SIGMOID

# Reactive Dashboards Using Apache Spark

Rahul Kumar

Software Developer

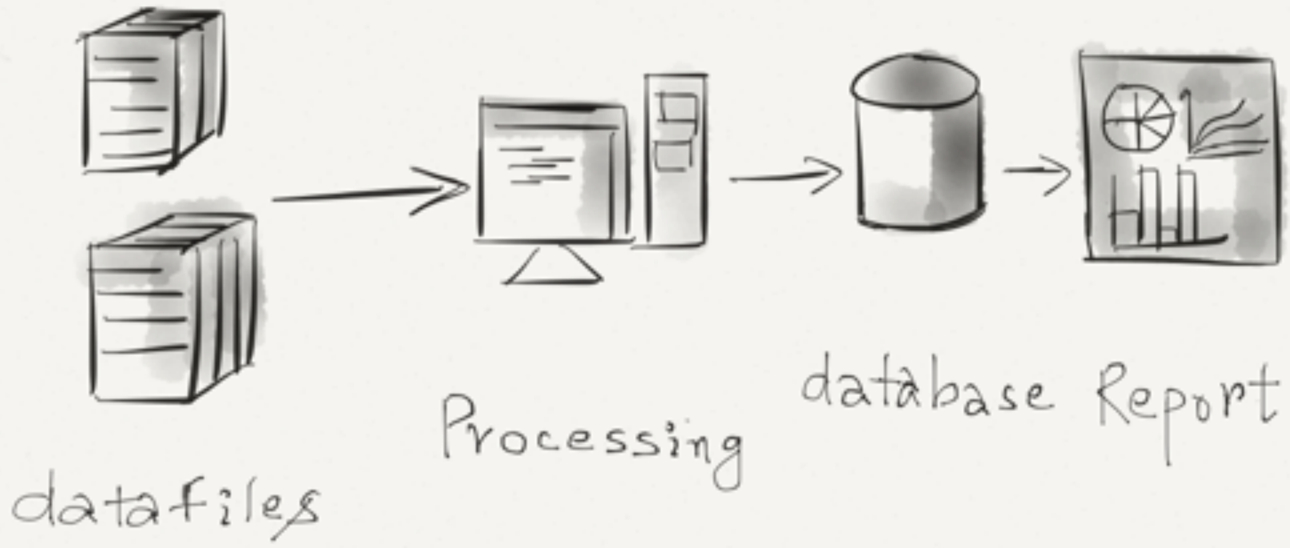
@rahul_kumar_aws

**LinuxCon, CloudOpen, ContainerCon North America 2015**

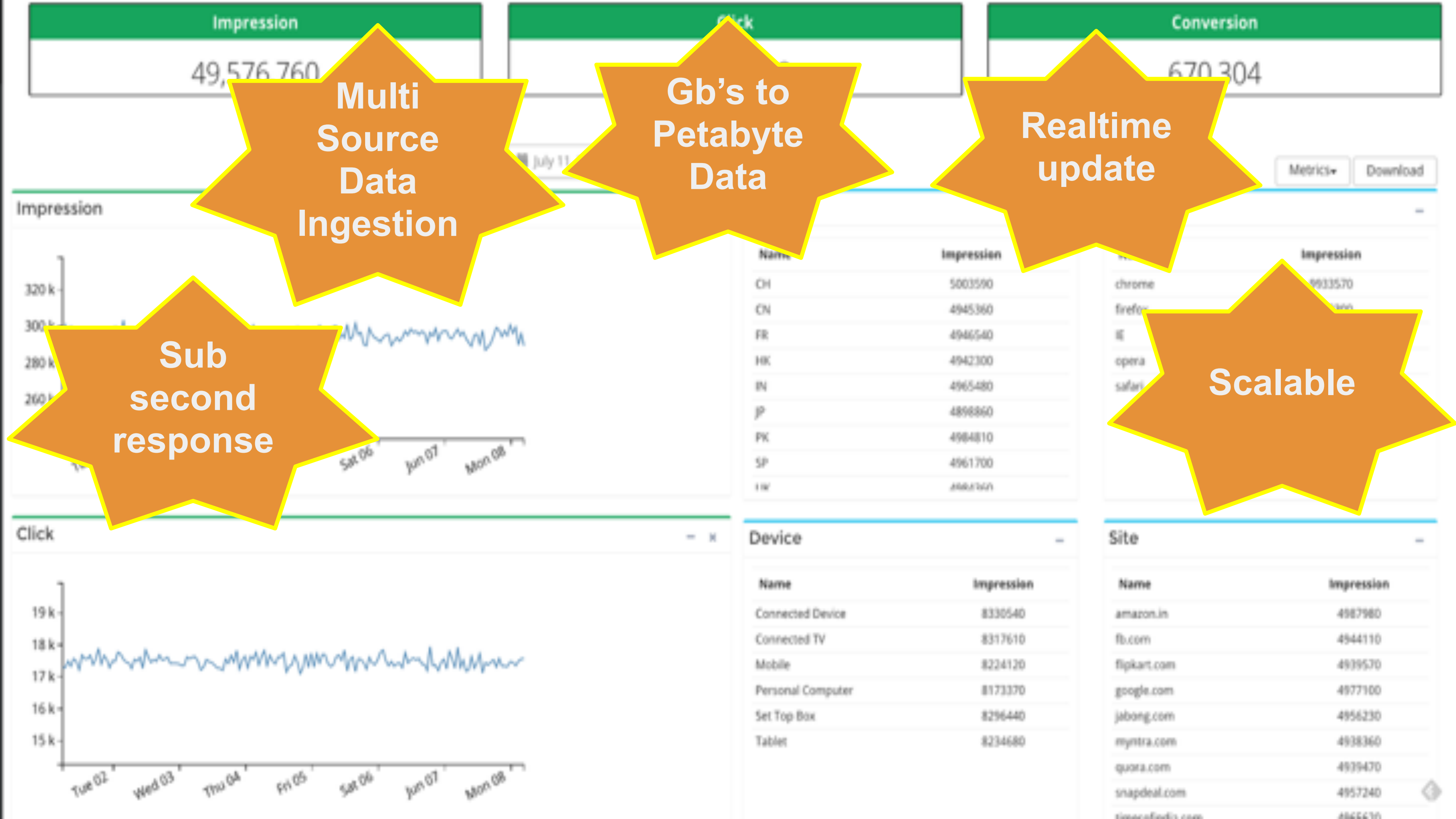
## Agenda

SIGMOID

- Big Data Introduction
- Apache Spark
- Introduction to Reactive Applications
- Reactive Platform
- Live Demo



## A typical database application



Multi Source Data Ingestion

Gb's to Petabyte Data

Realtime update

Sub second response

Scalable

| Name | Impression |
|---|---|
| CH | 5003590 |
| CN | 4945360 |
| FR | 4946540 |
| HK | 4942300 |
| IN | 4965480 |
| JP | 4898860 |
| PK | 4984810 |
| SP | 4961700 |
| UK | [illegible] |

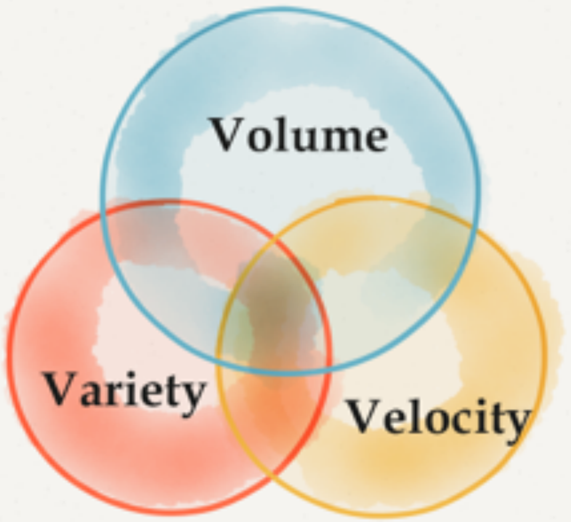
| | Impression |
|---|---|
| chrome | 4933570 |
| firefox | [illegible] |
| IE | |
| opera | |
| safari | |

Device

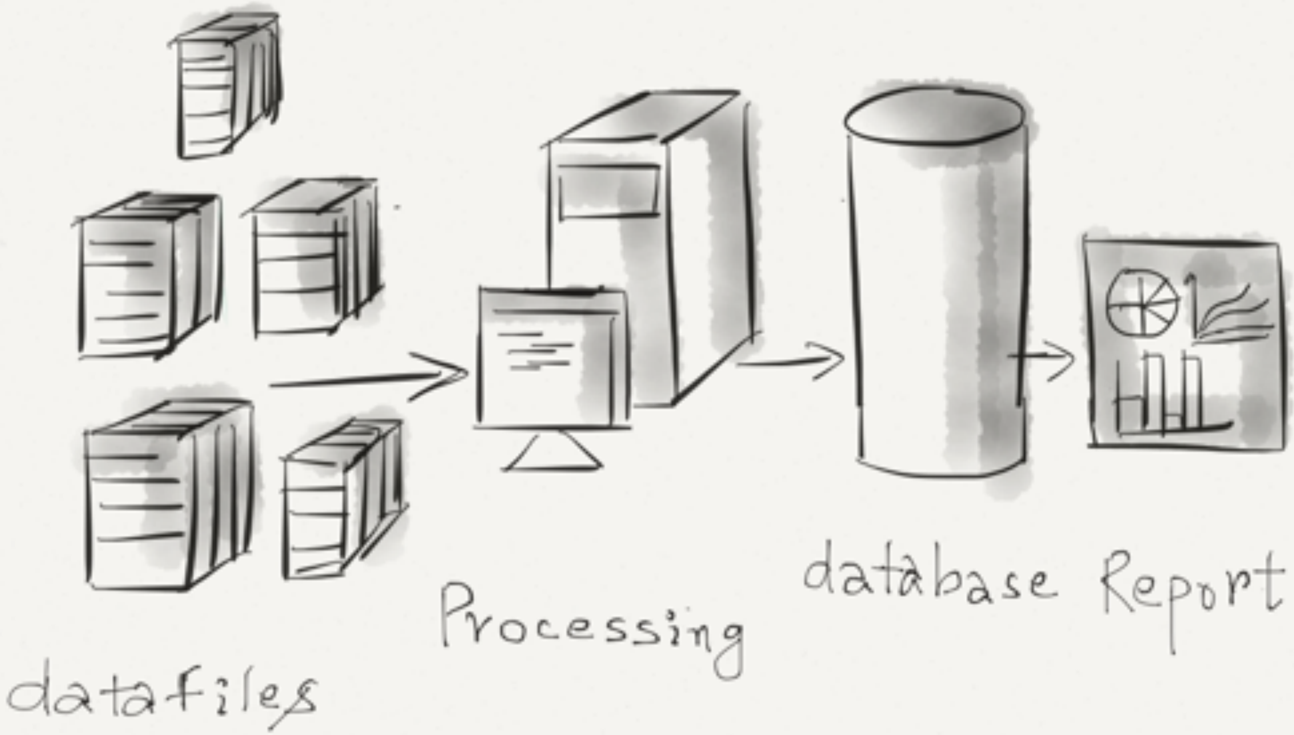
| Name | Impression |
|---|---|
| Connected Device | 8330540 |
| Connected TV | 8317610 |
| Mobile | 8224120 |
| Personal Computer | 8173370 |
| Set Top Box | 8296440 |
| Tablet | 8234680 |

Site

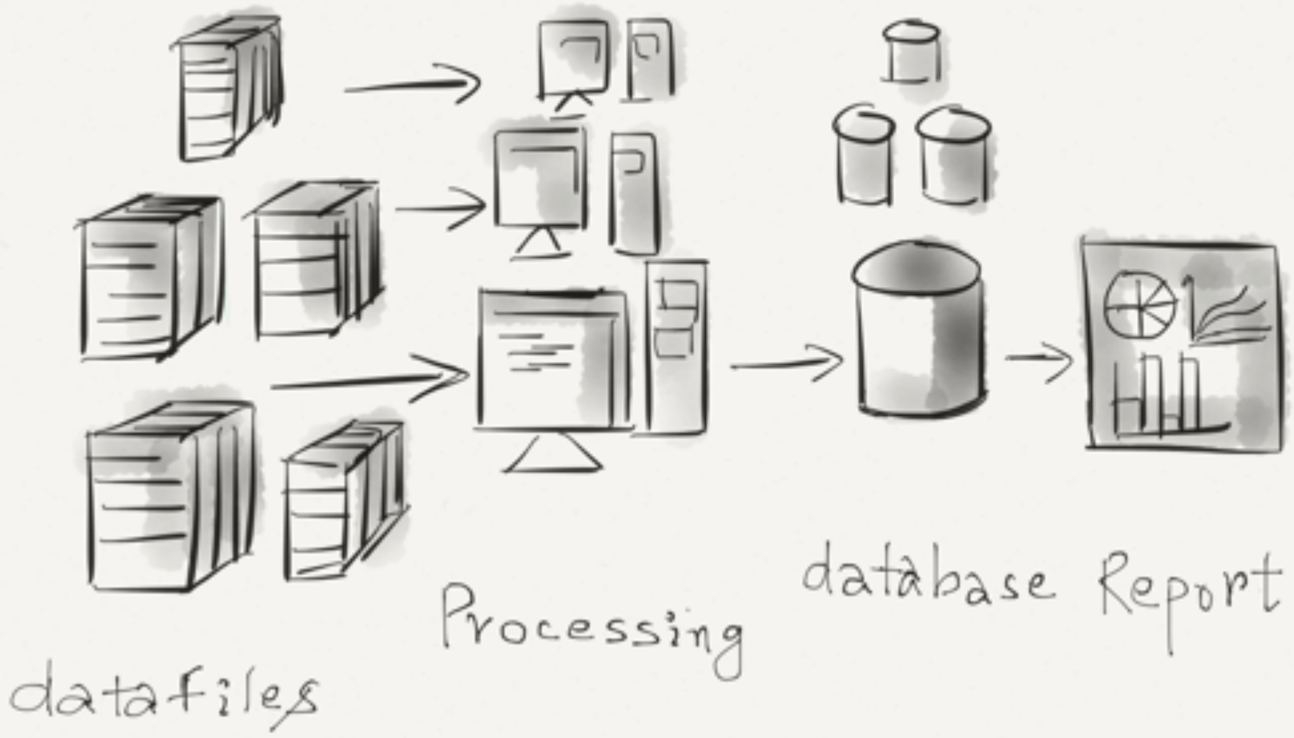
| Name | Impression |
|---|---|
| amazon.in | 4987980 |
| fb.com | 4944110 |
| flipkart.com | 4939570 |
| google.com | 4977100 |
| jabong.com | 4956230 |
| myntra.com | 4938360 |
| quora.com | 4939470 |
| snapdeal.com | 4957240 |
| timesofindia.com | [illegible] |



## Three V's of Big Data



Scale vertically (scale up)



## Scale horizontally (scale out)

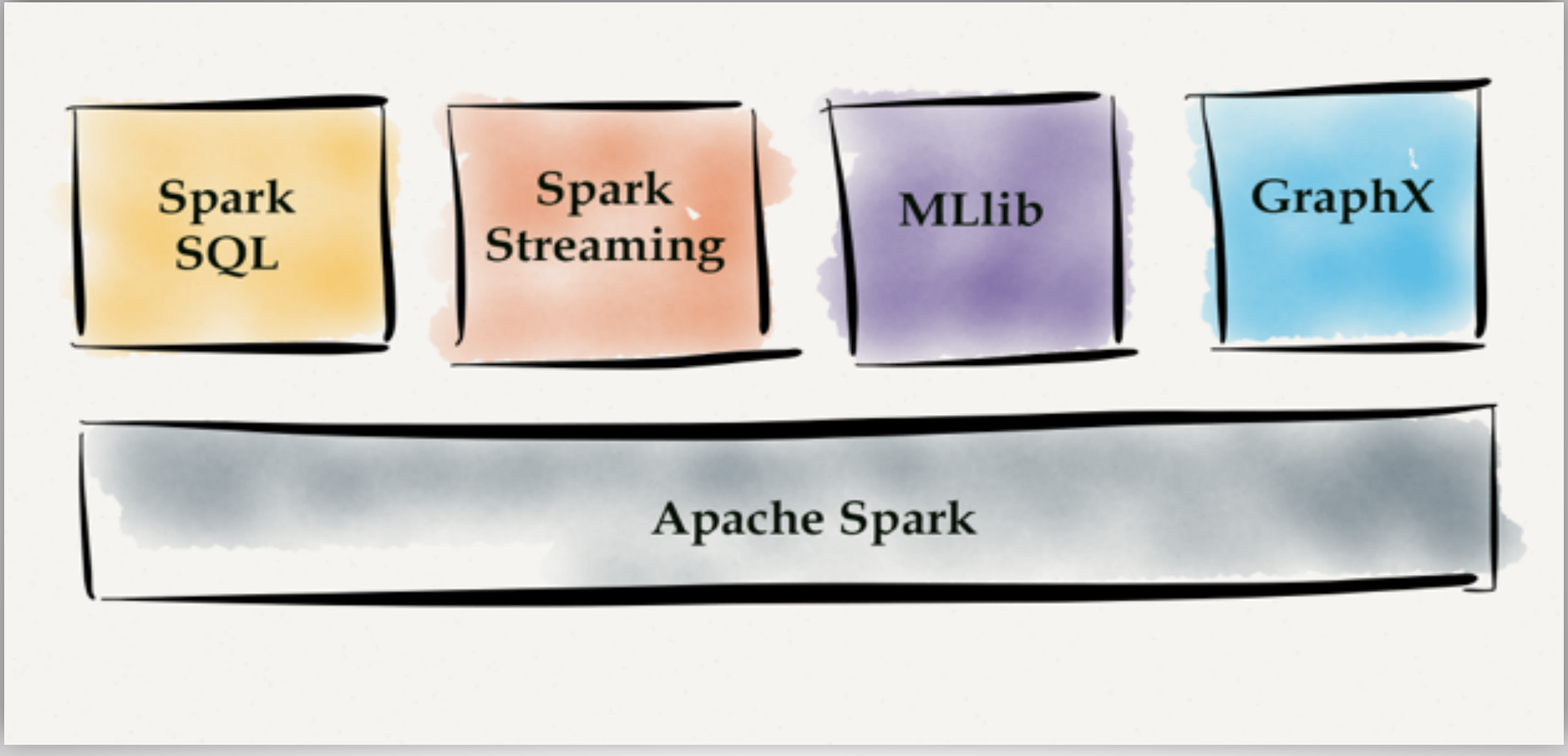
## Apache Spark

SIGMOID

**Apache Spark is a fast and general engine for large-scale data processing.**

- Speed
- Easy to Use
- Generality
- Runs Everywhere

# Apache Spark Stack



- Apache Spark Setup
- Interaction with Spark Shell
- Setup a Spark App
- RDD Introduction
- Deploy Spark app on Cluster

## Prerequisite for cluster setup

### Spark Cluster

**Spark runs on Java 6+, Python 2.6+ and R 3.1+.**

For the Scala API, Spark 1.4.1 uses Scala 2.10.

**Java 8**

```
sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java8-installer
```

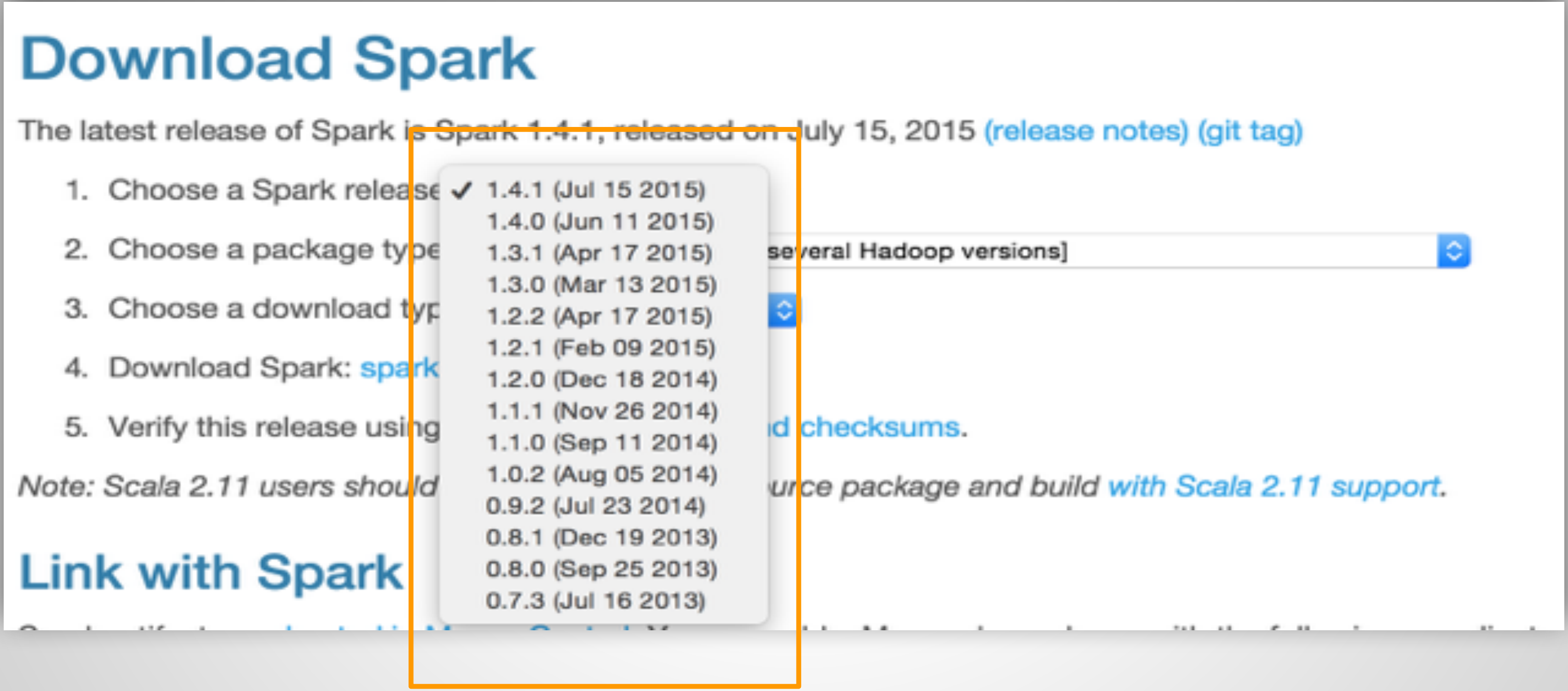
**Scala 1.10.4**

http://www.scala-lang.org/files/archive/scala-2.10.4.tgz

```
$tar -xvzf scala-2.10.4.tgz
vim ~/.bashrc
export SCALA_HOME=/home/ubuntu/scala-2.10.4
export PATH=$PATH:$SCALA_HOME/bin
```

## Spark Setup

http://spark.apache.org/downloads.html



# Download Spark

The latest release of Spark is Spark 1.4.1, released on July 15, 2015 (release notes) (git tag)

1. Choose a Spark release
2. Choose a package type
3. Choose a download typ
4. Download Spark: spark
5. Verify this release using

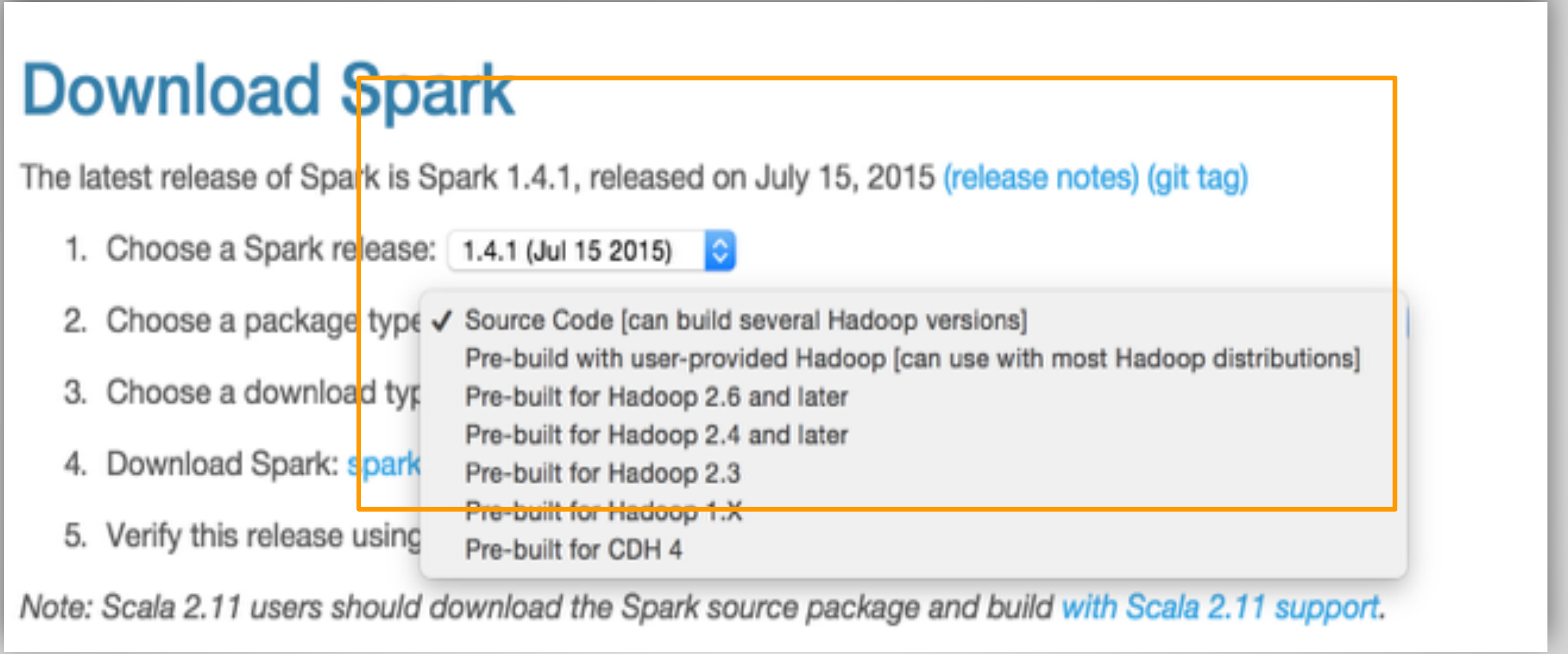
Note: Scala 2.11 users should

several Hadoop versions]

d checksums.

urce package and build with Scala 2.11 support.

## Link with Spark



# Download Spark

The latest release of Spark is Spark 1.4.1, released on July 15, 2015 (release notes) (git tag)

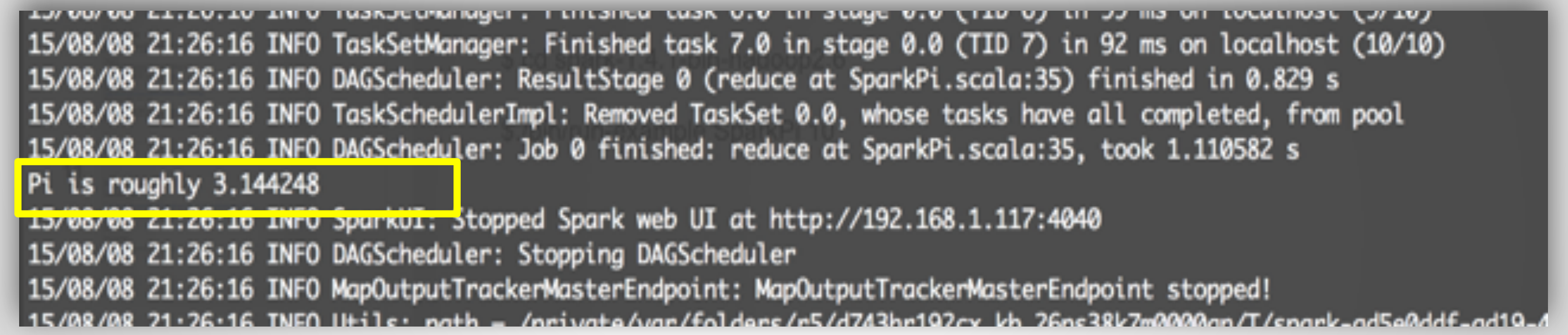
1. Choose a Spark release: 1.4.1 (Jul 15 2015)
2. Choose a package type
3. Choose a download typ
4. Download Spark: spark
5. Verify this release using

✓ Source Code [can build several Hadoop versions]
Pre-build with user-provided Hadoop [can use with most Hadoop distributions]
Pre-built for Hadoop 2.6 and later
Pre-built for Hadoop 2.4 and later
Pre-built for Hadoop 2.3
Pre-built for Hadoop 1.X
Pre-built for CDH 4

*Note: Scala 2.11 users should download the Spark source package and build with Scala 2.11 support.*

# Running Spark Example & Shell

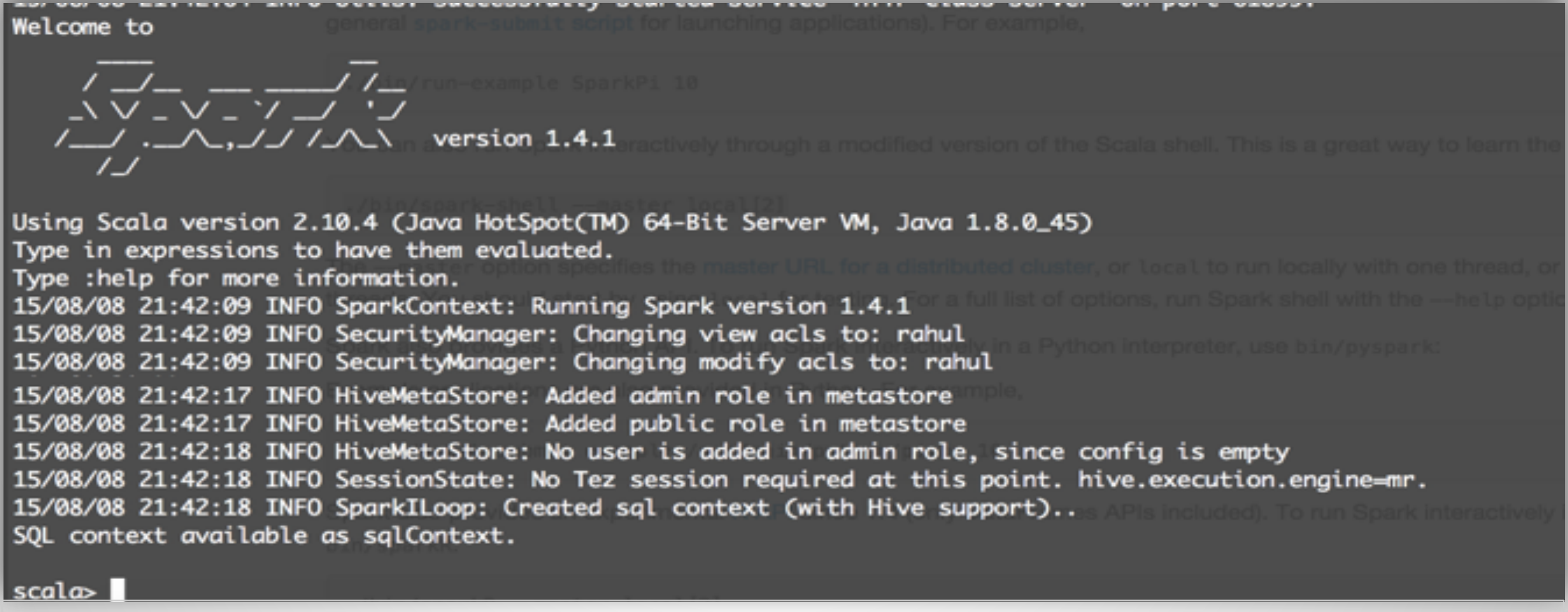
```
$ cd spark-1.4.1-bin-hadoop2.6

$./bin/run-example SparkPi 10
```



```
15/08/08 21:26:16 INFO TaskSetManager: Finished task 7.0 in stage 0.0 (TID 7) in 92 ms on localhost (10/10)
15/08/08 21:26:16 INFO DAGScheduler: ResultStage 0 (reduce at SparkPi.scala:35) finished in 0.829 s
15/08/08 21:26:16 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
15/08/08 21:26:16 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.scala:35, took 1.110582 s
Pi is roughly 3.144248
15/08/08 21:26:16 INFO SparkUI: Stopped Spark web UI at http://192.168.1.117:4040
15/08/08 21:26:16 INFO DAGScheduler: Stopping DAGScheduler
15/08/08 21:26:16 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
15/08/08 21:26:16 INFO Utils: path = /private/var/folders/r5/d743br192cx_kb_26ns38k7m0000gn/T/spark-ad5e0ddf-ad19-4
```

***cd spark-1.4.1-bin-hadoop2.6***

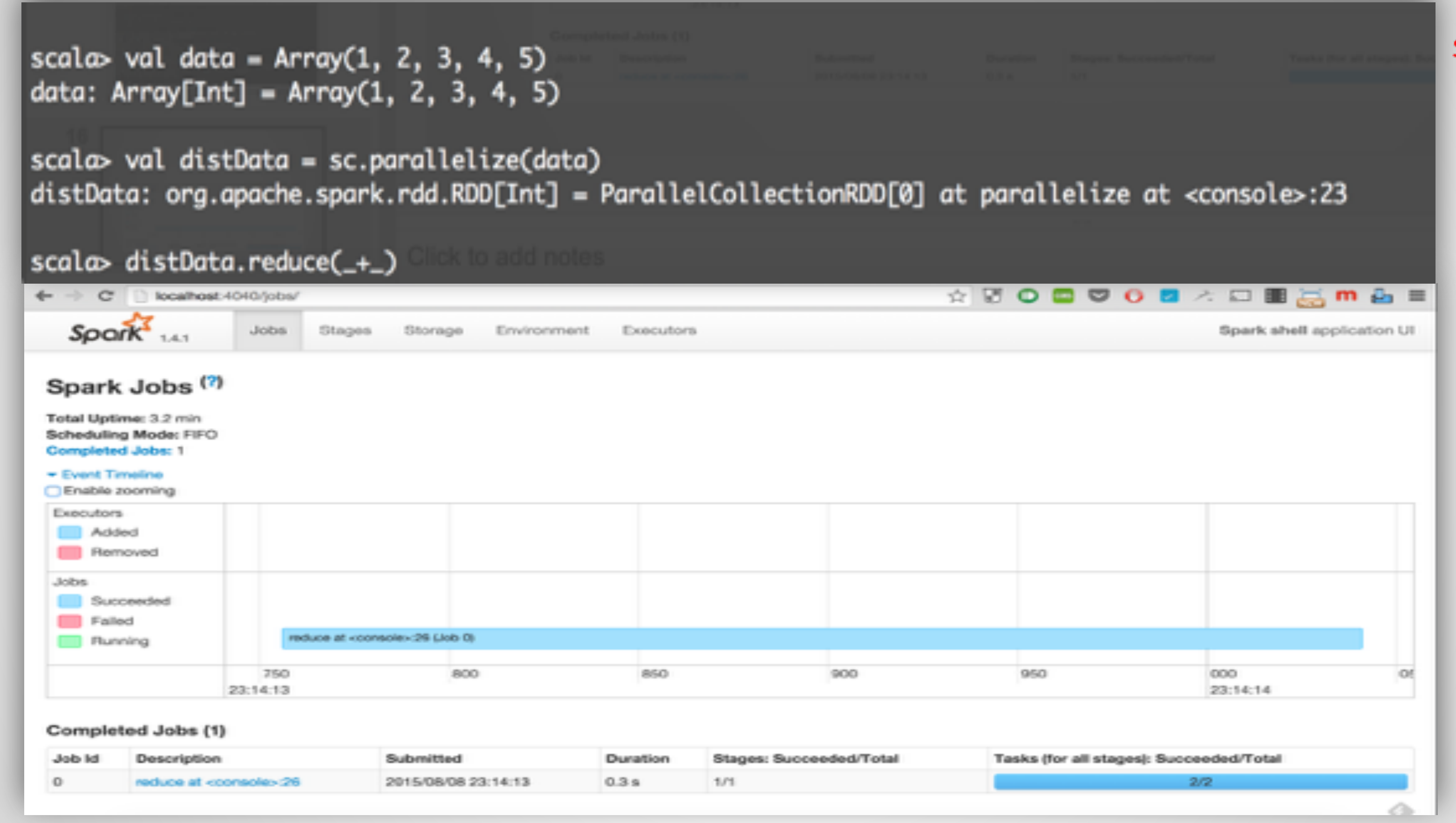
***spark-1.4.1-bin-hadoop2.6 $*** **./bin/spark-shell --master local[2]**



```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.4.1
      /_/

Using Scala version 2.10.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_45)
Type in expressions to have them evaluated.
Type :help for more information.
15/08/08 21:42:09 INFO SparkContext: Running Spark version 1.4.1
15/08/08 21:42:09 INFO SecurityManager: Changing view acls to: rahul
15/08/08 21:42:09 INFO SecurityManager: Changing modify acls to: rahul
15/08/08 21:42:17 INFO HiveMetaStore: Added admin role in metastore
15/08/08 21:42:17 INFO HiveMetaStore: Added public role in metastore
15/08/08 21:42:18 INFO HiveMetaStore: No user is added in admin role, since config is empty
15/08/08 21:42:18 INFO SessionState: No Tez session required at this point. hive.execution.engine=mr.
15/08/08 21:42:18 INFO SparkILoop: Created sql context (with Hive support)..
SQL context available as sqlContext.

scala>
```

The --master option specifies the master URL for a distributed cluster, or local to run locally with one thread, or local[N] to run locally with N threads.

SIGMOID



```
scala> val data = Array(1, 2, 3, 4, 5)
data: Array[Int] = Array(1, 2, 3, 4, 5)

scala> val distData = sc.parallelize(data)
distData: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:23

scala> distData.reduce(_+_)
```

localhost:4040/jobs/

Spark 1.4.1 | Jobs | Stages | Storage | Environment | Executors | Spark shell application UI

## Spark Jobs (?)

**Total Uptime:** 3.2 min
**Scheduling Mode:** FIFO
**Completed Jobs:** 1

▾ Event Timeline
☐ Enable zooming

Executors
🟦 Added
🟥 Removed

Jobs
🟦 Succeeded
🟥 Failed
🟩 Running

reduce at <console>:26 (Job 0)

750 23:14:13 | 800 | 850 | 900 | 950 | 000 23:14:14

**Completed Jobs (1)**

| Job Id | Description | Submitted | Duration | Stages: Succeeded/Total | Tasks (for all stages): Succeeded/Total |
|---|---|---|---|---|---|
| 0 | reduce at <console>:26 | 2015/08/08 23:14:13 | 0.3 s | 1/1 | 2/2 |

# RDD Introduction

Resilient
Distributed
Data Set

**Resilient Distributed Datasets** (RDDs), a ***distributed memory abstraction*** that lets programmers perform ***in-memory computations*** on large clusters in a ***fault-tolerant*** manner.

**RDD** shard the data over a cluster, like a virtualized, distributed collection.

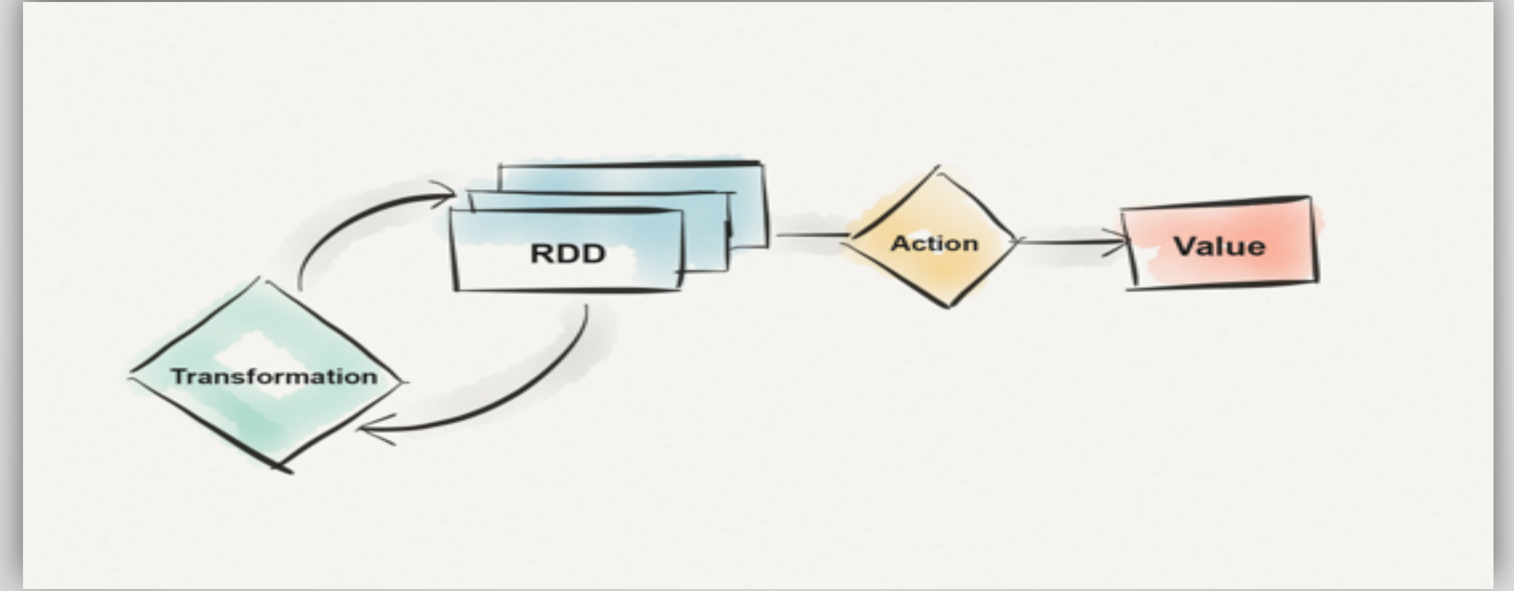
Users create **RDDs** in two ways: by **loading an external dataset**, or by **distributing a collection of objects** such as List, Map etc.

# RDD Operations

RDDs support two types of operations: **transformations** and **actions**.

Spark computes RDD only in a **lazy fashion.**

Only computation start when an **Action** call on RDD.



- **Simple SBT project setup** https://github.com/rahulkumar-aws/HelloWorld

```
$ mkdir HelloWorld
$ cd HelloWorld
$ mkdir -p src/main/scala
$ mkdir -p src/main/resources
$ mkdir -p src/test/scala
$ vim build.sbt
      name := “HelloWorld”

      version := “1.0”

      scalaVersion := “2.10.4”
$ mkdir project
$ cd project
$ vim build.properties
    sbt.version=0.13.8

$ vim scr/main/scala/HelloWorld.scala

 object HelloWorld { def main(args: Array[String]) = println("HelloWorld!") }
$ sbt run
```

## First Spark Application

**$git clone https://github.com/rahulkumar-aws/WordCount.git**

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
object SparkWordCount {

          def main(args: Array[String]): Unit = {

            val sc = new SparkContext("local","SparkWordCount")

            val wordsCounted = sc.textFile(args(0)).map(line=> line.toLowerCase)
                                 .flatMap(line => line.split("""\W+"""))
                                 .groupBy(word => word)
                                 .map{ case(word, group) => (word, group.size)}

            wordsCounted.saveAsTextFile(args(1))
            sc.stop()
          }
}
```

**$sbt "run-main ScalaWordCount src/main/resources/sherlockholmes.txt out"**

SIGMOID

## Launching Spark on Cluster

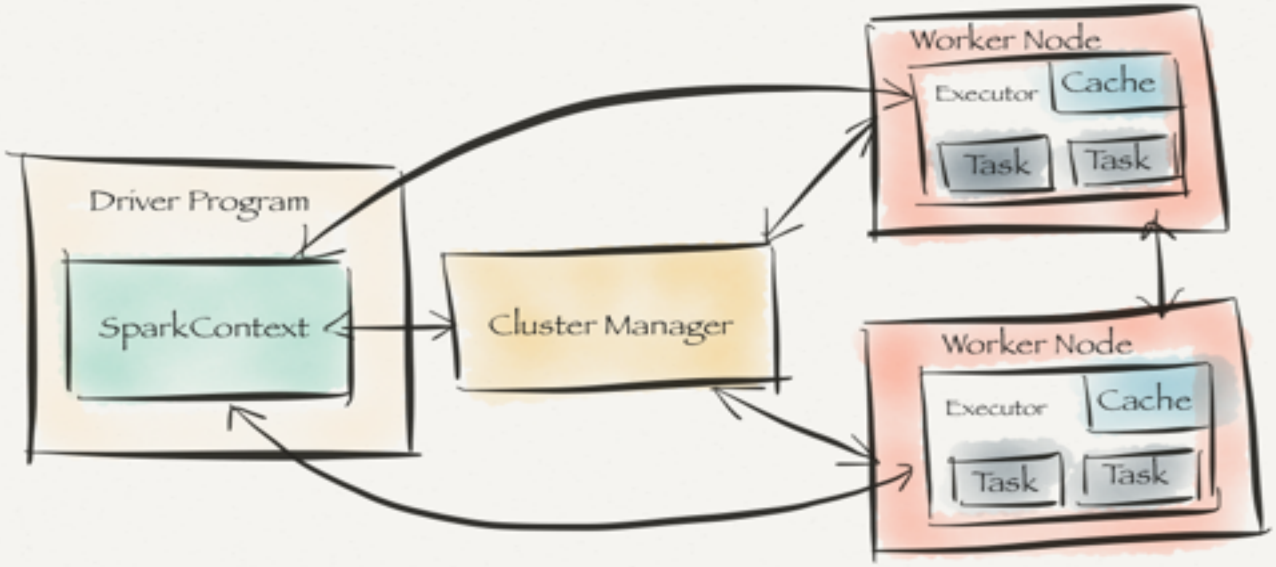
amazon web services™ EC2

Google Cloud Platform

Azure

MESOS

## Spark cluster components

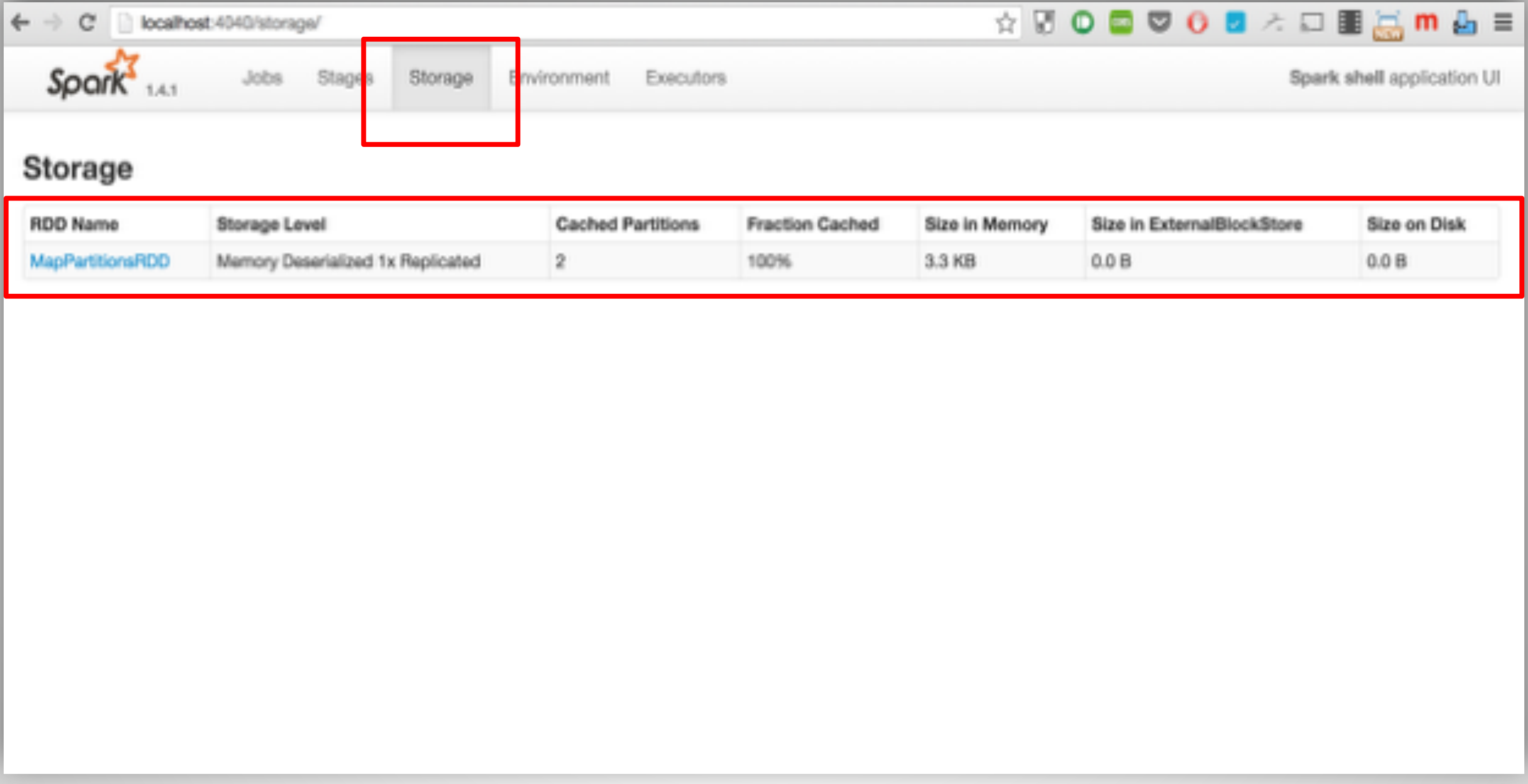


Cluster Manager  Can be Spark's own Standlone Cluster Manager or Mesos or YARN

## Spark Cache Introduction

Spark supports pulling data sets into a cluster-wide in-memory cache.

```
scala> val textFile = sc.textFile("README.md")

textFile: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[12] at textFile at <console>:21

scala> val linesWithSpark = textFile.filter(line => line.contains("Spark"))

linesWithSpark: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[13] at filter at
<console>:23

scala> linesWithSpark.cache()

res11: linesWithSpark.type = MapPartitionsRDD[13] at filter at <console>:23

scala> linesWithSpark.count()

res12: Long = 19
```



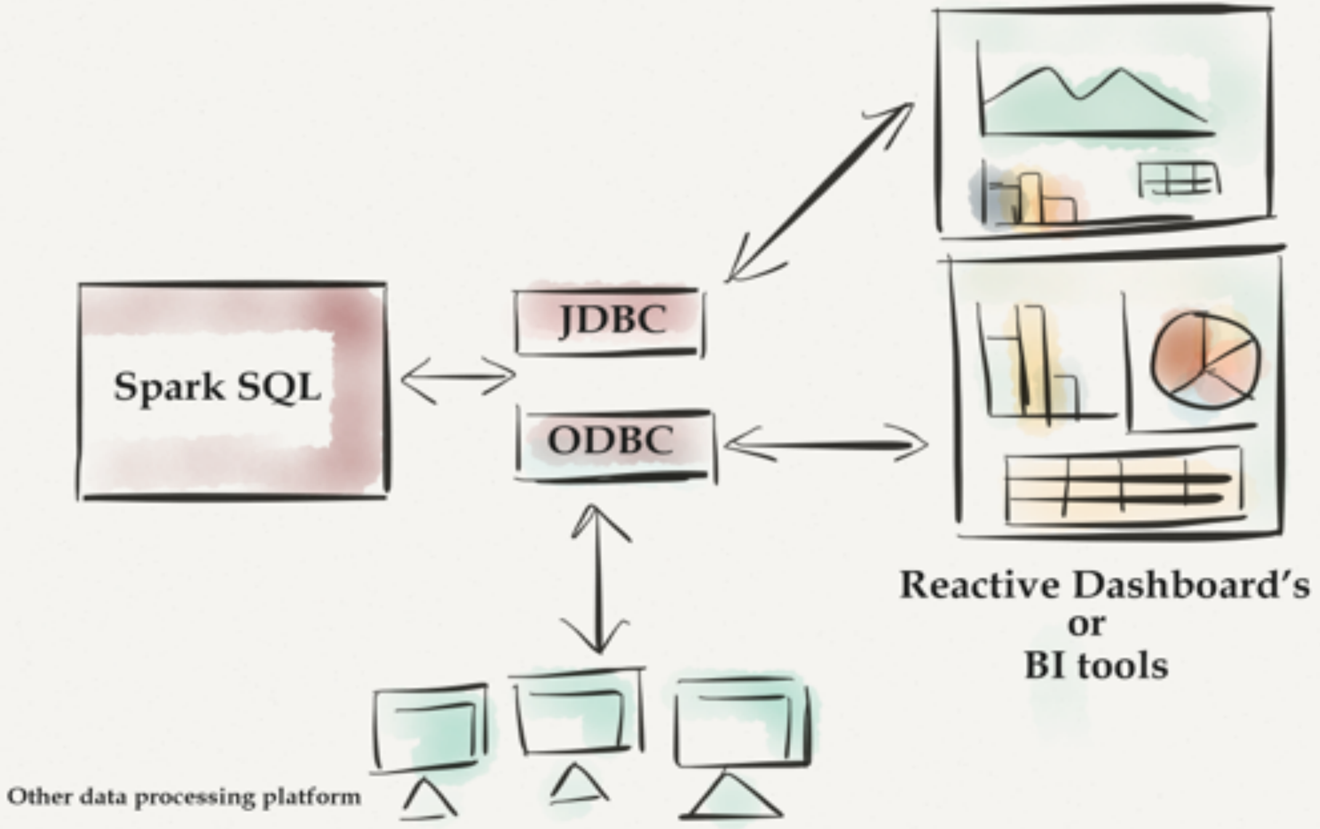
Storage

| RDD Name | Storage Level | Cached Partitions | Fraction Cached | Size in Memory | Size in ExternalBlockStore | Size on Disk |
|---|---|---|---|---|---|---|
| MapPartitionsRDD | Memory Deserialized 1x Replicated | 2 | 100% | 3.3 KB | 0.0 B | 0.0 B |

## Spark SQL Introduction

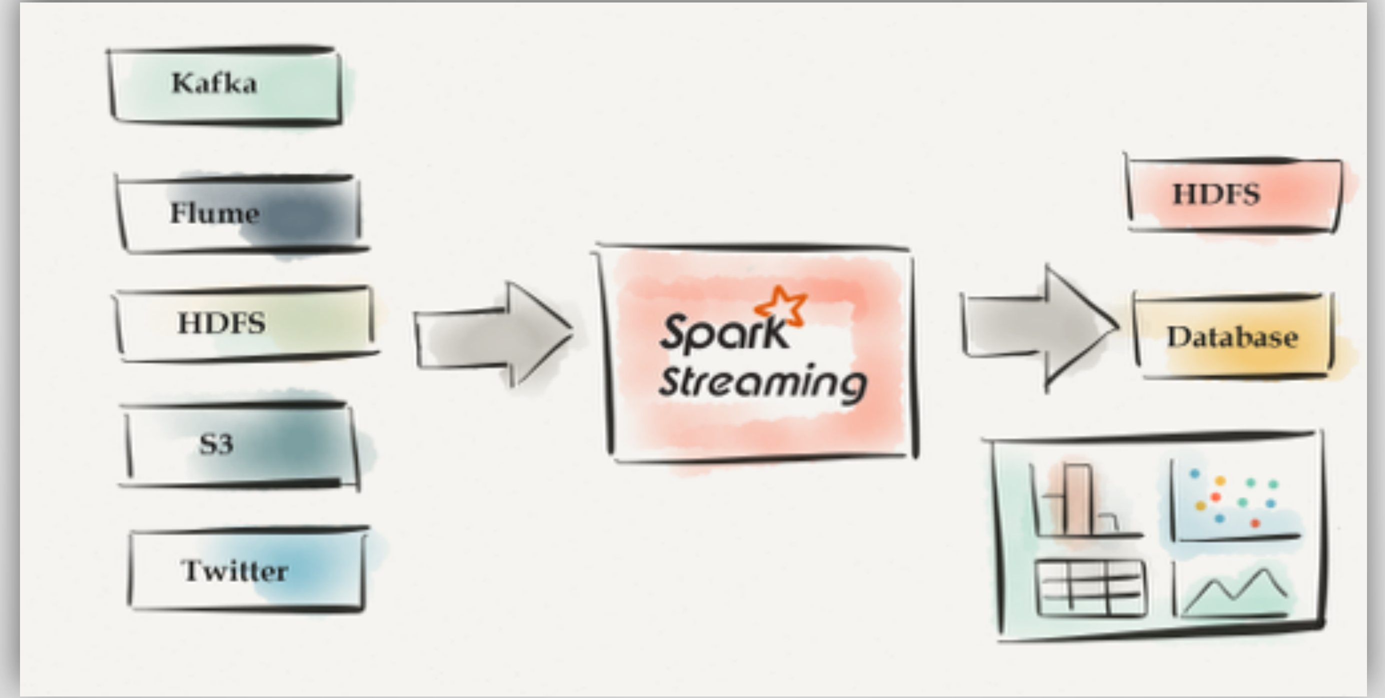
**Spark SQL** is Spark's module for working with structured data.

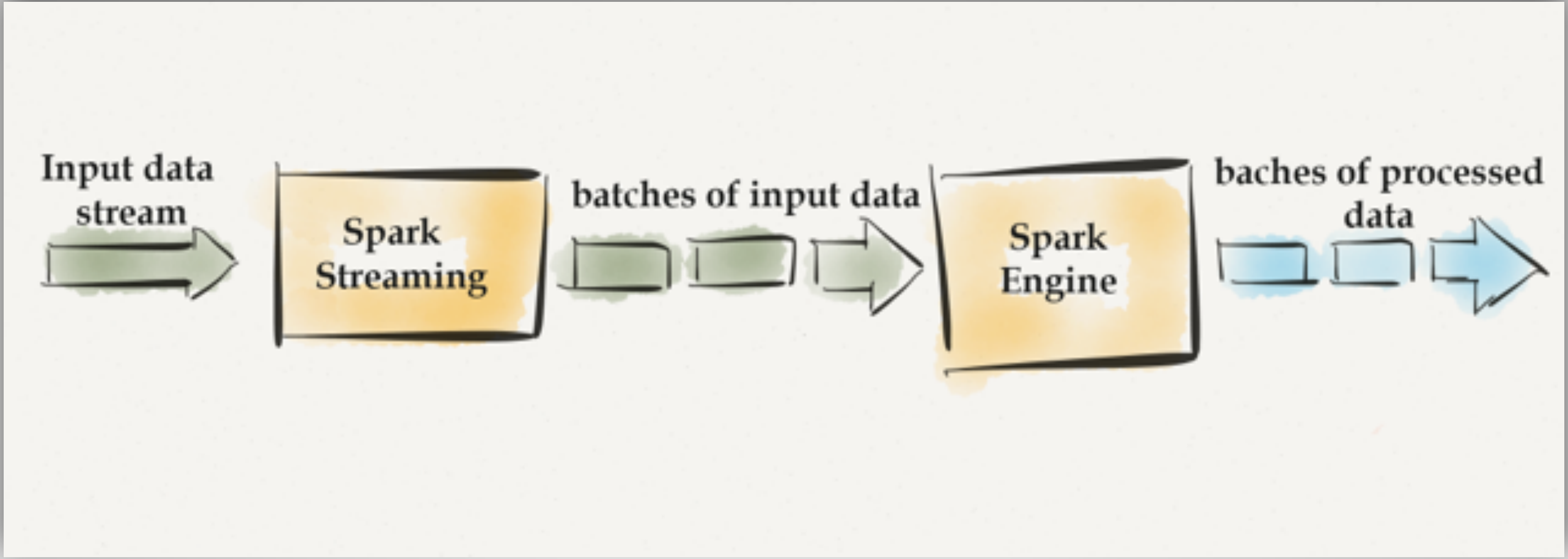
- Mix SQL queries with Spark programs
- Uniform Data Access, Connect to any data source
- DataFrames and SQL provide a common way to access a variety of data sources,
  including Hive,
  Avro,
  Parquet,
  ORC,
  JSON,
  and JDBC.
- Hive Compatibility Run unmodified Hive queries on existing data.
- Connect through JDBC or ODBC.


Spark SQL
JDBC
ODBC
Reactive Dashboard's
or
BI tools
Other data processing platform

# Spark Streaming Introduction

Spark Streaming is an extension of the core Spark API that enables **scalable**, **high-throughput**, **fault-tolerant stream processing** of live data streams.





**$git clone** [https://github.com/rahulkumar-aws/WordCount.git](https://github.com/rahulkumar-aws/WordCount.git)

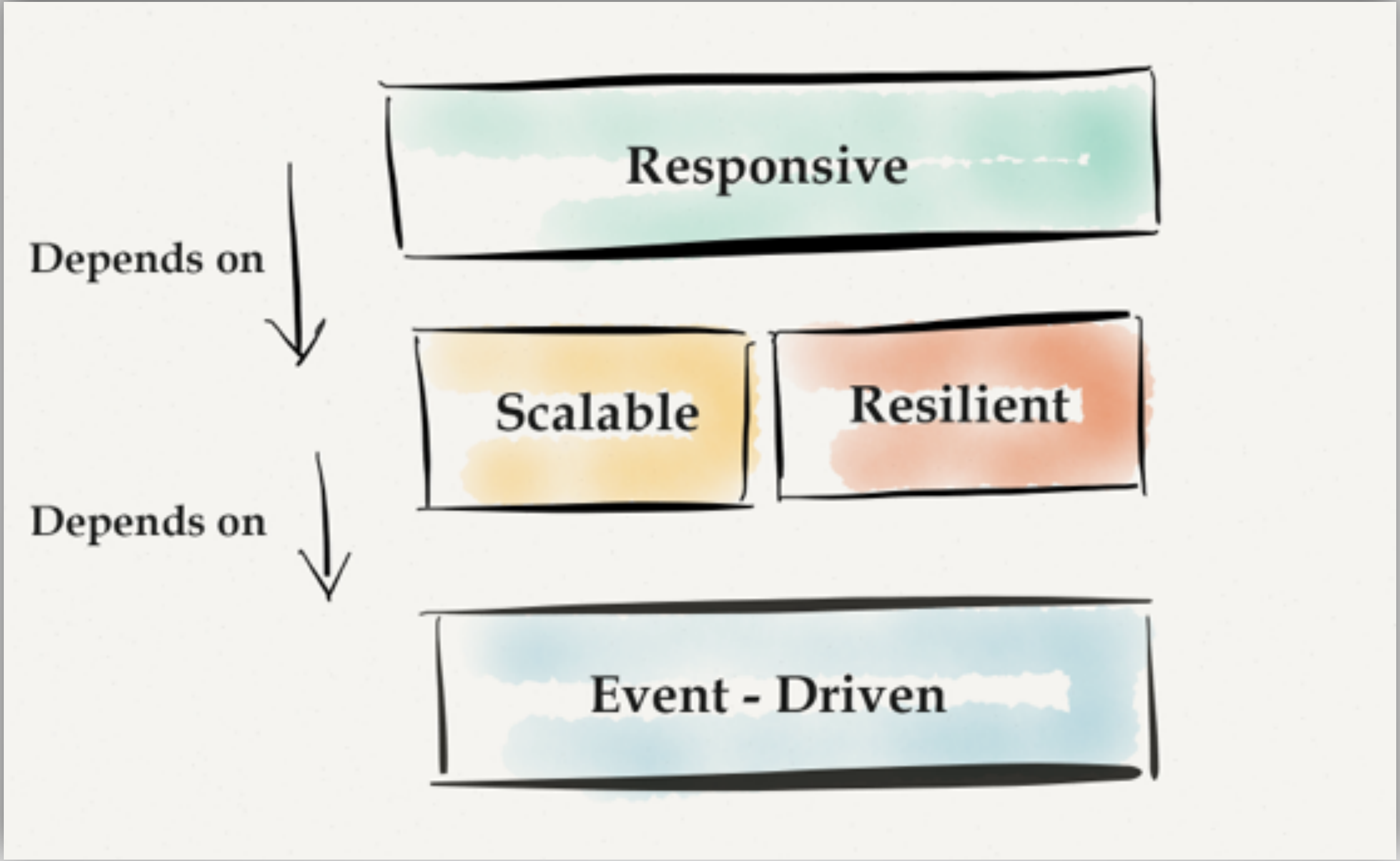
$ nc -lk 9999

sbt "run-main StreamingWordCount"

## Reactive Application

- Responsive
- Resilient
- Elastic
- Event Driven

http://www.reactivemanifesto.org


Responsive
Depends on
Scalable
Resilient
Depends on
Event - Driven

SIGMOID

SIGMOID

## Typesafe Reactive Platform

play akka Scala Java

## Play Framework

The High Velocity Web Framework For Java and Scala

- RESTful by default
- JSON is a first class citizen
- Web sockets, Comet, EventSource
- Extensive NoSQL & Big Data Support

https://www.playframework.com/download

https://downloads.typesafe.com/typesafe-activator/1.3.5/typesafe-activator-1.3.5-minimal.zip

## Akka

Akka is a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM.

- Simple Concurrency & Distribution
- Resilient by Design
- High Performance
- Elastic & Decentralized
- Extensible

Akka uses **Actor Model** that raise the abstraction level and provide a better platform to build **scalable**, **resilient** and **responsive applications.**

SIGMOID

## Demo

## References

https://www.cs.berkeley.edu/~matei/papers/2012/nsdi_spark.pdf

http://spark.apache.org/docs/latest/quick-start.html

Learning Spark Lightning-Fast Big Data Analysis

By Holden Karau, Andy Konwinski, Patrick Wendell, Matei Zaharia

https://www.playframework.com/documentation/2.4.x/Home

http://doc.akka.io/docs/akka/2.3.12/scala.html

# Thank You

SIGMOID

Rahul Kumar rahul.k@sigmoid.com @rahul_kumar_aws