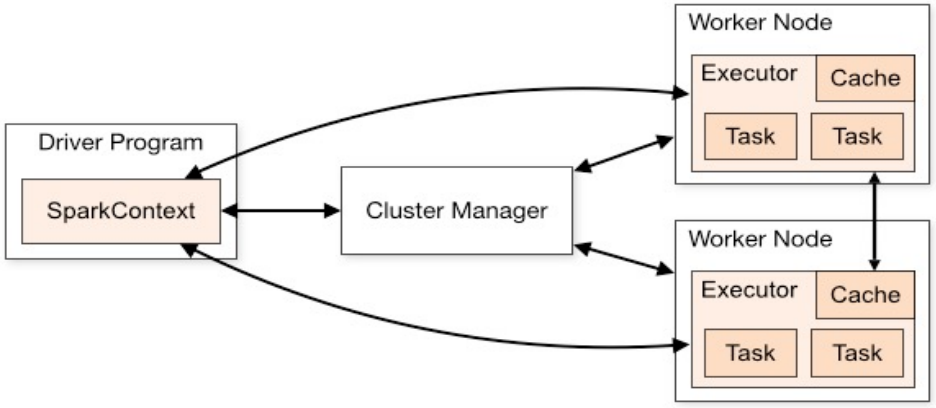
SPARK CLUSTER OVERVIEW

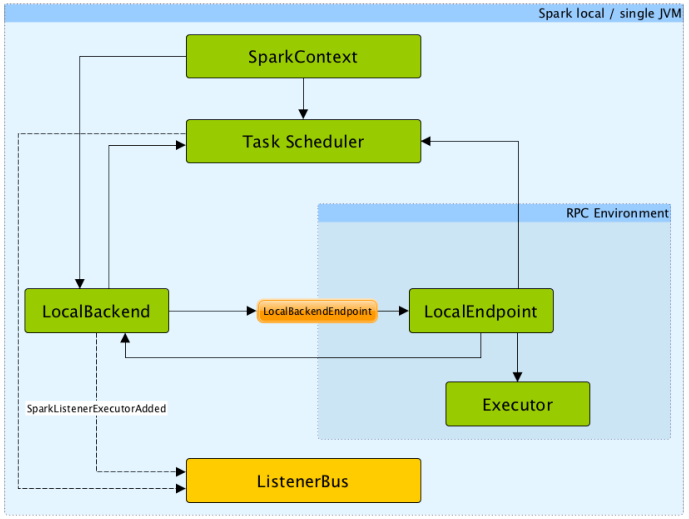
Spark Execution modes

It is possible to run a spark application using **cluster mode, local mode** (pseudo-cluster) or with an **interactive** shell (*pypsark* or *spark-shell*).

Cluster mode

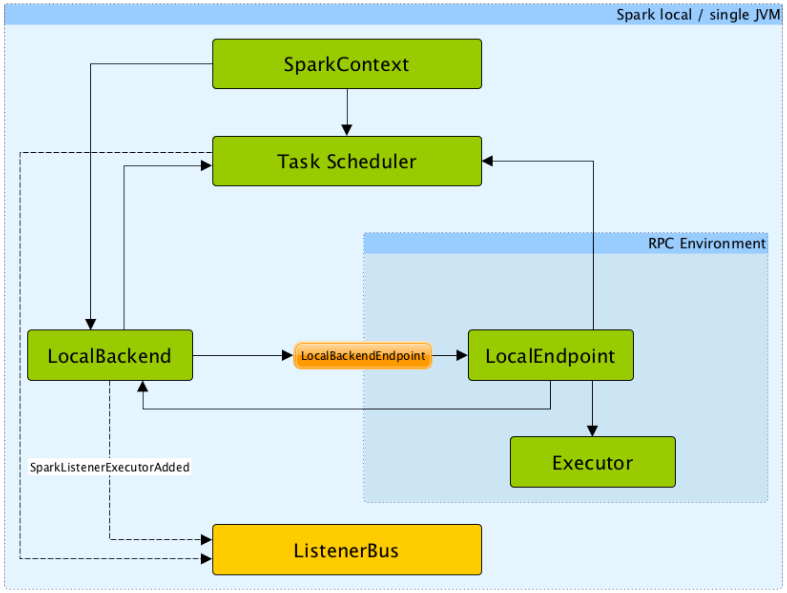


Local mode



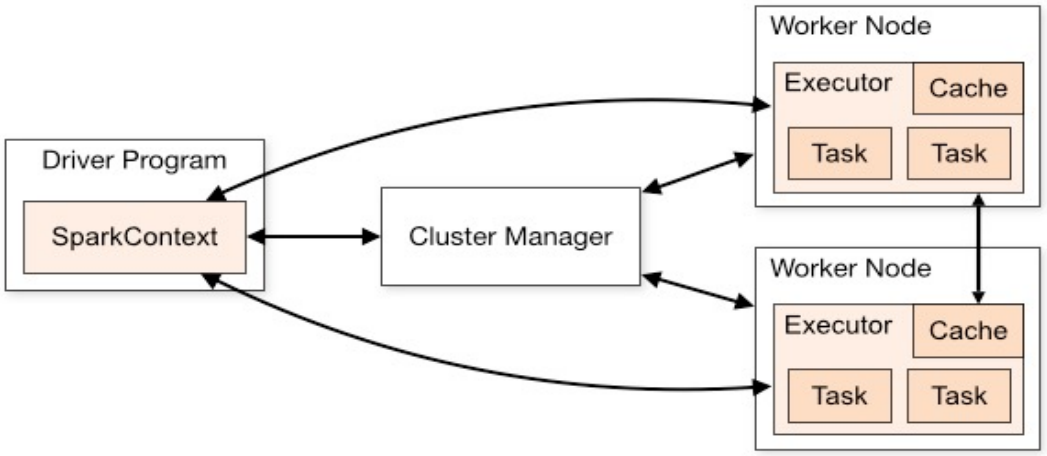
Spark Execution – Local mode

- In this non-distributed single-JVM deployment mode.
- Spark spawns all the execution components - driver, executor, LocalSchedulerBackend, and master - in the same single JVM.
- The default parallelism is the number of threads as specified in the master URL.



Spark Execution – Cluster mode

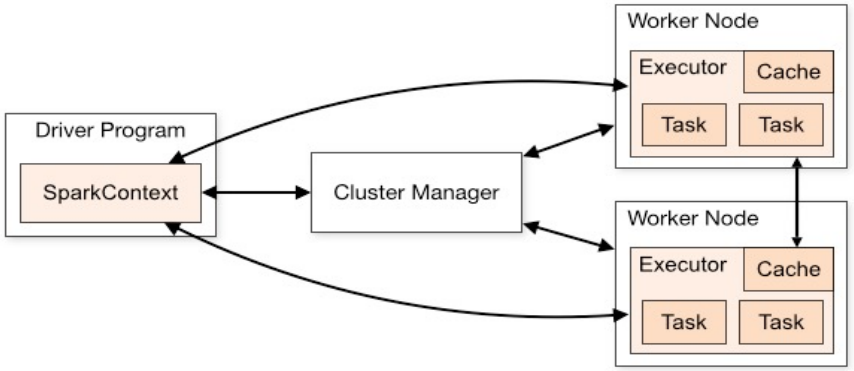
- **Standalone**: simplest way to deploy Spark on a private cluster
- Apache Mesos
- Hadoop YARN
- Kubernetes



Spark is agnostic to the underlying cluster manager

Spark Execution – Cluster mode

- Spark applications are run as independent sets of processes, coordinated by a SparkContext in a (*) *driver* program.
- The *context* connects to the cluster manager *which allocates resources*.

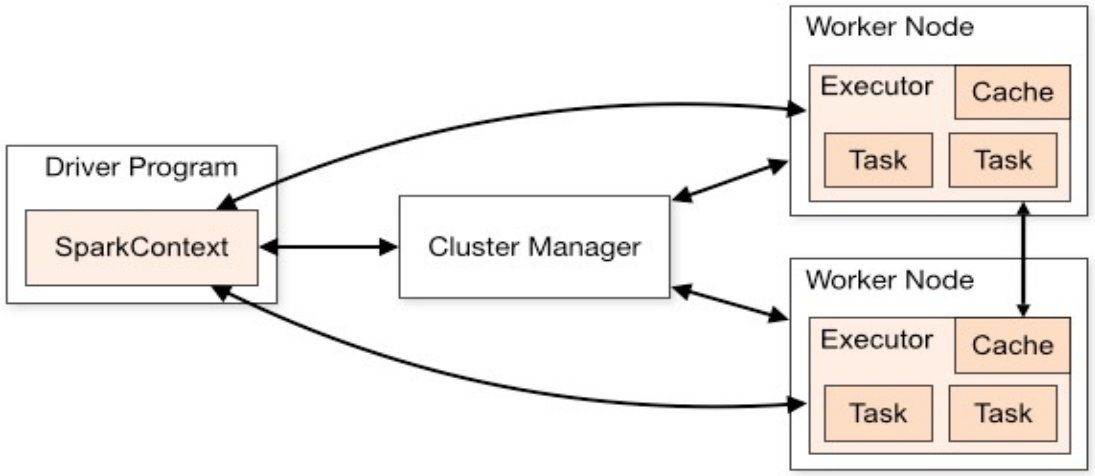


- Each *worker* in the cluster is managed by an *executor.*
- The *executor* manages computation as well as storage and caching on each machine.

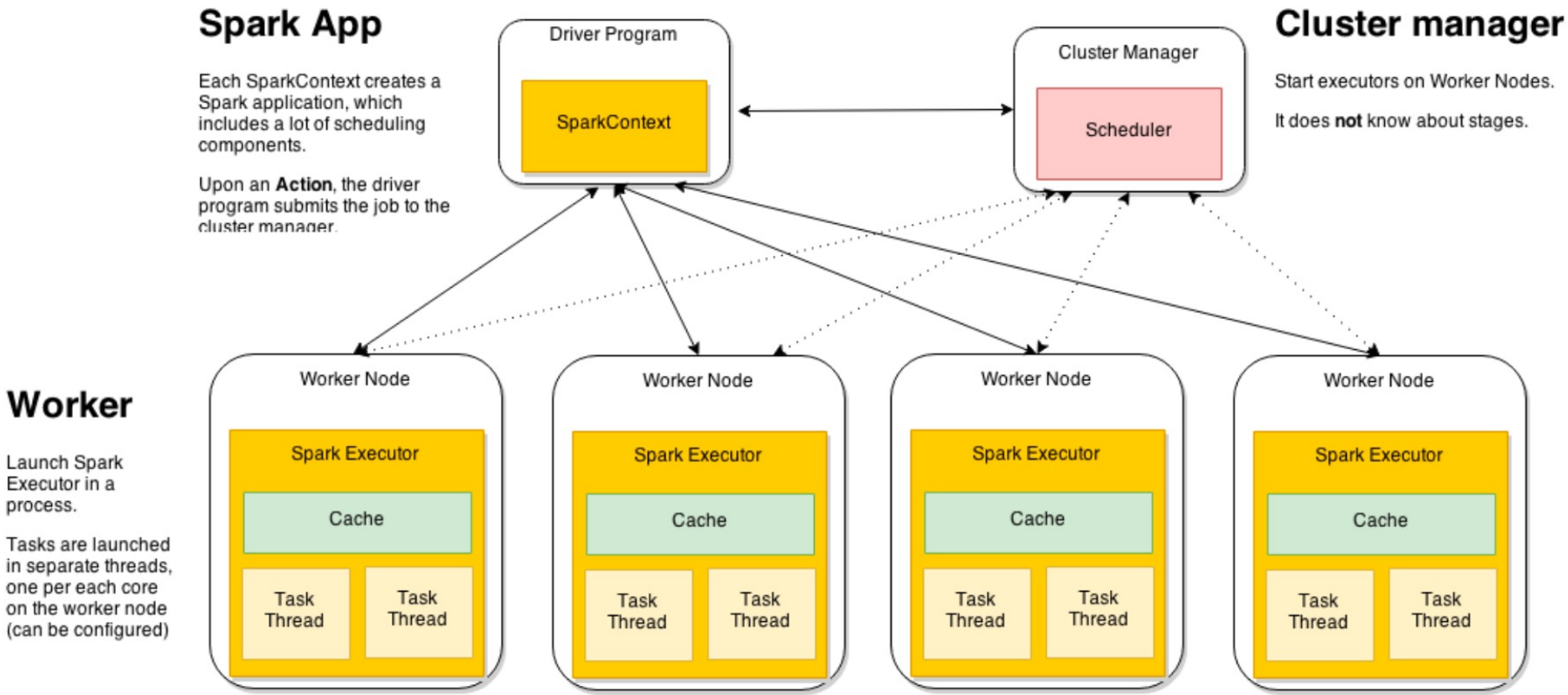
(*) *driver* → process running the *main()* function of the application and creating the *SparkContext*

Spark Execution – Cluster mode

- The application code is sent from the *driver* to the *executors*, and the executors specify the context and the various *tasks* to be run.
- The *driver* program must listen for and accept incoming connections from its executors throughout its lifetime.



Spark Execution – Cluster mode



Spark -- Standalone Cluster – Deploy modes

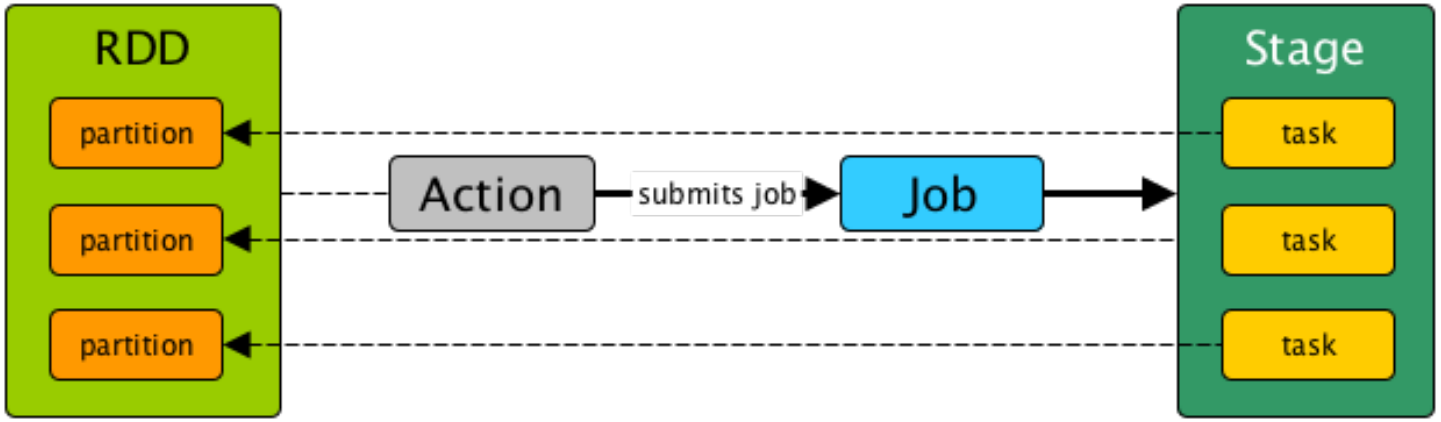
For standalone clusters supports two deploy modes.
They distinguish where the *driver* process runs:

- *Client mode (by default)*: the *driver* is launched in the *same process as the client* that submits the application.

- *Cluster mode*: the *driver* is launched from *one of the Worker processes* inside the cluster.
  - The client process exits as soon as it fulfils its responsibility of submitting the application without waiting for the application to finish.

Spark Components

- Task: individual unit of work sent to one executor over a sequences of partitions
- Job : set of tasks executed as a result of an action
- Stage: set of tasks in a job that can be executed in parallel – at partition level
- RDD: Parallel dataset with partitions
- DAG: Logical Graph of RDD operations



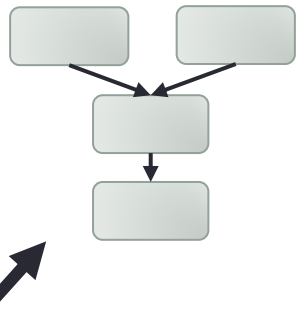
Job scheduling

RDD Objects

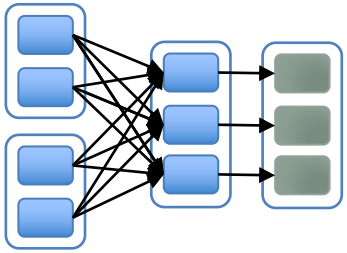
DAGScheduler

TaskScheduler

Worker

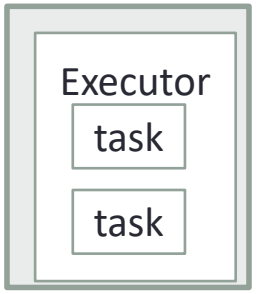


DAG



TaskSet

Tasks




```
rdd1.join(rdd2)
    .groupBy(…)
    .filter(…)
```

build operator DAG

split graph into *stages* of tasks

submit each stage as ready

launch tasks via cluster manager

retry failed or straggling tasks

execute tasks

Spark Application – wordcount.py

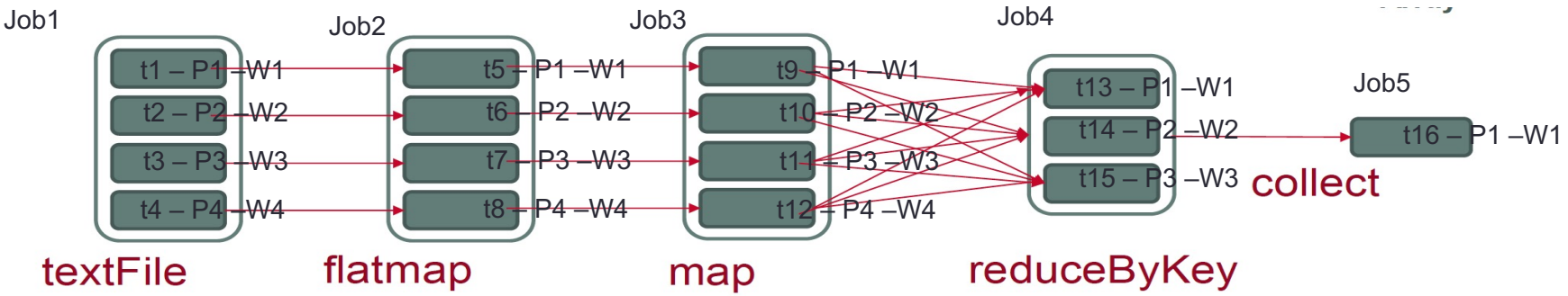
The application that we are going to create is a simple “wordcount”:

- Performs a ***textFile*** operation to read an input file in HDFS
- ***flatMap*** operation to split each line into words
- ***map*** operation to form (word, 1) pairs
- ***reduceByKey*** operation to sum the counts (all the ’1’) for each word

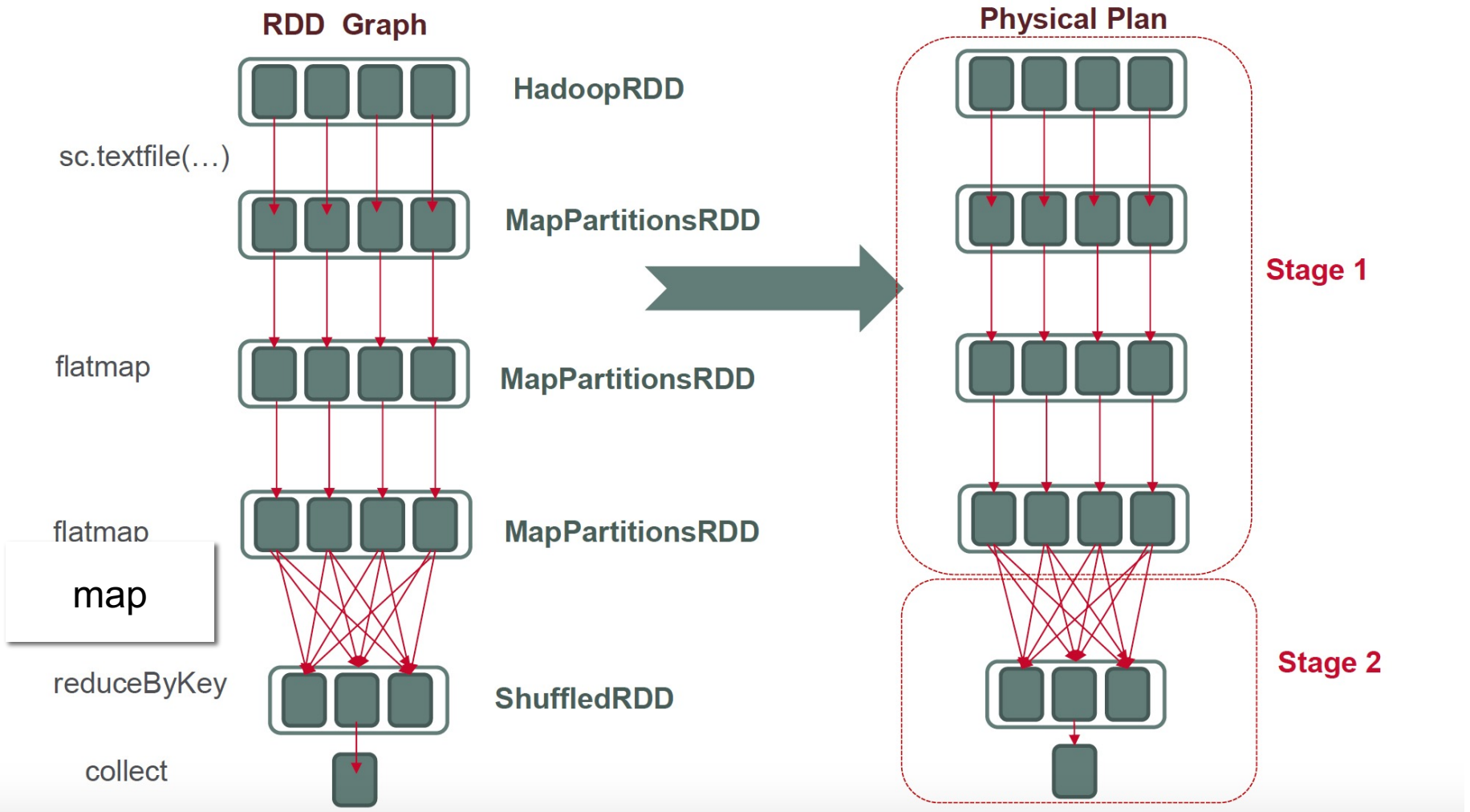
Spark Application – wordcount.py

```
import sys
from pyspark import SparkContext, SparkConf


if __name__ == "__main__":
    conf = SparkConf().setAppName("Spark Count")
    sc = SparkContext(conf=conf)
    inputFile = sys.argv[1]
    textFile = sc.textFile(inputFile)
    wordCounts = textFile.flatMap(lambda line: line.split()).\
        map(lambda word: (word, 1)).reduceByKey(lambda a, b: a+b)
    output=wordCounts.collect()
    for (word, count) in output:
        print("%s: %i" % (word, count))
```

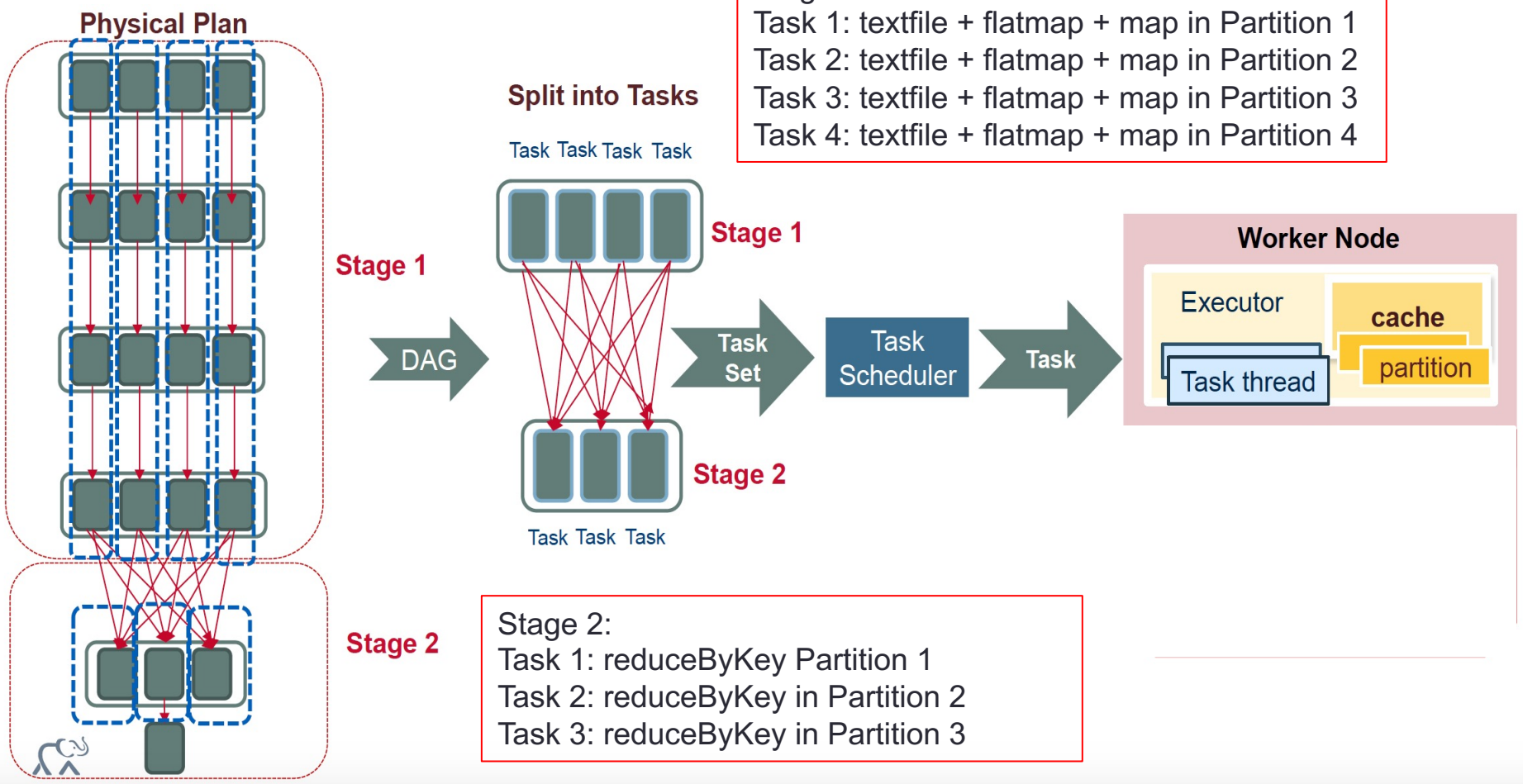


RDD DAG -> Physical Execution plan



Initial RDD distributed among 4 partitions. Final RDD distributed among 3 partitions

Execution plan -> Stages and Tasks



Operations that can run on the same partition are executed in stages

Running Spark Applications

- **Notebooks** are great for:
  - developing and testing quickly experiment with the data
  - demos and collaborating with other people

- **Spark-submit** jobs are more likely to be used in **production**.

Running Spark with Jupyter Notebooks

We are going to use Jupyter Notebooks for running our walkthroughs & lab exercises.

First we need to do the following steps:

- Copying all the material necessaire in our accounts in Cirrus
- Starting an interactive session in a node
- Starting a spark cluster (standalone) in that node
- Starting a Jupyter session connected with pyspark

All the information can be found at “Get_Started_Notebooks_Cirrus”:
https://github.com/EPCCed/prace-spark-for-data-scientists/blob/master/Get_Started_Notebooks_Cirrus.pdf

Submit job via spark-submit

spark-submit Syntax

```
spark-submit --option value \
  application jar | python file [application arguments]
```

Check the guide - Submitting Spark Applications:
https://github.com/EPCCed/prace-spark-for-data-scientists/blob/master/Spark_Applications/Submitting_Spark_Applications.pdf

Submit job via spark-submit

```
$SPARK_HOME/bin/spark-submit \
--class <main-class> \
--master <master-url> \
--deploy-mode <deploy-mode>  \
--conf \
….
<application-jar> [arguments] |
<python file >[arguments]
```

Some spark-submit options

- master – Determines how to run the job:
  - spark://r1i2n5:7077
  - local
- driver-memory
  - amount memory available for the driver process.
- executor-memory
  - amount of memory allocated to the executor process
- executor-cores
  - total number of cores allocated to the executor process
- total-executor-cores
  - Total number of cores available for all executors.

Note: https://spark.apache.org/docs/latest/submitting-applications.html

Cirrus

- High-performance computing cluster
- One of the EPSRC Tier-2 National HPC Services.
- 280 nodes: **36 Intel Xeon CPUs**, **hyper threading**, 256GB
  - Each node has (virtually) 72 cores
- 406 TB of storage- Lustre
- Link: http://www.cirrus.ac.uk/

https://cirrus.readthedocs.io/en/latest/user-guide/connecting.html

Cirrus

- Connecting to Cirrus

```
ssh [userID]@login.cirrus.ac.uk
```

- Two types of nodes:
  - Login – access to outside network
  - Computing – only network between nodes (no to outside world).

- For cloning the repository -> use the login node
  - git clone https://github.com/EPCCed/prace-spark-for-data-scientists.git

https://cirrus.readthedocs.io/en/latest/user-guide/connecting.html

Running jobs in Cirrus

- PBSPro to schedule jobs
  - Submission script to submit a job a queue
  - Interactive jobs → are also available
    - To submit a request for an interactive job reserving 1 nodes (72 physical cores) for 1 hour you would issue the following qsub command from the command line

```
qsub -IVl select=3:ncpus=36,walltime=05:00:00,place=scatter:excl -A y15 -q
$1-j oe
```

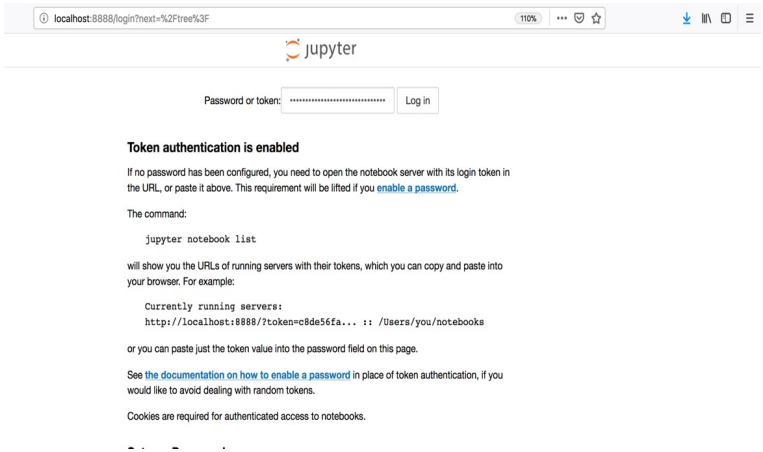
  - Your session will end:
    - It hits the requested **walltime**
    - Typing **exit command** within the session

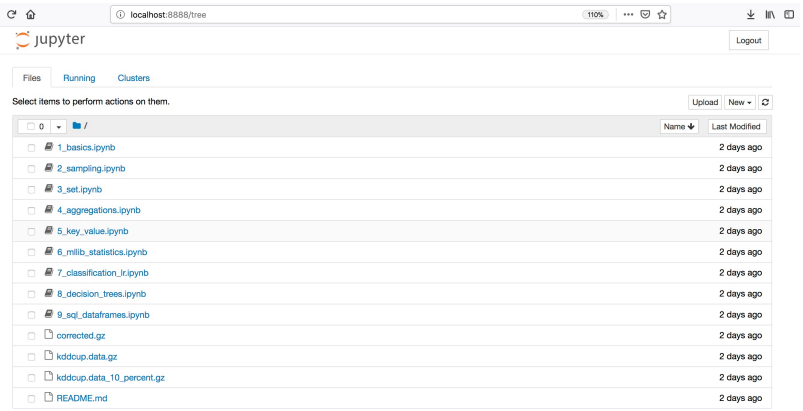
https://cirrus.readthedocs.io/en/latest/user-guide/batch.html#interactive-jobs

Jupyter notebooks

- Start the jupyter server:
  ./start_Jupyter_local.sh <master> → It will give you a token, like this one:
  http://0.0.0.0:8888/?token=2d5e554b2397355c334b8c3367503b06c4f6f95a26151795

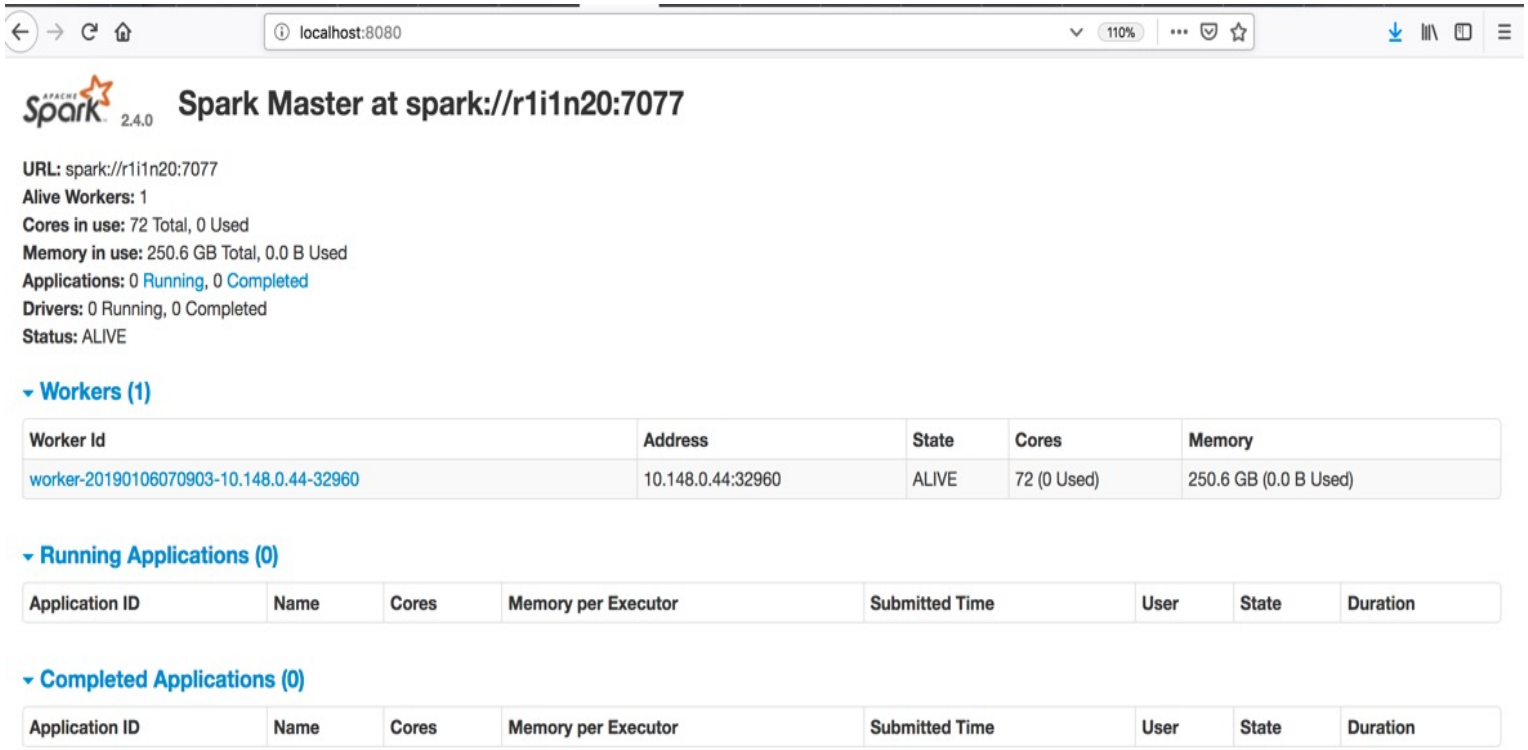
- Open **another terminal** and type the following command
  >> ssh **USER**@login.cirrus.ac.uk -L8888:**MASTER NODE**:8888
  Got to a Web browser and type → http://localhost:8888





All the information can be found at “Get_Started_Notebooks_Cirrus”:
https://github.com/EPCCed/prace-spark-for-data-scientists/blob/master/Get_Started_Notebooks_Cirrus.pdf

Master Spark UI

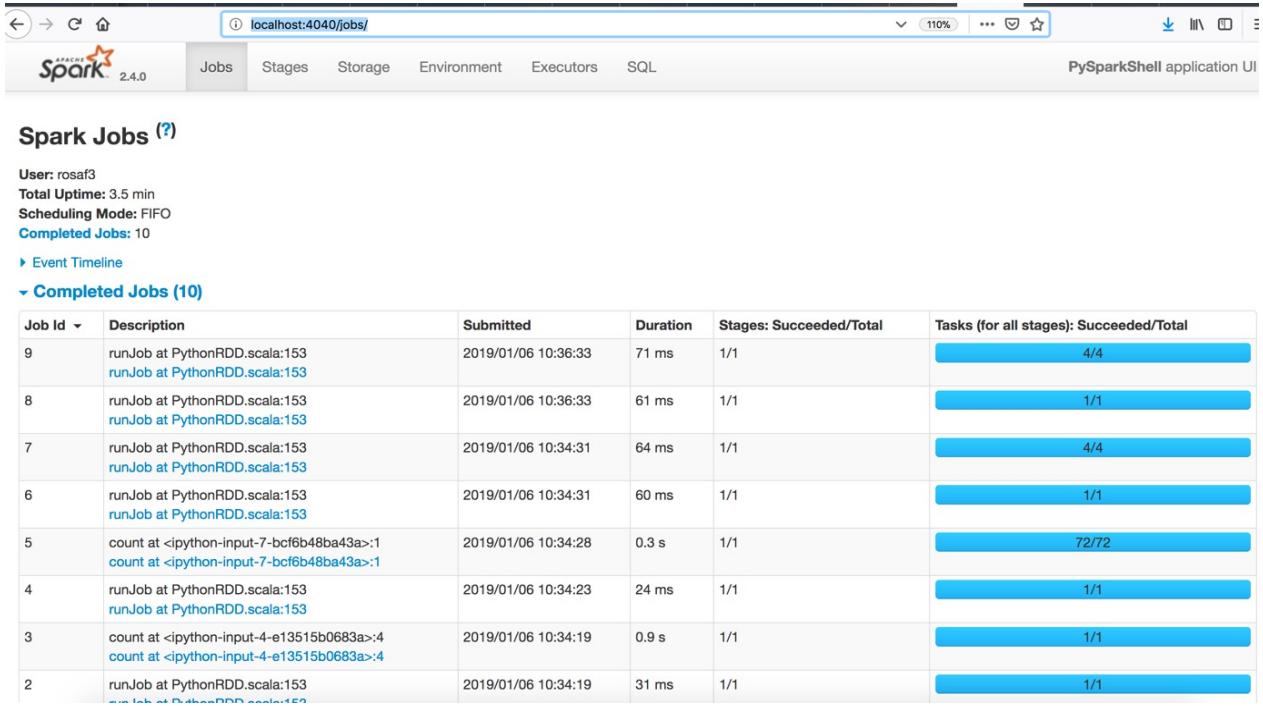


The spark setup can be monetarised via the **Master's web UI**

```
>> ssh USER@login.cirrus.ac.uk -L8080:MASTER NODE:8080
```

Got to a Web browser and type `localhost:8080`

Driver Spark UI



Every SparkContext launches a web UI (**Spark driver's web UI**),
by default on port 4040, that displays useful information about the application.

```
ssh USER@login.cirrus.ac.uk -L4040:DRIVER NODE:4040
```

web browser → localhost:4040

Running notebooks in your laptop

- **Prerequisites: Anaconda, Python3**
- Get Spark from the downloads page of the project website

  (https://blog.sicara.com/get-started-pyspark-jupyter-guide-tutorial-ae2fe84f594f)
- Check if pyspark is properly install → type `pyspark` in a terminal

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

- `>> git clone https://github.com/EPCCed/prace-spark-for-data-scientists.git`
- `>> cd walkthrough_examples`
- `>> export SPARK_HOME=[INSTALLATION_PATH]/spark-2.4.0-bin-hadoop2.7/`
- **`>> export PYSPARK_DRIVER_PYTHON=jupyter`**
- **`>> export PYSPARK_DRIVER_PYTHON_OPTS='notebook'`**
- `>> $SPARK_HOME/bin/pyspark`