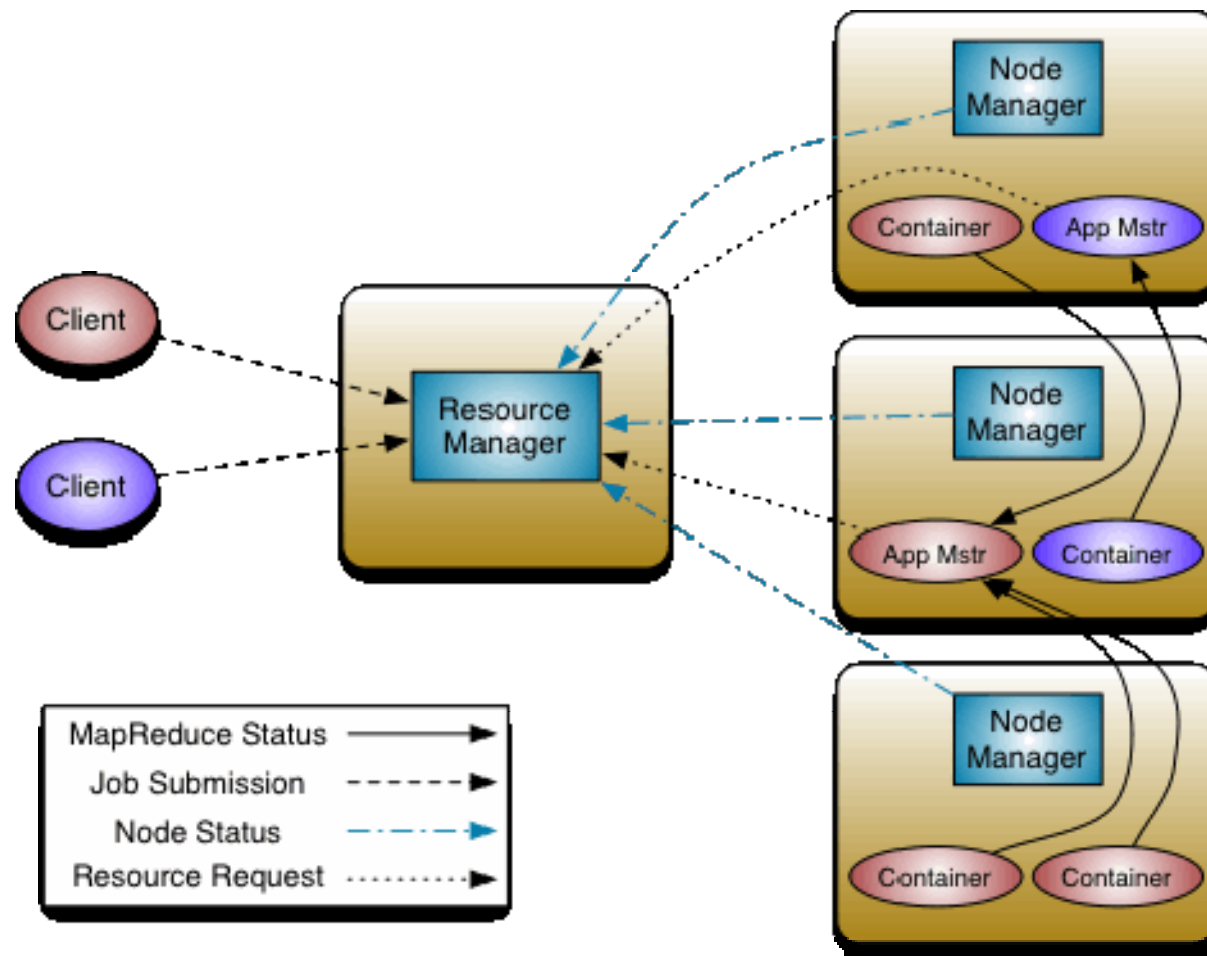
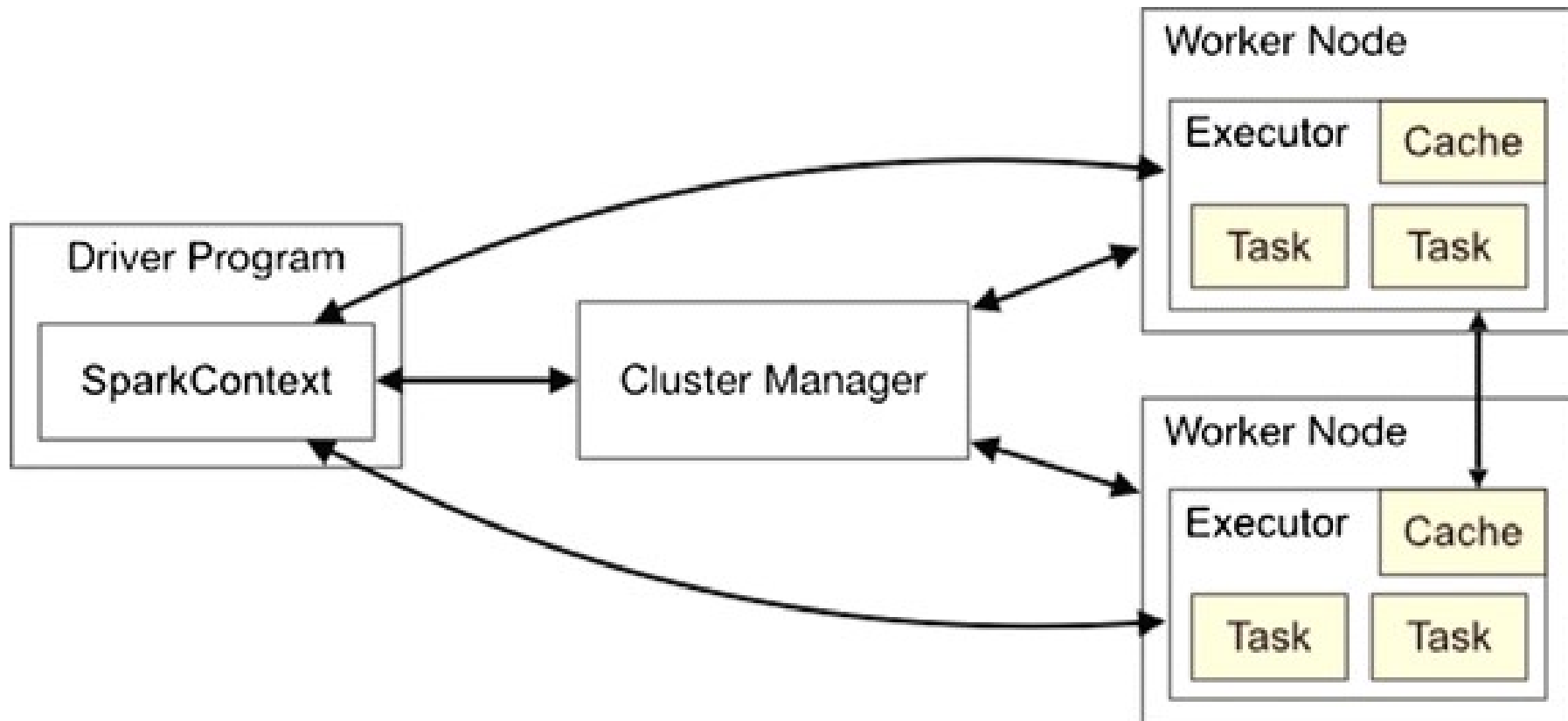


# An Introduction to Apache Spark

# Spark Architecture

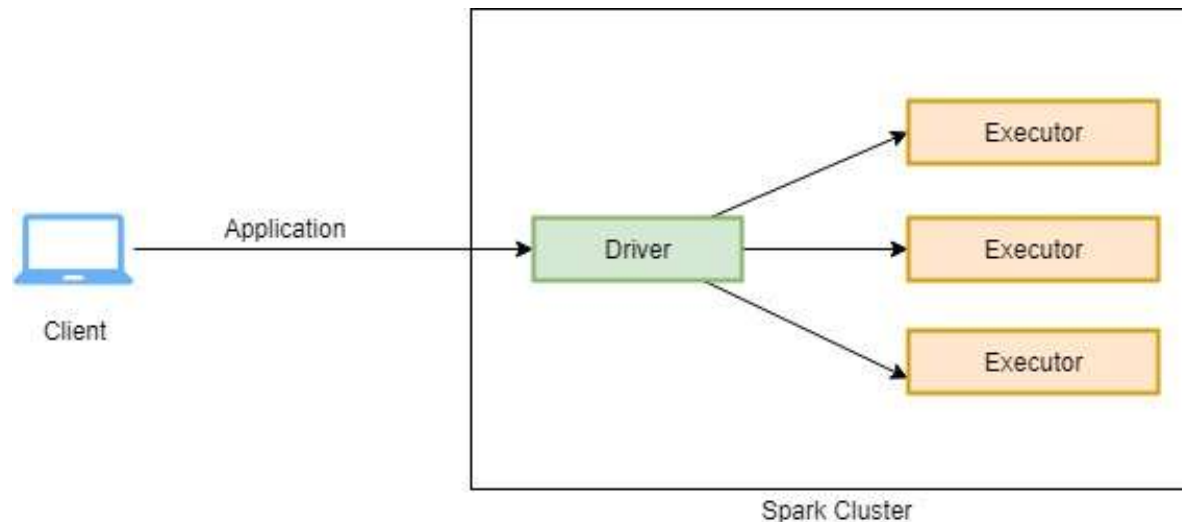


# Spark Architecture

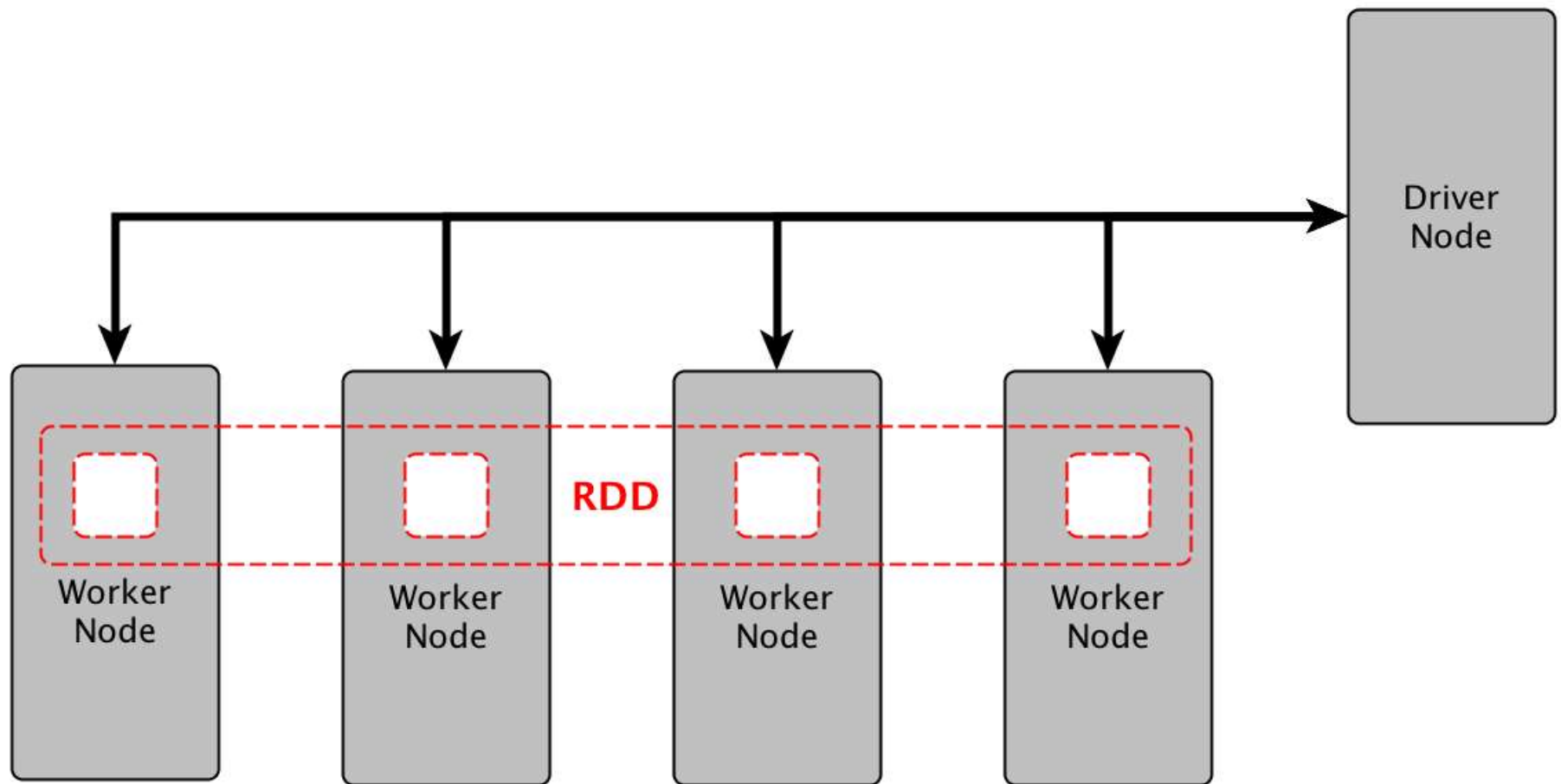


# Apache Spark Execution

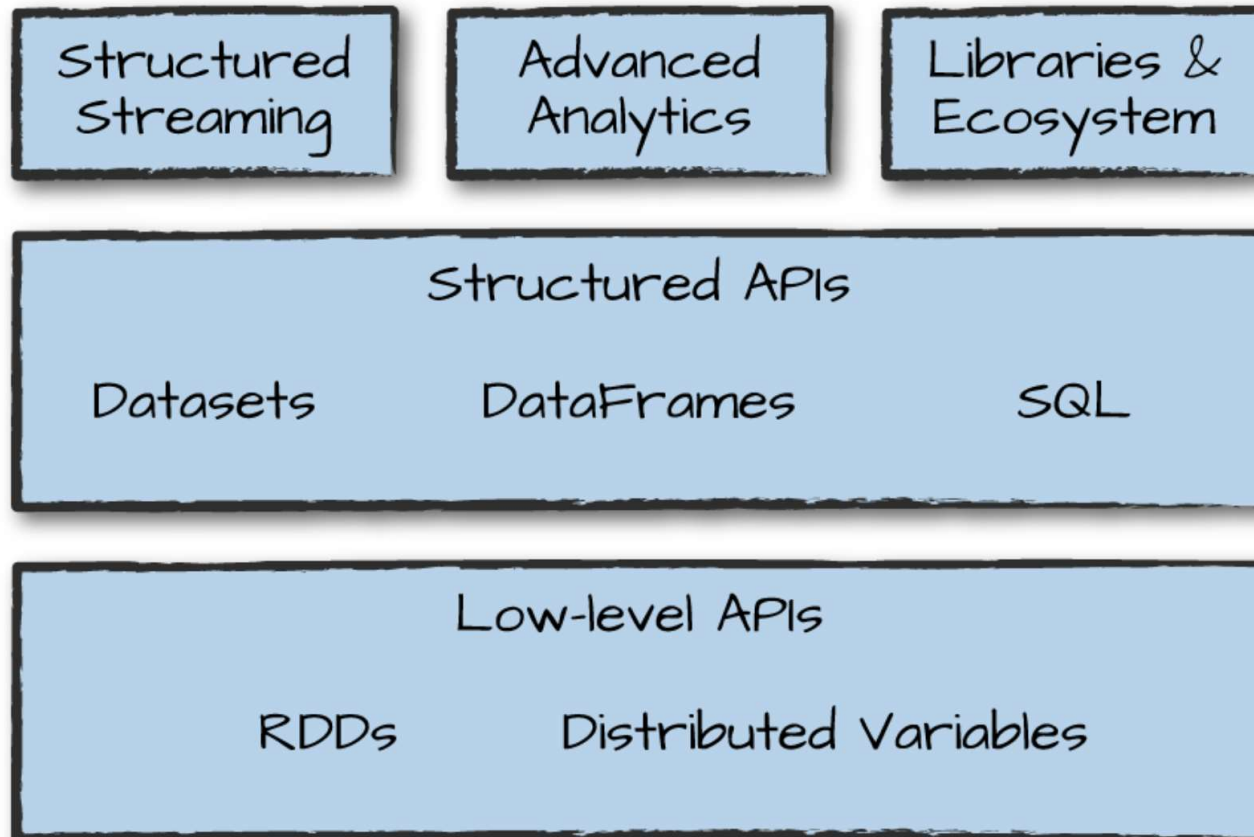
- For every application submitted on spark cluster, spark creates a dedicated Driver process and bunch of Executor processes.
- Driver process is responsible for analyzing, distributing, scheduling and monitoring of executor processes.
- Whereas the executor process is only responsible for running the task they were assigned by drivers and reporting the status back to the driver.



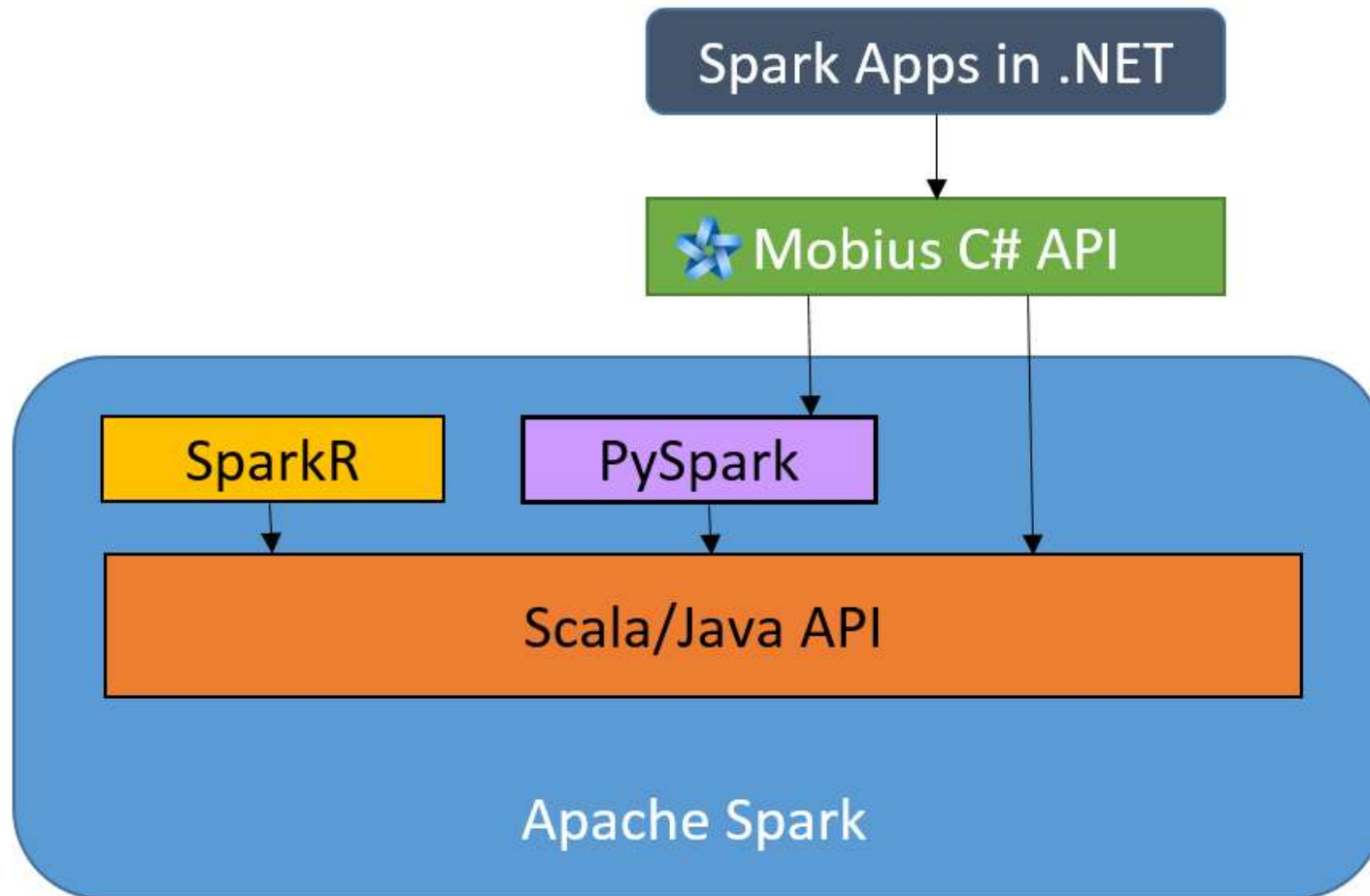
# Apache Spark Execution



# Spark's Language APIs



# Spark's Language APIs

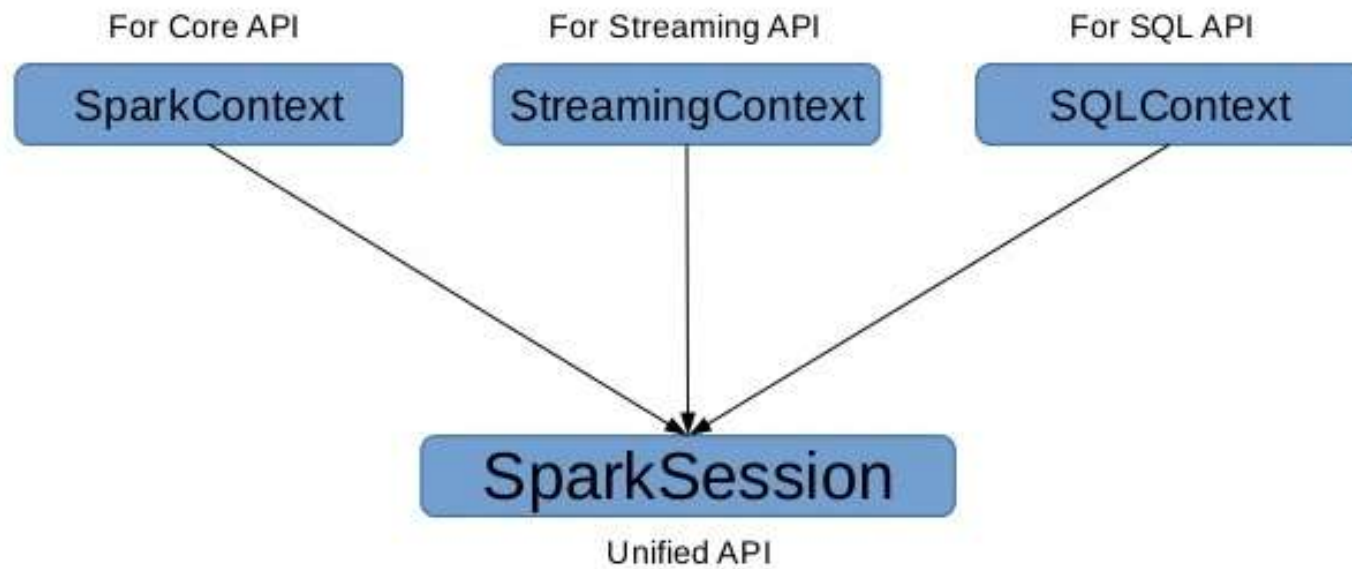


# SparkSession

- Before understanding spark-session let's understand the entry-point
- An entry-point is where control is transferred from the operating system to the provided program.
- Before 2.0 entry-point to spark-core was the sparkContext.
- Apache Spark is a powerful cluster computing engine, therefore it is designed for fast computation of big data.
- In previous versions, spark context is the entry point for spark
- For streaming we needed streamingContext.
- For SQL sqlContext and for hive hiveContext.
- So in spark 2.0, we have a new unified entry point.



# The SparkSession



# DataFrames

- In Apache Spark, a DataFrame is a distributed collection of rows
- It has below characteristics:
  - Immutable in nature
    - We can create DataFrame RDD once but can't change it.
  - Lazy Evaluations
    - Which means that a task is not executed until an action is performed.
  - Distributed

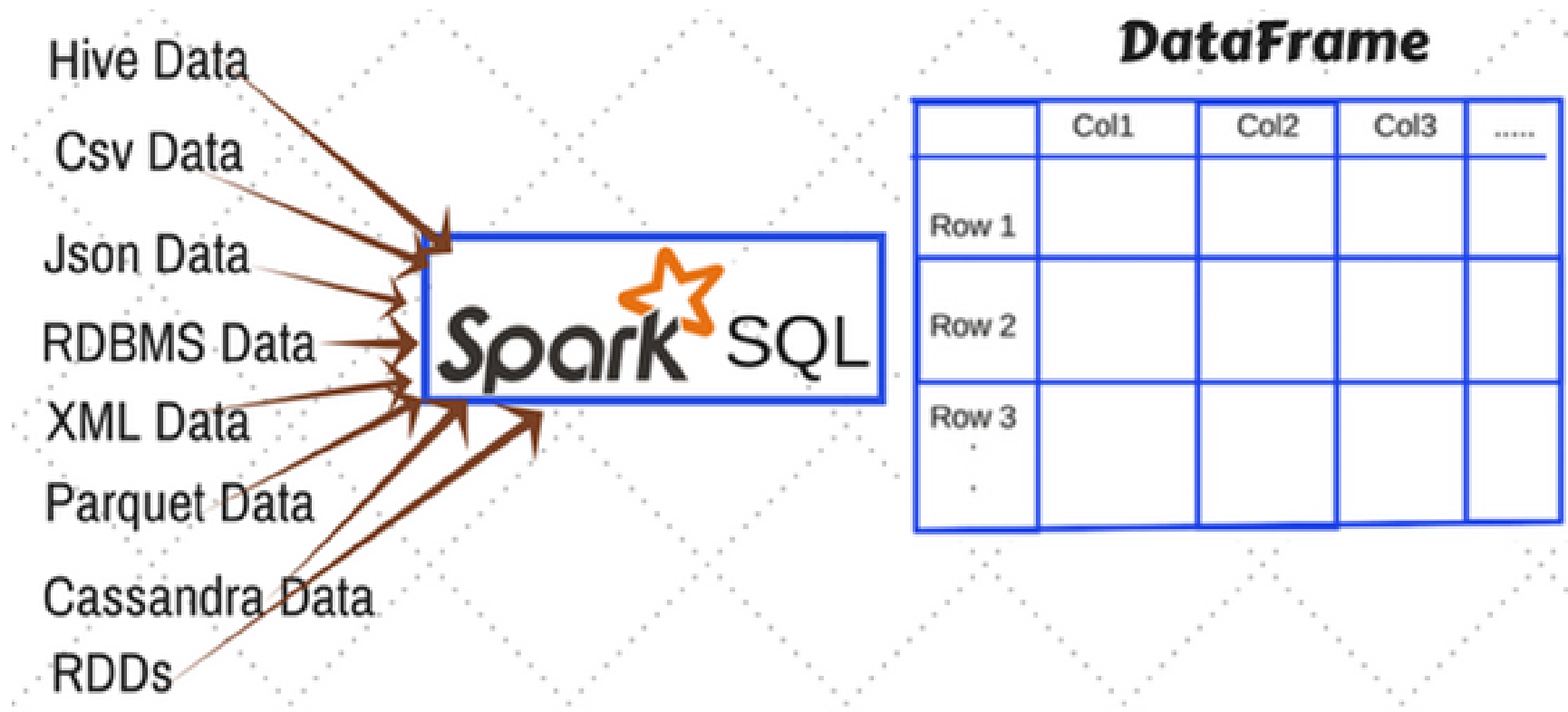
# Why DataFrames are Useful?

- Designed for processing large collection of data.
- Has the ability to handle petabytes of data.
- Has API support for different languages like
  - Python,
  - R,
  - Scala,
  - Java.

# Create a DataFrame

- Can be created using different data formats:
  - JSON
  - CSV
  - XML
  - Excel
- By loading data from Existing RDD
- By Programmatically specifying schema

# Ways to create DataFrame in Spark



# Creating DataFrame from RDD

- `from pyspark.sql import Row`
- `l = [('Ankit',25),('Jalfaizy',22),('saurabh',20),('Bala',26)]`
- `rdd = sc.parallelize(l)`
- `people = rdd.map(lambda x: Row(name=x[0], age=int(x[1])))`
- `schemaPeople = sqlContext.createDataFrame(people)`
- `type(schemaPeople)`
- `#Output:`
- `#pyspark.sql.dataframe.DataFrame`

# Thanks