

# Java-Success.com

Prepare to fast-track, choose & go places with 800+ Java & Big Data Q&As with lots of code & diagrams.

[Home](#) [Why? ▾](#) [300+ Java FAQs ▾](#) [300+ Big Data FAQs ▾](#) [Courses ▾](#)[👤 Membership ▾](#) [Your Career ▾](#)[Home](#) › [bigdata-success.com](#) › [Tutorials - Big Data](#) › [TUT - Starting Spark & Scala](#) ›

07: Getting started with Spark-submit

## 07: Getting started with Spark-submit

 Posted on [November 9, 2018](#)

This extends the tutorial [Setting up & getting started with Spark local mode with Sbt & Scala](#) and [Setting up Spark standalone on Mac](#).

Spark-submit is used to submit a Spark job. In this tutorial we will be using the “local” master in “client” mode. When you have a Hadoop installation with multiple nodes managed by the “YARN” resource manager, you can run it on “yarn” in “cluster” mode.

**Step 1:** Add the Spark dependency to “**build.sbt**” file.

```
1 organization := "com.sbt-tut"
```

### 300+ Java Interview FAQs

300+ Java FAQs



16+ Java Key Areas Q&amp;As



150+ Java Architect FAQs



80+ Java Code Quality Q&amp;As



150+ Java Coding Q&amp;As



### 300+ Big Data Interview FAQs

300+ Big Data FAQs



Tutorials - Big Data

TUT -  Starting Big Data

TUT - Starting Spark &amp; Scala

```
2
3 version:= "0.1"
4
5 scalaVersion := "2.11.12"
6
7 libraryDependencies += "org.apache.spark" %% "spark-core" %>
8
```

**Step 2:** From the command-line run the following command, and it may take some time to download the dependencies.

```
1 ~/projects/sbt-tutorial]$ sbt eclipse
2
```

**Step 3:** Write a very simple Spark code in Scala as shown below within a package named “com.scalaproject” and the Scala file is named “SimpleSpark.scala”.

```
1 package com.scalaproject
2
3 import org.apache.spark.sql.SparkSession;
4 import org.apache.spark.sql.Dataset;
5
6 object SimpleSpark {
7
8     def main(args: Array[String]): Unit = {
9         val spark = SparkSession.builder.appName("SimpleSpark")
10            .config("spark.master", "local")
11            .getOrCreate()
12
13         val list = List(1,2,3,4,5)
14         val rdd = spark.sparkContext.parallelize(list)
15
16         val multipleBy2Rdd = rdd.map(_ * 2)
17         println(multipleBy2Rdd.collect().toList)
18     }
19 }
20
21
```

**Step 4:** You can package it as a jar file in the command-line.

TUT - Starting with Python

TUT - Kafka

TUT - Pig

TUT - Apache Storm

TUT - Spark Scala on Zeppelin

TUT - Cloudera

TUT - Cloudera on Docker

TUT - File Formats

TUT - Spark on Docker

TUT - Flume

TUT - Hadoop (HDFS)

TUT - HBase (NoSQL)

TUT - Hive (SQL)

TUT - Hadoop & Spark

TUT - MapReduce

TUT - Spark and Scala

TUT - Spark & Java

TUT - PySpark on Databricks

TUT - Zookeeper

## 800+ Java Interview Q&As

300+ Core Java Q&As



300+ Enterprise Java Q&As



150+ Java Frameworks Q&As



120+ Companion Tech Q&As



Tutorials - Enterprise Java



```
1 ~/projects/sbt-tutorial]$ sbt package
2
```

projects/sbt-tutorial/target/scala-2.11/sbt-tutorial\_2.11-0.1.jar will be built.

**Step 5:** On a command-line run the spark-submit command.

```
1 ~/projects/sbt-tutorial]$ spark-submit --class com
2
```

## Output:

```
1 ....2018-11-09 23:01:35 INFO TaskSetManager:54 -
2 2018-11-09 23:01:35 INFO TaskSchedulerImpl:54 -
3 2018-11-09 23:01:35 INFO DAGScheduler:54 - Result
4 2018-11-09 23:01:35 INFO DAGScheduler:54 - Job 0
5 List(2, 4, 6, 8, 10)
6 2018-11-09 23:01:35 INFO SparkContext:54 - Invok
7 2018-11-09 23:01:35 INFO AbstractConnector:318 -
8 2018-11-09 23:01:35 INFO SparkUI:54 - Stopped Sp
9 2018-11-09 23:01:35 INFO MapOutputTrackerMasterE
10 .....
11
```

## The .bash\_profile

This assumes that Hadoop has been installed as per one of the other tutorials so that you can do \$(hadoop classpath). Also, when you install it with “brew install” as opposed downloading the \*.tar.gz and unzipping it.

```
1 //.....
2 export SPARK_HOME=/usr/local/Cellar/apache-spark/2
3 export SPARK_DIST_CLASSPATH=$(hadoop classpath)
4 export PATH=$SPARK_HOME/bin:${PATH}
5 //...
```

```
6
```

Activate the change:

```
1 $ source ~/.bash_profile
```

## Starting the Spark master

```
1 sudo spark-class org.apache.spark.deploy.master.Ma
2
```

Note down the **host:port** where the Spark master is listening via “http://localhost:8080”.

## Spark worker joins the Spark master

In a separate shell window start the worker node, specify the master host:location to join.

```
1 sudo spark-class org.apache.spark.deploy.worker.Wor
2
```

You can submit a job as shown below. If you don't specify the URL of the master, it defaults to “local[\*]”.

```
1
2 $ spark-submit --master spark://192.168.0.18:7077
3
```

◀ 06: Setting up Spark-shell on Mac & getting started

08. Setting up & getting started with Hadoop on Mac ▶

## Disclaimer

The contents in this Java-Success are copyrighted and from EmpoweringTech pty ltd. The EmpoweringTech pty ltd has the right to correct or enhance the current content without any prior notice. These are general advice only, and one needs to take his/her own circumstances into consideration. The EmpoweringTech pty ltd will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. Links to external sites do not imply endorsement of the linked-to sites. [Privacy Policy](#).

© 2022 [java-success.com](https://www.java-success.com)