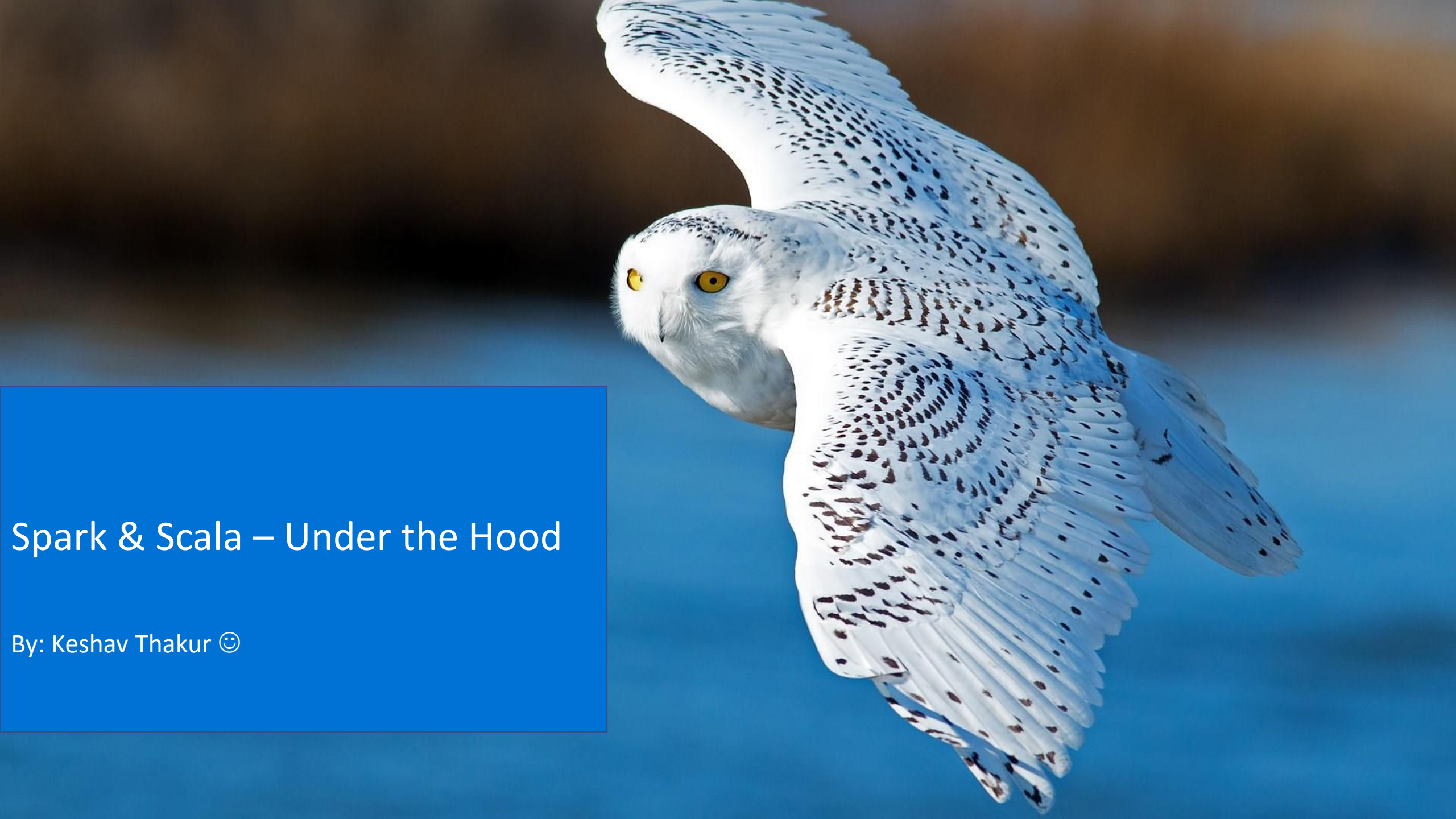


Spark & Scala – Under the Hood

By: Keshav Thakur ☺

Course Content

Day – 1 (Approx 3 Hours)

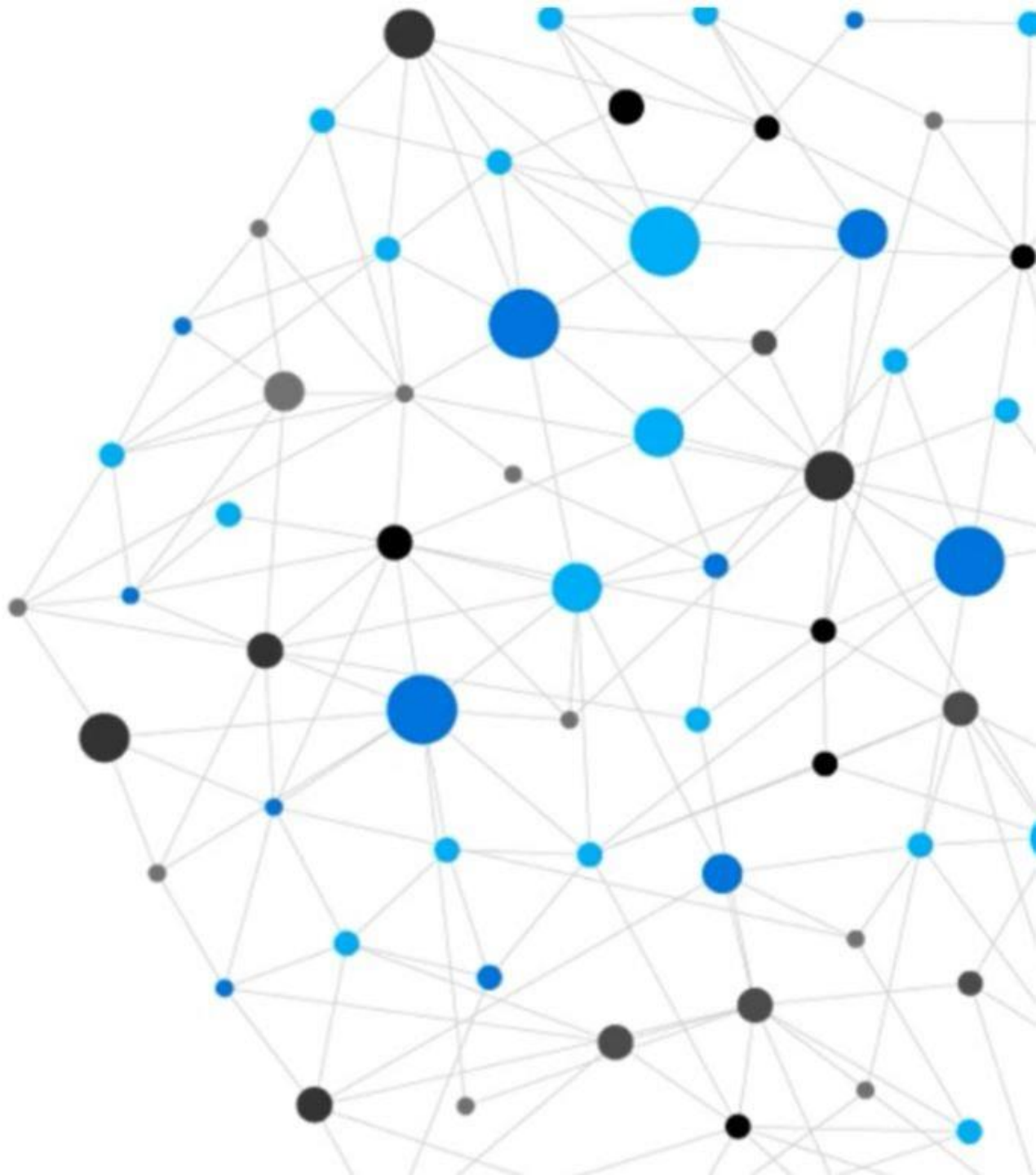
- Environment Setup – JAVA, Scala, Spark, SBT
- Big Data Analysis with Spark – Key Concepts
- -- Break – 10 Mins
- Spark Under the Hood – RDD's, Reduction Operation
- Demonstration – 20 Mins

Day – 2 (Approx 3 Hours)

- Spark SQL, Dataframe, Dataset
- Partitioning & Shuffling – What & Why it's important?
- -- Break – 10 Mins
- About Join's – 3 Possible Use Cases
- Demonstration – Use Case Implementation – 20 Mins

Day – 3 (Approx 4 Hours)

- Spark Optimization – Key Factors
- Debugging Spark Application – Spark UI Module
- -- Break – 10 Min
- Spark Cluster – Deciding Numbers & Configuration
- Interview Perspective – Q & A
- Demonstration – Final Use Case – 20 Mins



Big Data Analysis with Spark | Since 2009 in UC Berkeley's AMPLab

Big Data Processing using Spark:

Why Spark in Detail -- Alternative to Spark

Core Spark, Spark-SQL, Spark Streaming, Spark ML

Big Data Analysis with Scala & Spark

Tool Setup:

JAVA – Installing JDK -- https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html

SCALA – Installing Scala -- https://www.scala-lang.org/download/

sbt – Installing SBT -- https://www.scala-sbt.org/release/docs/Setup.html

Spark – Installing spark -- https://spark.apache.org/downloads.html

Hadoop – On Windows -- we need winutils.exe file -- <hadoop_version>/bin

create a folder **hadoop,** inside it create a folder named **bin**, paste the downloaded **winutils.exe** file

Download IntelliJ IDEA Community Edition

Install the Scala plugin

Setup the JDK

Creating a project - Click Create New Project on the Welcome Screen, then select Scala, and finally SBT Project.

Creating a Scala worksheet & Scala Class – Execute it.

Installing SCALA-IDE for Eclipse -- http://scala-ide.org/download/sdk.html

http://scala-ide.org/docs/current-user-doc/gettingstarted/index.html

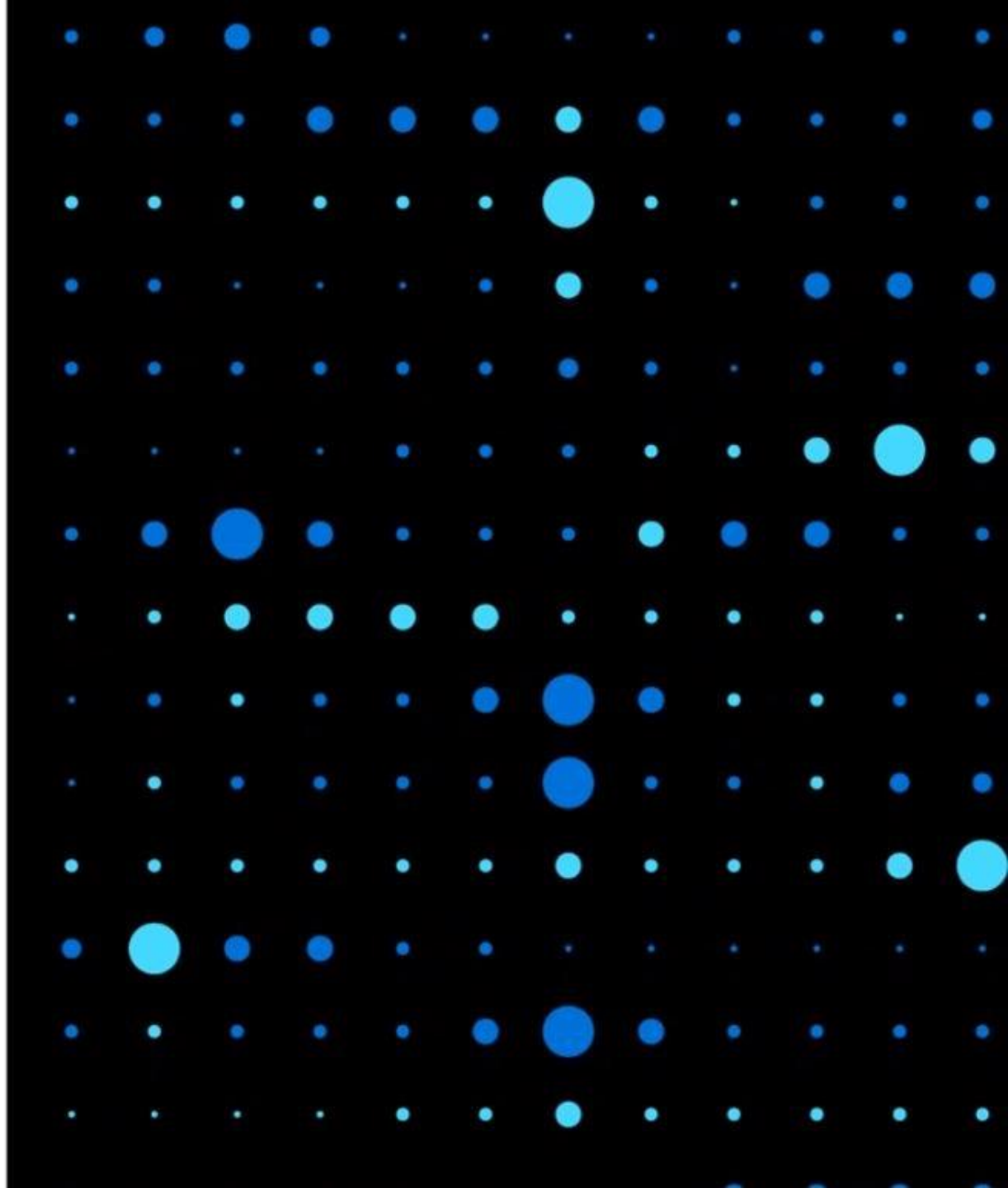
Else → https://community.cloud.databricks.com/login.html

Demonstration

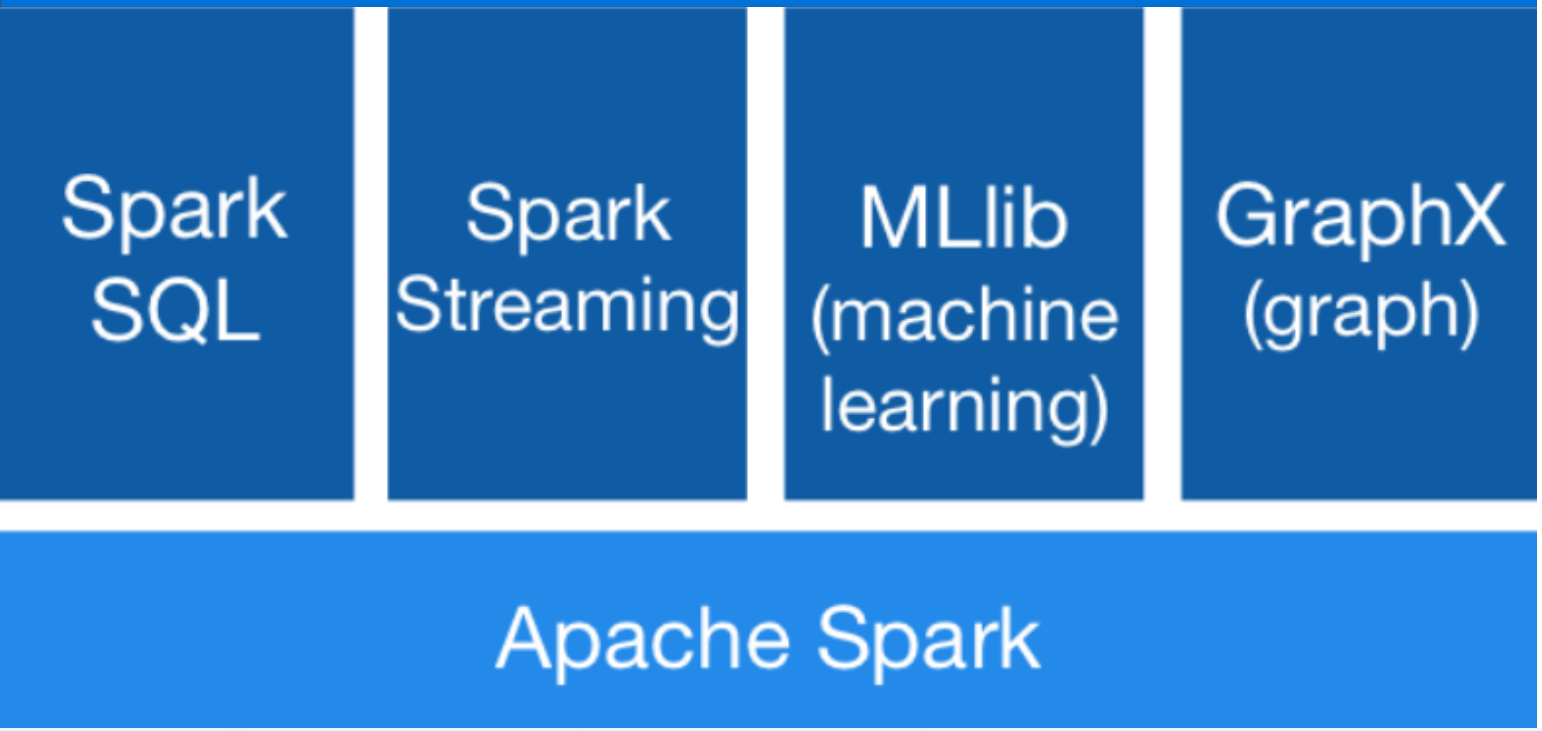
Setting up Environment: Spark
Scala Code Testing using Data bricks Community
https://community.cloud.databricks.com/login.html

https://medium.com/free-code-camp/learning-scala-from-0-60-part-i-dc095d274b78

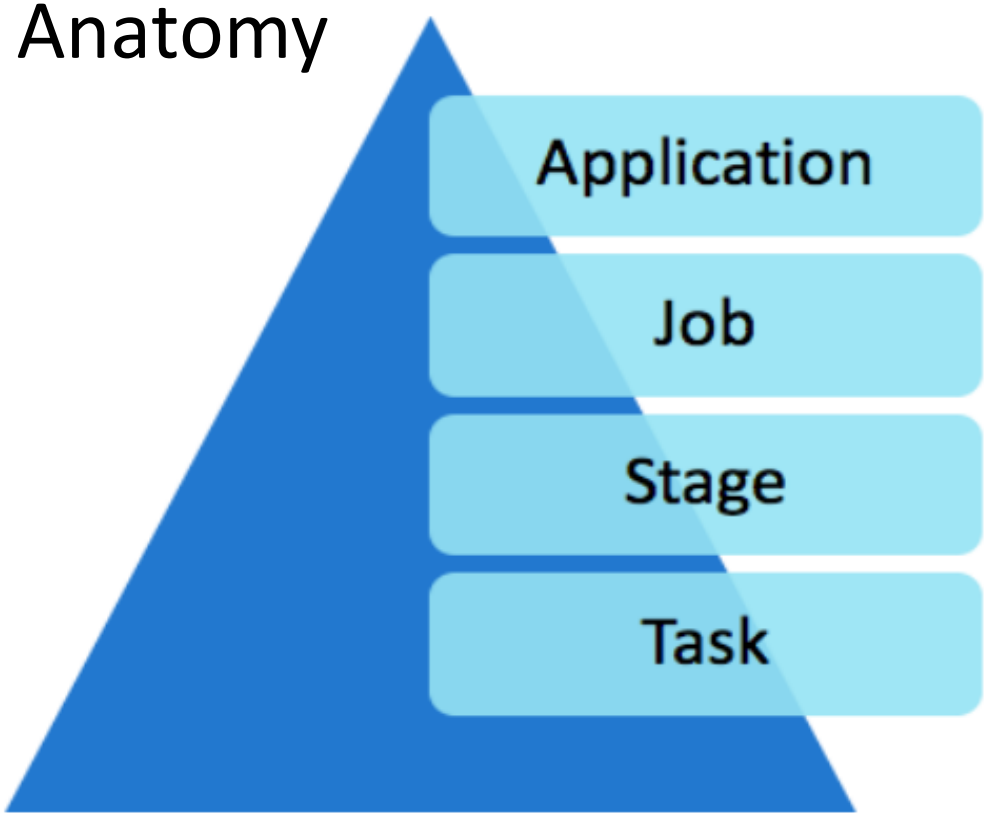
https://towardsdatascience.com/ultimate-pyspark-cheat-sheet-7d3938d13421



Big Data Analysis with Spark | Why?

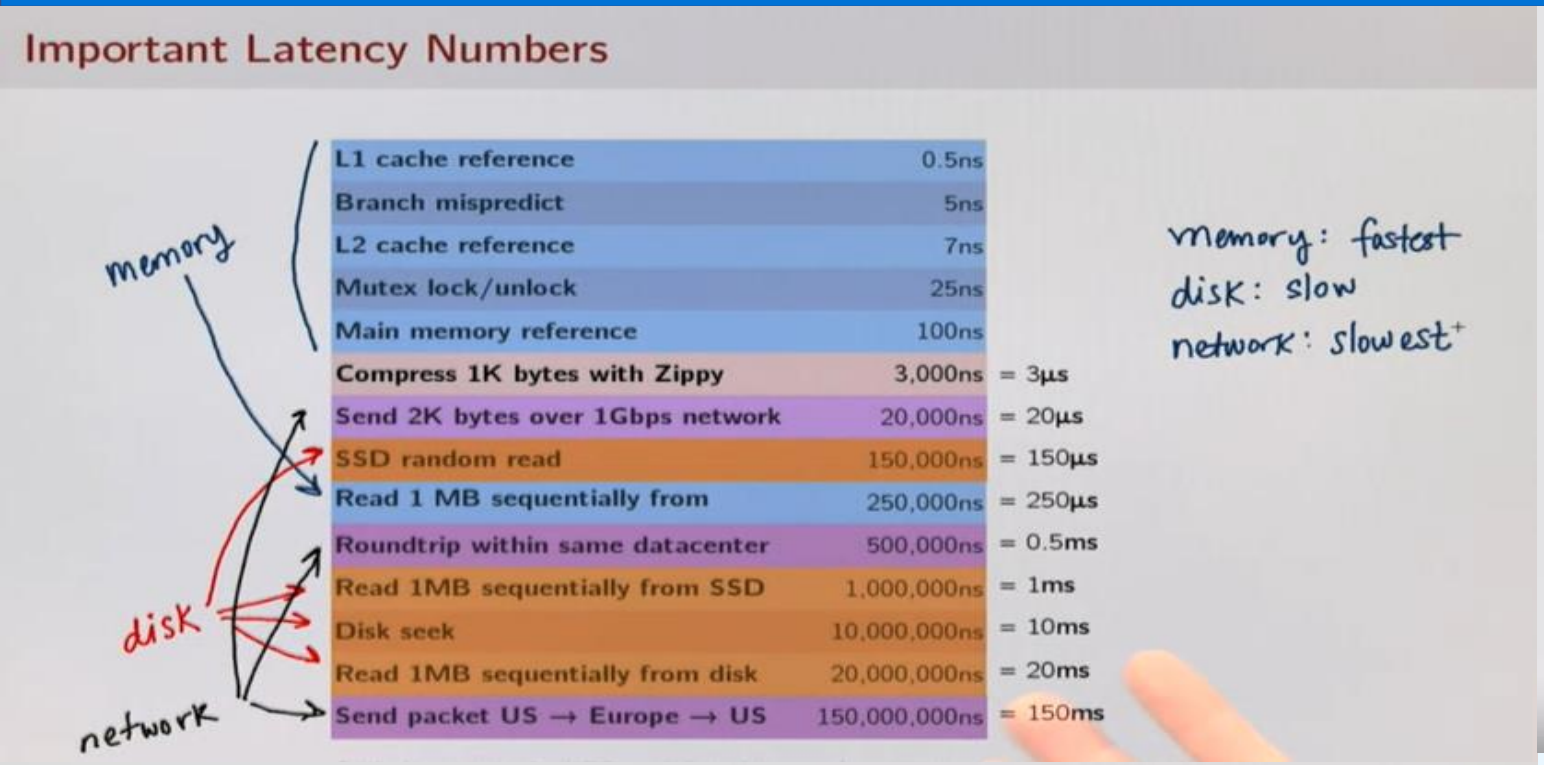


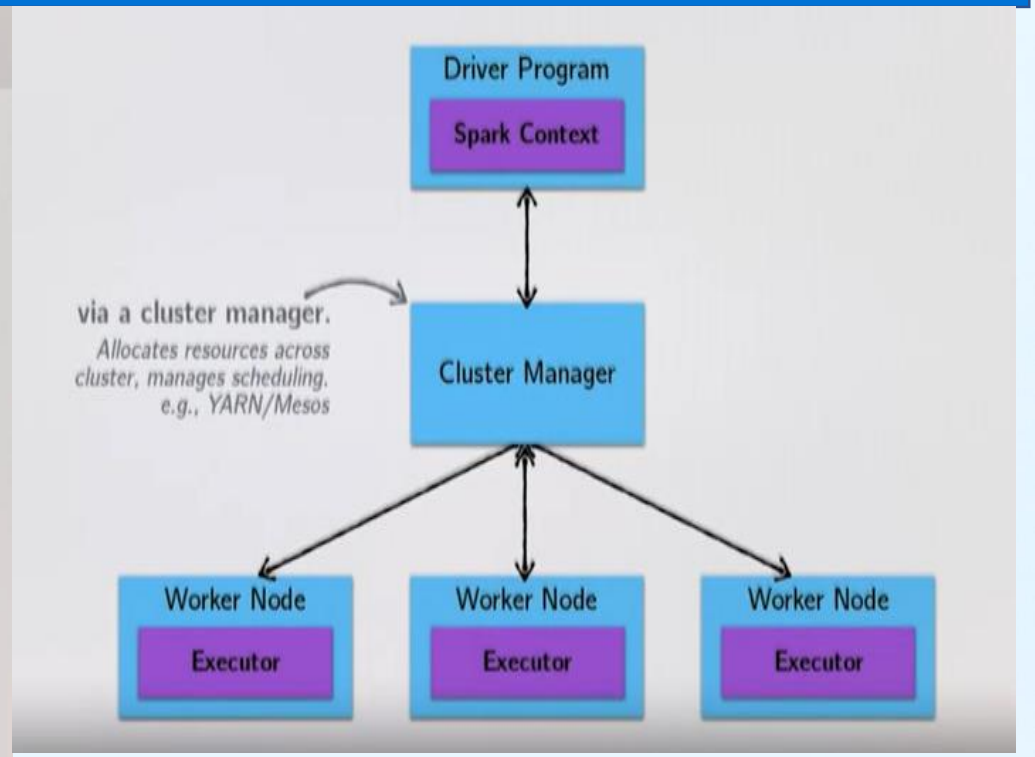
Anatomy

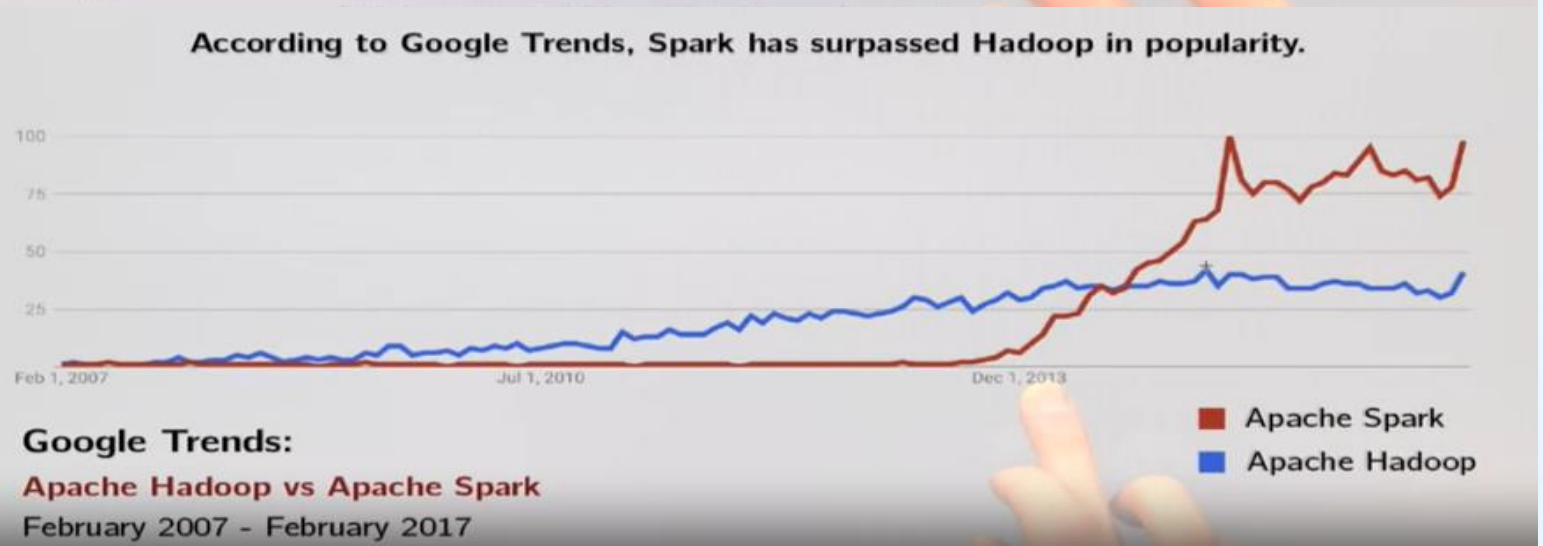


Data Parallel to Distributed Data Parallel – JVM Memory to the Multiple Worker Journey – Vertical to Horizontal Scaling
Better than R-Octave-MATLAB-Python…. Yet it support R-Python-JAVA-Scala
Lightening Fast Cluster computing – Data Science & Analytics an Ideal Use Case
Programming Problems well Managed → Concurrency, Thread starving, Partial Failure, Latency etc
Immutable, In-Memory, Fault Tolerant, Lazy
RDD's, Data frame, Datasets
Spark is better than these → Hive, Impala, MapReduce, Pig, Dremel, Tej, Storm, Giraph, Mahout, Drill etc …….. ????

Big Data Analysis with Spark | How it works under the Hood?



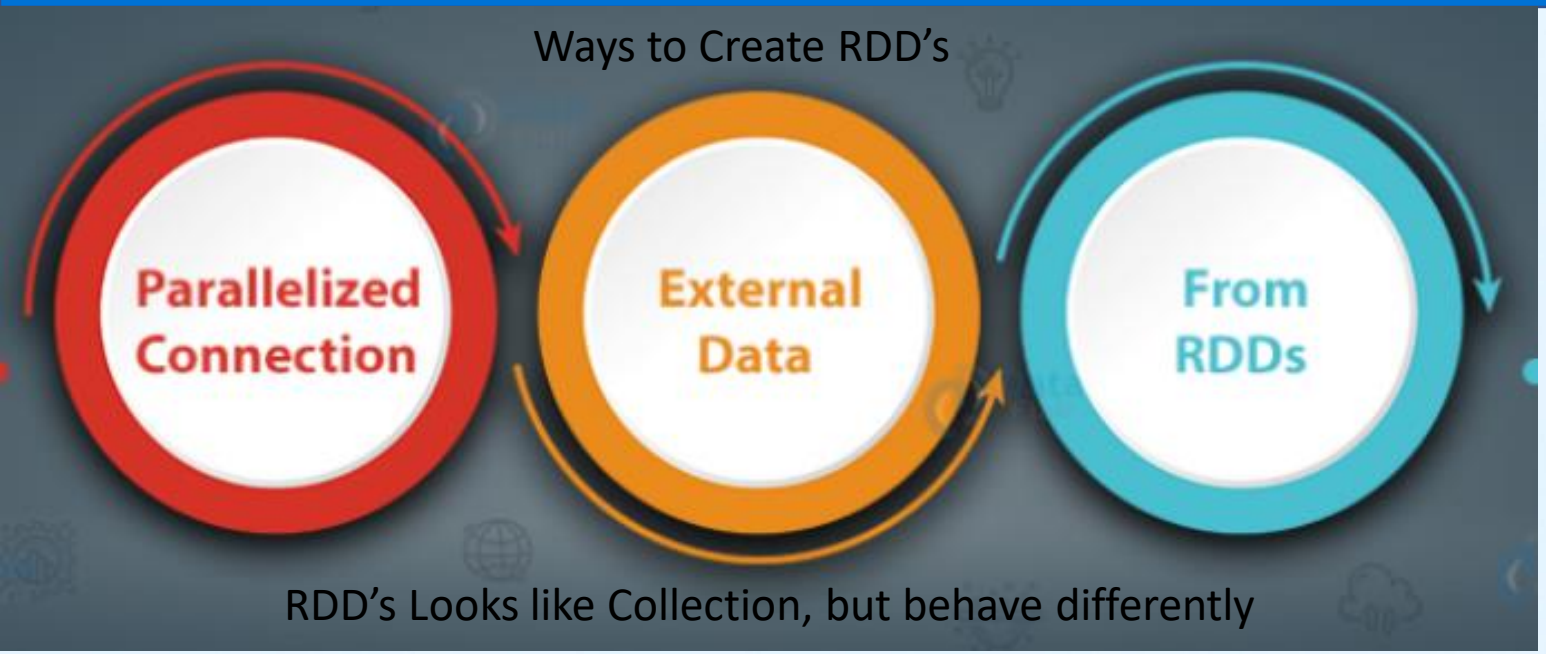




SparkContext(sc) is the entry point for Spark functionality. A Spark Context represents the connection to a Spark cluster and can be used to create RDDs in the cluster. Only one SparkContext should be active per JVM. -- https://spark.apache.org/docs/latest/api/scala/org/apache/spark/SparkContext.html

SparkSession(spark) is the entry to programming Spark with the Dataset and DataFrame API. It is one of the very first objects you create while developing a Spark SQL application HiveContext is a super set of the SQLContext

Big Data Analysis with Spark | RDD's → Group of Partitions → rdd.partitions.size




```
abstract class RDD[T] {
        def map[U] = {...}
        def flatMap[U] = {...}
        def filter = {...}
        def reduce = {...}
        def fold = {...}
        def aggregate = {...}
}
```

```
val sc = spark.sparkContext
val myRdd: RDD[(String, Int)] = sc.parallelize(
                        Seq(("Jon", 1),("Tyrion",2),("Bronn", 3)))
```

```
val sc = spark.sparkContext
val textFileRDD = sc.textFile("c://spark//data//sample.csv")
textFileRDD.collect.foreach(println)
```

```
Val countRDD = textFileRDD.flatMap(line=>line.split(" "))
                        .map(word => (word, 1))
                        .reduceByKey(_+_)
```

Transformations & Actions
Or Transformers & Accessors
Or Mapper & Reducers
Or Lazy & Eager

Transformation → Map, Flatmap, GroupBy
Action → count, collect, foreach, take, reduce, saveAsTextFile, saveAsSequenceFile,

Caching & Persistence

Cache→ Default Storage In-Memory
Persist → to any other storage type(memory or disk)