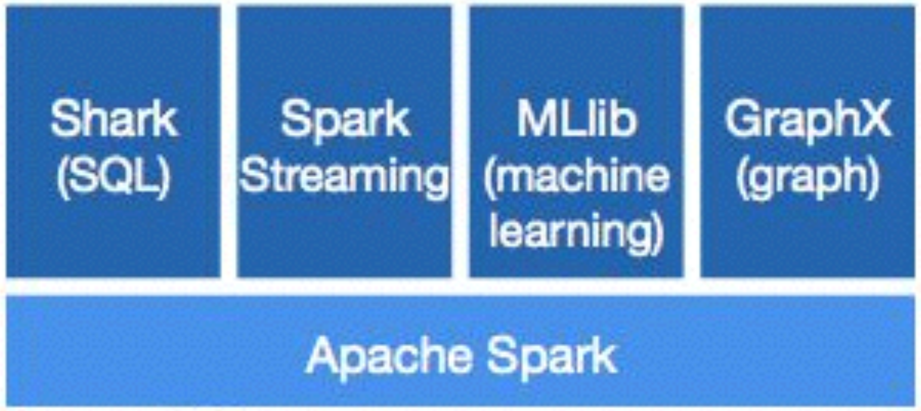
Scala/Spark

Wikipedia :

Apache Spark is an open-source cluster computing framework originally developed in the AMPLab at UC Berkeley. In contrast to Hadoop's two-stage disk-based MapReduce paradigm, Spark's in-memory primitives provide performance up to 100 times faster for certain applications.[1] By allowing user programs to load data into a cluster's memory and query it repeatedly, Spark is well-suited to machine learning algorithms.[2]
Spark requires a cluster manager and a distributed storage system. For cluster management, Spark supports standalone (native Spark cluster), Hadoop YARN, or Apache Mesos.[3] For distributed storage, Spark can interface with a wide variety, including Hadoop Distributed File System (HDFS),[4] Cassandra,[5] OpenStack Swift, and Amazon S3. Spark also supports a pseudo-distributed local mode, usually used only for development or testing purposes, where distributed storage is not required and the local file system can be used instead; in this scenario, Spark is running on a single machine with one executor per CPU core.

Project Components

1. Spark Core and Resilient Distributed Datasets (RDDs)
2. Spark SQL
3. Spark Streaming
4. MLlib Machine Learning Library
5. GraphX

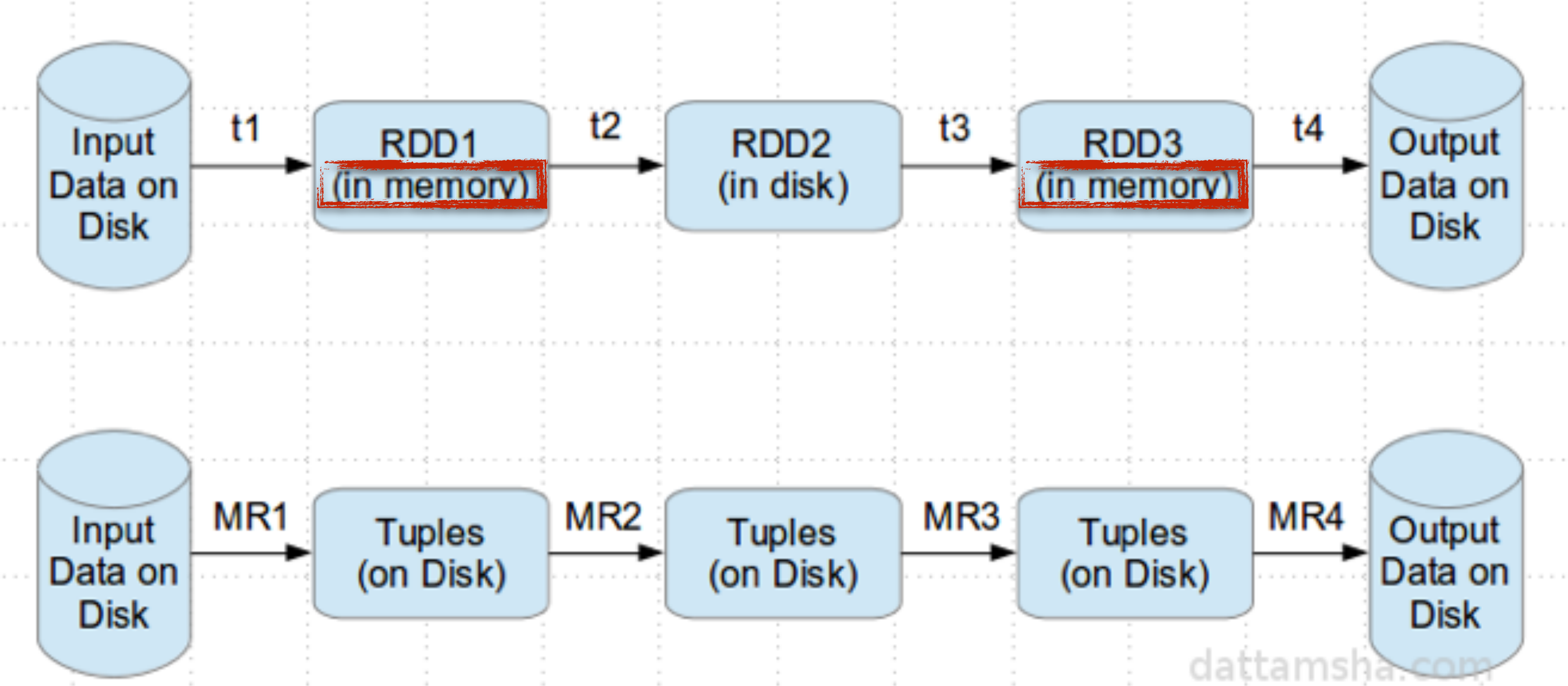


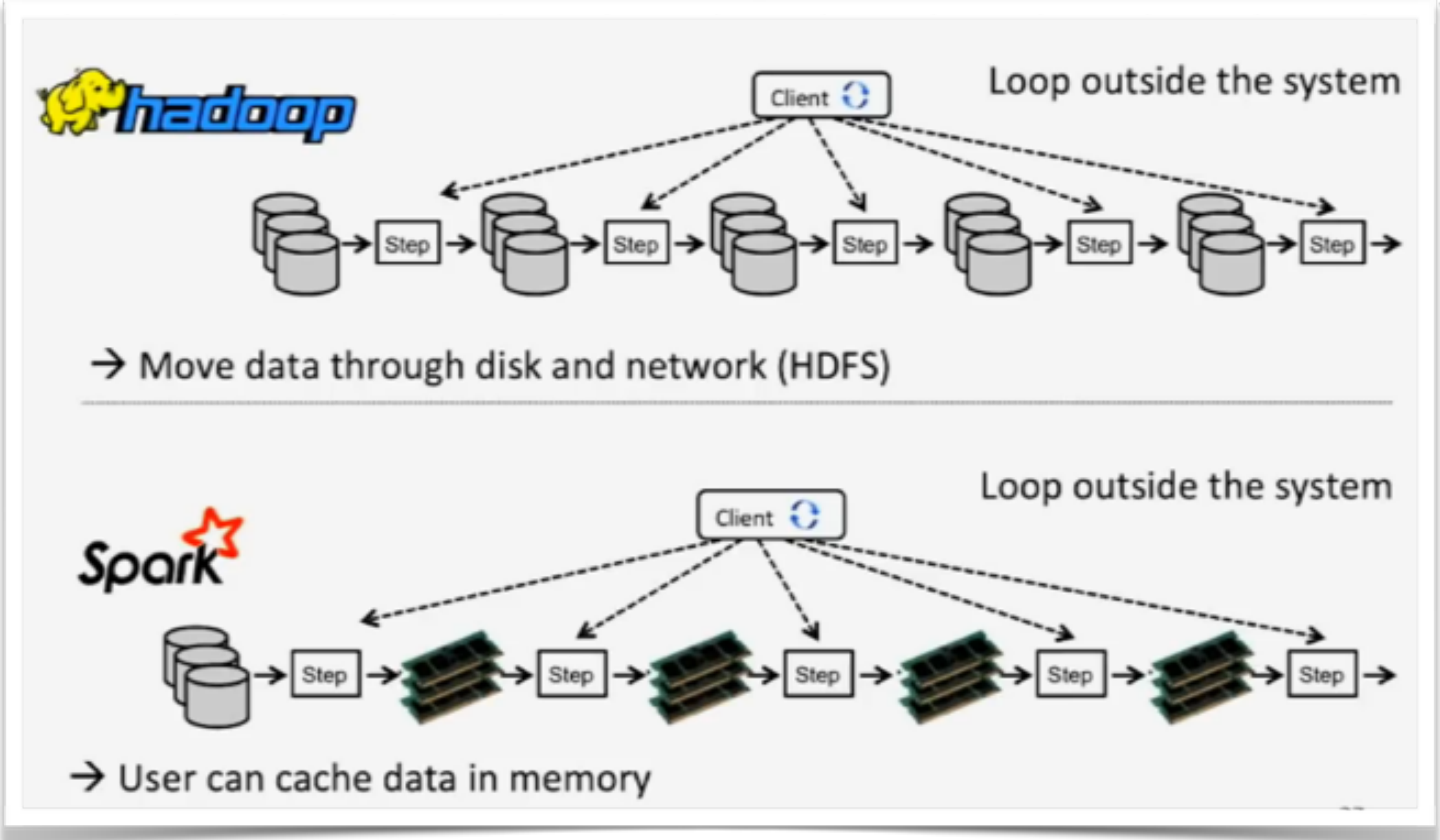
Shark (SQL)
Spark Streaming
MLlib (machine learning)
GraphX (graph)
Apache Spark

Features

- Java, Scala, Python, and R APIs.
- Scalability to over 8000 nodes in production.[11]
- Ability to cache datasets in memory for interactive data analysis: extract a working set, cache it, query it repeatedly.
- Interactive command line interface (in Scala or Python) for low-latency horizontally scalable data exploration.
- Higher level library for stream processing[12], through Spark Streaming.
- Support for structured and relational query processing (SQL), through Spark SQL.
- Higher level libraries for machine learning and graph processing.

Hadoop vs Spark

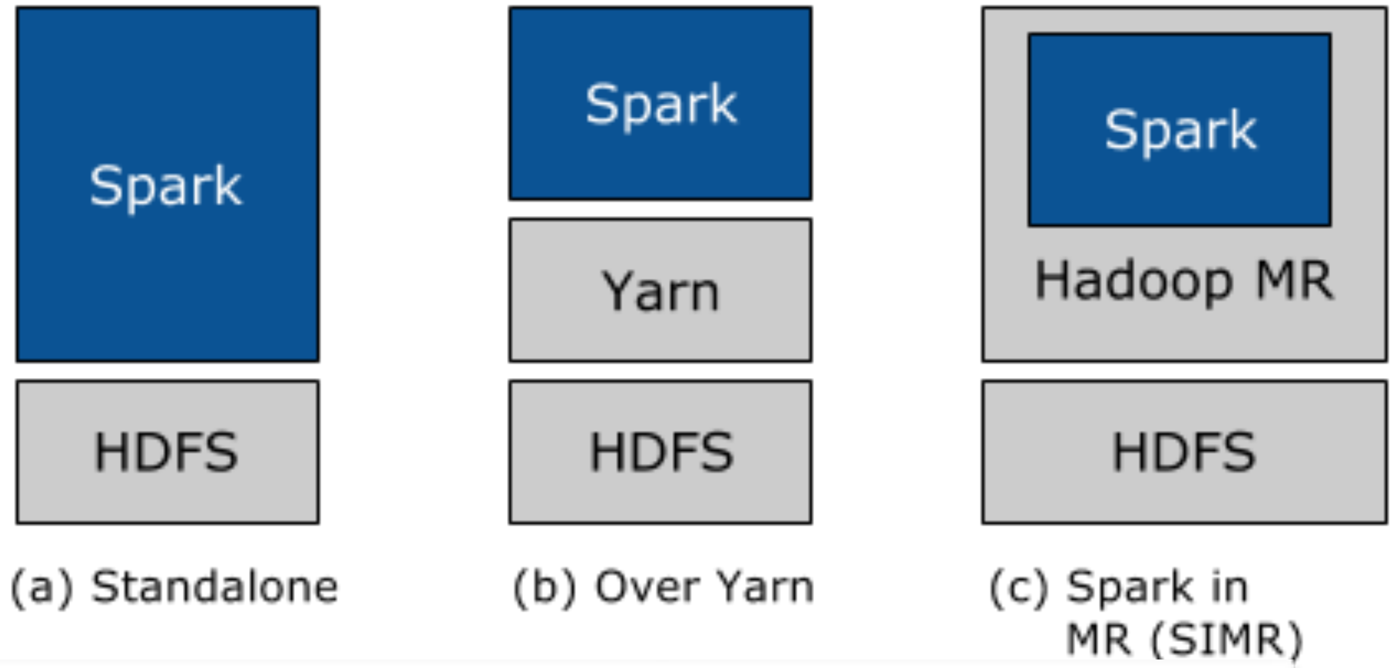


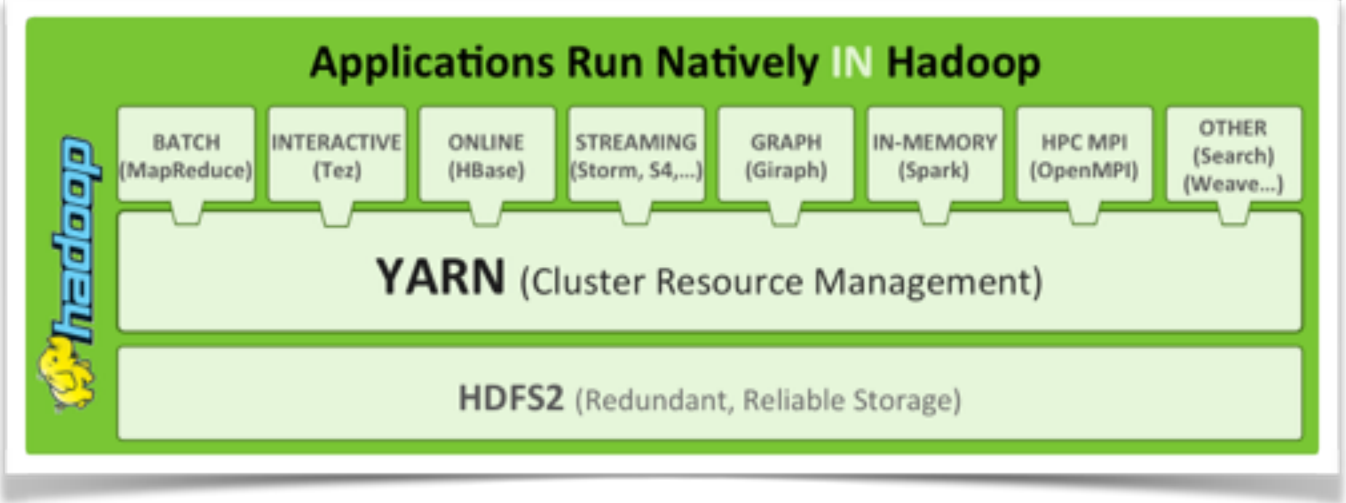


→ Move data through disk and network (HDFS)

→ User can cache data in memory

Spark and Hadoop : Working Together




Applications Run Natively IN Hadoop
BATCH (MapReduce)
INTERACTIVE (Tez)
ONLINE (HBase)
STREAMING (Storm, S4,...)
GRAPH (Giraph)
IN-MEMORY (Spark)
HPC MPI (OpenMPI)
OTHER (Search) (Weave...)
YARN (Cluster Resource Management)
HDFS2 (Redundant, Reliable Storage)
hadoop

Spark In MapReduce (SIMR): For the Hadoop users that are not running YARN yet, another option, in addition to the standalone deployment, is to use SIMR to launch Spark jobs inside MapReduce. With SIMR, users can start experimenting with Spark and use its shell within a couple of minutes after downloading it! This tremendously lowers the barrier of deployment, and lets virtually everyone play with Spark.

History

The design of Scala started in 2001 at the École Polytechnique Fédérale de Lausanne (EPFL) by Martin Odersky, following on from work on Funnel, a programming language combining ideas from functional programming and Petri nets.[10] Odersky had previously worked on Generic Java and javac, Sun's Java compiler.[10]
After an internal release in late 2003, Scala was released publicly in early 2004 on the Java platform,[11] and on the .NET platform in June 2004.[6][10][12] A second version (v2.0) followed in March 2006.[6] The .NET support was officially dropped in 2012.[13]
Although Scala had extensive support for functional programming from the beginning, Java remained a purely object oriented language until the introduction of lambda expressions with Java 8 in 2014.

Scala
Java
toptal

```
def products = orders.flatMap(o => o.products)
```

```
public List<Product>getProducts(){
   List<Product> products = new ArrayList<Product>();
   for (Order order : order) {
      products.addAll(order.getProducts());
   }
   return products;
}
```

- In Scala, the compiler is incredibly smart, so this avoids the developer needing to specify explicitly those things that the compiler can infer.
- Scala is ported on Spark (java libraries can be used)

Build Scala project with SBT

http://www.scala-sbt.org/0.13/tutorial/index.html

Scala crash course

https://www.sics.se/~amir/files/download/dic/scala.pdf

Examples Scala

- **Declare a list of integers as a variable called “myNumbers”:**

```
scala> val myNumbers = List(1, 2, 5, 4, 7, 3)
myNumbers: List[Int] = List(1, 2, 5, 4, 7, 3)
```

- **Declare a function, `cube`, that computes the cube (third power) of an Int.**

```
scala> def cube(a: Int): Int = a * a * a
cube: (a: Int)Int
```

- **Apply the function to `myNumbers` using the `map` function.**

```
scala> myNumbers.map(x => cube(x))
       res: List[Int] = List(1, 8, 125, 64, 343, 27)
       // Scala also provides some shorthand ways of writing this:
       // myNumbers.map(cube(_))
       // myNumbers.map(cube)
```

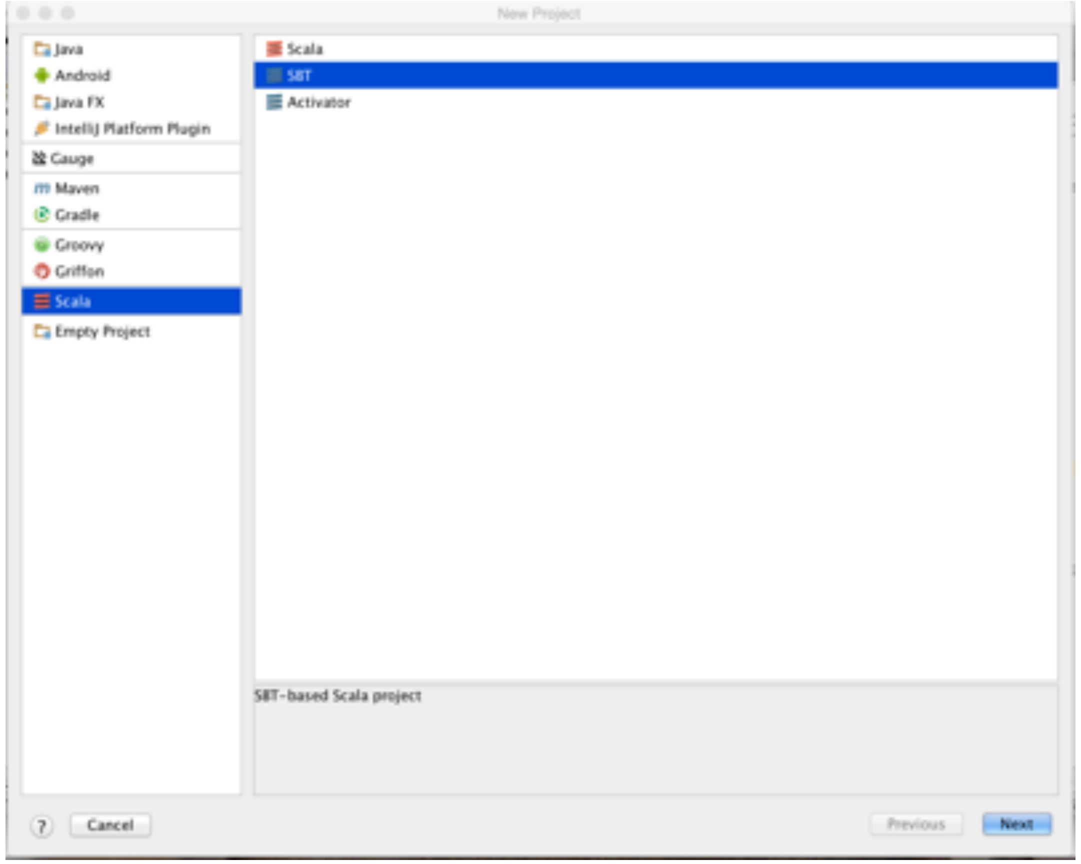
- **Then also try writing the function inline in a `map` call, using closure notation.**

```
scala> myNumbers.map{x => x * x * x}
res: List[Int] = List(1, 8, 125, 64, 343, 27)
```

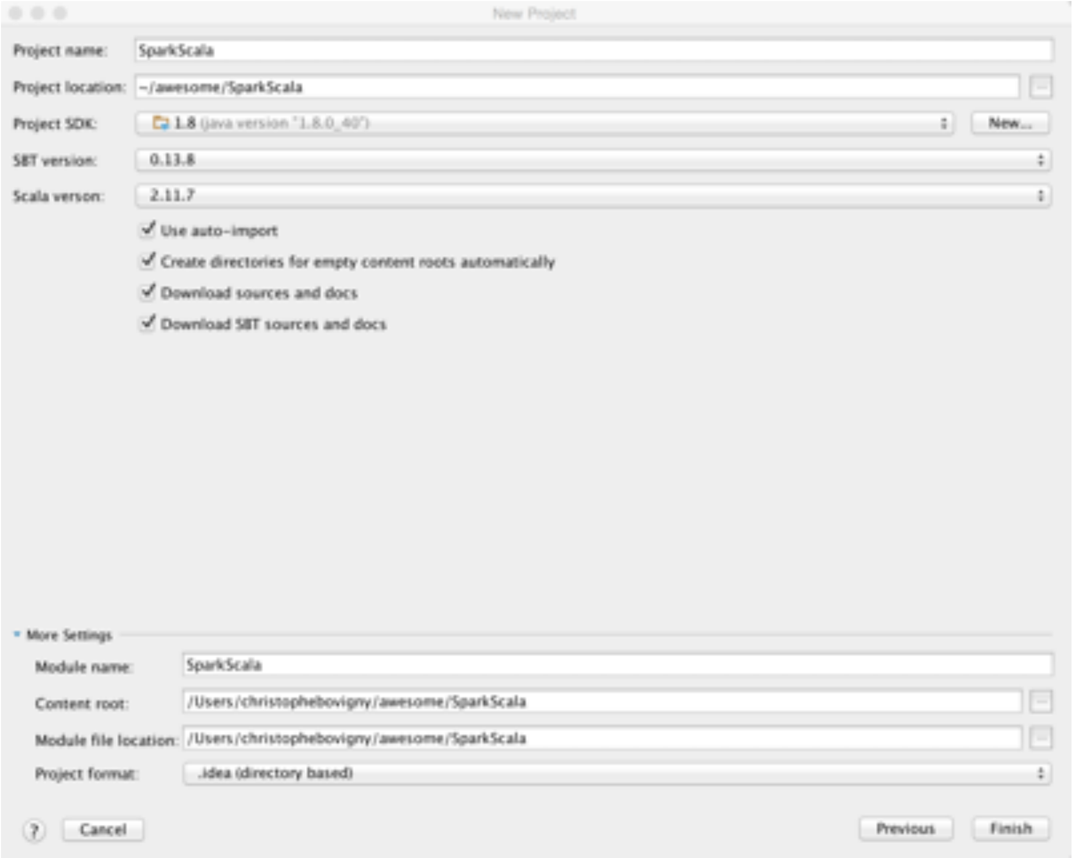
- **Define a `factorial` function that computes n! = 1 * 2 * … * n given input n. You can use either a loop or recursion, in our solution we use recursion (see steps 5-7 of First Steps to Scala). Then compute the sum of factorials in `myNumbers`.**

```
scala> def factorial(n:Int):Int = if (n==0) 1 else n * factorial(n-1) // From http://bit.ly/b2sVKI
factorial: (Int)Int
scala> myNumbers.map(factorial).sum
res: Int = 5193
```

Setup a new Project



SBT-based Scala project



Project name: SparkScala

Project location: ~/awesome/SparkScala

Project SDK: 1.8 (java version "1.8.0_40")

SBT version: 0.13.8

Scala verson: 2.11.7

☑ Use auto-import

☑ Create directories for empty content roots automatically

☑ Download sources and docs

☑ Download SBT sources and docs

More Settings

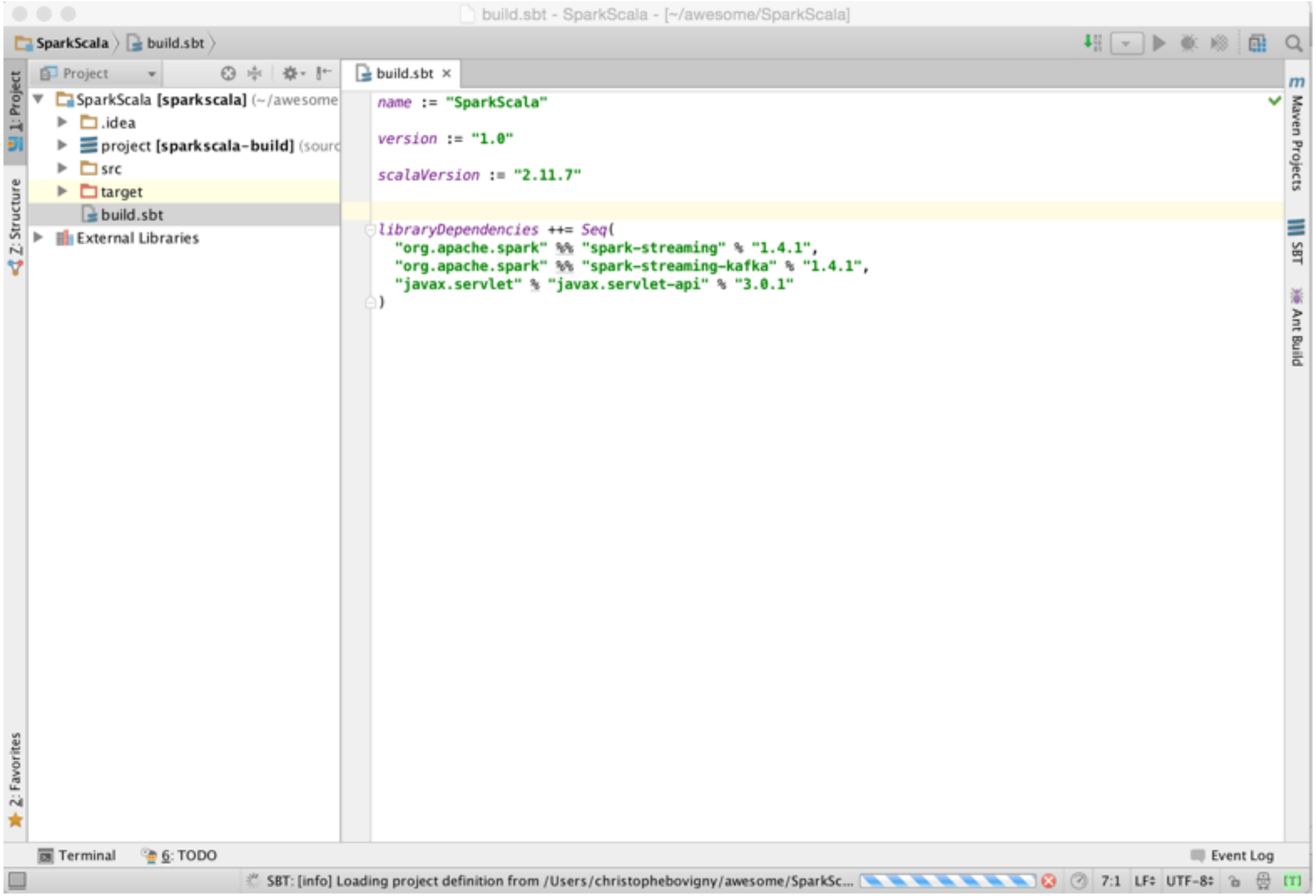
Module name: SparkScala

Content root: /Users/christophebovigny/awesome/SparkScala

Module file location: /Users/christophebovigny/awesome/SparkScala

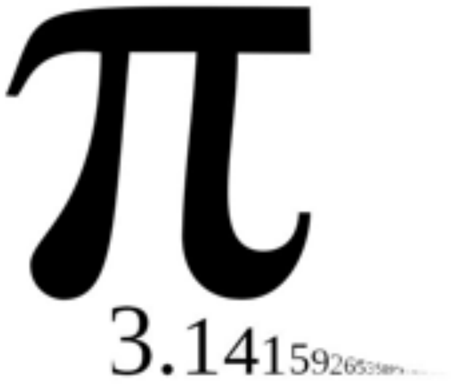
Project format: .idea (directory based)

Add dependencies

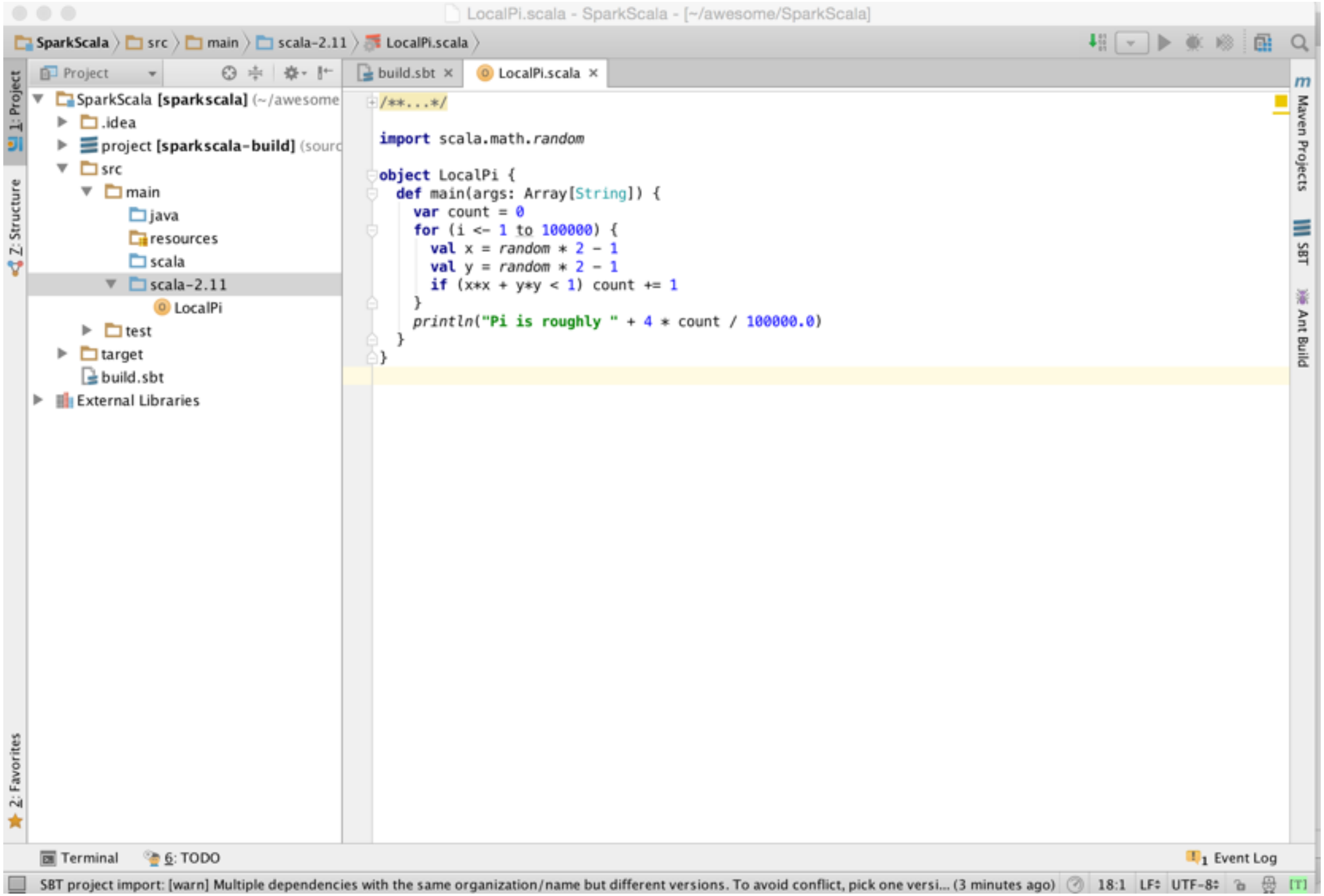


```
name := "SparkScala"

version := "1.0"

scalaVersion := "2.11.7"

libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-streaming" % "1.4.1",
  "org.apache.spark" %% "spark-streaming-kafka" % "1.4.1",
  "javax.servlet" % "javax.servlet-api" % "3.0.1"
)
```

```
SBT: [info] Loading project definition from /Users/christophebovigny/awesome/SparkSc...
```



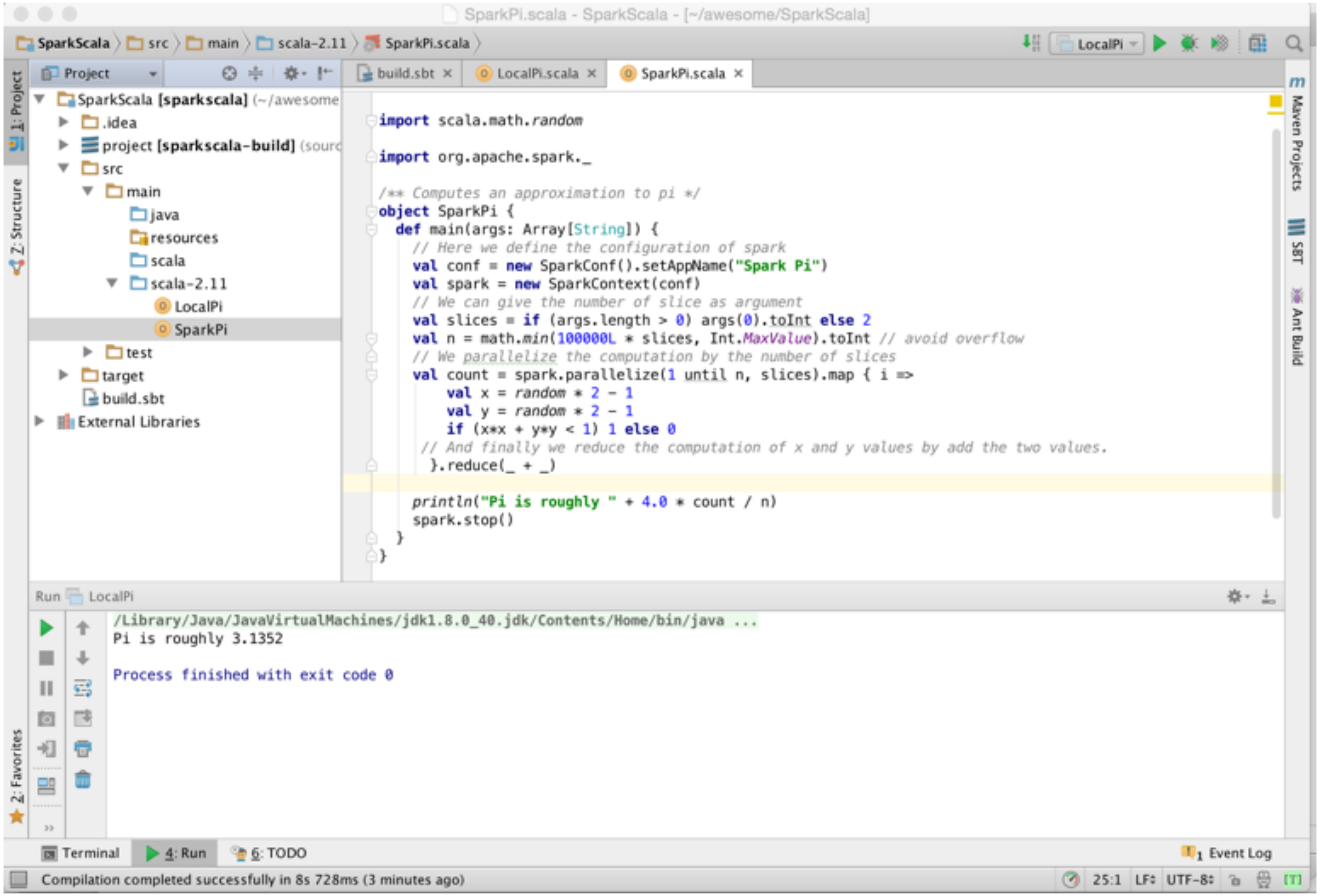
Pi / Local



```
/**...*/

import scala.math.random

object LocalPi {
  def main(args: Array[String]) {
    var count = 0
    for (i <- 1 to 100000) {
      val x = random * 2 - 1
      val y = random * 2 - 1
      if (x*x + y*y < 1) count += 1
    }
    println("Pi is roughly " + 4 * count / 100000.0)
  }
}
```

SBT project import: [warn] Multiple dependencies with the same organization/name but different versions. To avoid conflict, pick one versi... (3 minutes ago)

Pi / Spark

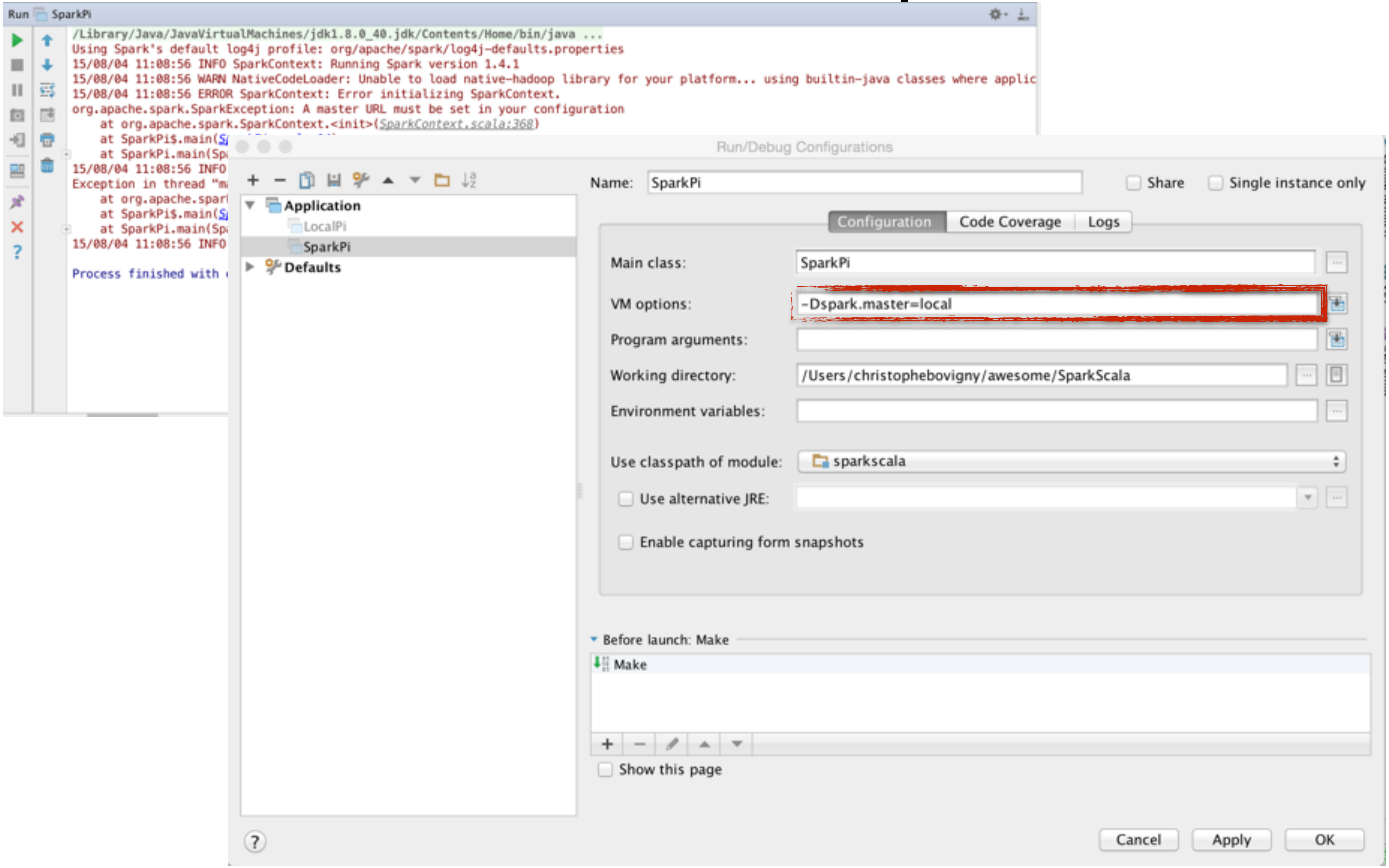


```scala
import scala.math.random

import org.apache.spark._

/** Computes an approximation to pi */
object SparkPi {
  def main(args: Array[String]) {
    // Here we define the configuration of spark
    val conf = new SparkConf().setAppName("Spark Pi")
    val spark = new SparkContext(conf)
    // We can give the number of slice as argument
    val slices = if (args.length > 0) args(0).toInt else 2
    val n = math.min(100000L * slices, Int.MaxValue).toInt // avoid overflow
    // We parallelize the computation by the number of slices
    val count = spark.parallelize(1 until n, slices).map { i =>
        val x = random * 2 - 1
        val y = random * 2 - 1
        if (x*x + y*y < 1) 1 else 0
     // And finally we reduce the computation of x and y values by add the two values.
      }.reduce(_ + _)

    println("Pi is roughly " + 4.0 * count / n)
    spark.stop()
  }
}
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_40.jdk/Contents/Home/bin/java ...
Pi is roughly 3.1352

Process finished with exit code 0
```

Compilation completed successfully in 8s 728ms (3 minutes ago)

Run Pi /Spark



```
/Library/Java/JavaVirtualMachines/jdk1.8.0_40.jdk/Contents/Home/bin/java ...
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
15/08/04 11:08:56 INFO SparkContext: Running Spark version 1.4.1
15/08/04 11:08:56 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
15/08/04 11:08:56 ERROR SparkContext: Error initializing SparkContext.
org.apache.spark.SparkException: A master URL must be set in your configuration
    at org.apache.spark.SparkContext.<init>(SparkContext.scala:368)
    at SparkPi$.main(S
    at SparkPi.main(Sp
15/08/04 11:08:56 INFO
Exception in thread "m
    at org.apache.spar
    at SparkPi$.main(S
    at SparkPi.main(Sp
15/08/04 11:08:56 INFO

Process finished with
```

Name: SparkPi

□ Share □ Single instance only

Main class: SparkPi

VM options: -Dspark.master=local

Program arguments:

Working directory: /Users/christophebovigny/awesome/SparkScala

Environment variables:

Use classpath of module: sparkscala

□ Use alternative JRE:

□ Enable capturing form snapshots

Before launch: Make

Make

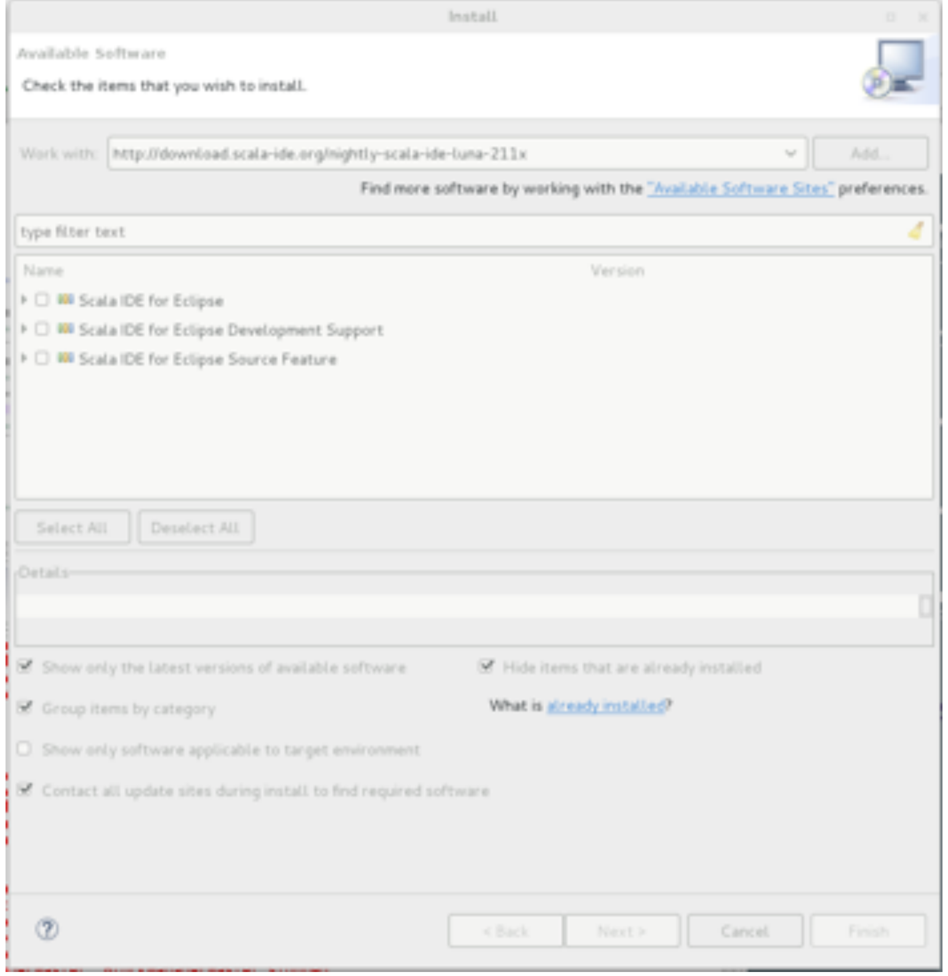
□ Show this page

Setup a new Project

- Install sbt for your machine

www.scala-sbt.org/release/tutorial/Setup.html

- In eclipse Help -> Install New Software



Available Software

Check the items that you wish to install.

Work with: http://download.scala-ide.org/nightly-scala-ide-luna-211x

Find more software by working with the "Available Software Sites" preferences.

- □ Scala IDE for Eclipse
- □ Scala IDE for Eclipse Development Support
- □ Scala IDE for Eclipse Source Feature

☑ Show only the latest versions of available software

☑ Hide items that are already installed

☑ Group items by category

What is already installed?

□ Show only software applicable to target environment

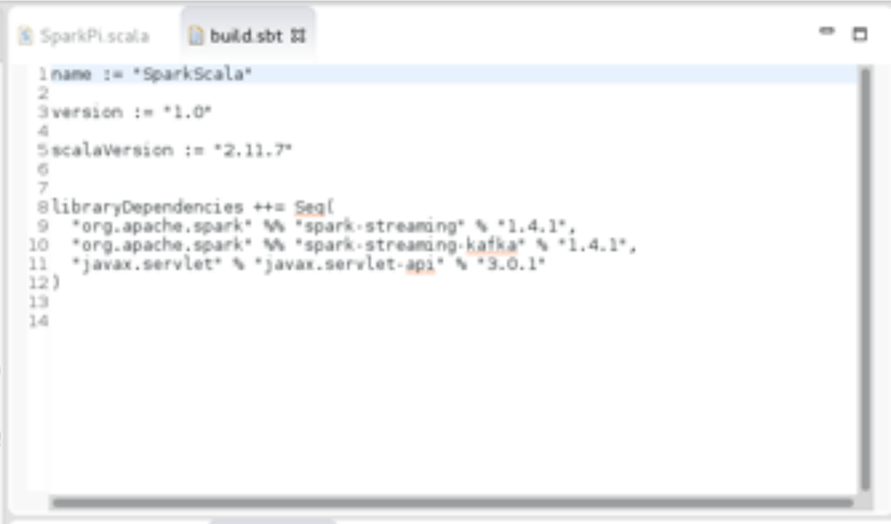
☑ Contact all update sites during install to find required software

Setup a new Project

- Create the project folder structure : *mkdir -p src/main/scala*

- Create a file in src/main/scala : *echo ‘object Hi { def main(args: Array[String]) = println(« Hi !»)’ > hw.scala*

- To allow managing dependencies, project name, Scala Version etc… create a file named *build.sbt* in the project root.



```
name := "SparkScala"

version := "1.0"

scalaVersion := "2.11.7"


libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-streaming" % "1.4.1",
  "org.apache.spark" %% "spark-streaming-kafka" % "1.4.1",
  "javax.servlet" % "javax.servlet-api" % "3.0.1"
)
```

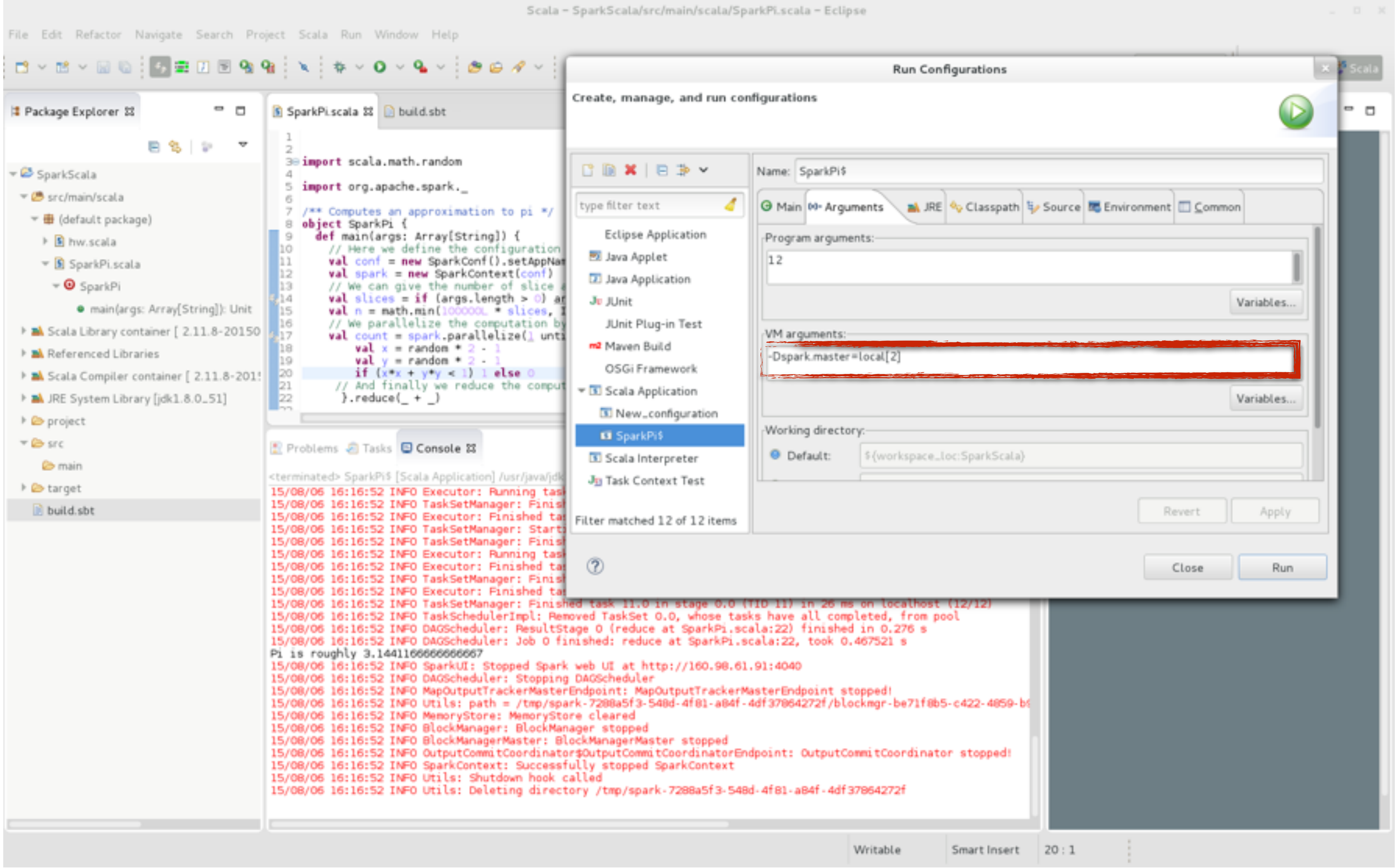
Setup a new Project

- Open $ ~/.sbt/0.13/plugins/build.sbt
  *addSbtPlugin(«com.typesafe.sbteclipse » %
  « sbteclipse-plugin » % 2.2.0)*

- In the project root folder : *sbt eclipse with-source=true*

- File->Import-> Existing Projects into Workspace

Run Spark Project



```
import scala.math.random

import org.apache.spark._

/** Computes an approximation to pi */
object SparkPi {
  def main(args: Array[String]) {
    // Here we define the configuration
    val conf = new SparkConf().setAppNam
    val spark = new SparkContext(conf)
    // We can give the number of slice
    val slices = if (args.length > 0) ar
    val n = math.min(100000L * slices, I
    // We parallelize the computation by
    val count = spark.parallelize(1 unti
        val x = random * 2 - 1
        val y = random * 2 - 1
        if (x*x + y*y < 1) 1 else 0
     // And finally we reduce the comput
      }.reduce(_ + _)
```

Name: SparkPi$

Program arguments: 12

VM arguments: -Dspark.master=local[2]

Working directory:
● Default: ${workspace_loc:SparkScala}

```
<terminated> SparkPi$ [Scala Application] /usr/java/jdk
15/08/06 16:16:52 INFO Executor: Running tas
15/08/06 16:16:52 INFO TaskSetManager: Finis
15/08/06 16:16:52 INFO Executor: Finished ta
15/08/06 16:16:52 INFO TaskSetManager: Start
15/08/06 16:16:52 INFO TaskSetManager: Finis
15/08/06 16:16:52 INFO Executor: Running tas
15/08/06 16:16:52 INFO Executor: Finished ta
15/08/06 16:16:52 INFO TaskSetManager: Finis
15/08/06 16:16:52 INFO Executor: Finished ta
15/08/06 16:16:52 INFO TaskSetManager: Finished task 11.0 in stage 0.0 (TID 11) in 26 ms on localhost (12/12)
15/08/06 16:16:52 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
15/08/06 16:16:52 INFO DAGScheduler: ResultStage 0 (reduce at SparkPi.scala:22) finished in 0.276 s
15/08/06 16:16:52 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.scala:22, took 0.467521 s
Pi is roughly 3.1441166666666667
15/08/06 16:16:52 INFO SparkUI: Stopped Spark web UI at http://160.98.61.91:4040
15/08/06 16:16:52 INFO DAGScheduler: Stopping DAGScheduler
15/08/06 16:16:52 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
15/08/06 16:16:52 INFO Utils: path = /tmp/spark-7288a5f3-548d-4f81-a84f-4df37864272f/blockmgr-be71f8b5-c422-4859-b
15/08/06 16:16:52 INFO MemoryStore: MemoryStore cleared
15/08/06 16:16:52 INFO BlockManager: BlockManager stopped
15/08/06 16:16:52 INFO BlockManagerMaster: BlockManagerMaster stopped
15/08/06 16:16:52 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
15/08/06 16:16:52 INFO SparkContext: Successfully stopped SparkContext
15/08/06 16:16:52 INFO Utils: Shutdown hook called
15/08/06 16:16:52 INFO Utils: Deleting directory /tmp/spark-7288a5f3-548d-4f81-a84f-4df37864272f
```

Scala Spark Examples

- https://github.com/apache/spark/tree/master/examples/src/main/scala/org/apache/spark/examples
- http://spark.apache.org