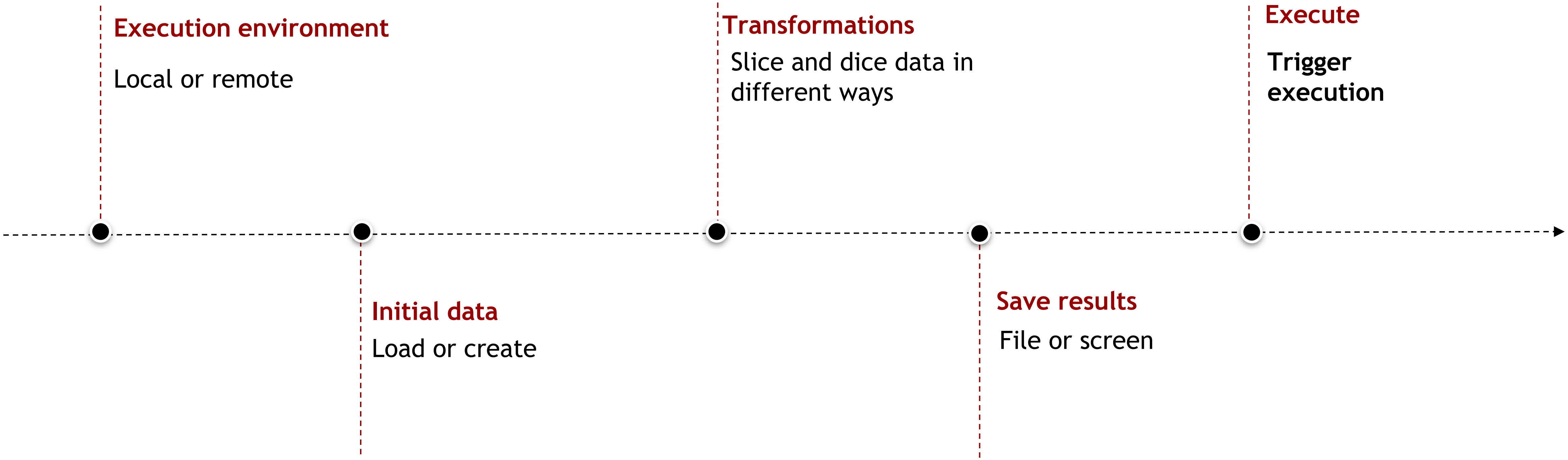
Master Components

- **Dispatcher**
  - Accepts job submissions (REST, CLI, Web UI).
  - Assigns unique Job IDs & forwards to JobManager.
- **JobManager**
  - Orchestrates job scheduling, checkpointing, state management, recovery.
  - Converts **JobGraph → Execution Graph**.
  - Coordinates execution & failover handling.
- **ResourceManager**
  - Integrates with **YARN, Kubernetes, AWS**.
  - Allocates resources & assigns task slots.
  - Enables **elastic scaling**.

Worker Component

- **TaskManagers**
  - Worker nodes running **slots** for task execution.
  - Execute operators & maintain **local + checkpointed state.**
  - Handle distributed data processing.

Basic Anatomy of a Flink Program



Execution environment

Local or remote

Initial data

Load or create

Transformations

Slice and dice data in different ways

Save results

File or screen

Execute

Trigger execution

ANY SUGGESTIONS?

THANK YOU