DATABRICKS

Write Once, Run Anywhere

Pat McDonough

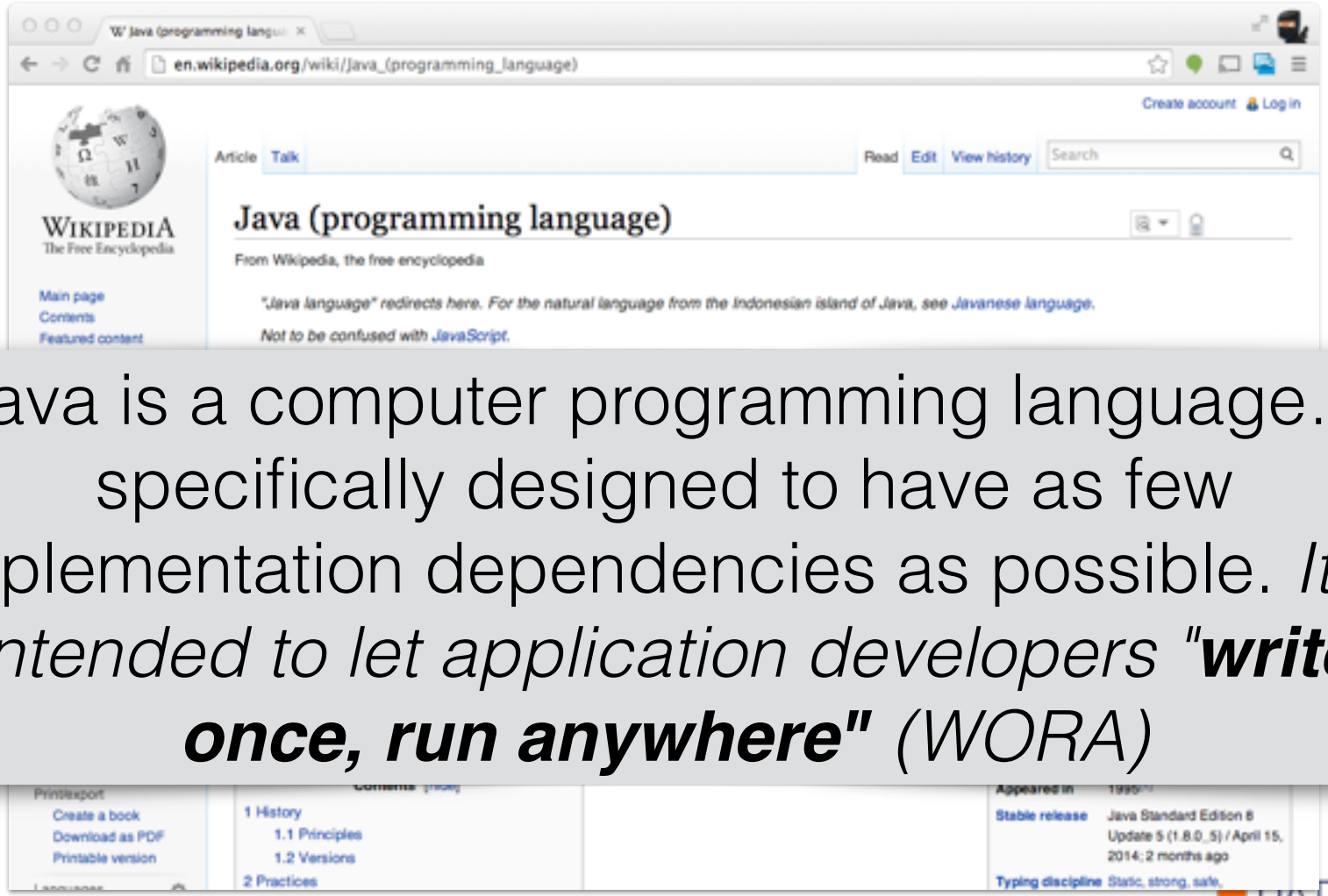
Write Once, Run Anywhere…

Write Once, Run Anywhere…

You Might Have Heard This Before!

Java, According to Wikipedia

Java, According to Wikipedia

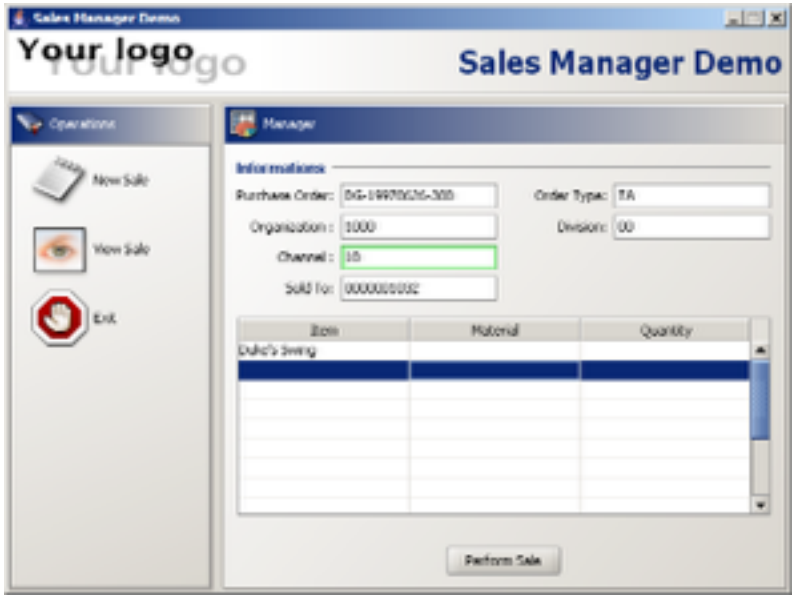


Java is a computer programming language… specifically designed to have as few implementation dependencies as possible. *It is intended to let application developers "**write once, run anywhere**" (WORA)*

Java & “WORA” in the First Decade

Java Client Applications

- Apps with GUIs (AWT or Swing) could be deployed to any OS with a JVM



DATABRICKS

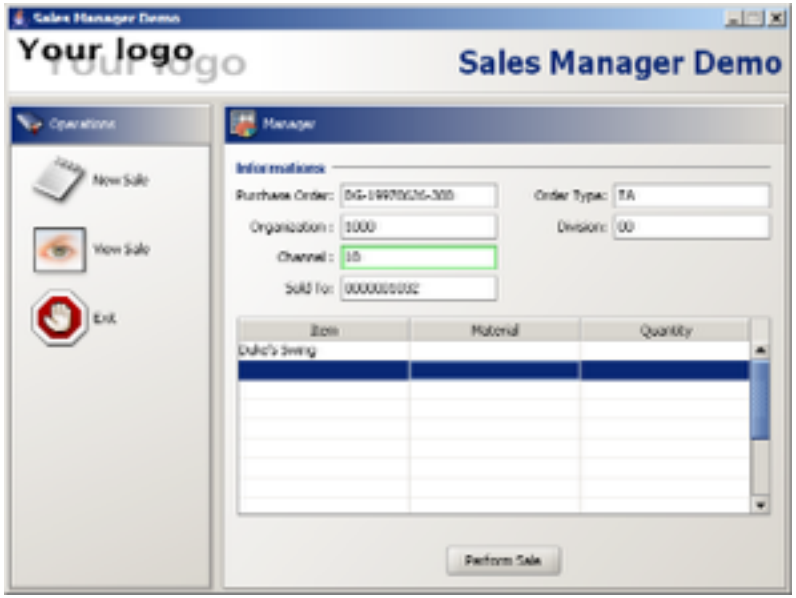
Java & “WORA” in the First Decade

Java Client Applications

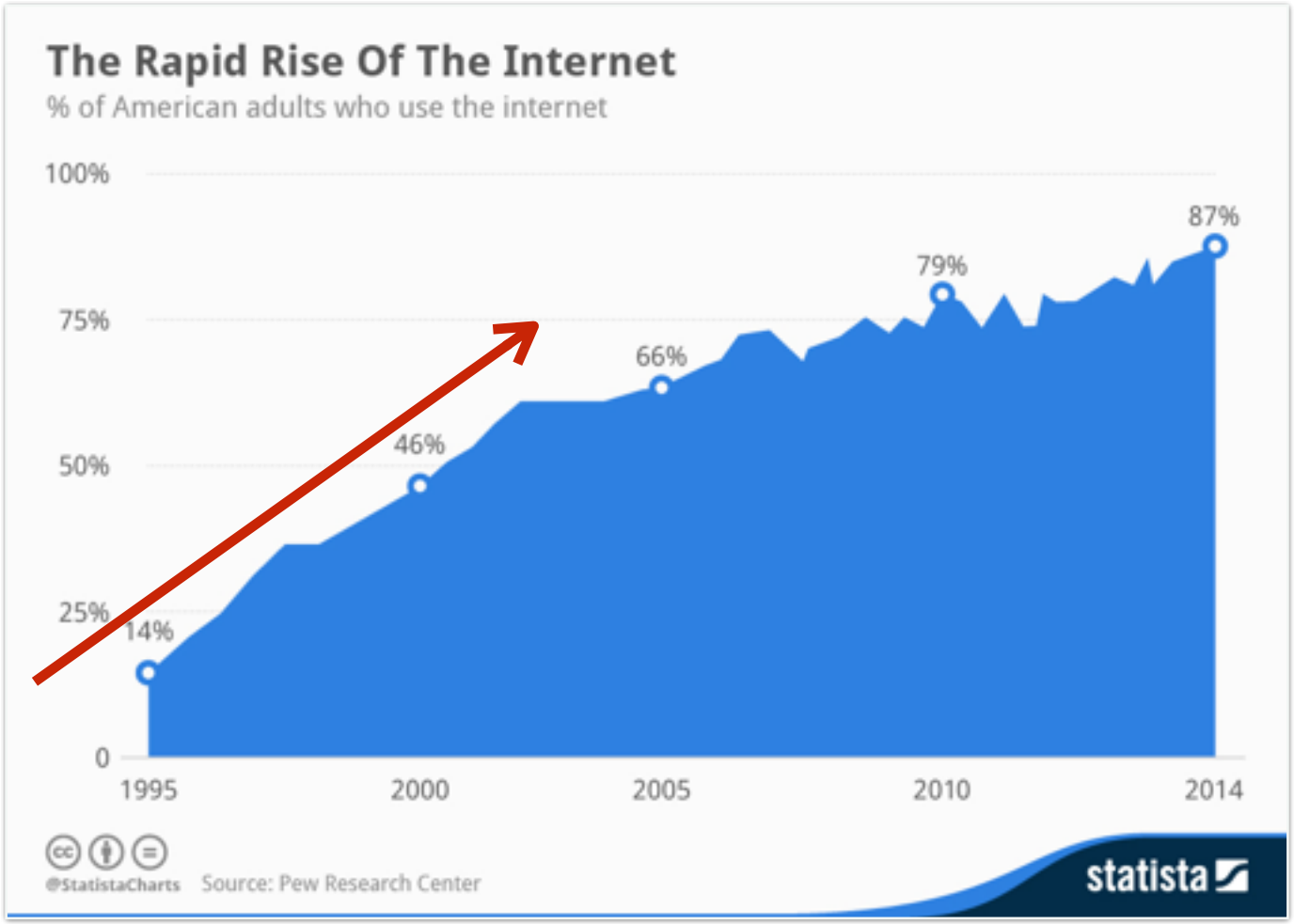
- Apps with GUIs (AWT or Swing) could be deployed to any OS with a JVM

Neat!

- … but not all that useful - people don’t want non-native GUIs



Java & “WORA” in the First Decade



Java & “WORA” in the First Decade

Applets

- A way to deliver “rich” GUIs to many different platforms through the browser

[Insert Ugly Applet Here]

Java & “WORA” in the First Decade

Applets

- A way to deliver “rich” GUIs to many different platforms through the browser

Neat!

[Insert Ugly Applet Here]

Java & “WORA” in the First Decade

Applets

- A way to deliver “rich” GUIs to many different platforms through the browser

Neat!

- … but basically ended at producing many gimmicky website animations

[Insert Ugly Applet Here]

Java & “WORA” in the First Decade

Back-end Applications

- Windows Desktop for an IDE
- Unix Server for Production

Neat!

- … And actually useful too

Java & “WORA” in the First Decade

- Back-end Java starts to formalize around standards —> **J2EE**
  - Core libraries, deployment formats, etc.
- Vendors Offer J2EE App Servers
- Ironically, this immediately lead to no more WORA
  - Specific App Servers required a specific SDK
  - … or even a specific IDE

Java & “WORA” in the First Decade

- Fixing WORA on the back-end:
  - Fall back to the *Least Common Denominator* —> **Servlets** (usually via Tomcat)

- **Spring** comes about to dominate as the SDK of choice for Java back-end applications
  - *“…specifically designed to have as few implementation dependencies as possible”*

So yes, you've heard this before

So yes, you've heard this before

Which examples apply to the state of Big Data Ecosystem?

Important Changes Since Then

- ~~Vendor Standards~~ Open Source
- Data has overwhelmed us
- Distributed Systems Are The New Standard (specifically, Data Parallel systems)

Big Data Platforms Are Everywhere Now… But Where Are the ***Big Data Applications***?

"Big Data Applications" don't exist very far beyond connecting ODBC/JDBC or simple ETL integrations

Why?

- Too many disparate systems to piece together
- Complicated matrix of compile-time and runtime dependencies across distributions
    - i.e. each distribution effectively has it's own SDK

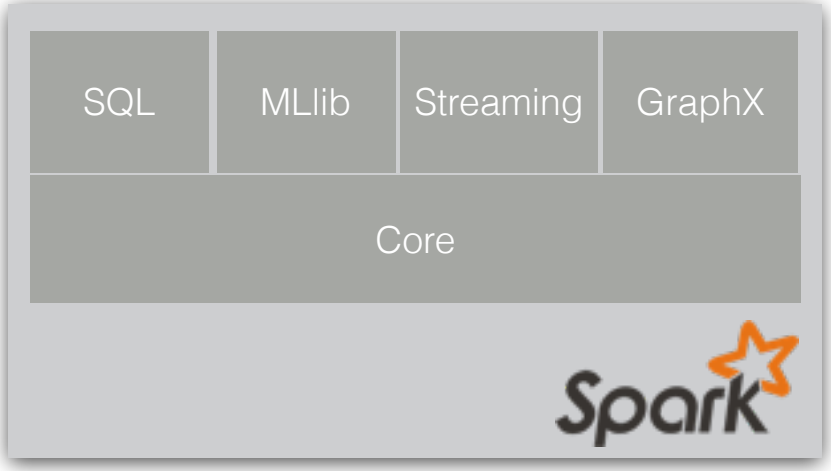
The Big Data Ecosystem Needs a Common SDK

The Big Data Ecosystem Needs a Common SDK

Apache Spark is the answer

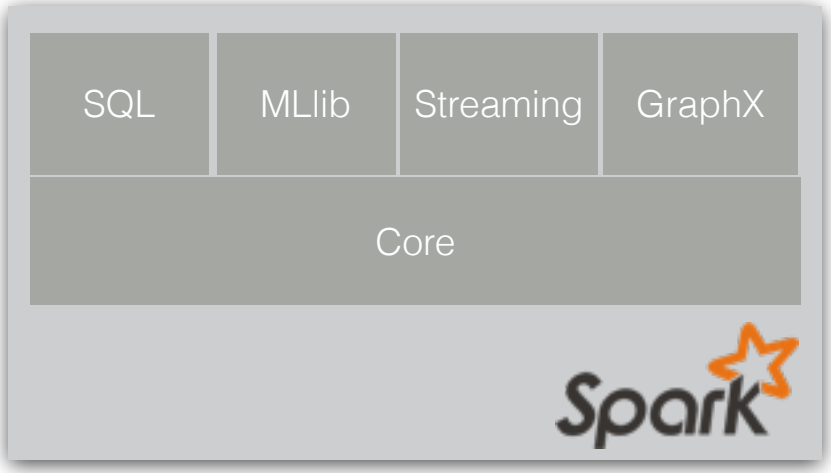
Spark

An SDK for Big Data Applications



Spark

An SDK for Big Data Applications

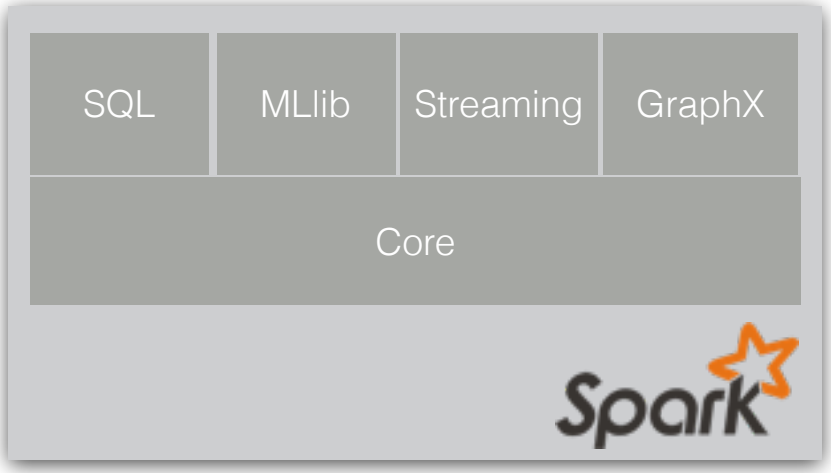


Unified System With Libraries to Build a Complete Solution

Full-featured Programming Environment

Spark

An SDK for Big Data Applications



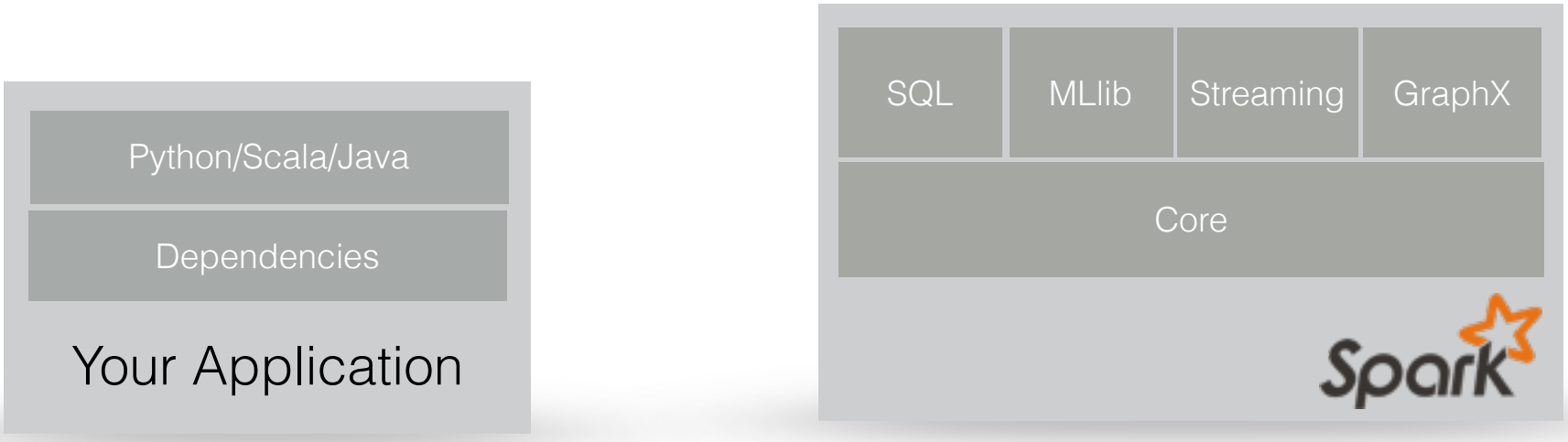
Unified System With Libraries to Build a Complete Solution

Full-featured Programming Environment

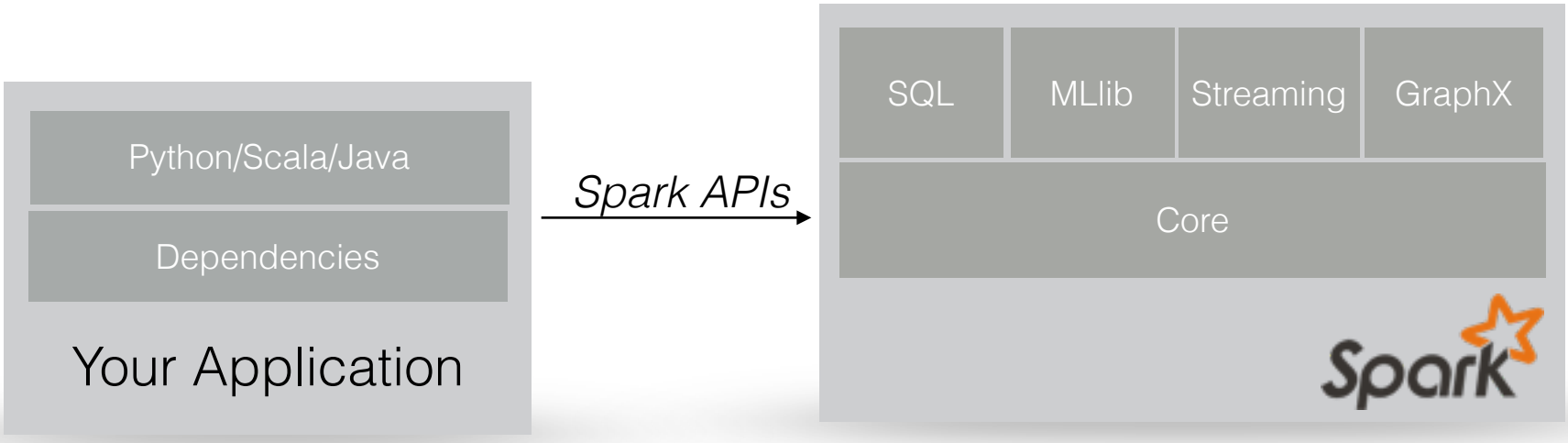
Single, Consistent Interface for Developers to Write Against

Runtimes available on several platforms

Develop Big Data Applications



Develop Big Data Applications



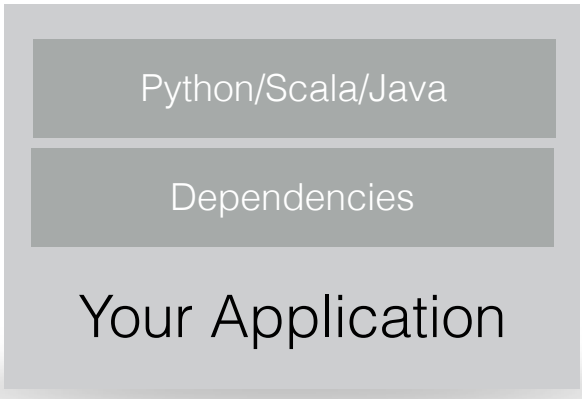
Develop Applications…

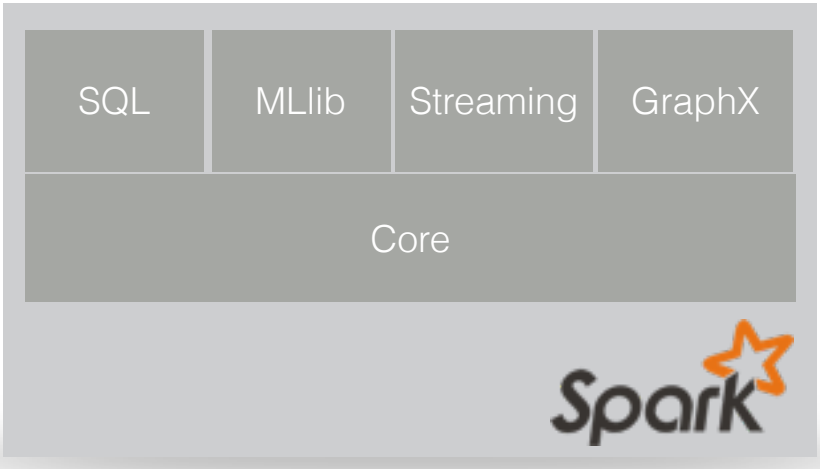
… using your preferred language,

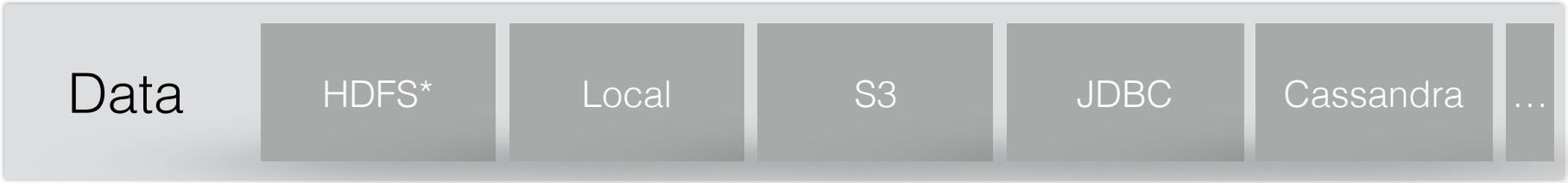
… using existing libraries,

… using Spark's Public APIs
(SparkContext, RDDs)

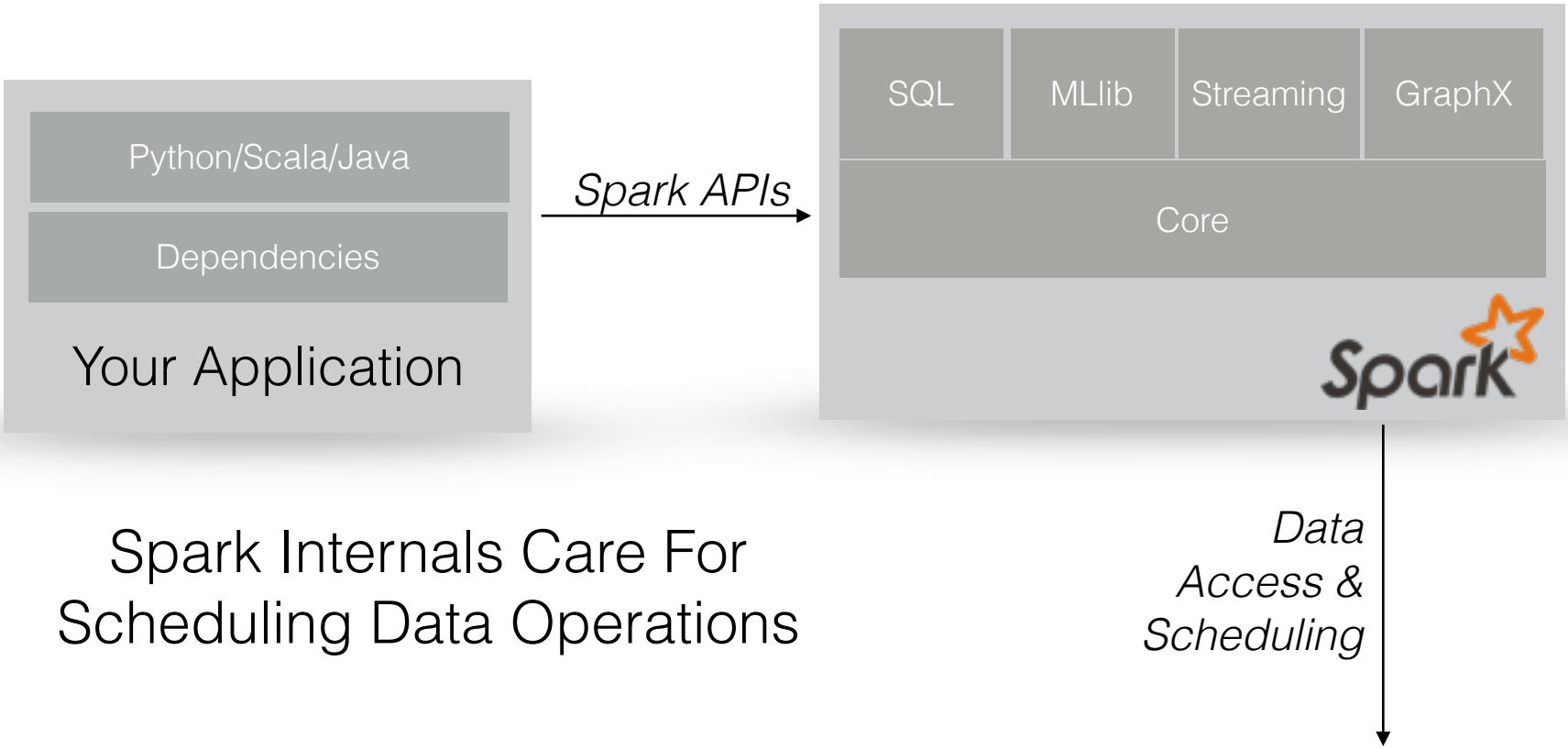
Work With Data



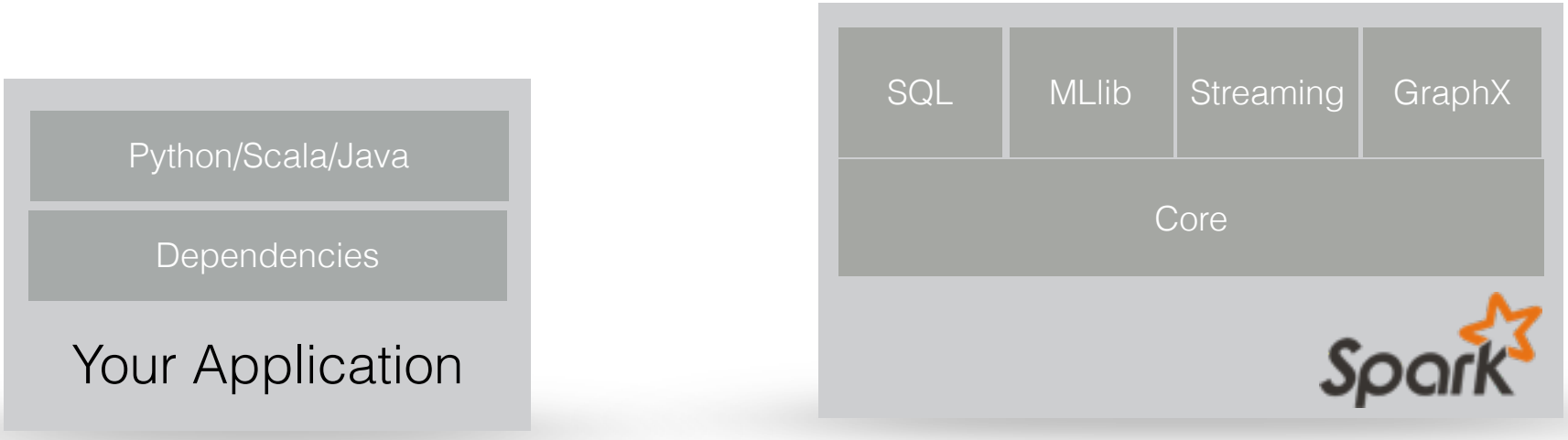




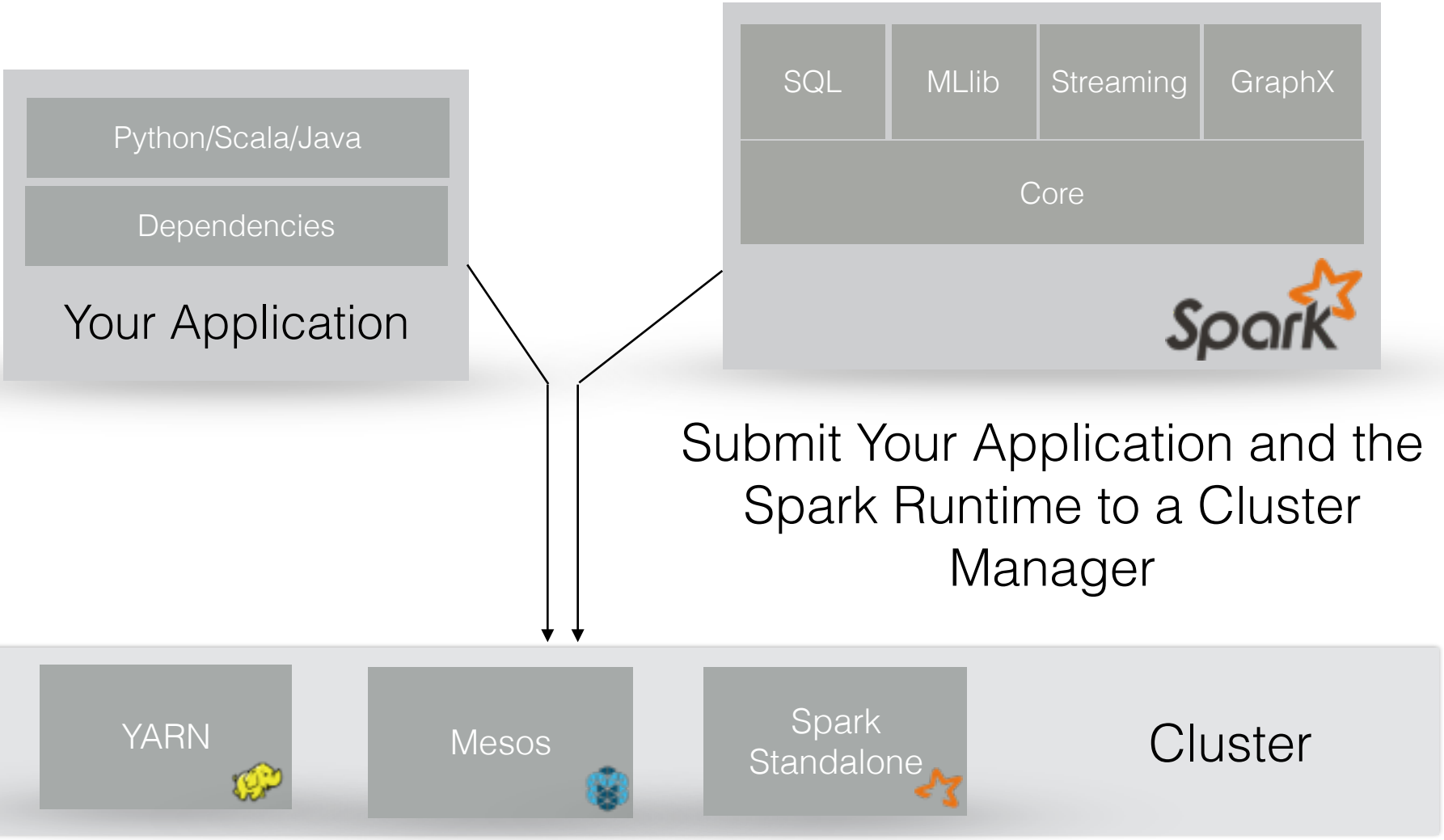
Work With Data



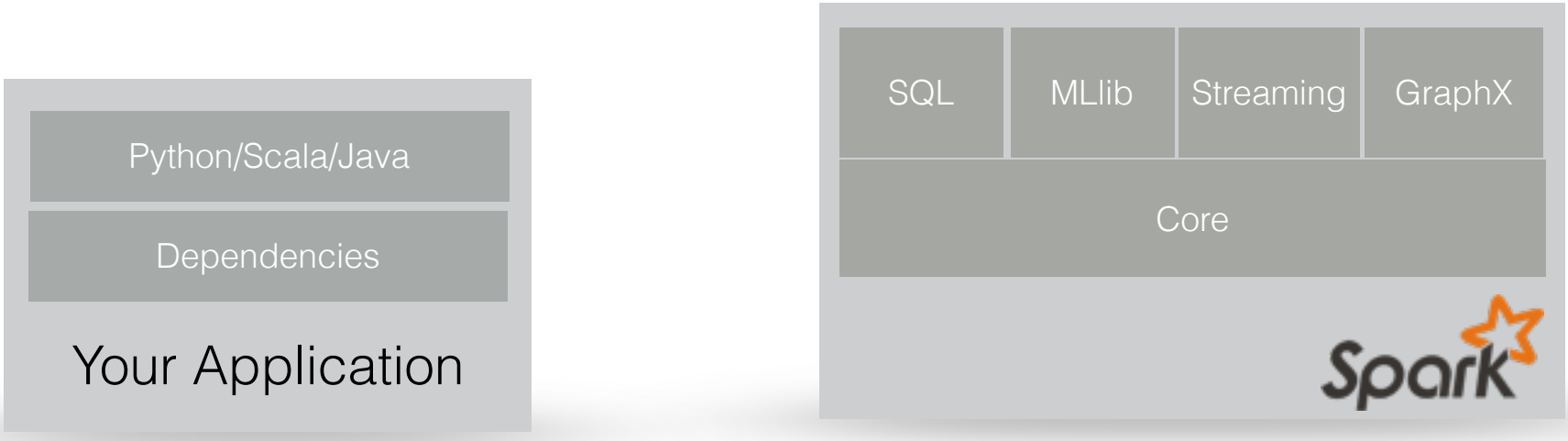
Run Your Applications



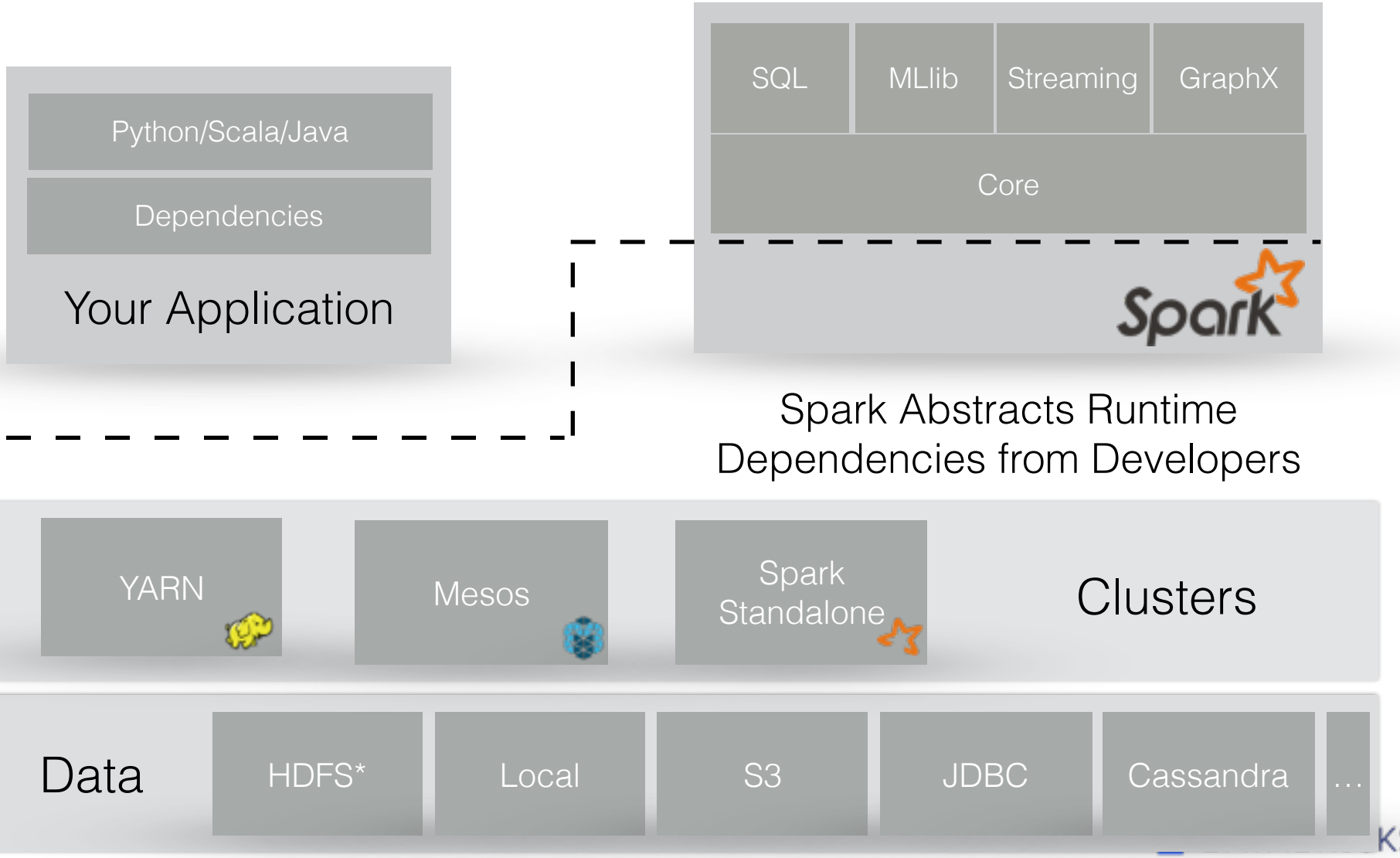
Run Your Applications



The Complete Picture

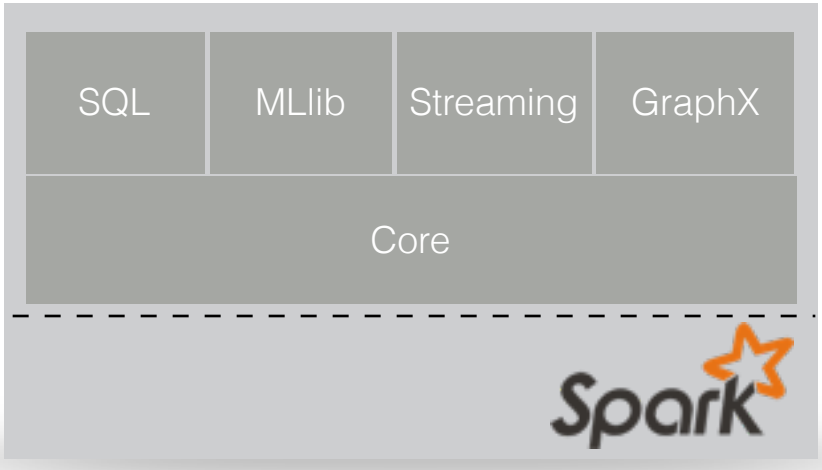


The Complete Picture



How Spark Handles Hadoop Dependencies

- The Spark library is compiled with compatibility to a specific Hadoop version
- At runtime, Spark uses reflection to find any Hadoop classes it needs



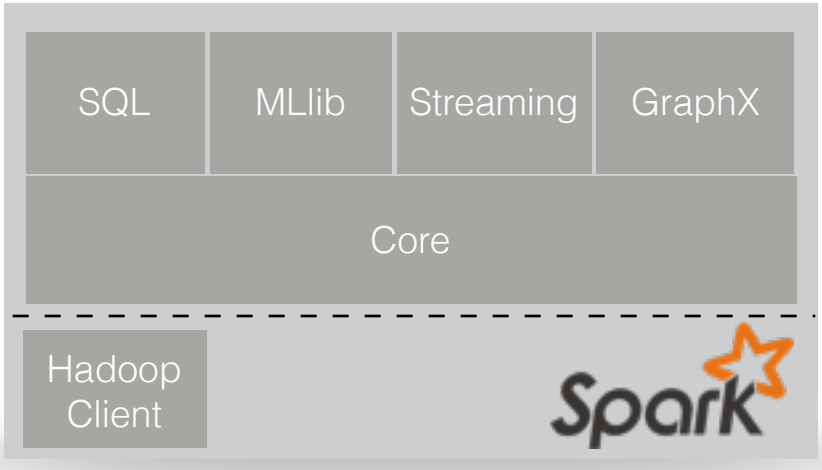
Examples:

```
# Apache Hadoop 2.2.X
mvn -Pyarn -Phadoop-2.2 \
-Dhadoop.version=2.2.0 \
-DskipTests clean package
```

```
# CDH 4.2.0 with MapReduce v1
mvn -Dhadoop.version=2.0.0-
mr1-cdh4.2.0 -DskipTests \
clean package
```

How Spark Handles Hadoop Dependencies

- The Spark library is compiled with compatibility to a specific Hadoop version
- At runtime, Spark uses reflection to find any Hadoop classes it needs



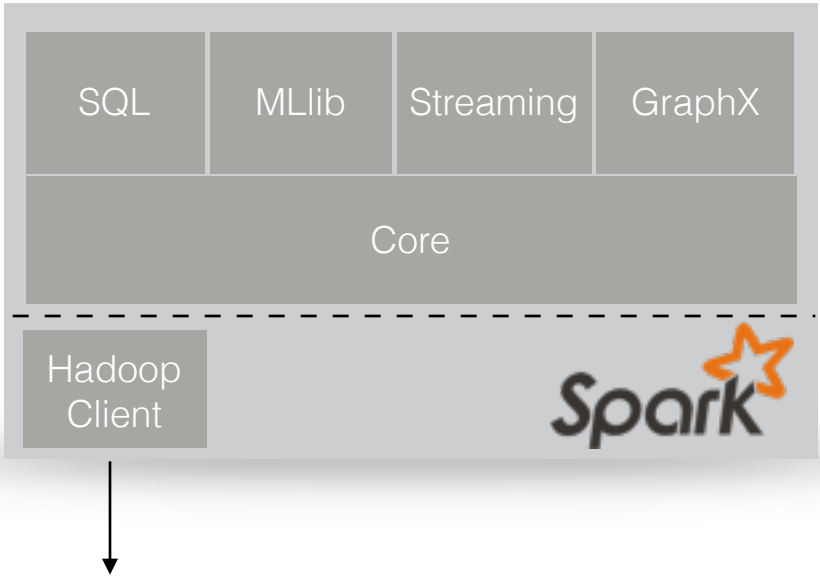
Examples:

```
# Apache Hadoop 2.2.X
mvn -Pyarn -Phadoop-2.2 \
-Dhadoop.version=2.2.0 \
-DskipTests clean package
```

```
# CDH 4.2.0 with MapReduce v1
mvn -Dhadoop.version=2.0.0-
mr1-cdh4.2.0 -DskipTests \
clean package
```

Spark Support Included on Big Data Platforms

- While this build process is very easy, it's even easier to have the runtime pre-built…
- Platform support also indicates stronger integration testing, supported, and integrated management tools



cloudera MAPR DATASTAX

Hortonworks IBM Pivotal ORACLE

DATABRICKS

Spark 1.0

Spark-Submit

- Spark-submit provides a consistent manner to launch jobs regardless of which platform
- Includes an important clean-up to make configurations more consistent

```
# Run on a Spark standalone cluster
./bin/spark-submit \
  --class org.apache.spark.examples.SparkPi \
  --master spark://207.184.161.138:7077 \
  --executor-memory 20G \
  --total-executor-cores 100 \
  /path/to/examples.jar \
  1000

# Run on a YARN cluster
export HADOOP_CONF_DIR=XXX
./bin/spark-submit \
  --class org.apache.spark.examples.SparkPi \
  --master yarn-cluster \
  --executor-memory 20G \
  --num-executors 50 \
  /path/to/examples.jar \
  1000
```

Spark SQL

- We actually wrestled with the name a bit because it's not only about SQL
- SparkSQL introduces SchemaRDDs and an Optimizer (Catalyst)
- SQL is actually not the only developer interface - there is also a DSL
- This provides a deeper integration for any structured data

```
val sqlContext = new org.apache.spark.sql.SQLContext(sc)
import sqlContext._
val people: RDD[Person] = ... // An RDD of case class objects…

// The following is the same as
// 'SELECT name FROM people WHERE age >= 10 AND age <= 19'
val teenagers = people.where('age >= 10).where('age <= 19).select('name)
```

Databricks Is Committed to Growing Apache Spark's Developer Ecosystem

- Developer Training, Online Materials, Free Resources
- Strong Commitment to Open Source
- Certification Programs

We're Hiring!

http://databricks.com/about-us

- Evangelists
- Trainers
- Solutions Architects
- Software Engineers