# APACHE FLINK

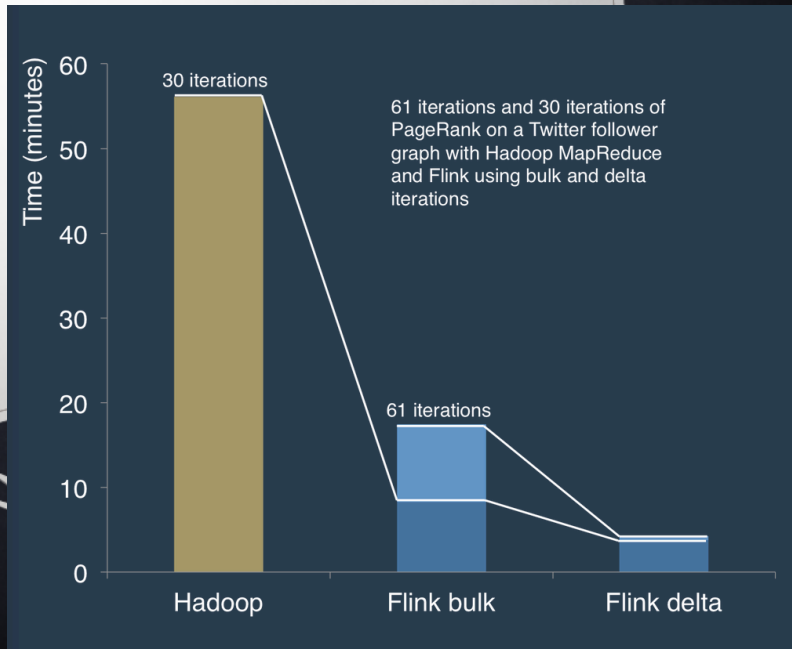
*By: Sheefali Tewari*

# WHAT IS APACHE FLINK?

- Fast and reliable large-scale data processing engine

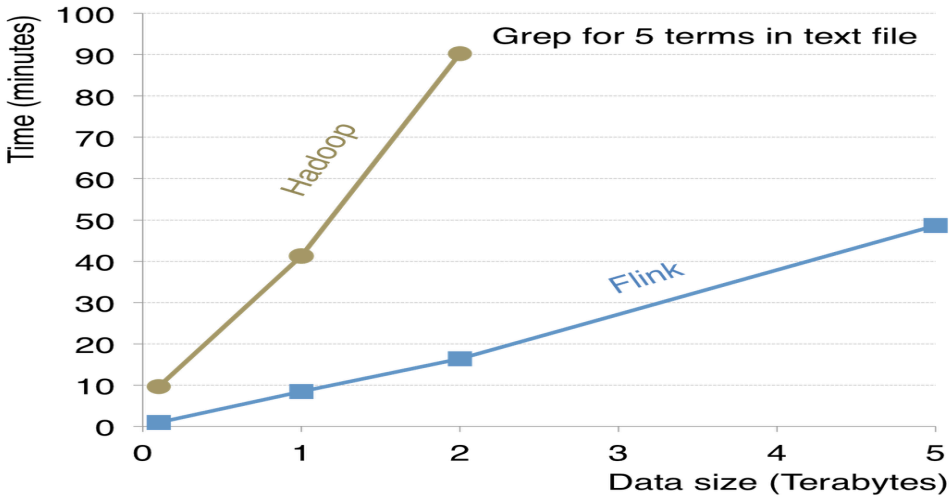
# PROPERTIES

- Fast
  - State-of-the-art performance exploiting in-memory processing and data streaming
  - Makes system extremely fast for data-intensive and iterative jobs



# PROPERTIES

- Reliable and scalable
  - Designed to perform well even when the cluster's memory runs out
  - Contains its own memory management component, serialization framework, and type inference engine
  - Tested on clusters of 100s of machines, Google Compute Engine, and Amazon EC2



# PROPERTIES

- Expressive
  - Can be written in Java and Scala
    - Native Java and Scala data types without packing them into key-value pairs, logical field addressing, and a wealth of operators
  - Executable on a cluster

**Word Count in Flink's Scala API**

```
case class Word (word: String, frequency: Int)

val counts = text
  .flatMap {line => line.split(" ").map(word => Word(word,1))}
  .groupBy("word").sum("frequency")
```

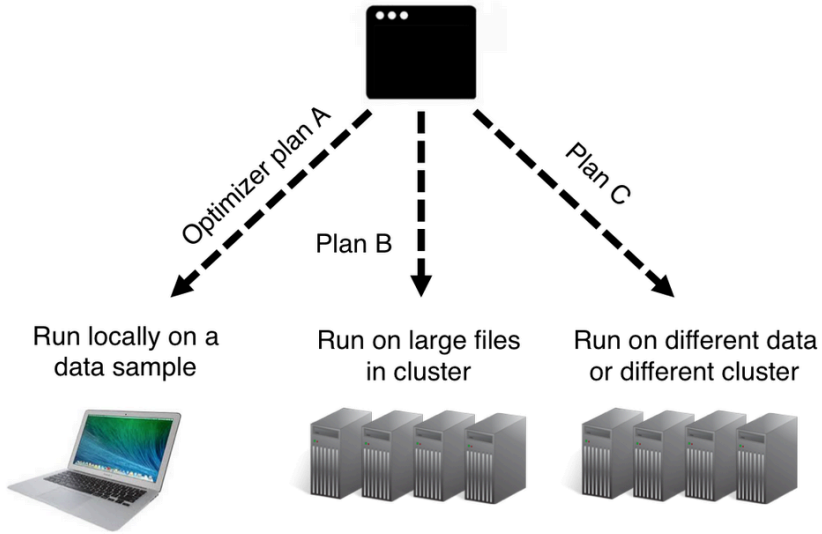
**Transitive Closure**

```
case class Path (from: Long, to: Long)

val tc = edges.iterate(10) { paths: DataSet[Path] =>
  val next = paths
    .join(edges).where("to").equalTo("from") {
      (path, edge) => Path(path.from, edge.to)
    }
    .union(paths).distinct()
  next
}
```

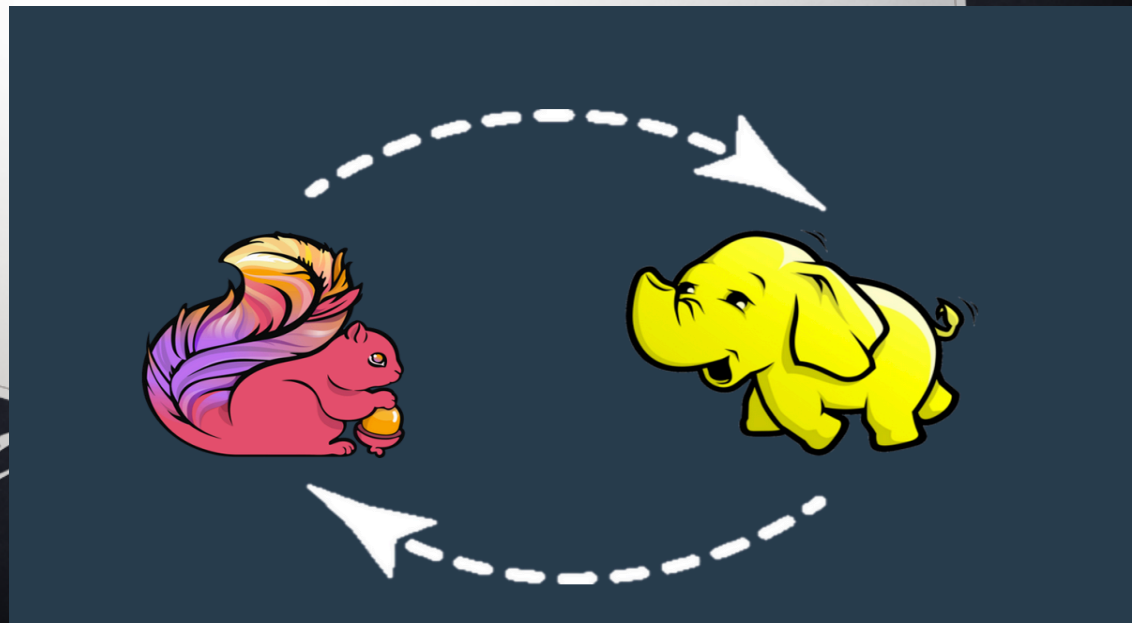
# PROPERTIES

- Easy to use
  - Few configuration parameters required
  - Cost-based optimizer built in takes care of finding the best way to execute the program in any environment



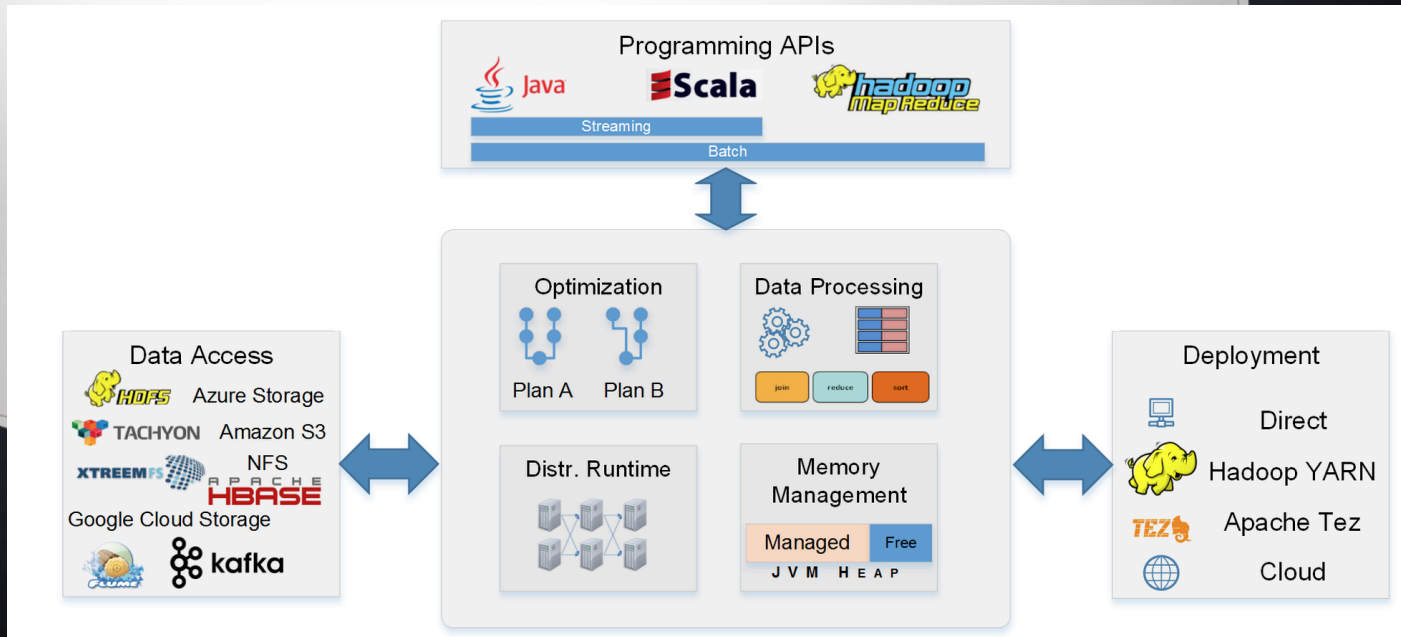
# PROPERTIES

- Hadoop-compatible
  - Runs on YARN and HDFS and has a Hadoop compatibility package
  - Can run legacy MapReduce operators unmodified and faster
  - Supports all Hadoop input and output formats and data types



# SYSTEM OVERVIEW

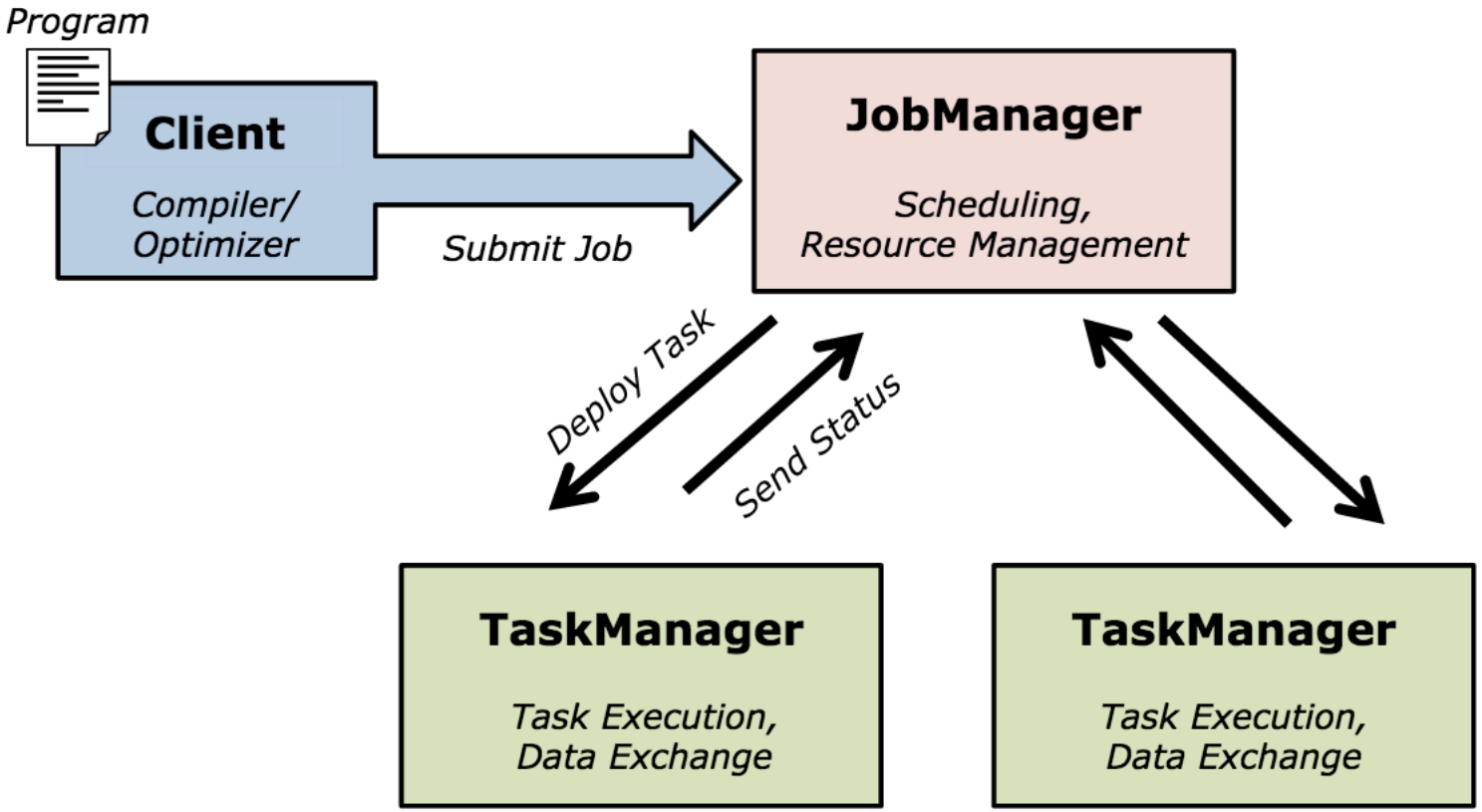
- Flink contains APIs in Java and Scala for analyzing data from batch and streaming data sources, as well as its own optimizer and distributed runtime with custom memory management



# ARCHITECTURE AND PROCESS MODEL

- Processes
  - When the Flink system starts, it brings up JobManager and one or more TaskManagers
  - JobManager = coordinator of the Flink system
  - TaskManager = Workers that execute parts of the parallel programs
  - When starting the system in local mode, a single JobManager and TaskManager are brought up within the same JVM
  - Once program is submitted → client is created that performs the pre-processing and turns the program into the parallel data flow form that is executed by the JobManager and TaskManagers

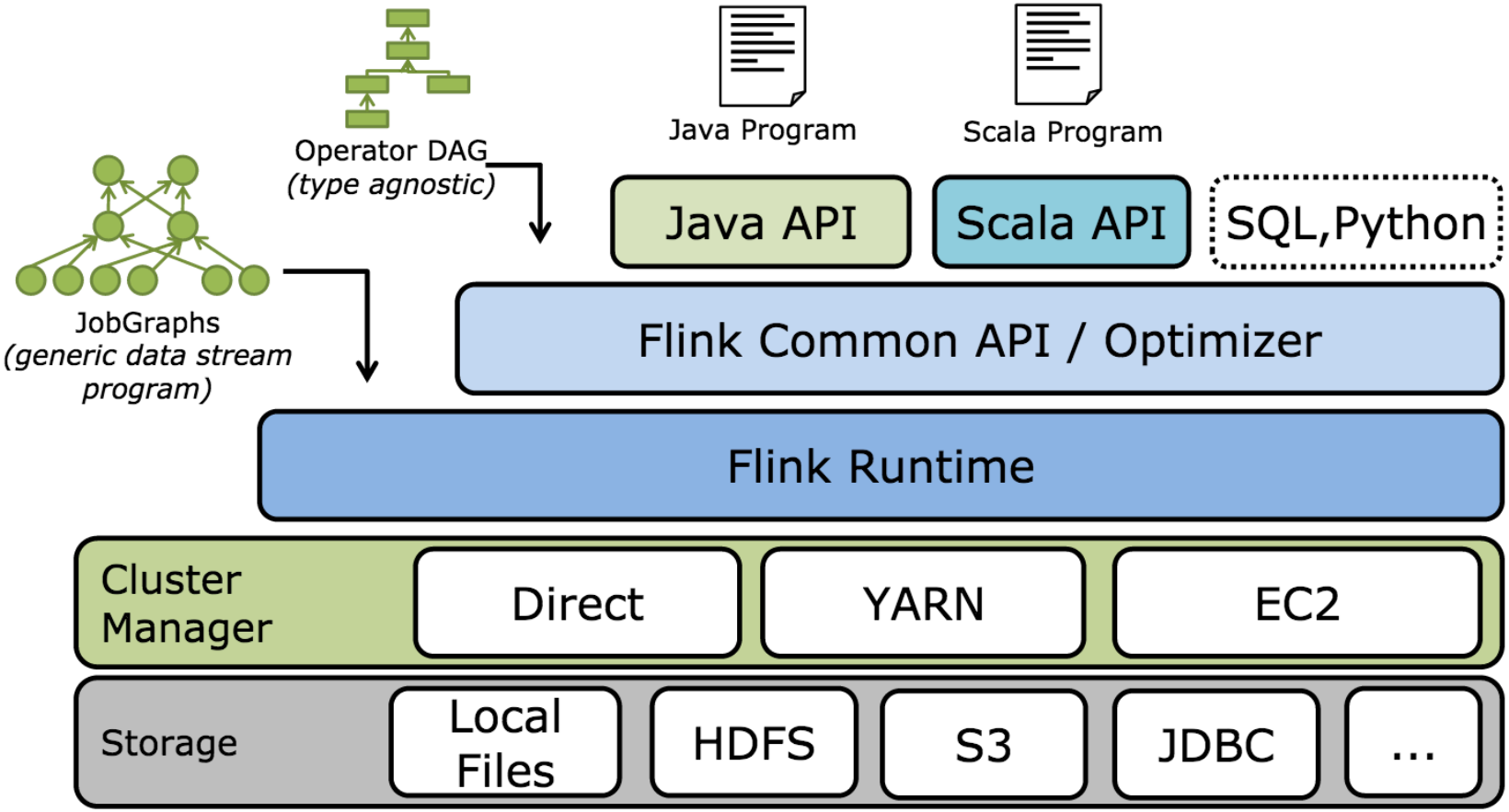
# ARCHITECTURE AND PROCESS MODEL



# ARCHITECTURE AND PROCESS MODEL

- Component stack
  - Different layers of the stack build on top of each other and raise the abstraction level of the program representations they accept:
    - Runtime
      - This layer receives a program in the form of a JobGraph
        - JobGraph → generic parallel data flow with arbitrary tasks that consume and produce data streams
    - Optimizer and common api
      - This layer takes programs in the form of operator DAGs
      - Operators are specific (e.g., Map, Join, Filter, Reduce, etc.) but are data type agnostic
      - Concrete types and their interaction with the runtime is specified by the higher layers
    - API layer
      - This layer implements multiple APIs that create operator DAGs for their programs
      - Each API needs to provide utilities (serializers, comparators) that describe the interaction between its data types and runtime

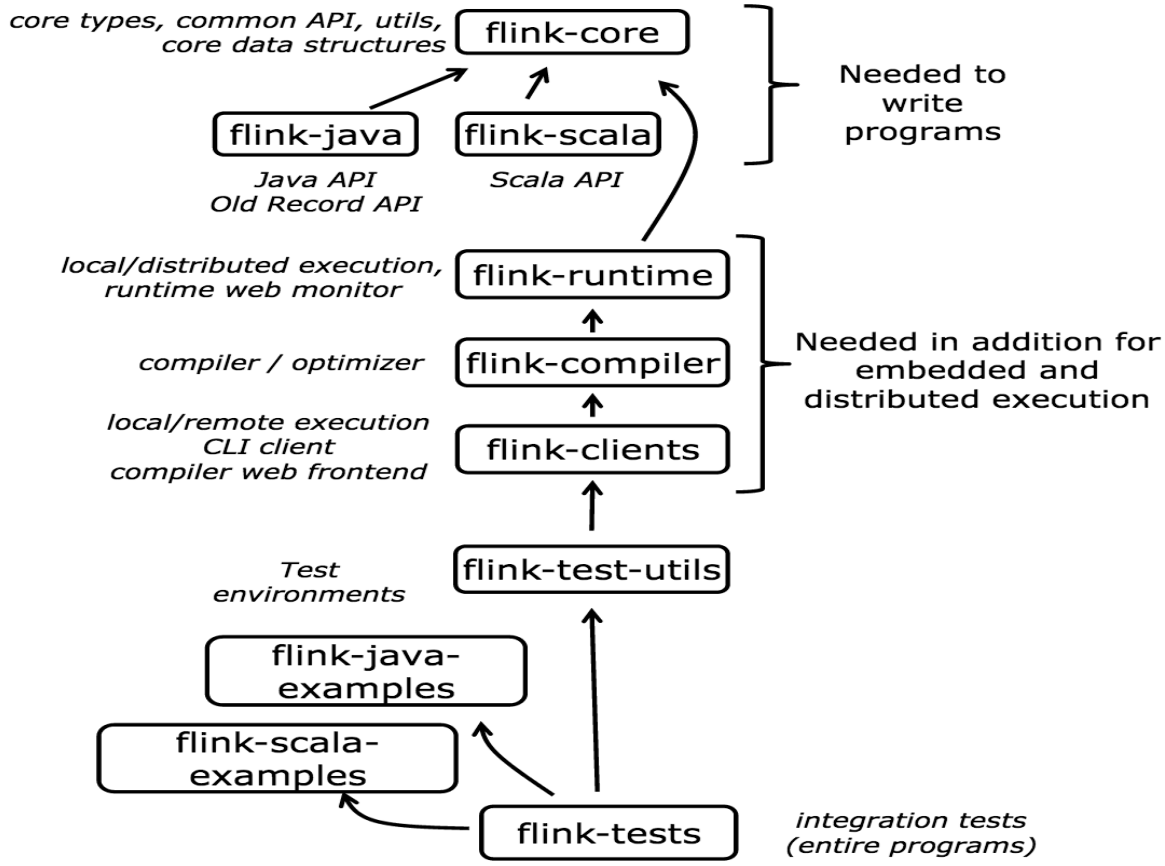
# ARCHITECTURE AND PROCESS MODEL



# ARCHITECTURE AND PROCESS MODEL

- Projects and dependencies
  - Flink system code is divided into multiple sub-projects
  - Goal is to reduce the number of dependencies that a project implementing a Flink program needs, as well as to facilitate easier testing of small sub-modules

# ARCHITECTURE AND PROCESS MODEL



# ARCHITECTURE AND PROCESS MODEL

- Projects and dependencies (continued)
  - Flink currently contains these sub-projects
    - Flink-dist: The distribution project
      - Defines how to assemble the compiled code, scripts, and other resources into the final folder structure that is ready to use
    - Flink-addons: Series of projects that are in an early version
      - Currently contains projects for YARN support, JDBC data sources and sinks, hadoop compatibility, graph specifics, and HBase connectors
    - Flink-quickstart: Scripts, maven archetypes, and example programs for the quickstarts and tutorials

# SOURCE

- http://flink.apache.org/