Apache Flink Conceptual Architecture

Team Flink Force

Apache Flink

Powerful and versatile framework for developing real-time and batch processing applications

Key Stakeholders

1

Dependents

2

Contributors

3

Apache Software Foundation

Historical Evolution

2010-2014

Inception and Recognition

- Apache Flink started as a fork of Stratosphere's distributed execution engine in 2010.
- Joined Apache incubator in March 2014 and was renamed to Apache Flink

2015-2016

Milestone

- 2015: Flink Streaming API for real-time data processing was introduced by Apache Flink 0.9
- 2016: Flink 1.0 was released which solidified stability and introduced a SQL-like querying API

2016-2021

Performance and Evolution

- 2016-2018: Performance, fault tolerance, dynamic scaling, and event time processing were prioritized.
- 2018-2021: query optimization, added Python support, and ensuring exactly-once processing semantics

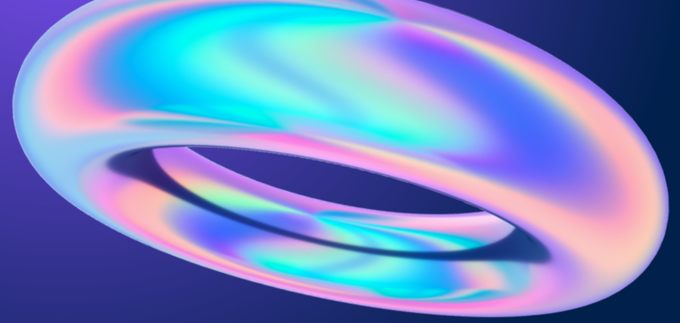
2021-2023

Recent Advances

- 2021-2023 (Versions 1.15-1.17): Delete and Update API in Flink SQL for batch processing and the “Gateway mode” for SQL Client interaction.

Contribution

Github	JIRA	Stack Overflow	Wiki Documentation	Mentorship
Development discussion platform for developers.	A systematic approach to problem resolution through issue tracking	Q & A Forum for common questions	A resource for users to deepen their understanding	Mentorship and support program for newcomers.

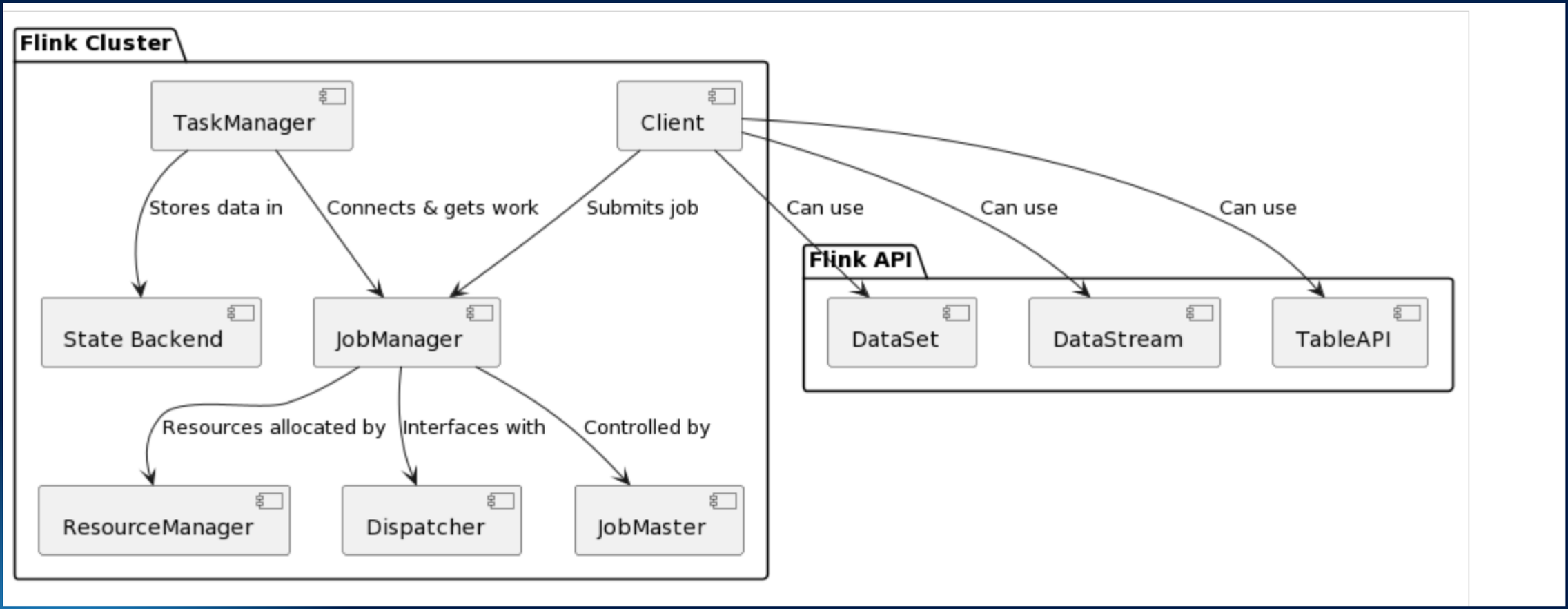


Architecture of Apache Flink & Component Analysis

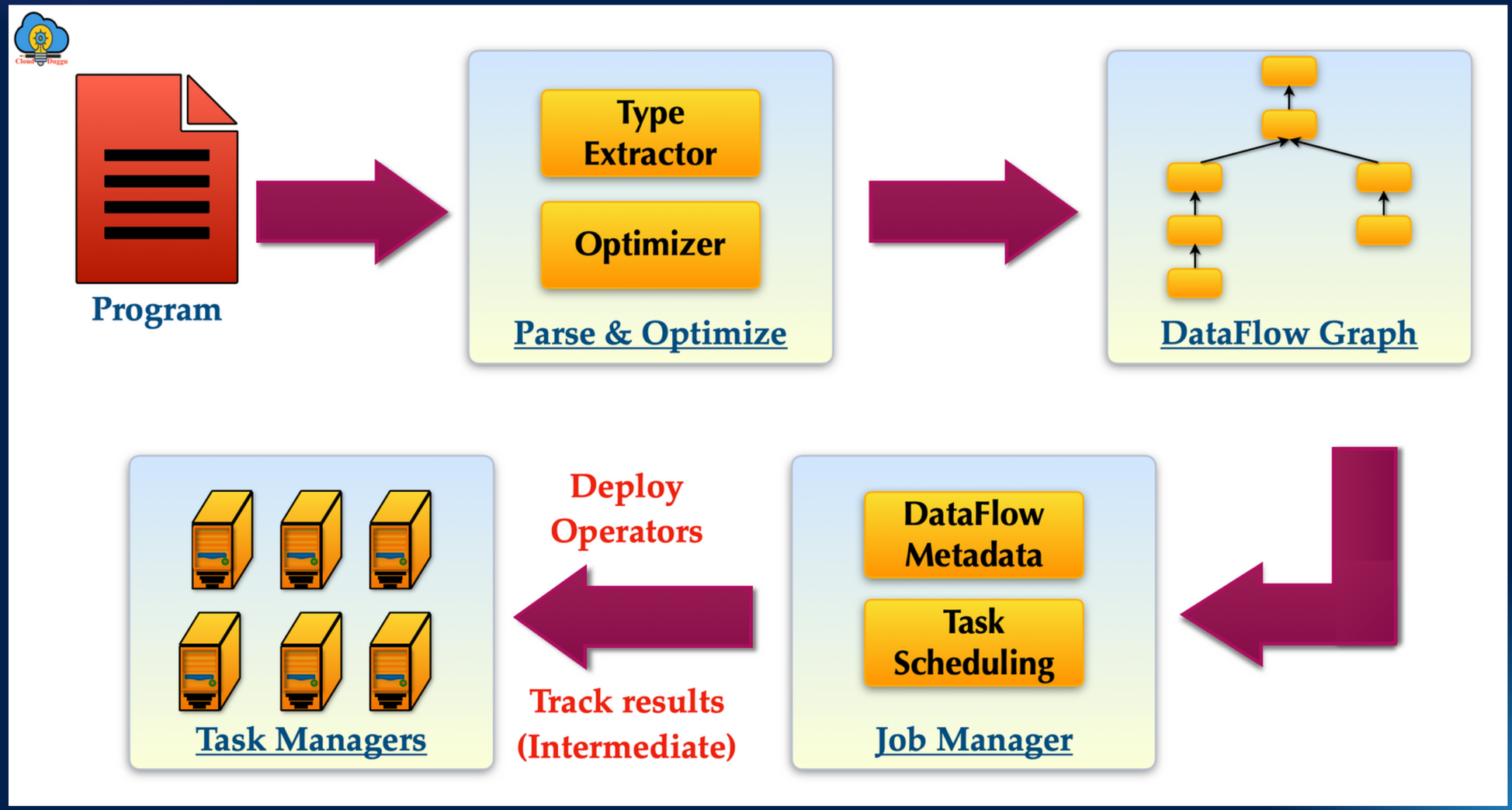
Technology Stack

- Core languages
  - Java (85.9%)
  - Scala (9.9%)
  - Python (2.8%)
  - Typescript (0.4%)
  - Other (1.4%)

Component Breakdown
Client-Server Interactions



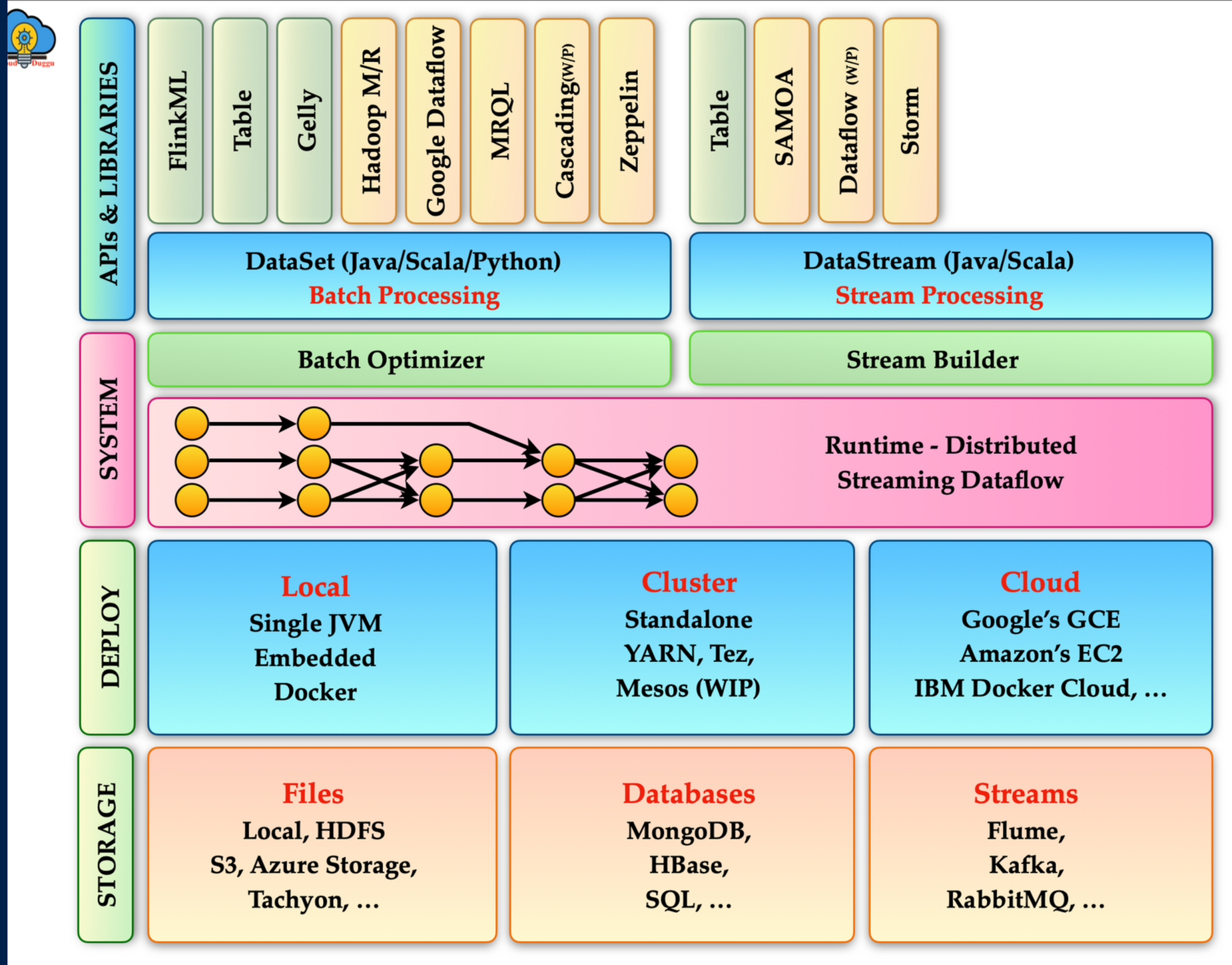
Inter-Component Data Flow



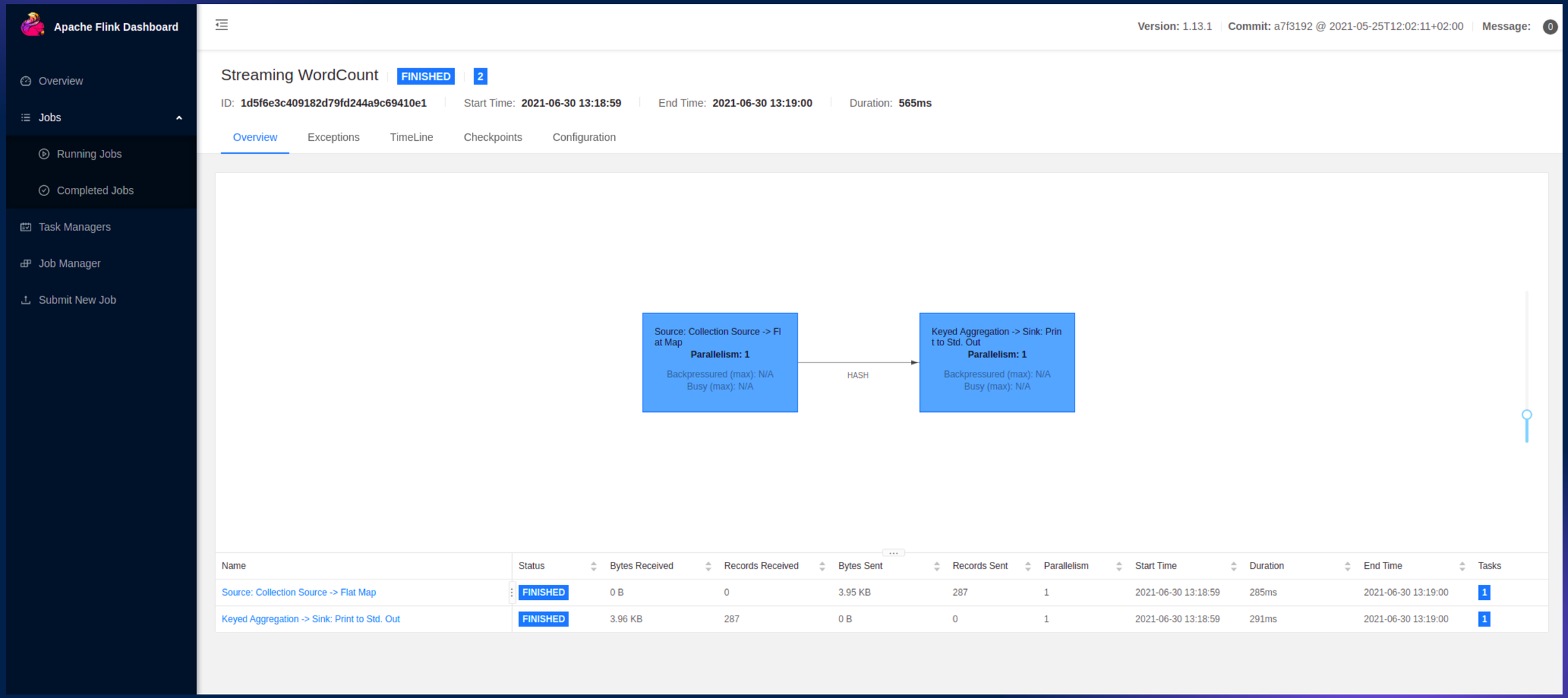
Architectural Styles

- Layered - Core Technology Stack
- Distributed - Scalability & Storage
- Repository - Durable State Storage
- Pipe & Filter - Batch & Stream Processing
- Client Server - Server Based System
- Implicit Invocation - Event Driven

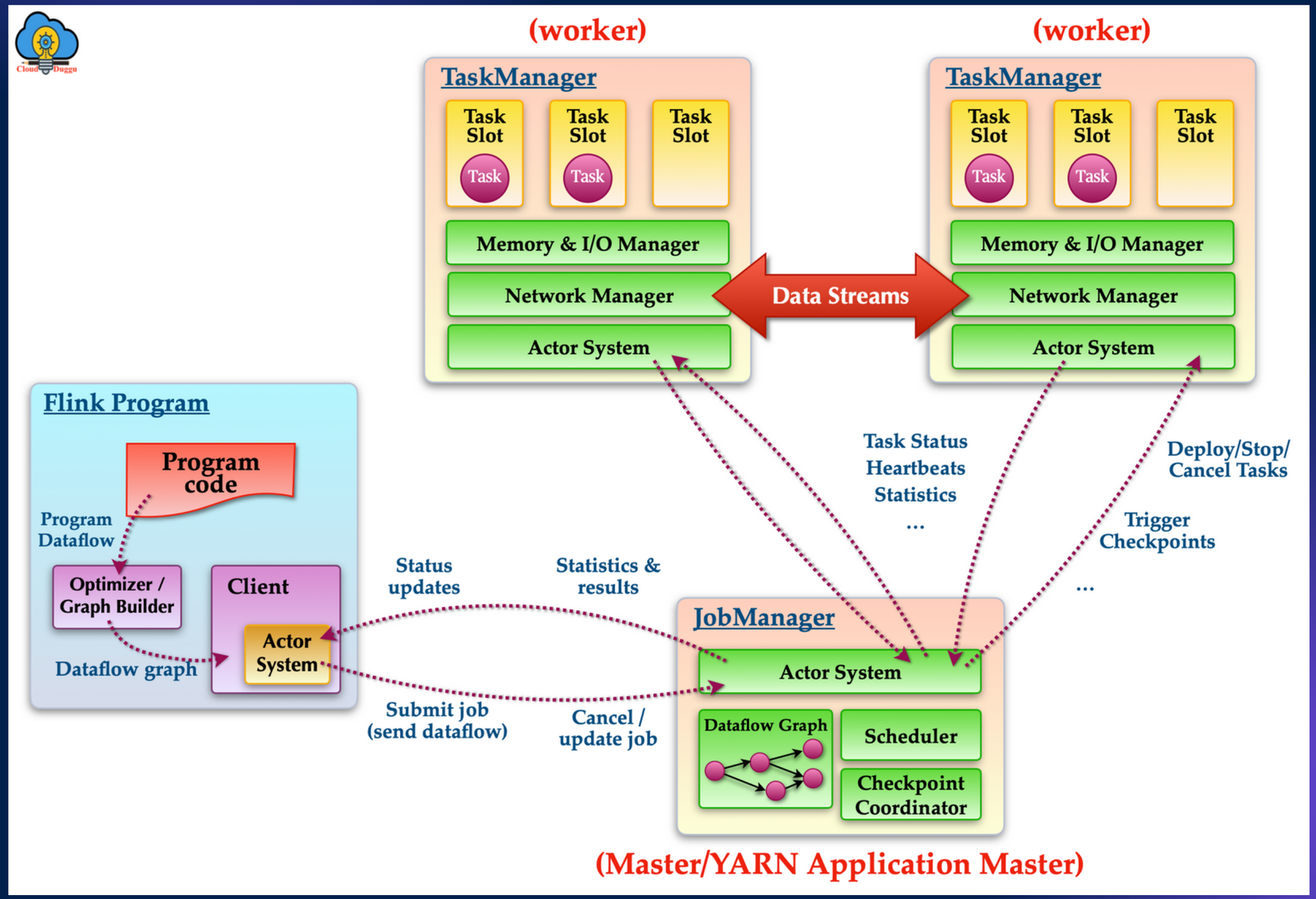
Layered System “Tech Stack”



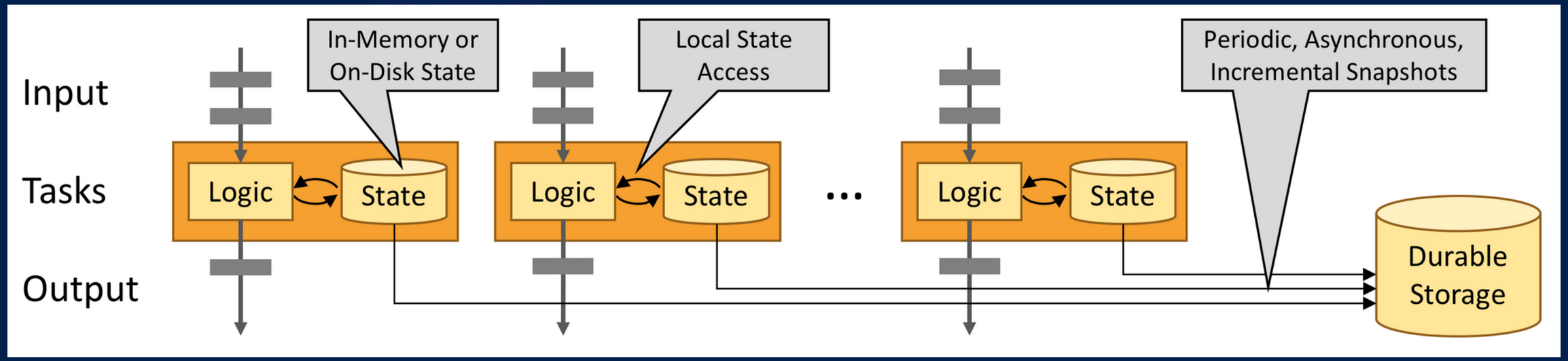
Flink Web Dashboard



Distributed System

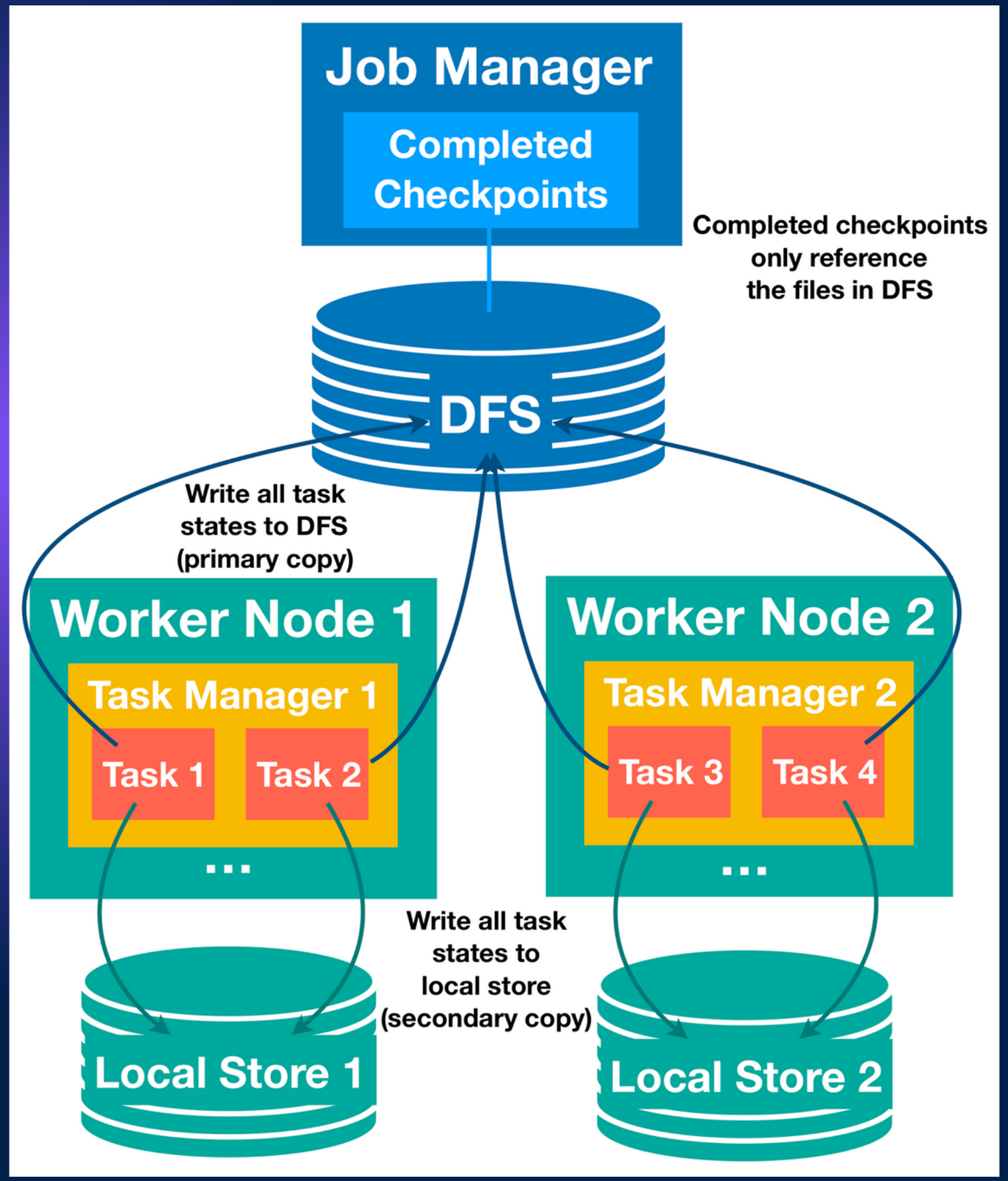


Repository



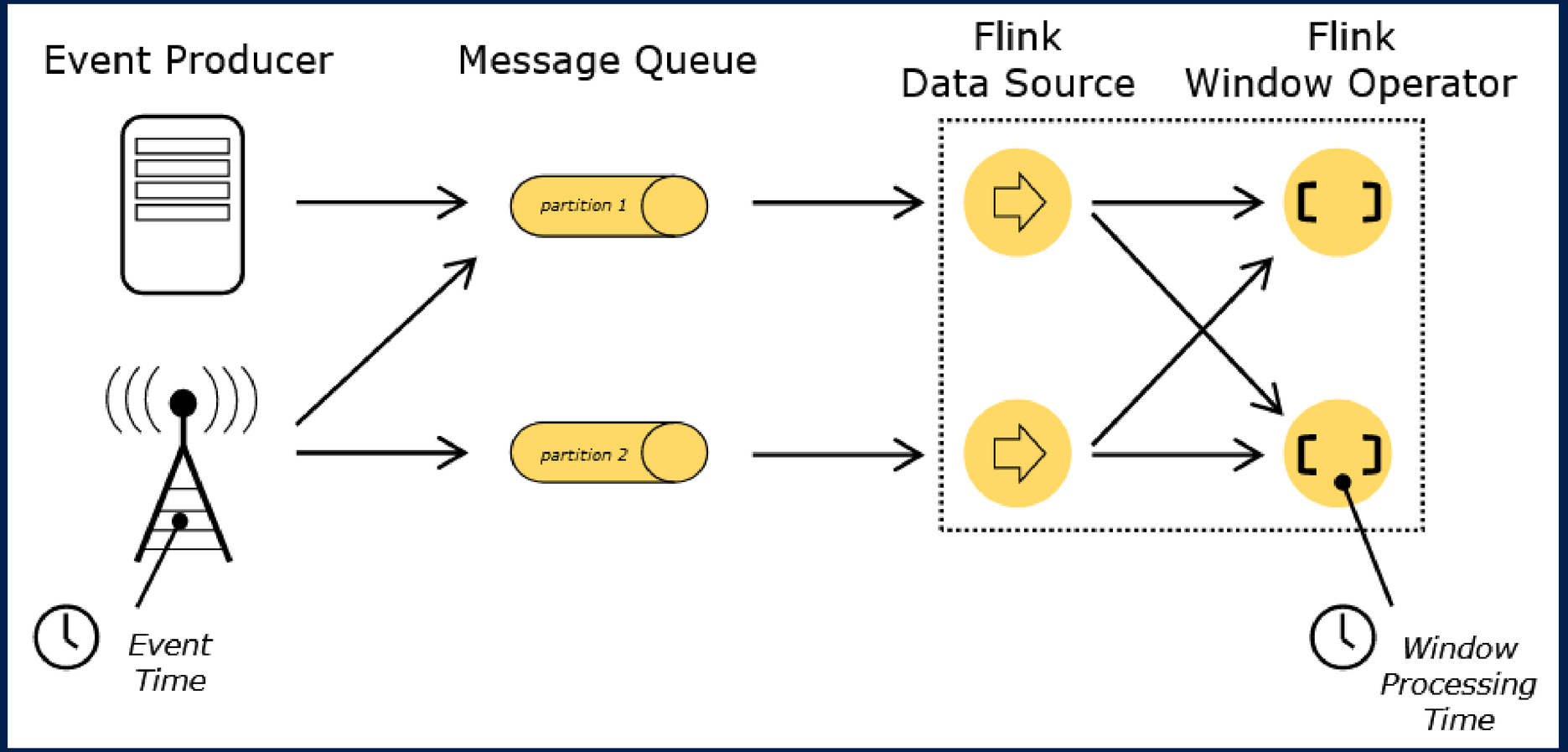
Guarantees “exactly-once” state consistency

- use of in memory state snapshots
- specialized disk data structures for backups



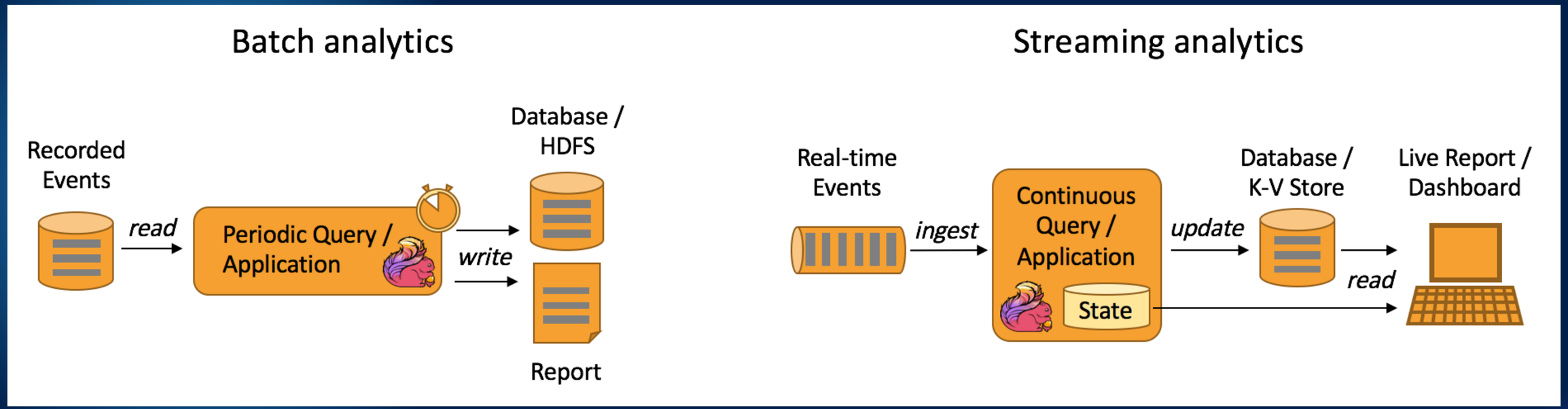
Distributed Storage & Execution + Repository

Pipe & Filter

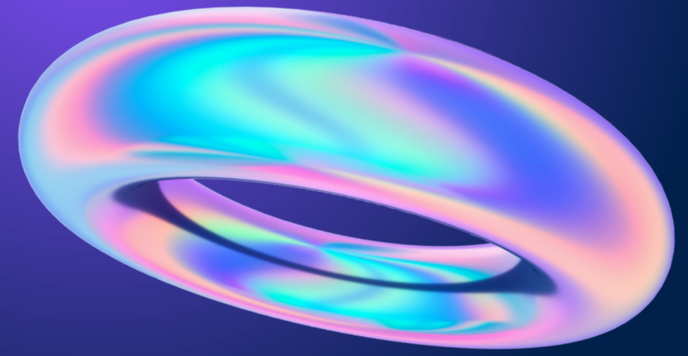


Flink serves as the component for applications to stream data from any sources to any sink

Implicit Invocation - Event Driven

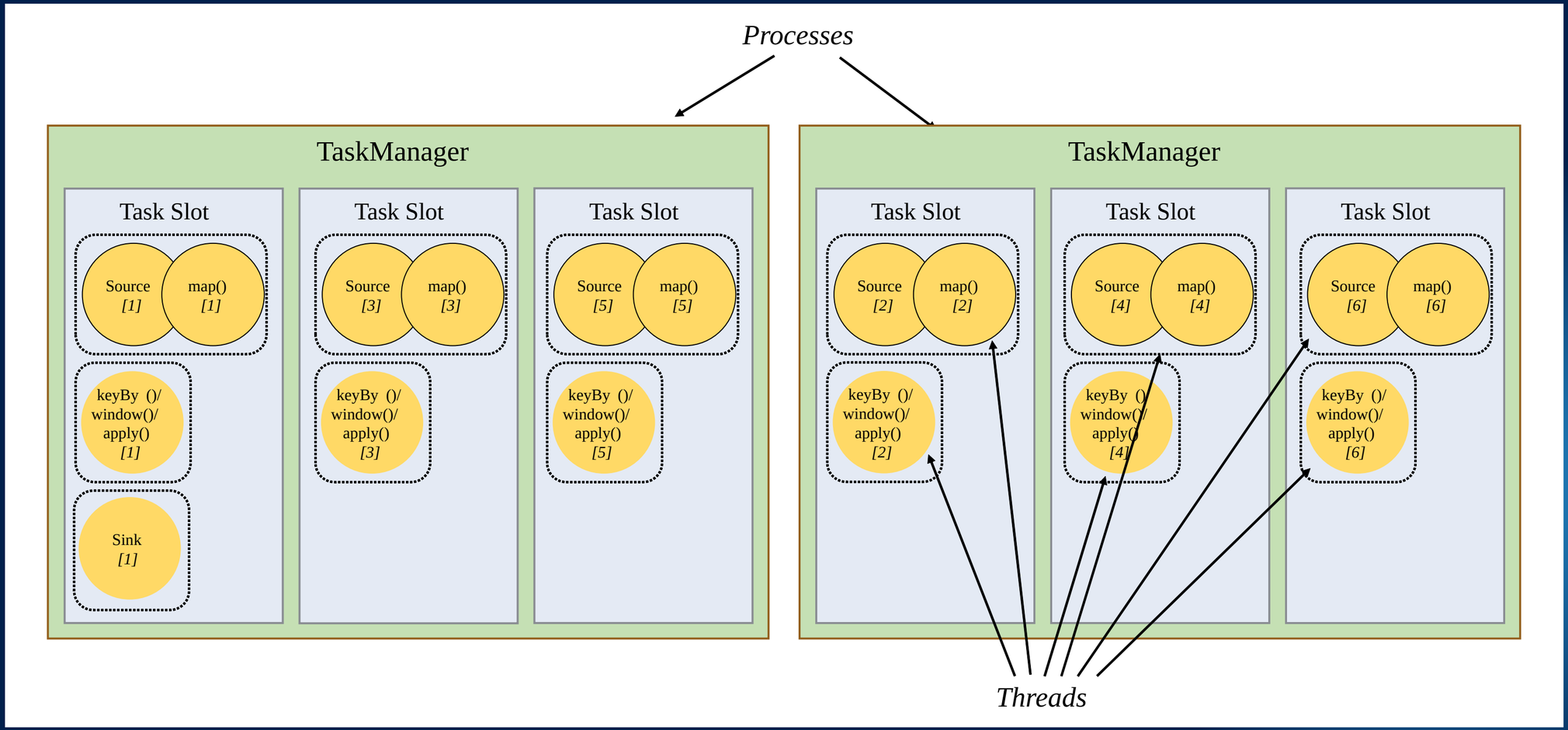


Flink applications respond to events in a stateful manner and have specific time-based semantics

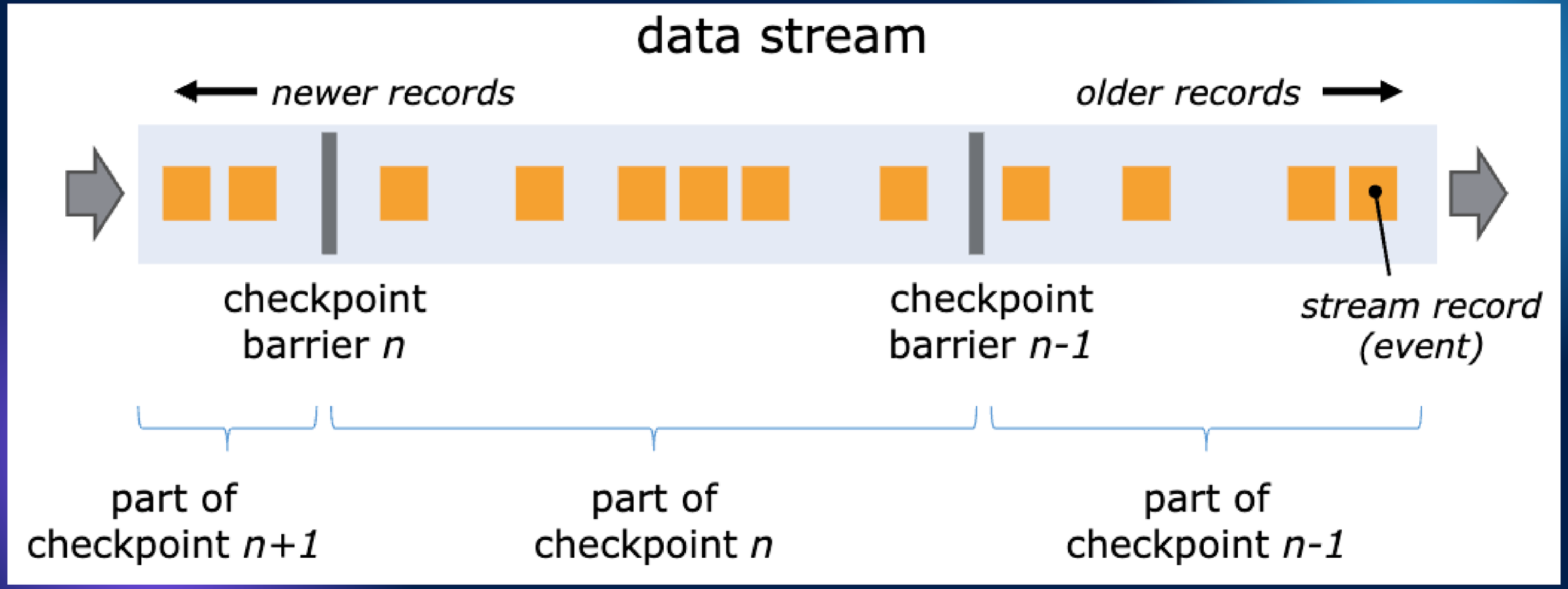


Concurrency, Responsibility Analysis, & Quality Attributes

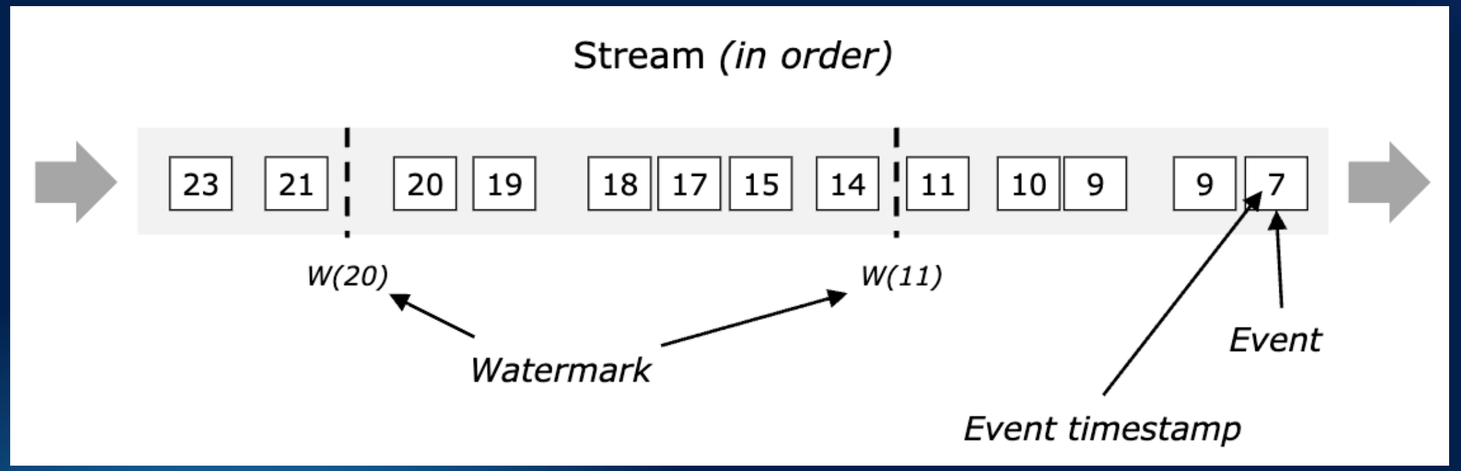
Concurrency Aspects - Operator Parallelism

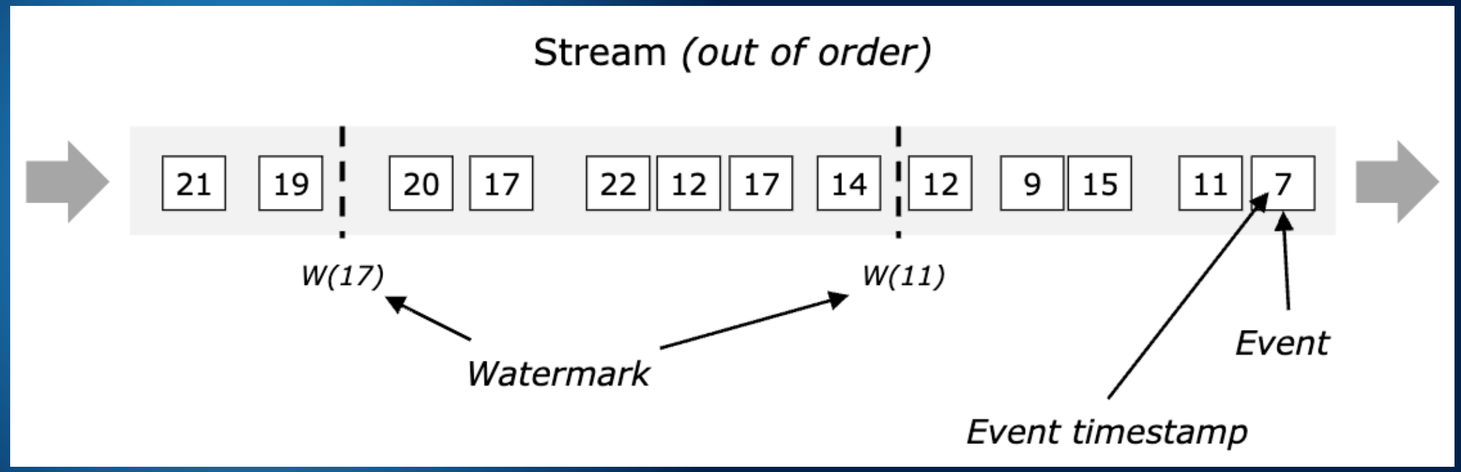


Concurrency Aspects - Checkpointing

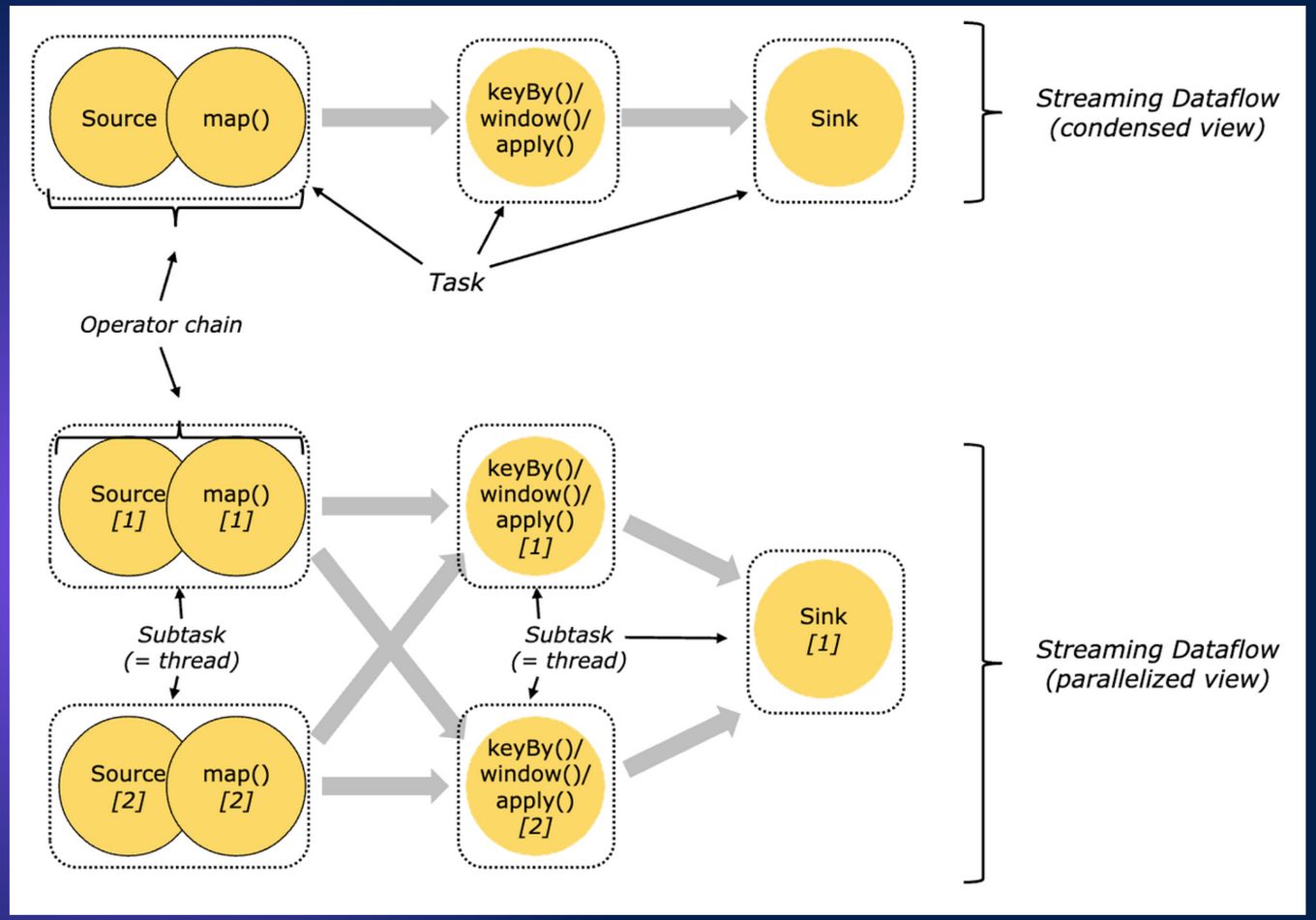


Watermarks


Stream (in order)
23
21
20
19
18
17
15
14
11
10
9
9
7
W(20)
W(11)
Watermark
Event
Event timestamp


Stream (out of order)
21
19
20
17
22
12
17
14
12
9
15
11
7
W(17)
W(11)
Watermark
Event
Event timestamp

Job and Task Managers
Task and Operator Chaining



Division of Responsibilities

- Open-source project
- Committers
- PMC (Project Manager Committee)
- Release Manager

Security Measures

- Security features
  - Authentication
  - Authorization mechanisms
  - Data encryption
- Apache Software Foundation's licensing provisions
- Individual Contributor License Agreement (ICLA)

Quality Attributes

Performance 

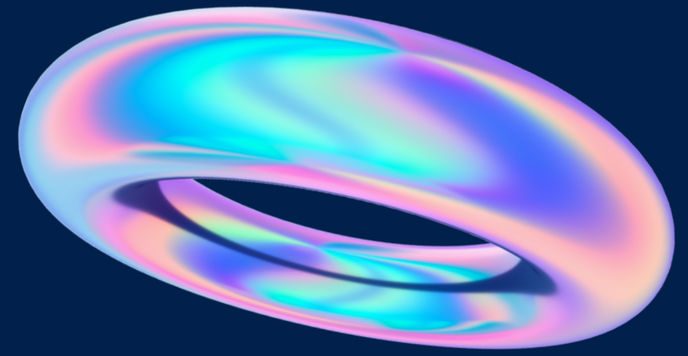
real-time data processing, complex event processing and Batch data processing

 Scalability

clustering and operational in any-scale system

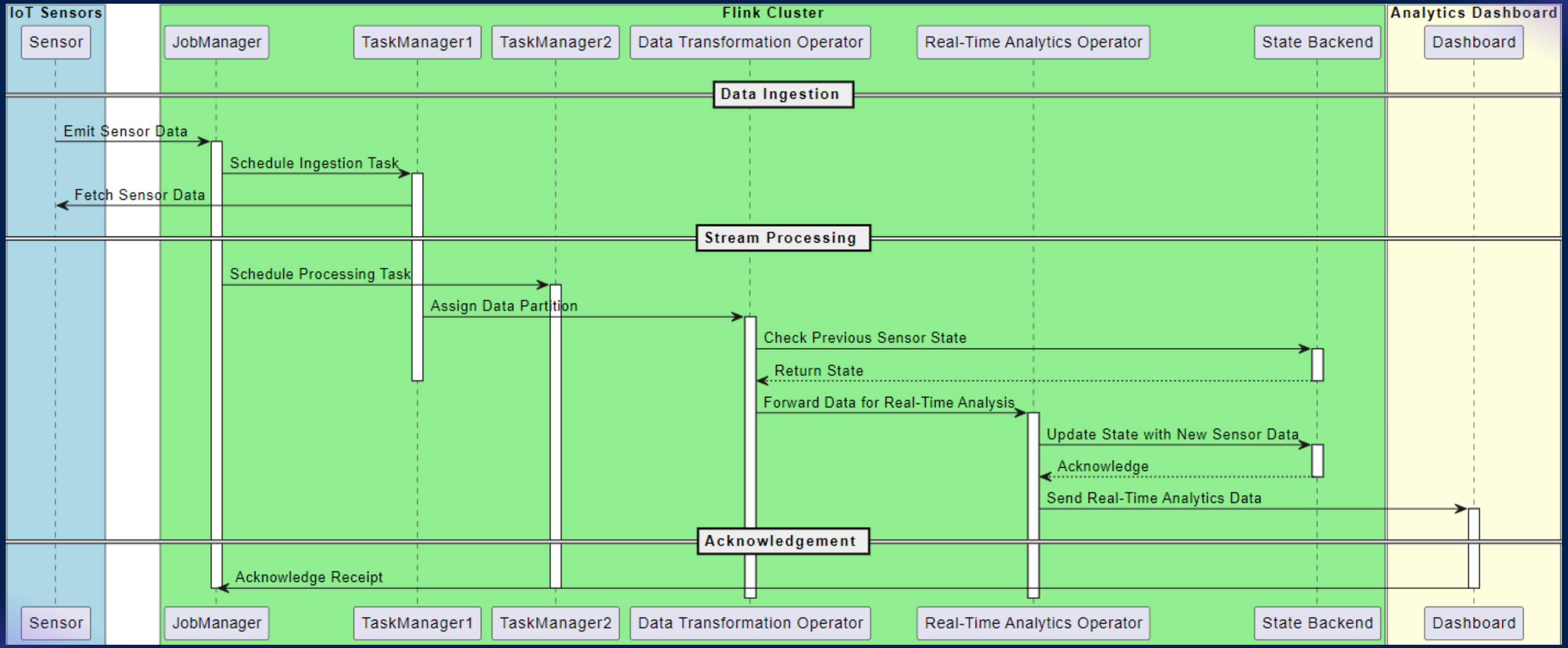
Reliability

Fault tolerance and monitoring

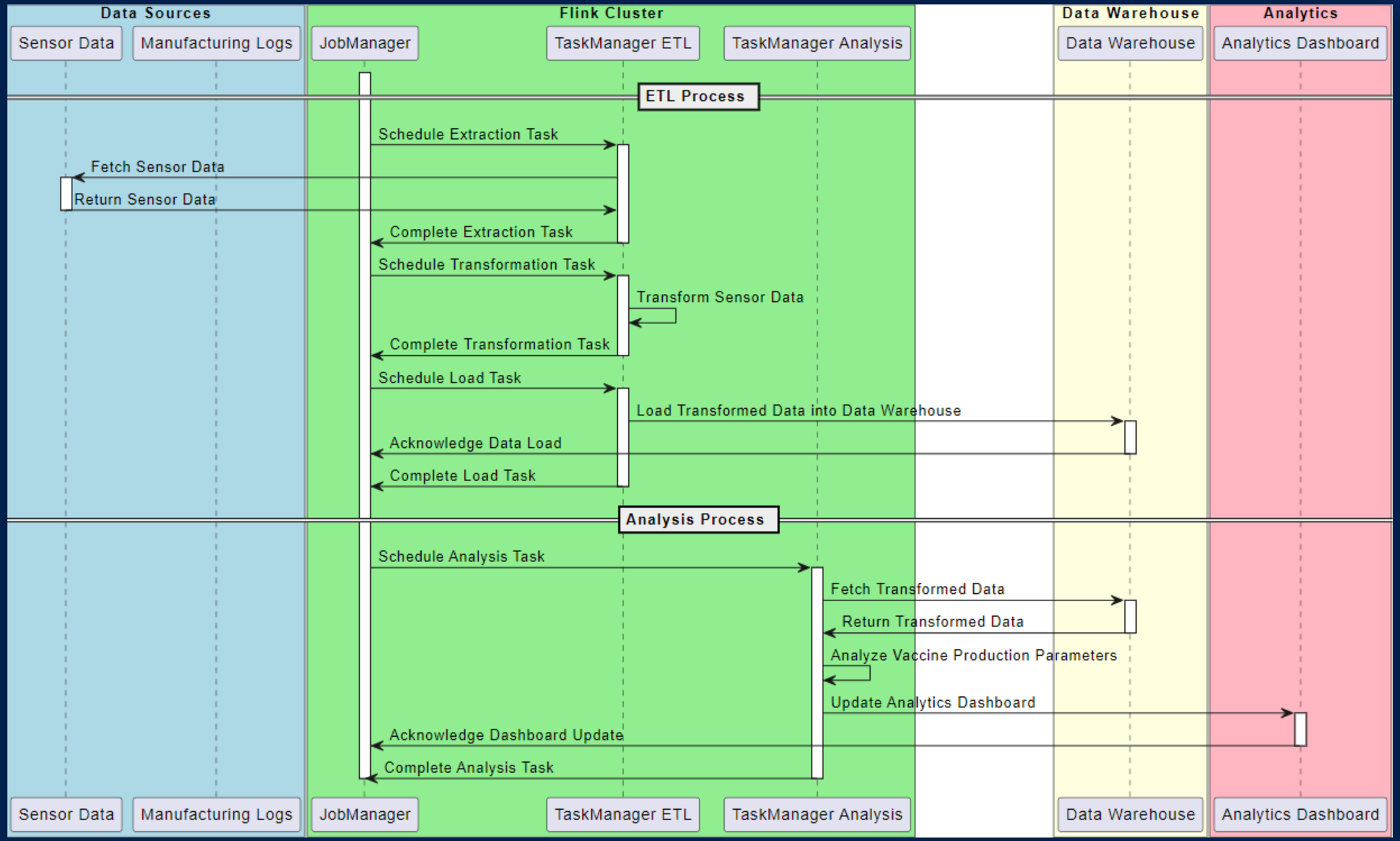


Use Cases

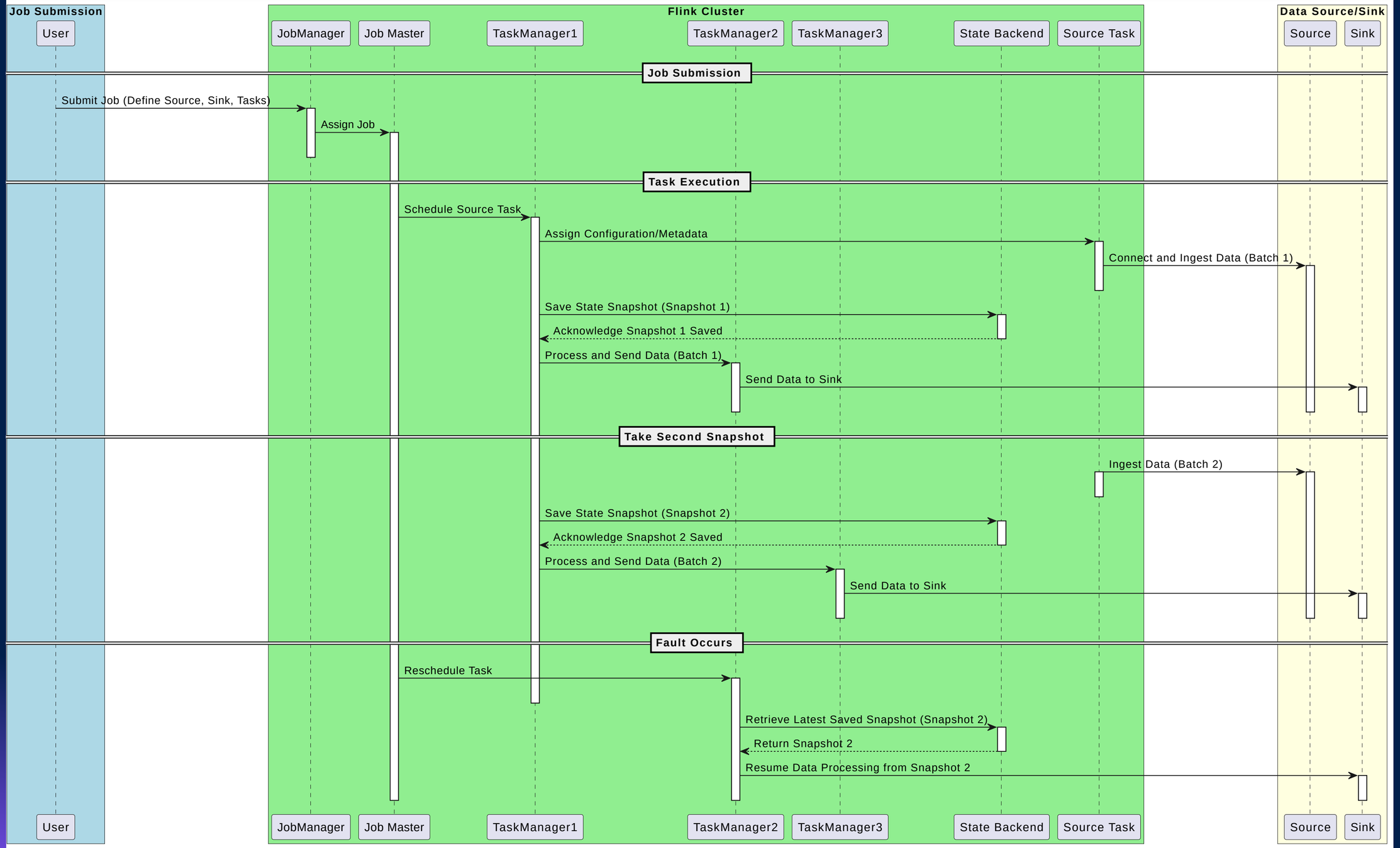
Stream Processing



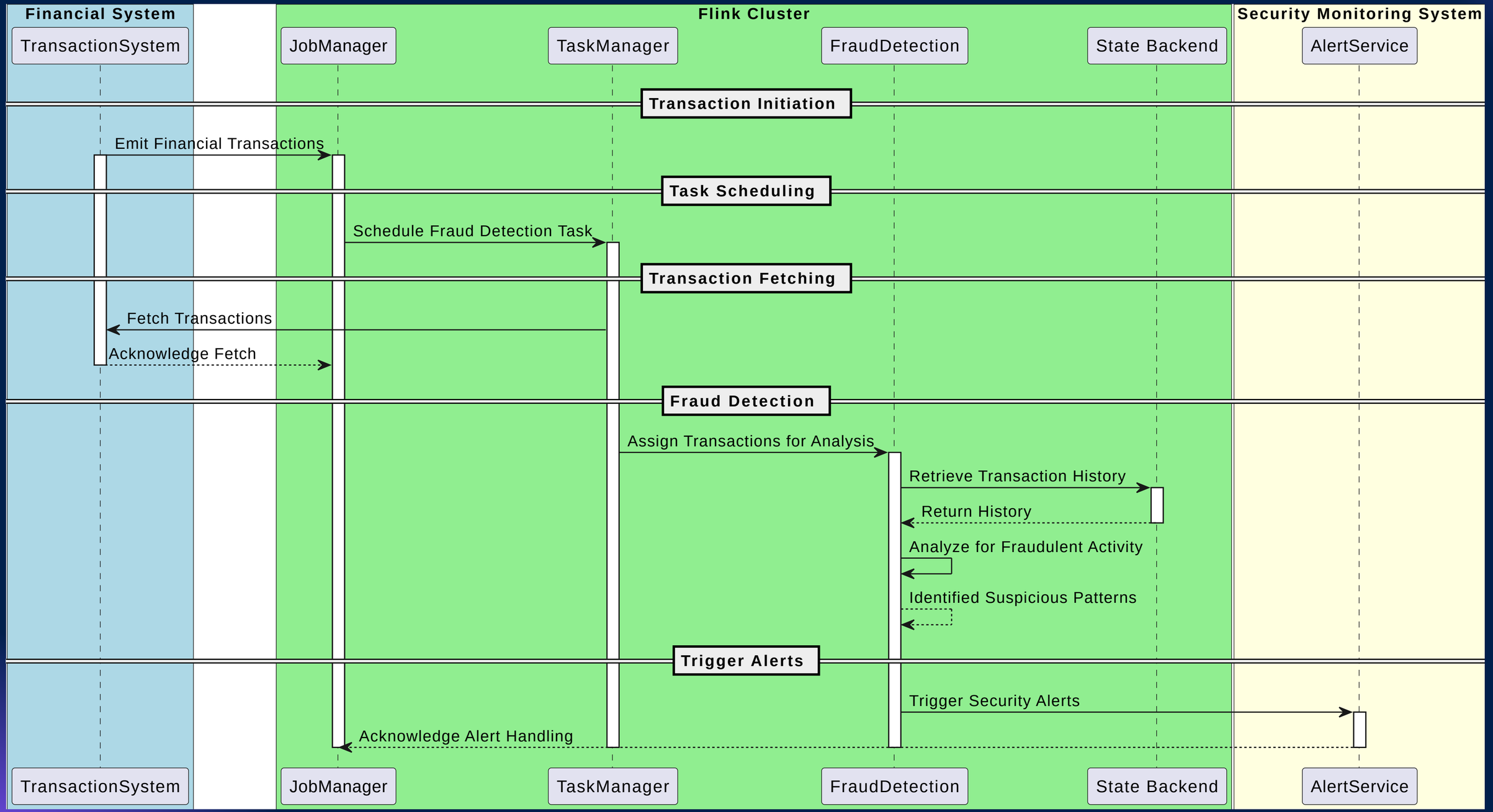
Batch Processing



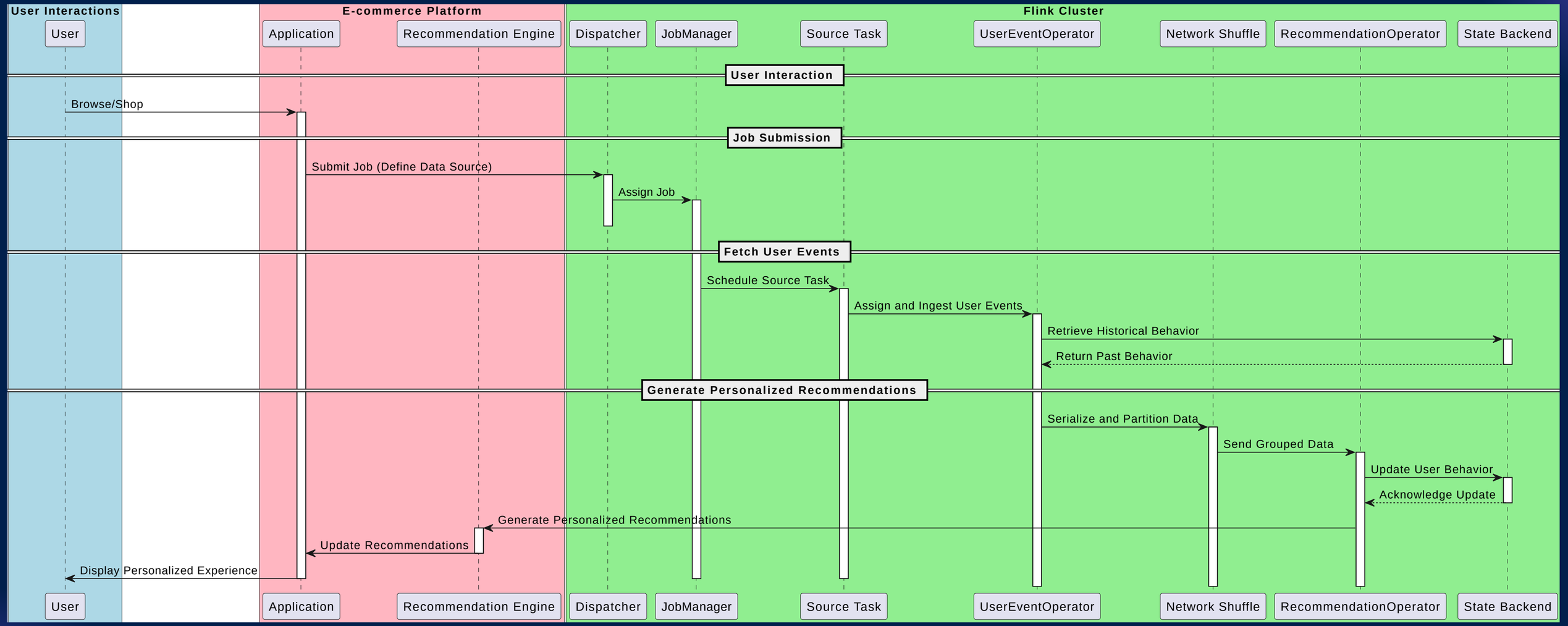
Fault Tolerance



Fraud Detection



Stateful Computations - User Recommendations



Thank You!

Any Questions?

References:

[1] https://flink.apache.org
[2] https://github.com/apache/flink
[3] https://www.apache.org
[4] https://cwiki.apache.org/confluence/display/FLINK/Flink+Bylaws
[5] https://docs.cloudera.com/csa/1.11.0/security/topics/csa-flink-security-overview.html
[6] https://flink.apache.org/powered-by/
[7] https://thenewstack.io/3-reasons-why-you-need-apache-flink-for-stream-processing/
[8] https://www.uber.com/blog/building-scalable-streaming-pipelines/
[9] https://aws.amazon.com/managed-service-apache-flink/
[10] https://flink.apache.org/how-to-contribute/overview/
[11] https://nightlies.apache.org/flink/flink-docs-release-1.17/docs/dev/datastream/operators/
[12] https://nightlies.apache.org/flink/flink-docs-release-1.17/docs/dev/datastream/execution/parallel/
[13] https://nightlies.apache.org/flink/flink-docs-release-1.17/docs/concepts/stateful-stream-processing/
[14] https://nightlies.apache.org/flink/flink-docs-release-1.17/docs/concepts/time/
[15] https://nightlies.apache.org/flink/flink-docs-release-1.17/docs/concepts/flink-architecture/
[16] https://medium.com/@BitrockIT/apache-flink-and-kafka-stream-a-comparative-analysis-f8cb5b946ec3#:~:text=Kafka%20Streams%20is%20a%20partially,data%20processing%2C%20and%20data%20analytics.
[17] https://nightlies.apache.org/flink/flink-docs-release-1.17/
[18] https://apache.googlesource.com/flink/+/release-1.0.2/flink-runtime-web/README.md
[19] https://www.cloudduggu.com/flink/architecture/

Contributing Members:

Rafael Dolores
Walid AlDari
Alex Arnold
Zachary Ross
Hashir Jamil
Nabaa Gaziy
Maaz Sidiqqui