Spark Introduction

course Esilv 2024

arnaud.nauwynck@gmail.com

this document:

https://github.com/Arnaud-Nauwynck/presentations/
/pres-bigdata/9-spark-intro

Spark (Recent) History & Ancestors

MapReduce @Google

2002
@Google

2004 Google
Paper published

2014 Google
No more used of MapReduce

hadoop

2006 @Yahoo
Hadoop implementation

2008 Apache Open-Source

2012 Yarn (v2)

2021 MapReduce bashing
... HDFS & Hadoop also
HortonWork bought by Cloudera
HDInsight @Azure…very bad choice
.. To be abandonned

MPI
1995 Message Passing Interface

APACHE Spark™

2010 Spark paper

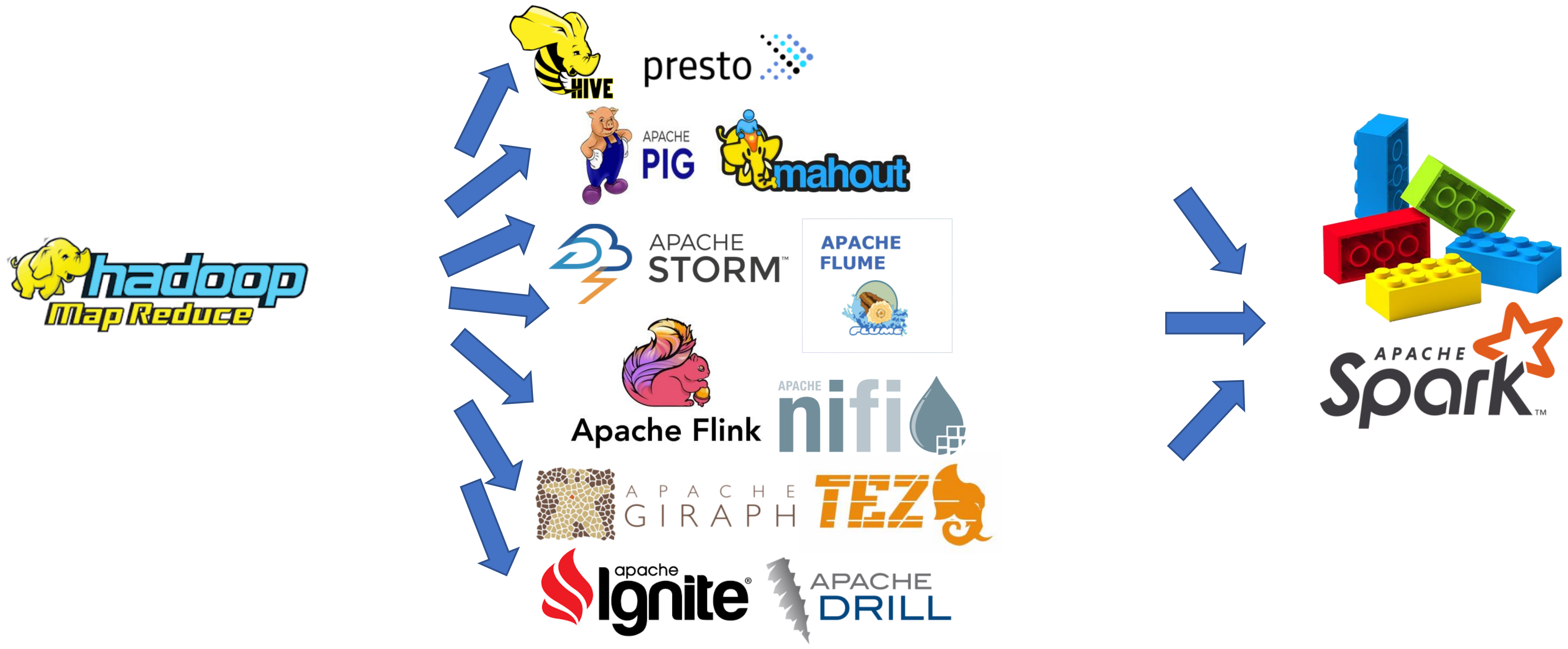
2013 Apache top-level

??

2015
Kubernetes

2020
Spark on K8s

Simple => Many Specific Systems => Unified

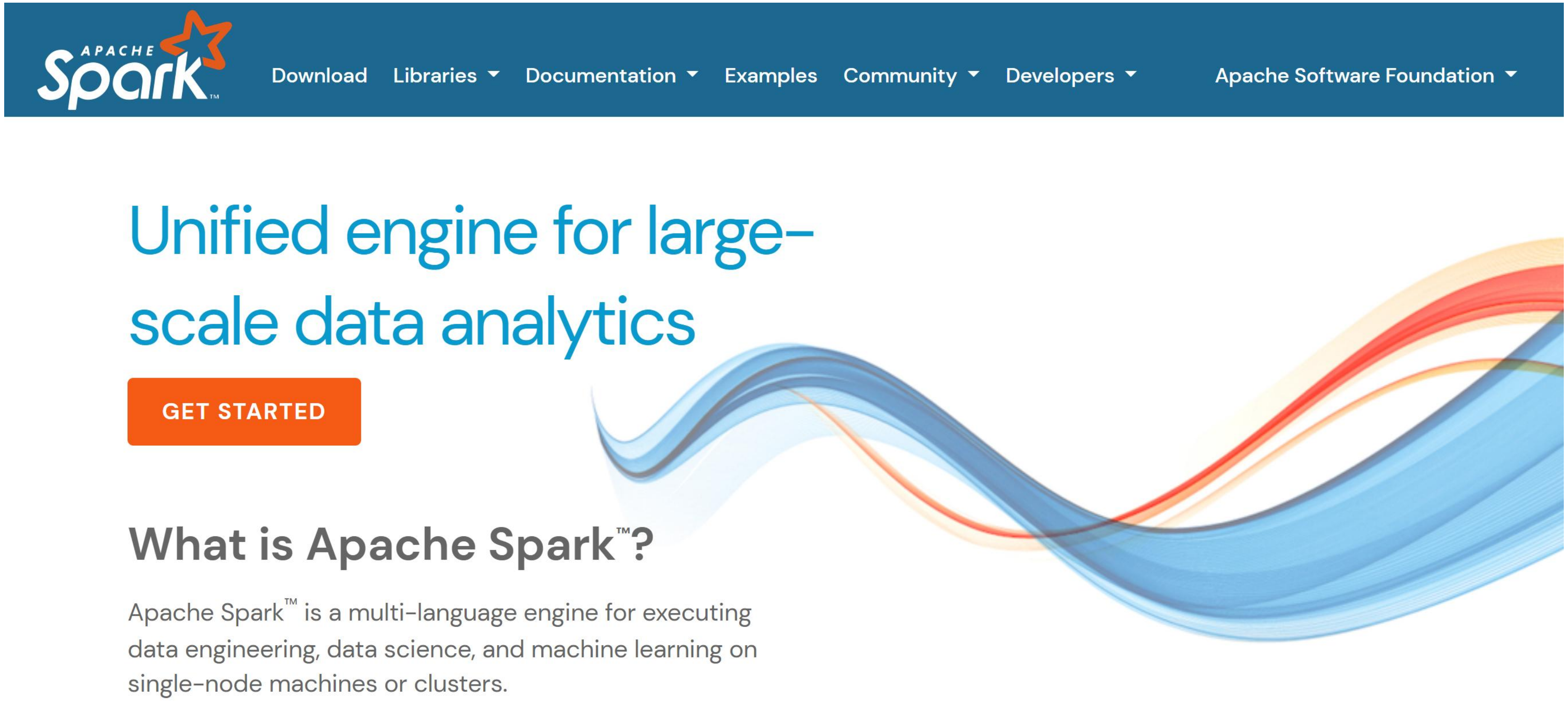


« **Simple** » ecosystem
(verbose ineficient &
complex java code)

« **Bazard** » ecosystem
(**Too MANY TOO** SPECIFIC
redundant, complexes)

"Unified" ecosystem
Simple
+ extensible modules

Spark = « **Unified Engine** »

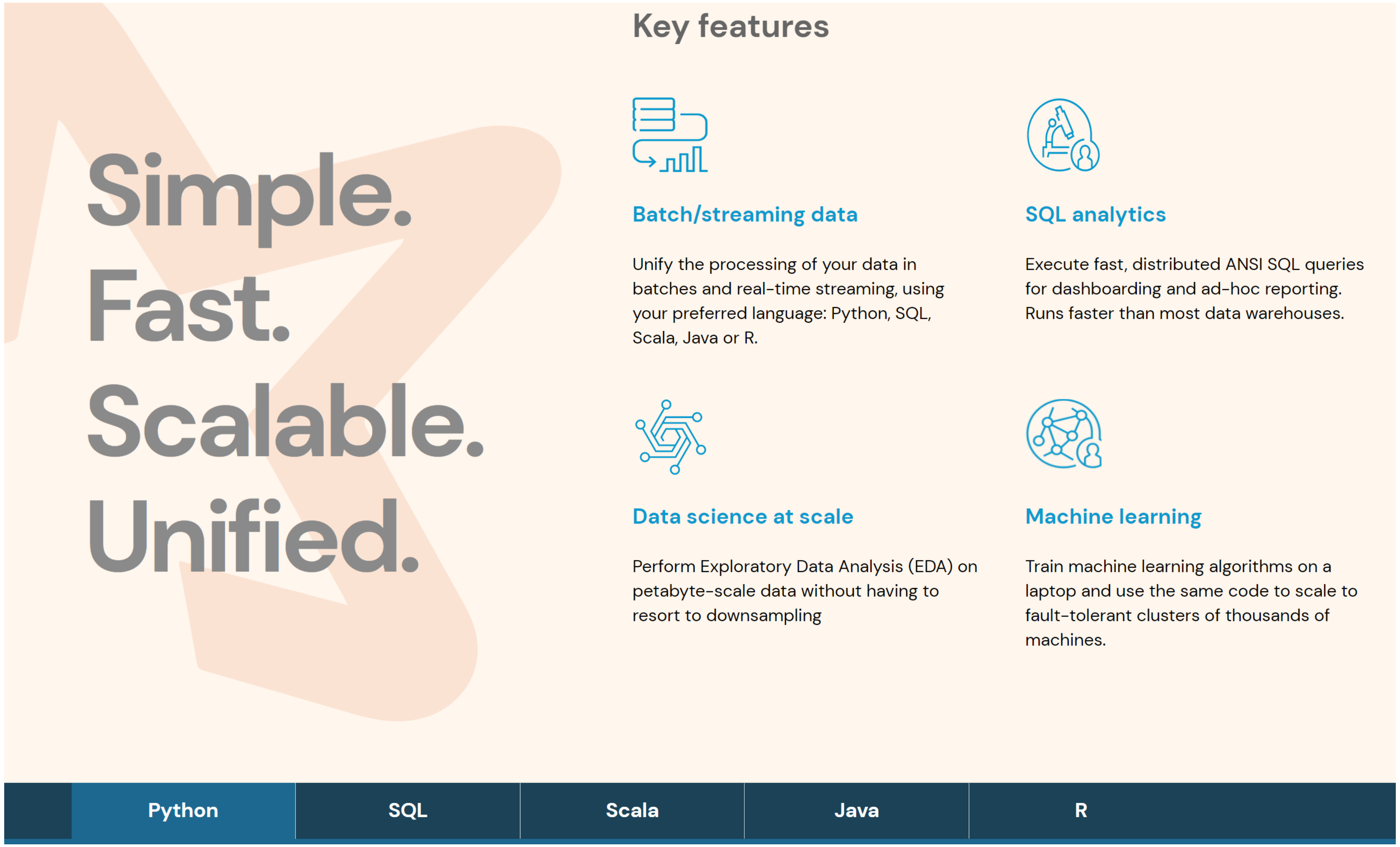


Unified engine for large-scale data analytics

What is Apache Spark™?

Apache Spark™ is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters.

Multi Purposes – Multi Langages



Key features

Simple.
Fast.
Scalable.
Unified.

Batch/streaming data

Unify the processing of your data in batches and real-time streaming, using your preferred language: Python, SQL, Scala, Java or R.

SQL analytics

Execute fast, distributed ANSI SQL queries for dashboarding and ad-hoc reporting. Runs faster than most data warehouses.

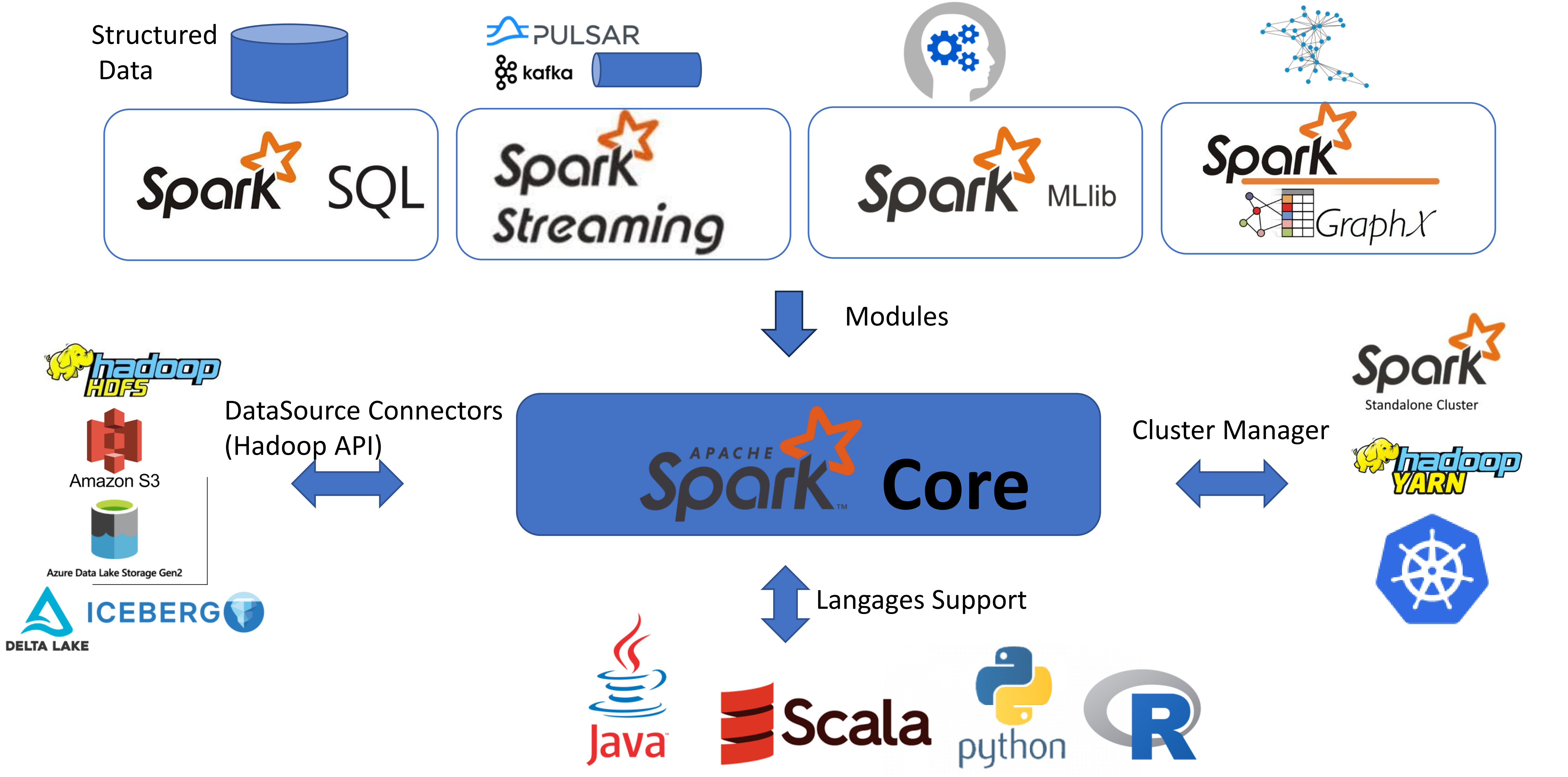
Data science at scale

Perform Exploratory Data Analysis (EDA) on petabyte-scale data without having to resort to downsampling

Machine learning

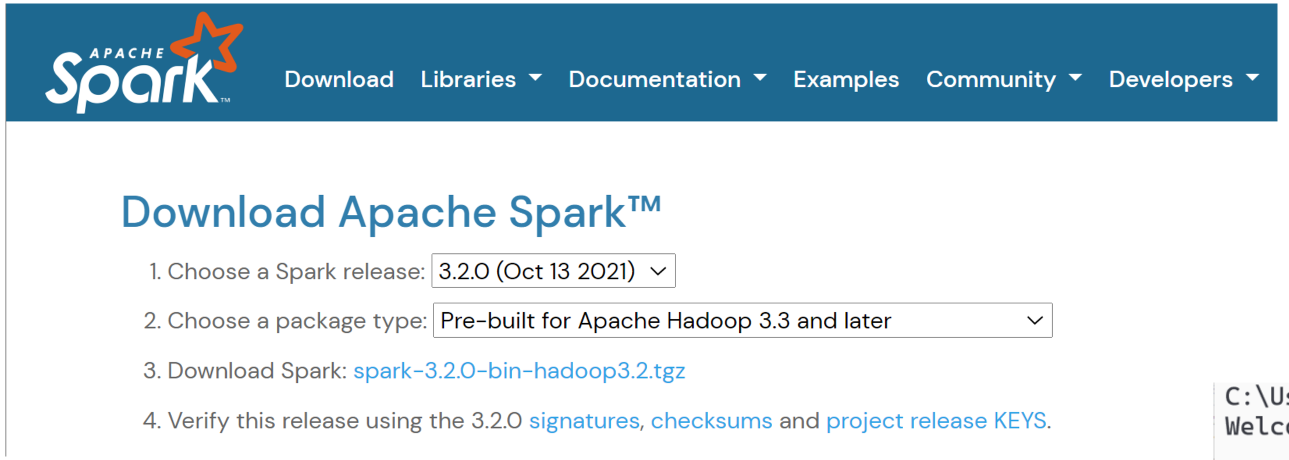
Train machine learning algorithms on a laptop and use the same code to scale to fault-tolerant clusters of thousands of machines.

Spark-Core + ...


Structured Data
PULSAR
kafka
Spark SQL
Spark Streaming
Spark MLlib
Spark GraphX
Modules
hadoop HDFS
Amazon S3
Azure Data Lake Storage Gen2
ICEBERG
DELTA LAKE
DataSource Connectors (Hadoop API)
APACHE Spark Core
Cluster Manager
Spark Standalone Cluster
hadoop YARN
Langages Support
Java
Scala
python
R

Getting Started

1/ Download



Download Apache Spark™

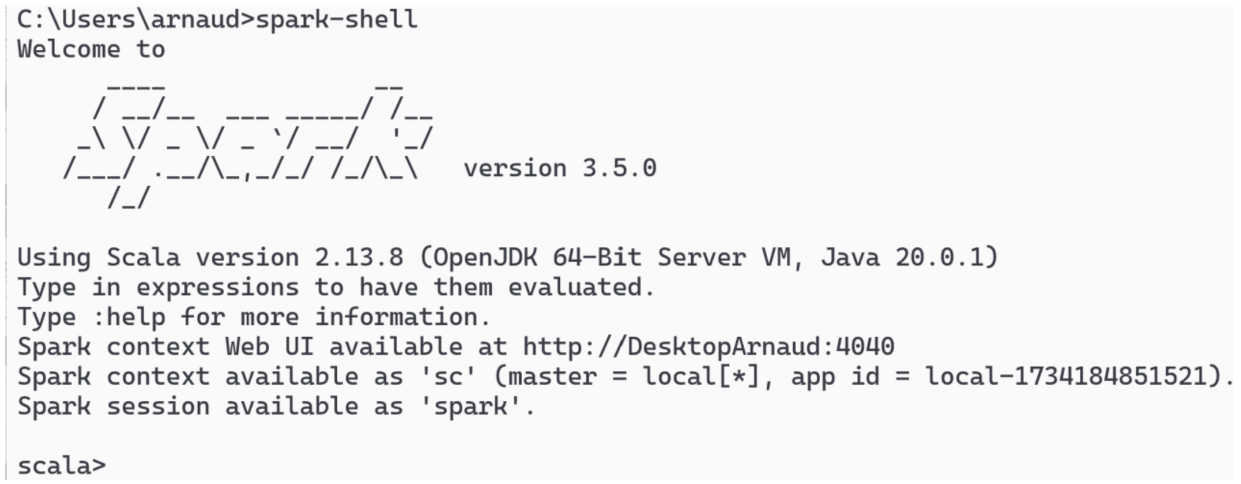
1. Choose a Spark release: 3.2.0 (Oct 13 2021)
2. Choose a package type: Pre-built for Apache Hadoop 3.3 and later
3. Download Spark: spark-3.2.0-bin-hadoop3.2.tgz
4. Verify this release using the 3.2.0 signatures, checksums and project release KEYS.

2/ Unzip + add to PATH

3/ launch

C:> bin\spark-shell

(or spark-submit, or spark-sql)

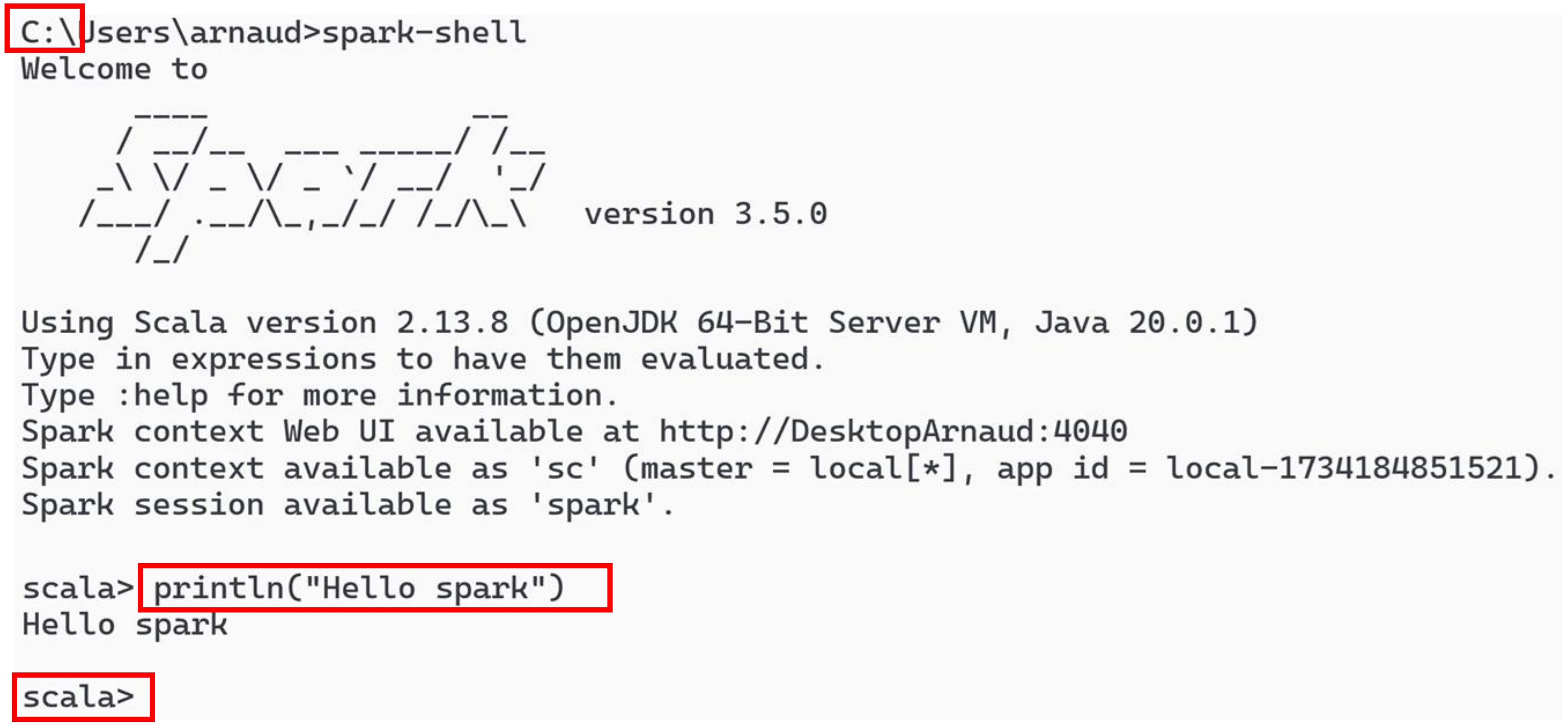


```
C:\Users\arnaud>spark-shell
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.0
      /_/

Using Scala version 2.13.8 (OpenJDK 64-Bit Server VM, Java 20.0.1)
Type in expressions to have them evaluated.
Type :help for more information.
Spark context Web UI available at http://DesktopArnaud:4040
Spark context available as 'sc' (master = local[*], app id = local-1734184851521).
Spark session available as 'spark'.

scala>
```

scala>
scala> println(« Hello spark »);

launch spark-shell
from terminal

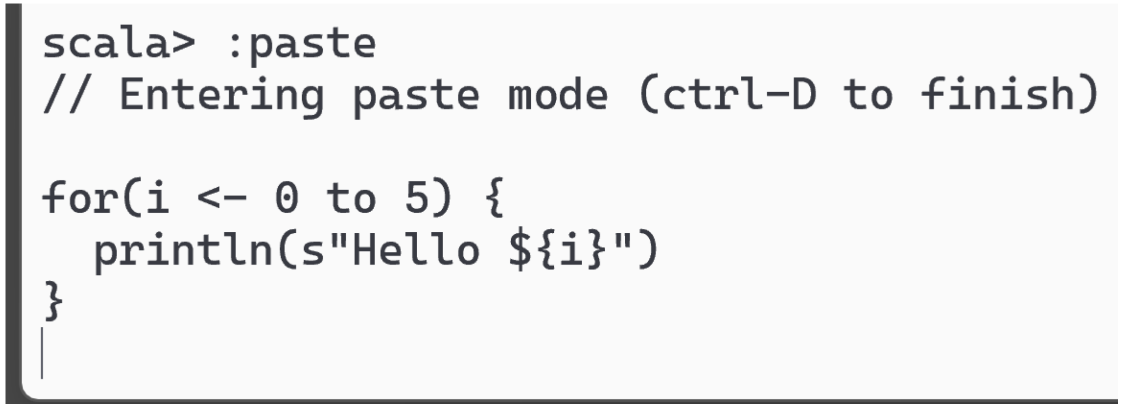
inside spark
type scala code



```
C:\Users\arnaud>spark-shell
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.0
      /_/

Using Scala version 2.13.8 (OpenJDK 64-Bit Server VM, Java 20.0.1)
Type in expressions to have them evaluated.
Type :help for more information.
Spark context Web UI available at http://DesktopArnaud:4040
Spark context available as 'sc' (master = local[*], app id = local-1734184851521).
Spark session available as 'spark'.

scala> println("Hello spark")
Hello spark

scala>
```

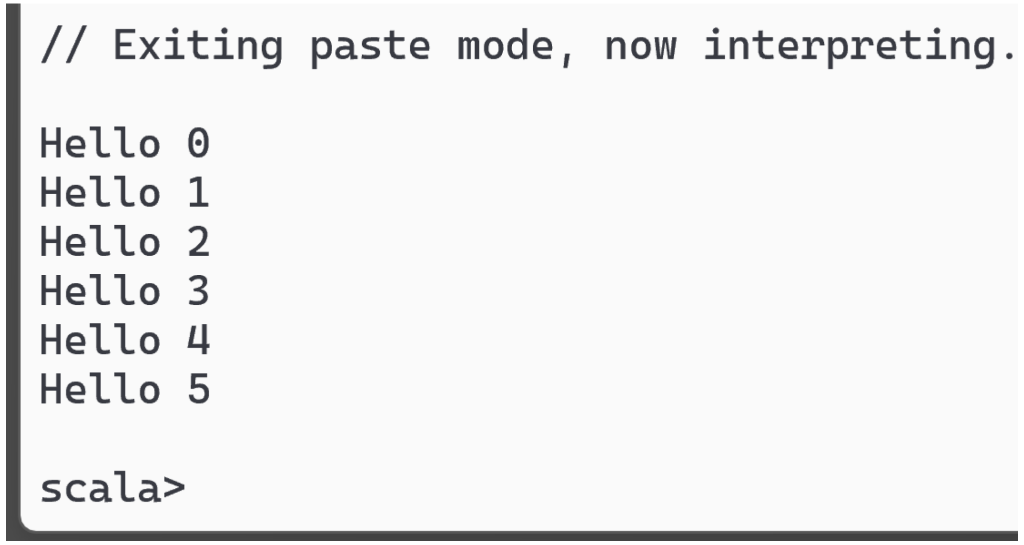
Spark-shell>   SCALA code

:paste



```
scala> :paste
// Entering paste mode (ctrl-D to finish)

for(i <- 0 to 5) {
  println(s"Hello ${i}")
}
```

Ctrl-D

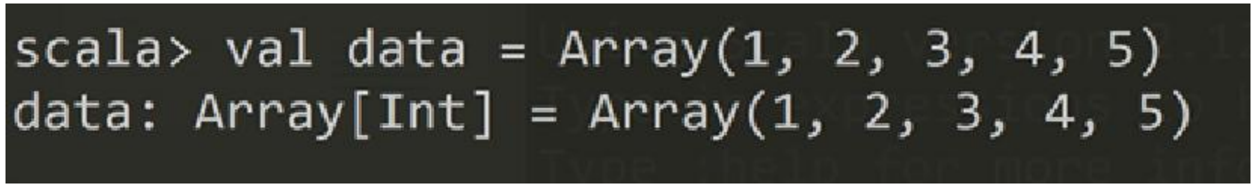


```
// Exiting paste mode, now interpreting.

Hello 0
Hello 1
Hello 2
Hello 3
Hello 4
Hello 5

scala>
```

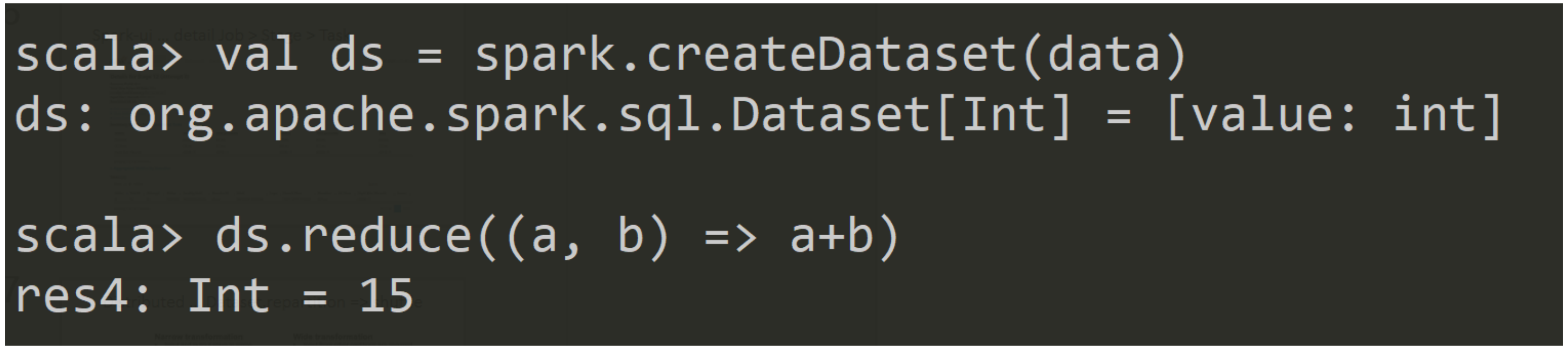
spark-shell> help

```
scala> :help
All commands can be abbreviated, e.g., :he instead of :help.
:help [command]          print this summary or command-specific help
:completions <string>    output completions for the given string
:imports [name name ...] show import history, identifying sources of names
:implicits [-v]          show the implicits in scope
:javap <path|class>      disassemble a file or class name
:line <id>|<line>        place line(s) at the end of history
:load <path>             interpret lines in a file
:paste [-raw] [path]     enter paste mode or paste a file
:power                   enable power user mode
:quit                    exit the REPL
:replay [options]        reset the REPL and replay all previous commands
:require <path>          add a jar to the classpath
:reset [options]         reset the REPL to its initial state, forgetting all session entries
:save <path>             save replayable session to a file
:sh <command line>       run a shell command (result is implicitly => List[String])
:settings <options>      update compiler options, if possible; see reset
:silent                  disable/enable automatic printing of results
:type [-v] <expr>        display the type of an expression without evaluating it
:kind [-v] <type>        display the kind of a type. see also :help kind
:warnings                show the suppressed warnings from the most recent line which had any

Useful default key bindings:
  TAB           code completion
  CTRL-ALT-T    show type at cursor, hit again to show code with types/implicits inferred.

scala>
```

ds = spark.createDataSet(..)



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



```
scala> val ds = spark.createDataset(data)
ds: org.apache.spark.sql.Dataset[Int] = [value: int]

scala> ds.reduce((a, b) => a+b)
res4: Int = 15
```

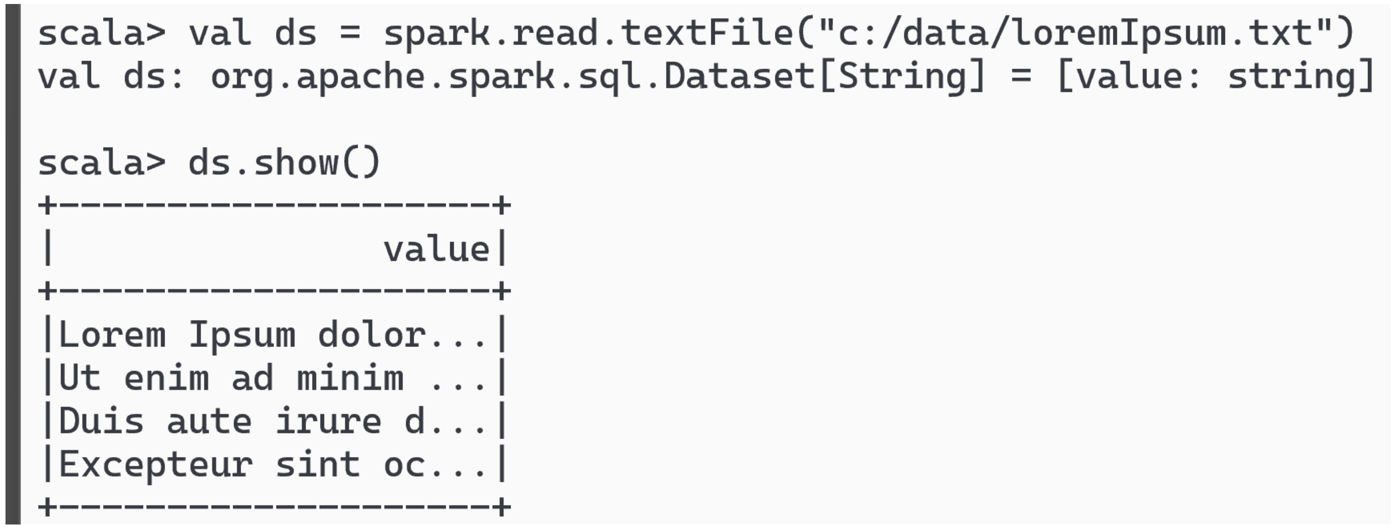
ds = spark.read.textFile(..)

loremIpsum.txt

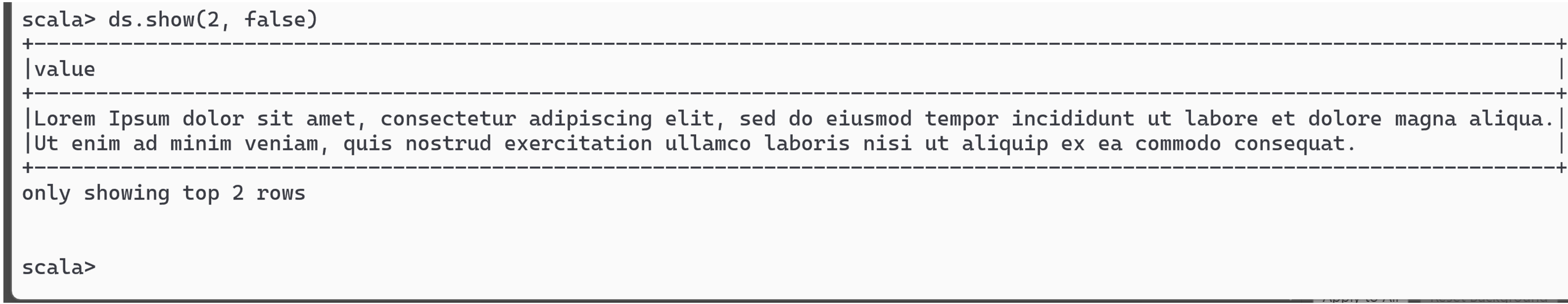
```
Lorem Ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
```

```
scala> val ds=spark.read.textFile("c:/data/loremIpsum.txt")
ds: org.apache.spark.sql.Dataset[String] = [value: string]

scala> ds.count  // count lines
res1: Long = 4
```

dataset.show ()
default   show(20 /*line*/, true /*truncate*/)



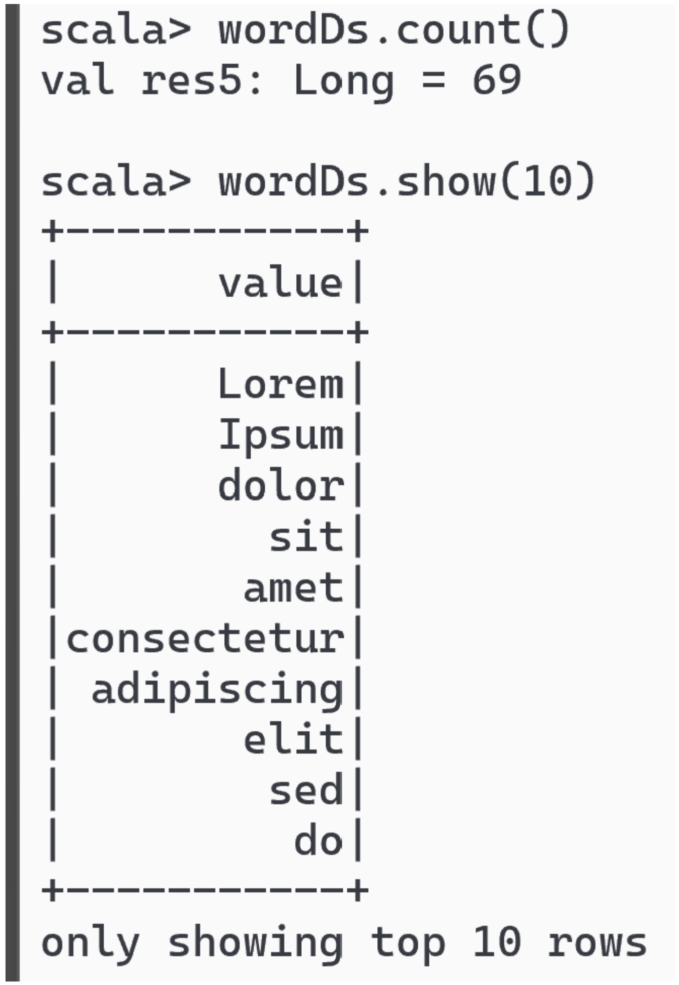
```
scala> val ds = spark.read.textFile("c:/data/loremIpsum.txt")
val ds: org.apache.spark.sql.Dataset[String] = [value: string]

scala> ds.show()
+--------------------+
|               value|
+--------------------+
|Lorem Ipsum dolor...|
|Ut enim ad minim ...|
|Duis aute irure d...|
|Excepteur sint oc...|
+--------------------+
```



```
scala> ds.show(2, false)
+---------------------------------------------------------------------------------------------------------------------------+
|value                                                                                                                      |
+---------------------------------------------------------------------------------------------------------------------------+
|Lorem Ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.|
|Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.                |
+---------------------------------------------------------------------------------------------------------------------------+
only showing top 2 rows


scala>
```

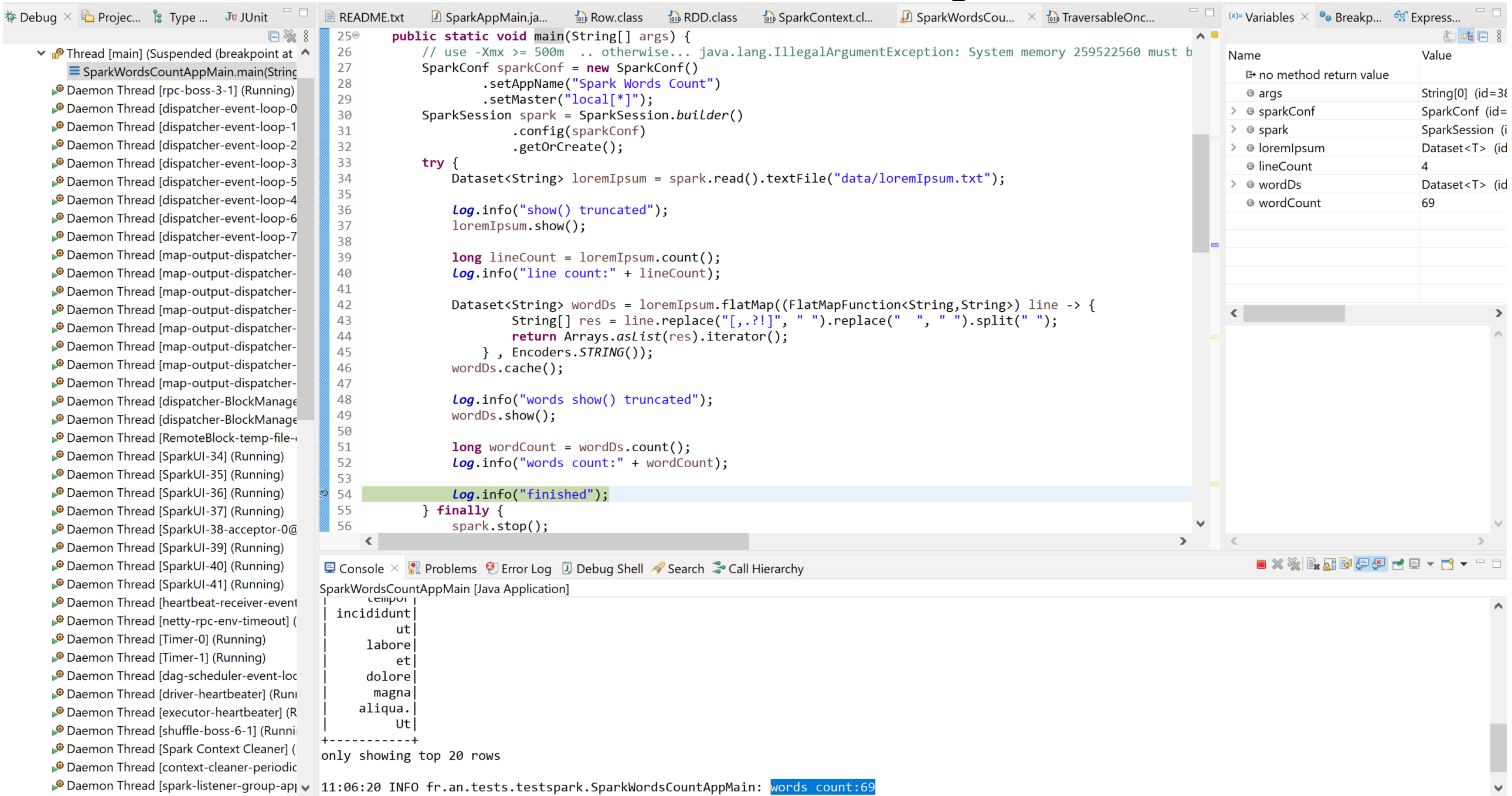
Words Count : flatMap(...).count

```
scala> val wordDs = ds.flatMap(line =>
        line.replaceAll("[,;.:!?]", " ")
                .replaceAll("  ", " ")
                .split(" ")
        )
words: .. Dataset[String] = [value: string]

scala> wordDs.count
res3: Long = 69

scala> wordDs.show(10, false)
```



```
scala> wordDs.count()
val res5: Long = 69

scala> wordDs.show(10)
+-----------+
|      value|
+-----------+
|      Lorem|
|      Ipsum|
|      dolor|
|        sit|
|       amet|
|consectetur|
| adipiscing|
|       elit|
|        sed|
|         do|
+-----------+
only showing top 10 rows
```

Words Count ... local[*] Debug in Java IDE



Thread [main] (Suspended (breakpoint at
SparkWordsCountAppMain.main(String
Daemon Thread [rpc-boss-3-1] (Running)
Daemon Thread [dispatcher-event-loop-0
Daemon Thread [dispatcher-event-loop-1
Daemon Thread [dispatcher-event-loop-2
Daemon Thread [dispatcher-event-loop-3
Daemon Thread [dispatcher-event-loop-5
Daemon Thread [dispatcher-event-loop-4
Daemon Thread [dispatcher-event-loop-6
Daemon Thread [dispatcher-event-loop-7
Daemon Thread [map-output-dispatcher-
Daemon Thread [map-output-dispatcher-
Daemon Thread [map-output-dispatcher-
Daemon Thread [map-output-dispatcher-
Daemon Thread [map-output-dispatcher-
Daemon Thread [map-output-dispatcher-
Daemon Thread [map-output-dispatcher-
Daemon Thread [map-output-dispatcher-
Daemon Thread [dispatcher-BlockManage
Daemon Thread [dispatcher-BlockManage
Daemon Thread [RemoteBlock-temp-file-
Daemon Thread [SparkUI-34] (Running)
Daemon Thread [SparkUI-35] (Running)
Daemon Thread [SparkUI-36] (Running)
Daemon Thread [SparkUI-37] (Running)
Daemon Thread [SparkUI-38-acceptor-0@
Daemon Thread [SparkUI-39] (Running)
Daemon Thread [SparkUI-40] (Running)
Daemon Thread [SparkUI-41] (Running)
Daemon Thread [heartbeat-receiver-event
Daemon Thread [netty-rpc-env-timeout] (
Daemon Thread [Timer-0] (Running)
Daemon Thread [Timer-1] (Running)
Daemon Thread [dag-scheduler-event-loo
Daemon Thread [driver-heartbeater] (Runi
Daemon Thread [executor-heartbeater] (R
Daemon Thread [shuffle-boss-6-1] (Runni
Daemon Thread [Spark Context Cleaner] (
Daemon Thread [context-cleaner-periodic
Daemon Thread [spark-listener-group-ap

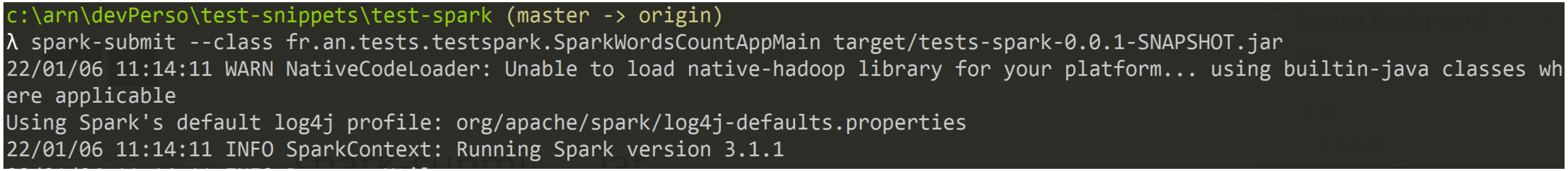
```java
    public static void main(String[] args) {
        // use -Xmx >= 500m  .. otherwise... java.lang.IllegalArgumentException: System memory 259522560 must b
        SparkConf sparkConf = new SparkConf()
                .setAppName("Spark Words Count")
                .setMaster("local[*]");
        SparkSession spark = SparkSession.builder()
                    .config(sparkConf)
                    .getOrCreate();
        try {
            Dataset<String> loremIpsum = spark.read().textFile("data/loremIpsum.txt");

            Log.info("show() truncated");
            loremIpsum.show();

            long lineCount = loremIpsum.count();
            Log.info("line count:" + lineCount);

            Dataset<String> wordDs = loremIpsum.flatMap((FlatMapFunction<String,String>) line -> {
                    String[] res = line.replace("[,.?!]", " ").replace("  ", " ").split(" ");
                    return Arrays.asList(res).iterator();
                } , Encoders.STRING());
            wordDs.cache();

            Log.info("words show() truncated");
            wordDs.show();

            long wordCount = wordDs.count();
            Log.info("words count:" + wordCount);

            Log.info("finished");
        } finally {
            spark.stop();
```

Name	Value
no method return value	
args	String[0] (id=3
sparkConf	SparkConf (id=
spark	SparkSession (i
loremIpsum	Dataset<T> (id
lineCount	4
wordDs	Dataset<T> (id
wordCount	69

SparkWordsCountAppMain [Java Application]

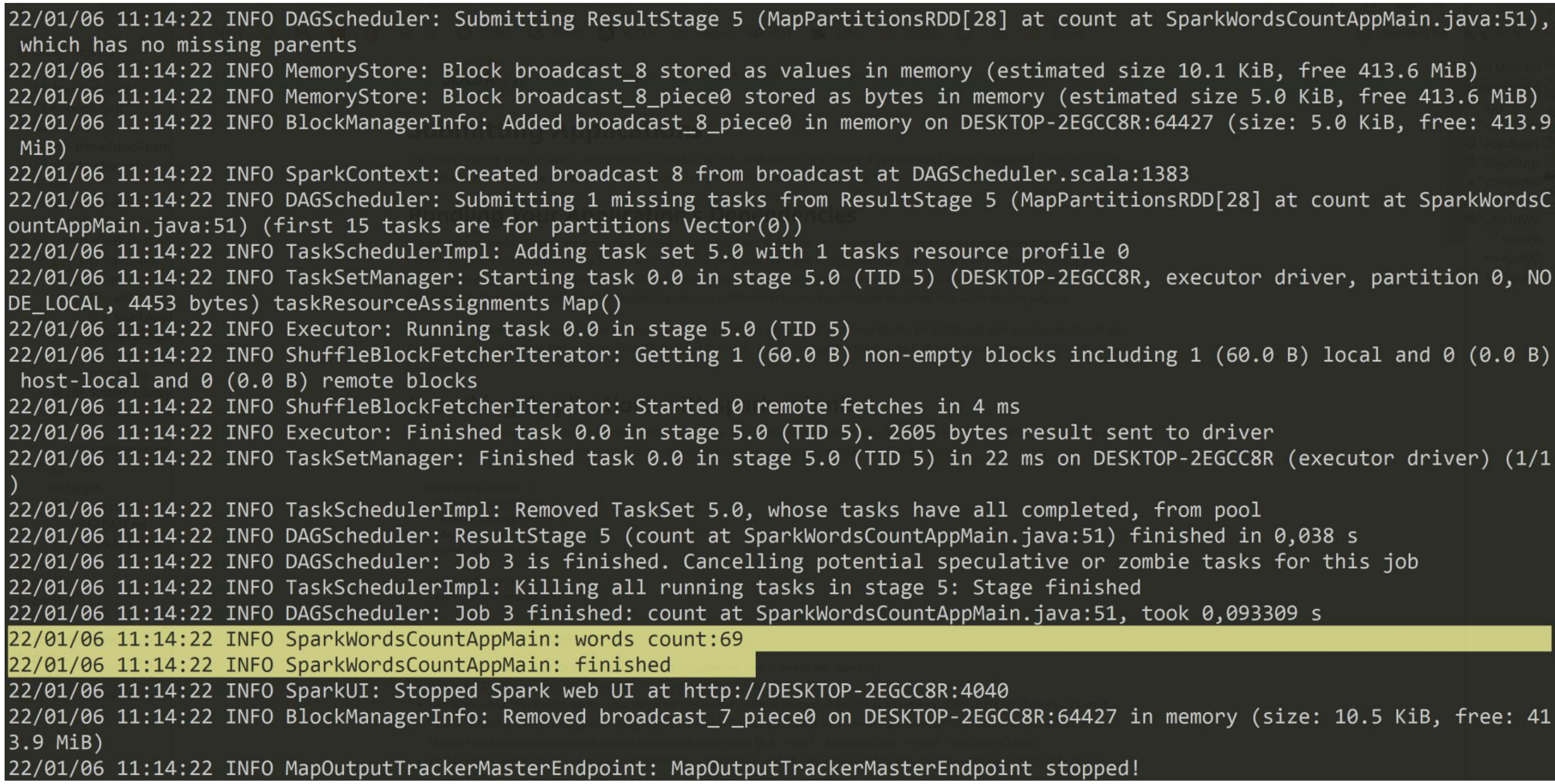
```
|   tempor|
| incididunt|
|         ut|
|     labore|
|         et|
|     dolore|
|      magna|
|    aliqua.|
|         Ut|
+-----------+
only showing top 20 rows

11:06:20 INFO fr.an.tests.testspark.SparkWordsCountAppMain: words count:69
```

spark-submit --class <<MainClass>> <<JarFile>>



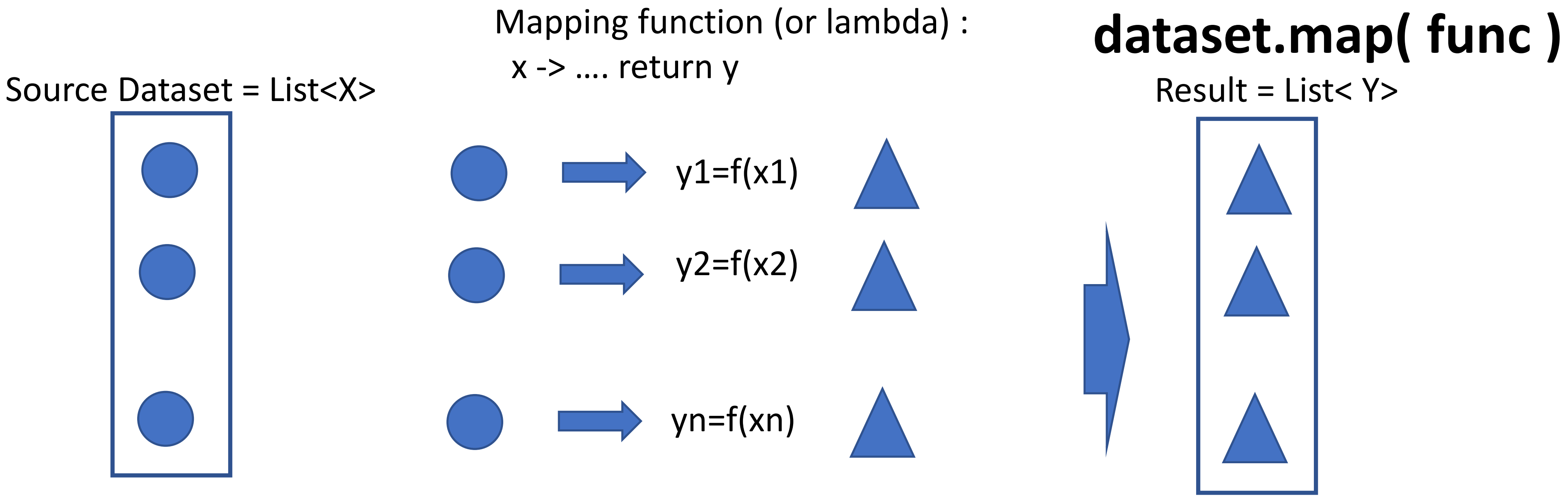
```
c:\arn\devPerso\test-snippets\test-spark (master -> origin)
λ spark-submit --class fr.an.tests.testspark.SparkWordsCountAppMain target/tests-spark-0.0.1-SNAPSHOT.jar
22/01/06 11:14:11 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes wh
ere applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
22/01/06 11:14:11 INFO SparkContext: Running Spark version 3.1.1
```

… skipped 1000 lines LOG …



```
22/01/06 11:14:22 INFO DAGScheduler: Submitting ResultStage 5 (MapPartitionsRDD[28] at count at SparkWordsCountAppMain.java:51),
 which has no missing parents
22/01/06 11:14:22 INFO MemoryStore: Block broadcast_8 stored as values in memory (estimated size 10.1 KiB, free 413.6 MiB)
22/01/06 11:14:22 INFO MemoryStore: Block broadcast_8_piece0 stored as bytes in memory (estimated size 5.0 KiB, free 413.6 MiB)
22/01/06 11:14:22 INFO BlockManagerInfo: Added broadcast_8_piece0 in memory on DESKTOP-2EGCC8R:64427 (size: 5.0 KiB, free: 413.9
 MiB)
22/01/06 11:14:22 INFO SparkContext: Created broadcast 8 from broadcast at DAGScheduler.scala:1383
22/01/06 11:14:22 INFO DAGScheduler: Submitting 1 missing tasks from ResultStage 5 (MapPartitionsRDD[28] at count at SparkWordsC
ountAppMain.java:51) (first 15 tasks are for partitions Vector(0))
22/01/06 11:14:22 INFO TaskSchedulerImpl: Adding task set 5.0 with 1 tasks resource profile 0
22/01/06 11:14:22 INFO TaskSetManager: Starting task 0.0 in stage 5.0 (TID 5) (DESKTOP-2EGCC8R, executor driver, partition 0, NO
DE_LOCAL, 4453 bytes) taskResourceAssignments Map()
22/01/06 11:14:22 INFO Executor: Running task 0.0 in stage 5.0 (TID 5)
22/01/06 11:14:22 INFO ShuffleBlockFetcherIterator: Getting 1 (60.0 B) non-empty blocks including 1 (60.0 B) local and 0 (0.0 B)
 host-local and 0 (0.0 B) remote blocks
22/01/06 11:14:22 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 4 ms
22/01/06 11:14:22 INFO Executor: Finished task 0.0 in stage 5.0 (TID 5). 2605 bytes result sent to driver
22/01/06 11:14:22 INFO TaskSetManager: Finished task 0.0 in stage 5.0 (TID 5) in 22 ms on DESKTOP-2EGCC8R (executor driver) (1/1
)
22/01/06 11:14:22 INFO TaskSchedulerImpl: Removed TaskSet 5.0, whose tasks have all completed, from pool
22/01/06 11:14:22 INFO DAGScheduler: ResultStage 5 (count at SparkWordsCountAppMain.java:51) finished in 0,038 s
22/01/06 11:14:22 INFO DAGScheduler: Job 3 is finished. Cancelling potential speculative or zombie tasks for this job
22/01/06 11:14:22 INFO TaskSchedulerImpl: Killing all running tasks in stage 5: Stage finished
22/01/06 11:14:22 INFO DAGScheduler: Job 3 finished: count at SparkWordsCountAppMain.java:51, took 0,093309 s
22/01/06 11:14:22 INFO SparkWordsCountAppMain: words count:69
22/01/06 11:14:22 INFO SparkWordsCountAppMain: finished
22/01/06 11:14:22 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-2EGCC8R:4040
22/01/06 11:14:22 INFO BlockManagerInfo: Removed broadcast_7_piece0 on DESKTOP-2EGCC8R:64427 in memory (size: 10.5 KiB, free: 41
3.9 MiB)
22/01/06 11:14:22 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
```

resultDs = dataset.map(x -> {... return y; })

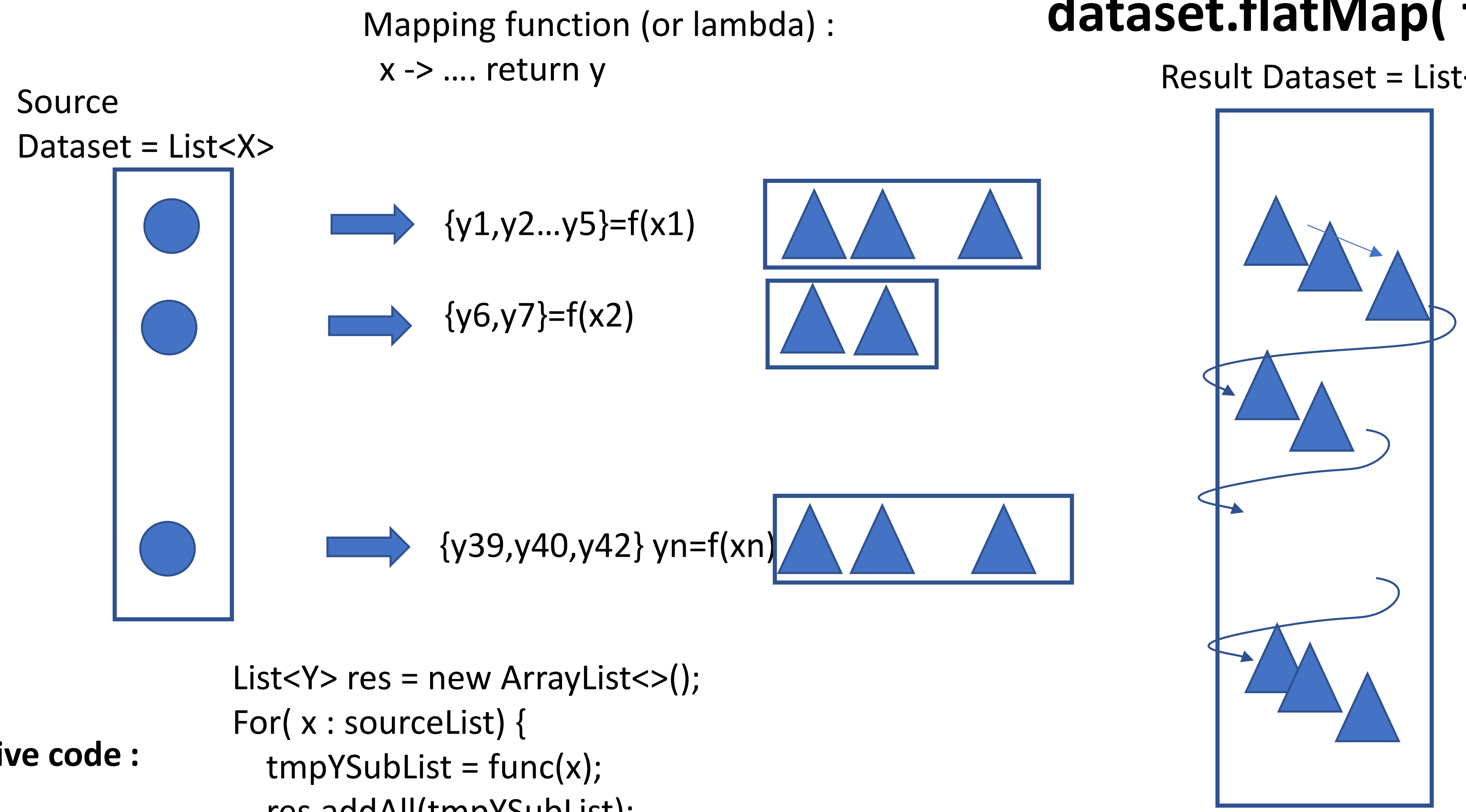


« Equivalent » imperative code :

```
List<Y> res = new ArrayList<>();
For( x : sourceList) {
   y = func(x);
   res.add(y);
}
return res;
```

resultDs = dataset.flatMap(x -> {... return list<y>; })

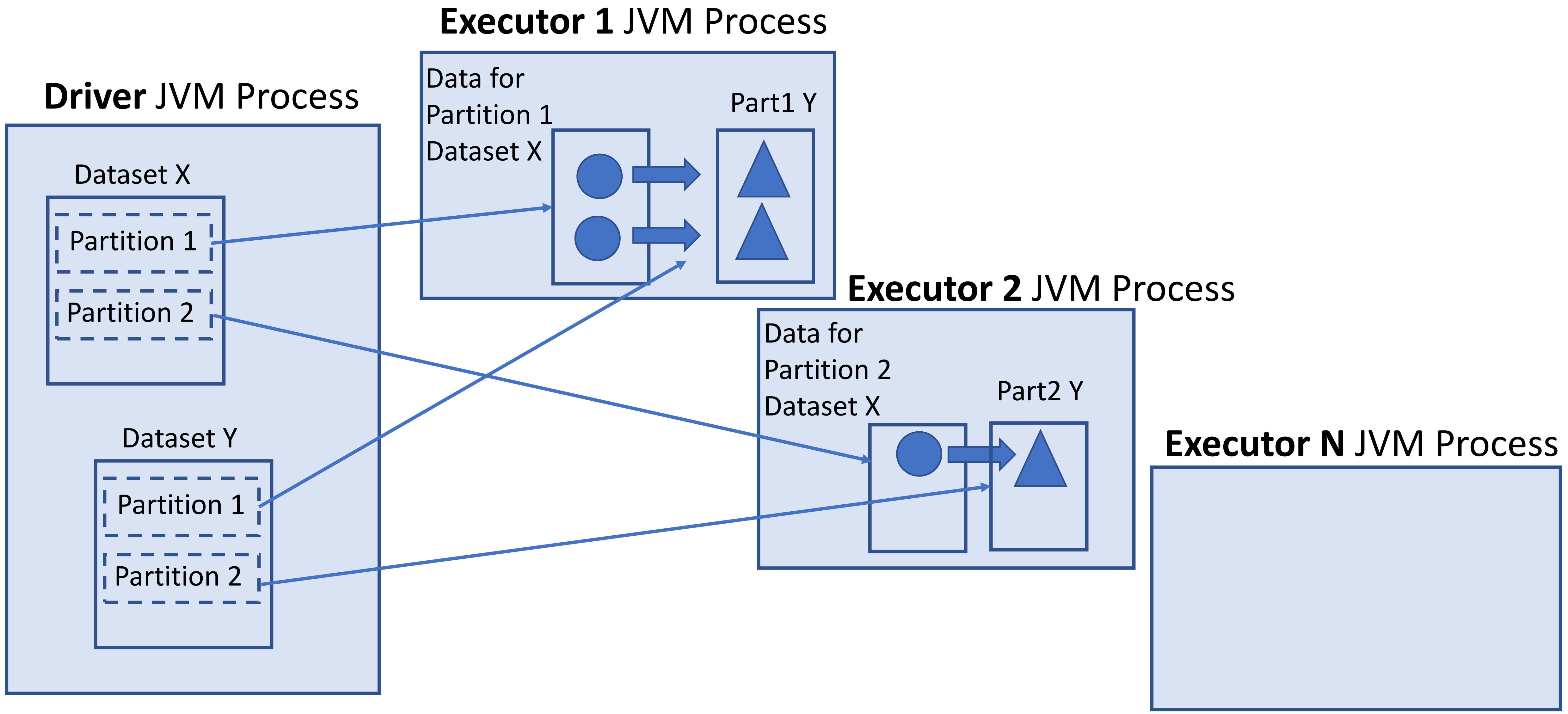
dataset.flatMap(func)



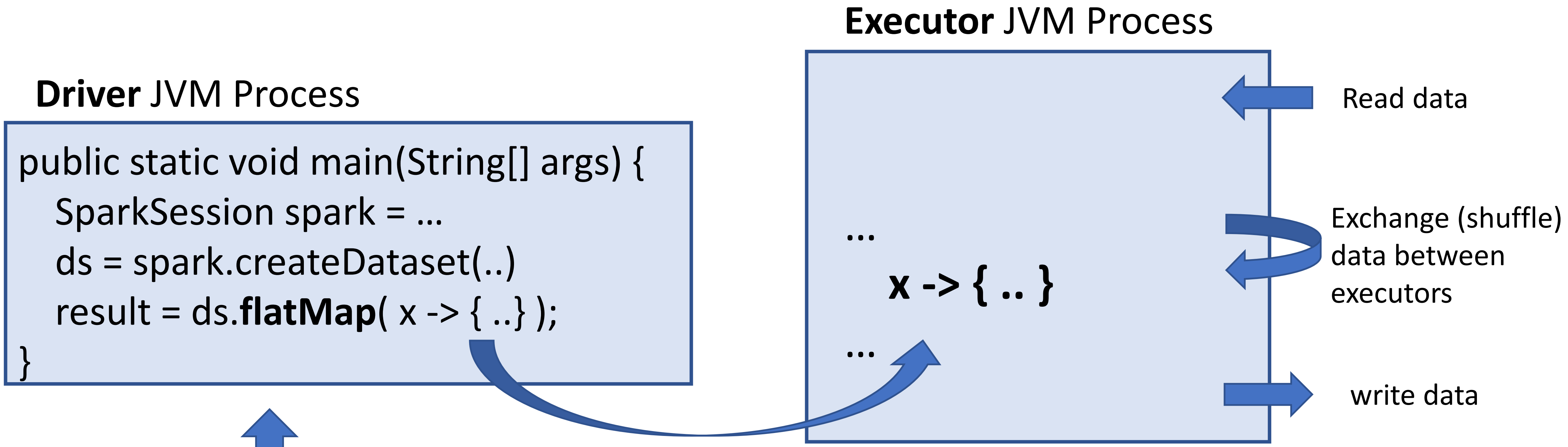
« Equivalent » imperative code :

```
List<Y> res = new ArrayList<>();
For( x : sourceList) {
   tmpYSubList = func(x);
   res.addAll(tmpYSubList);
}
return res;
```

Dataset: « logical View » of « partitions » on Driver … « data » allocated in-memory on Executors


Executor 1 JVM Process
Data for Partition 1 Dataset X
Part1 Y
Driver JVM Process
Dataset X
Partition 1
Partition 2
Executor 2 JVM Process
Data for Partition 2 Dataset X
Part2 Y
Dataset Y
Partition 1
Partition 2
Executor N JVM Process

Driver : Drives the main() program
Executors : Execute the functions on data

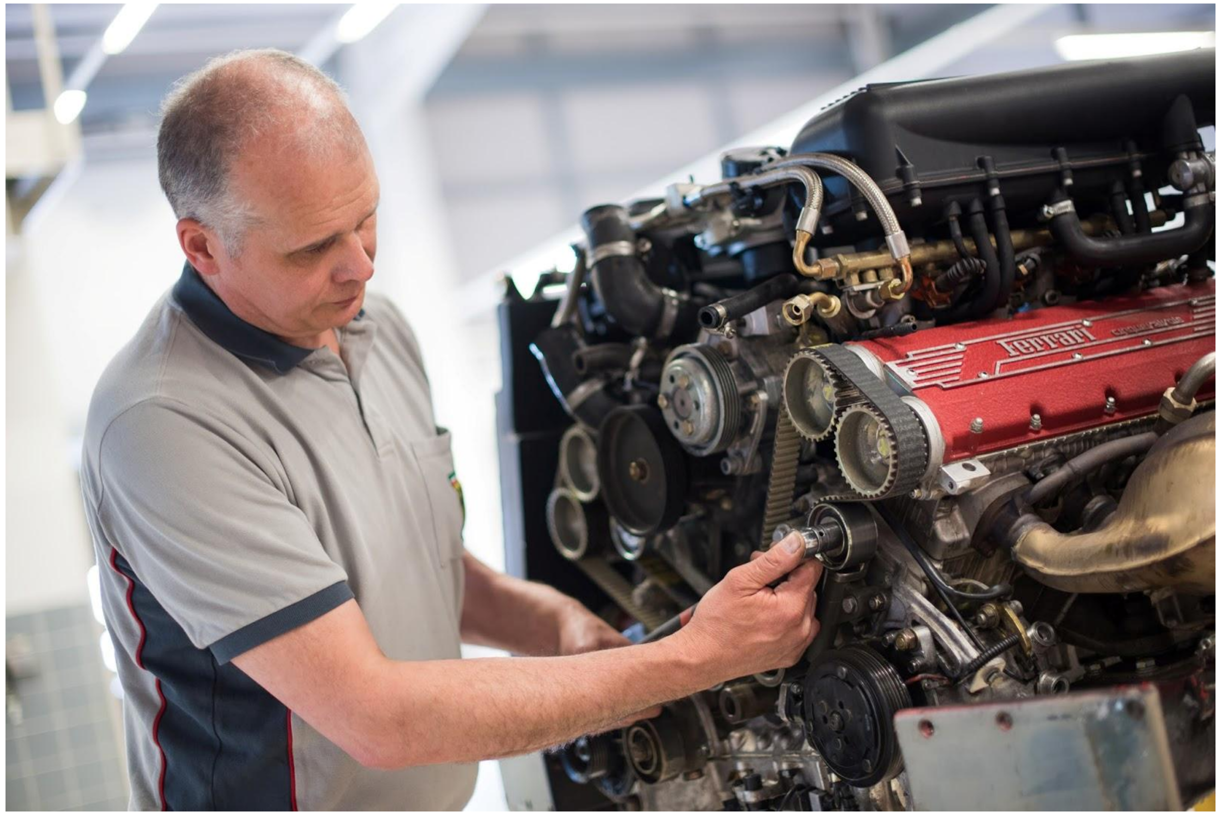


```
public static void main(String[] args) {
   SparkSession spark = …
   ds = spark.createDataset(..)
   result = ds.flatMap( x -> { ..} );
}
```

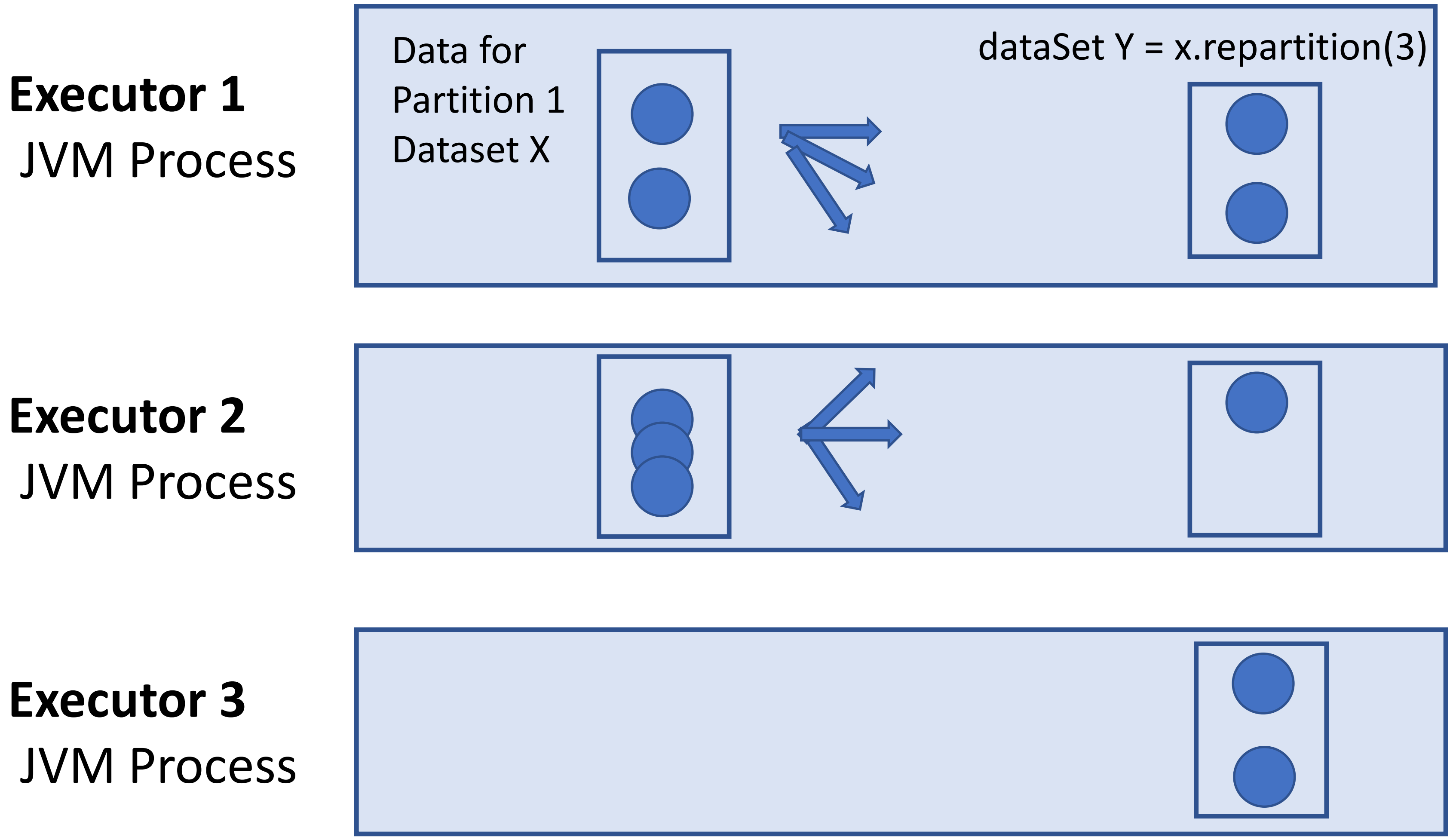
Your main code
Using Spark **Dataset API**

Code fragment in map() is « serialized »
+ sent to executors
Run on CPUs of multiple executors

See the difference ?
Driver Api / Executor Engine
Code Logic / Data+Cpu Internal



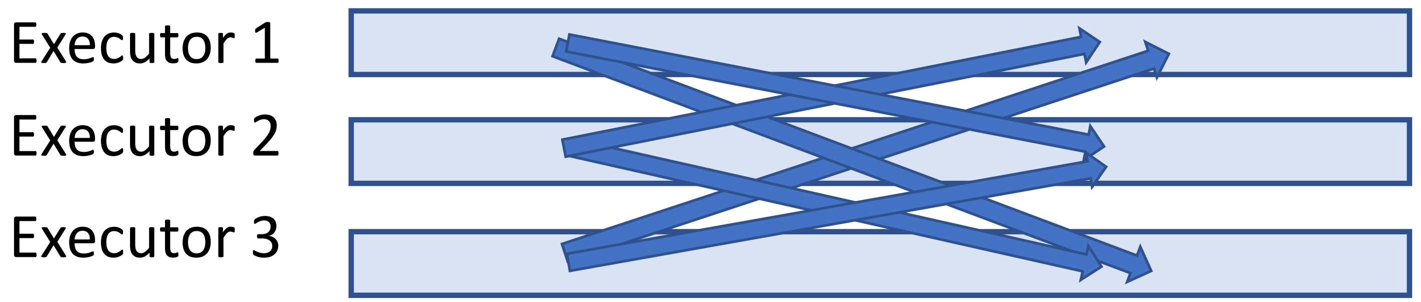
Exchange / Shuffle / Repartition / Coalesce / Distribute Wide transformation / Reduce



Wide vs Narrow Transformation

Wide Transformation (Exchange) between

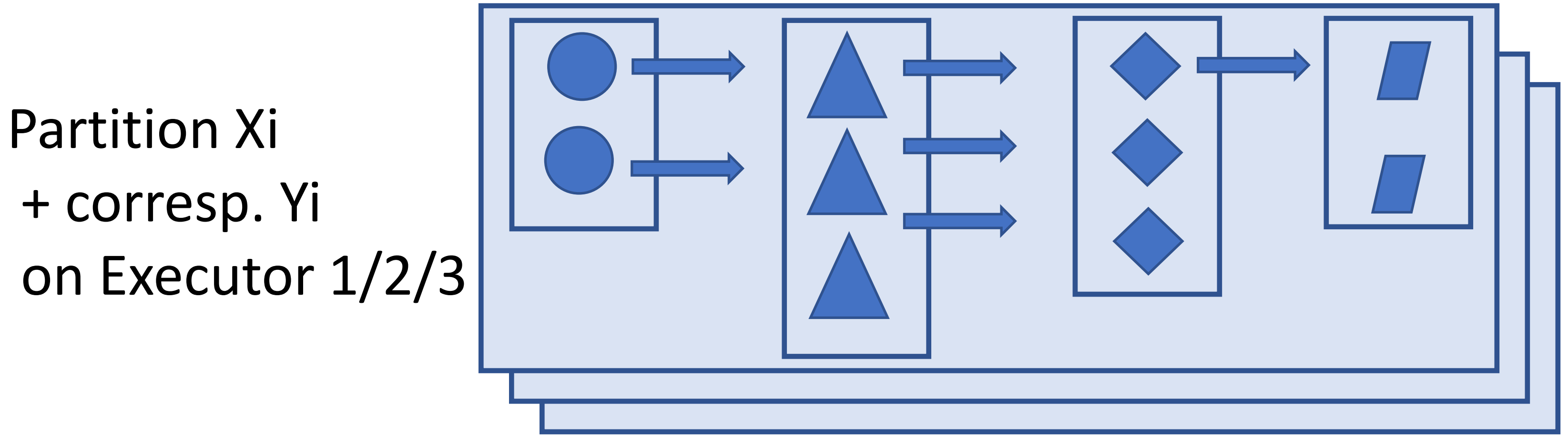
dataSet Y = x.repartition(3).shuffle(..) .reduce(..) .sortBy()



Network exchange
Bewteen executors
… serialization/deserialization of byte data

Narrow Transformation

dataSet Y = x.map(func1) .flatMap(func2).filter(pred) .first(10) .sortWithinPartition ()



No data exchange
only in-memory pointer,
within same thread/process
… computation changes
but same logical partitionning

Things get complex…
« repartition() .flatMap() .explain() »

```
scala> loremIpsum.flatMap(line => line.replaceAll("[,;.:!?]", " ").replaceAll("  ", " ").split(" ")).explain
== Physical Plan ==
*(1) SerializeFromObject [… AS value#116]
+- MapPartitions org.apache.spark.sql.Dataset$$Lambda$3735/14922651@1f3e32c, obj#115: java.lang.String
   +- DeserializeToObject value#12.toString, obj#114: java.lang.String
      +- FileScan text [value#12] Batched: false, DataFilters: [], Format: Text,
          Location: InMemoryFileIndex[file:/c:/data/loremIpsum.txt], PartitionFilters: [], PushedFilters: [], ReadSchema: struct<value
```

```
scala> loremIpsum.repartition(3).flatMap(line => line.replaceAll("[,;.:!?]", " ").replaceAll("  ", " ").split(" ")).explain
== Physical Plan ==
*(1) SerializeFromObject [staticinvoke(…  ) AS value#112]
+- MapPartitions org.apache.spark.sql.Dataset$$Lambda$3735/14922651@18e4389, obj#111: java.lang.String
   +- DeserializeToObject value#12.toString, obj#110: java.lang.String
      +- Exchange RoundRobinPartitioning(3), REPARTITION_WITH_NUM, [id=#235]
         +- FileScan text [value#12] Batched: false, DataFilters: [], Format: Text,
             Location: InMemoryFileIndex[file:/c:/data/loremIpsum.txt], PartitionFilters: [], PushedFilters: [], ReadSchema: struct<valu
```

Confusing Questions at first Glance…

1/ RDD vs DataSet vs DataFrame ?
2/ meaning of Sql / Job / Task / Staging / Action ?
3/ Driver vs Executor … where is executed my code ?
4/ Batch and Streaming api ?
5/ SQL or Code ? Functional API in java-scala-python ?
6/ Spark-shell / spark-sql / spark-submit / spark-thrift server / spark-history server ?
7/ Master = yarn/standone/k8s + mode = Client vs Cluster vs …
8/ use Spark-ui / Console / logs?
9/ Performance diagnostic?
DAG ? Metrics ? Shuffle ? SpillToDisk ? SkewedPartition?
10/ Optimize or add more resources ?