

Get Started with PySpark and Jupyter Notebook in Cirrus

Jupyter Notebook is a popular application that enables you to edit, run and share Python code into a web view. It allows you to modify and re-execute parts of your code in a very flexible way. That's why Jupyter is a great tool to test and prototype programs. We are going to use Jupyter Notebooks for running our walkthroughs and lab exercises. But first we need to do some steps for:

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

Retrieving all the material:

1. Open a new terminal and login into cirrus: - replace USER by your USERNAME in cirrus
 - a. ssh USER@login.cirrus.ac.uk
2. Copy the spark source and the other necessary scripts into your \$HOME directory:
 - a. cp /lustre/home/shared/y15/spark/**spark-2.4.0-bin-hadoop2.7.tar** \$HOME/.
 - b. cp /lustre/home/shared/y15/spark/**start_interactive.sh** \$HOME/.
 - c. cp /lustre/home/shared/y15/spark/**start_spark.sh** \$HOME/.
 - d. cp /lustre/home/shared/y15/spark/**stop_spark.sh** \$HOME/.
3. Decompress **spark-2.4.0-bin-hadoop2.7.tar**
 - a. tar -zxvf **spark-2.4.0-bin-hadoop2.7.tar**
4. Clone the repository in your \$HOME:
 - a. git clone <https://github.com/EPCCed/prace-spark-for-data-scientists.git>

Starting an interactive session:

5. Start an interactive session in a node, using the **start_interactive** script. This script requests by default one node from the y15 reservation for 5 hours:
 - a. Run **start_interactive.sh** script indicating your reservation number (e.g. **R1197436**)

```
./start_interactive.sh R1197436
```

Note: If you want to modify the reservation time, you need to change the **walltime in the script**.

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

This will give you an interactive session into a node (e.g. node **r1i1n20**) and you will see something like this:

```
[USERNAME@r1i1n20 ~]$ ./start_interactive.sh R1197436  
qsub: waiting for job 399686.indy2-login0 to start  
qsub: job 399686.indy2-login0 ready  
[USERNAME@r1i1n20 ~]$ → You will need the name of the node assigned in the step 8
```

Note: `qstat -u USERNAME` → it gives you the number of nodes and cores allocated.

Starting a Spark cluster:

6. Start the spark cluster in the node assigned to us (e.g. `r1i1n20`) using the **`start_spark`** script. It is very important to use **`source`** to export the environment variables into your PATH. By default, this script starts **one master process and one slave process on the same node** (it also starts the history server). **The slave process runs one worker instance by default, which is configured to use all available CPUs.**

```
[USERNAME@r1i1n20 ~]$ source ./start_spark.sh
1 node(s) assigned
Autoloading gcc/6.2.0
starting org.apache.spark.deploy.master.Master, logging to
/lustre/home/y15/USERNAME/spark-2.4.0-bin-hadoop2.7/logs/spark-USERNAME-
org.apache.spark.deploy.master.Master-1-r1i1n20.out
failed to launch: nice -n 0 /lustre/home/y15/USERNAME/spark-2.4.0-bin-
hadoop2.7/bin/spark-class org.apache.spark.deploy.master.Master --host r1i1n20 --
port 7077 --webui-port 8080
full log in /lustre/home/y15/USERNAME/spark-2.4.0-bin-hadoop2.7/logs/spark-
USERNAME-org.apache.spark.deploy.master.Master-1-r1i1n20.out
r1i1n20: Warning: Permanently added 'r1i1n20,10.148.0.44' (ECDSA) to the list of
known hosts.
r1i1n20: starting org.apache.spark.deploy.worker.Worker, logging to
/lustre/home/y15/USERNAME/spark-2.4.0-bin-hadoop2.7/logs/spark-USERNAME-
org.apache.spark.deploy.worker.Worker-1-r1i1n20.out
starting org.apache.spark.deploy.history.HistoryServer, logging to
/lustre/home/y15/USERNAME/spark-2.4.0-bin-hadoop2.7/logs/spark-USERNAME-
org.apache.spark.deploy.history.HistoryServer-1-r1i1n20.out
```

Note 1: We can also configure the number of worker instances launched by each slave and the number of CPUs bound to each worker. This is done by passing arguments to the `setup_spark_env.py` script called from within `start_spark.sh`, see below for the actual command.

```
python $SPARK_CONF_DIR/setup_spark_env.py "SPARK_WORKER_INSTANCES=10"
"SPARK_WORKER_CORES=2"
```

Note 2: To shutdown spark and close the interactive session, simply run `./stop_spark.sh` followed by `exit`.

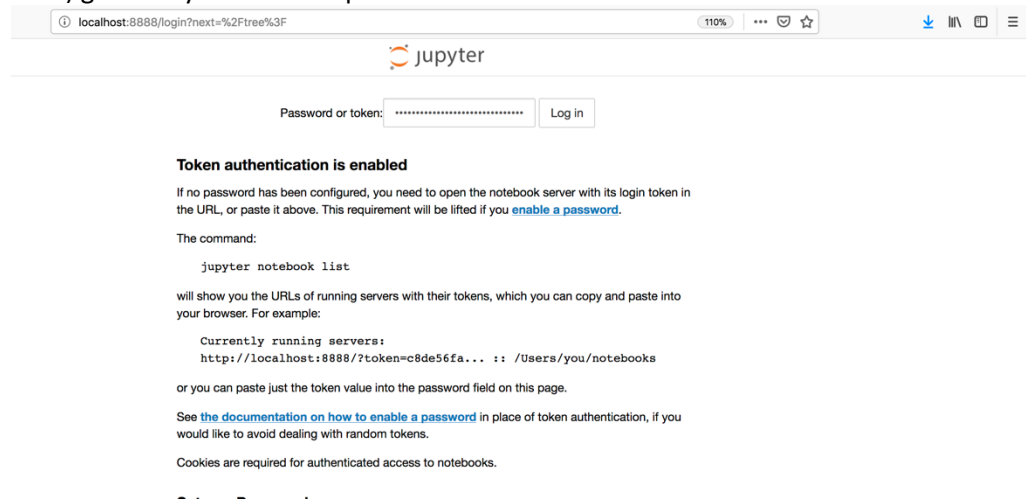
```
[USERNAME@r1i1n20 ~]$ ./stop_spark.sh
stopping org.apache.spark.deploy.master.Master
stopping org.apache.spark.deploy.history.HistoryServer
r1i1n20: stopping org.apache.spark.deploy.worker.Worker
[USERNAME@r1i1n20 ~]$ exit
```

Starting a Jupyter session connected with Pyspark:

7. Go to **`prace-spark-for-data-scientists`** to start the jupyter server:
 - a. Run the **`script start_Jupyter_local.sh`** indicating your MASTER node (which is the node assigned to us in the interactive session- e.g. `r1i1n20`)
 - b. **`./start_Jupyter_local.sh r1i1n20`** → This will give you a **token**, which will be used in step 8.b, like this one:
<http://0.0.0.0:8888/?token=2d5e554b2397355c334b8c3367503b06c4f6f95a26151795>

8. Open **another terminal** and type the following command but replacing first USER by your USERNAME in cirrus, and NODE by the node that has been assigned to you in the interactive session.

- a. `ssh USER@login.cirrus.ac.uk -L8888:MASTER NODE:8888`
- b. web browser → <http://localhost:8888>
- c. It will start a jupyter session, where you will have to type the token (only the first time) given to you in the step 7.



9. Finally, the monitoring of the spark setup can be done via the **Master's web UI**. Simply launch another terminal session and run something like

- a. `ssh USER@login.cirrus.ac.uk -L8080:MASTER NODE:8080`
- b. Web browser → `localhost:8080`

As follows, we can see different configurations of our spark cluster, depending on the configuration used in *start_interactive* (number of nodes), *start_spark* (number of workers, and number of cores per worker) scripts.

Example 1: Using the default values in both the scripts: **the master and a worker processes** are located in the **same node**. And the worker process is configured with 72 cores.

Spark Master at spark://r1i1n20:7077

URL: spark://r1i1n20:7077

Alive Workers: 1

Cores in use: 72 Total, 0 Used

Memory in use: 250.6 GB Total, 0.0 B Used

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20190106070903-10.148.0.44-32960	10.148.0.44:32960	ALIVE	72 (0 Used)	250.6 GB (0.0 B Used)

Running Applications (0)


Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

Example 2: We have asked for one node (in *start_interactive*), but we have configured the spark cluster (*start_spark*) to have 10 workers with 2 cores per worker.

```
python $SPARK_CONF_DIR/setup_spark_env.py "SPARK_WORKER_INSTANCES=10"
"SPARK_WORKER_CORES=2"
```

 **Spark Master at spark://r1i2n35:7077**

URL: spark://r1i2n35:7077
 Alive Workers: 10
 Cores in use: 20 Total, 0 Used
 Memory in use: 2.4 TB Total, 0.0 B Used
 Applications: 0 Running, 0 Completed
 Drivers: 0 Running, 0 Completed
 Status: ALIVE

▼ Workers (10)

Worker Id	Address	State	Cores	Memory
worker-20190106101403-10.148.0.217-46260	10.148.0.217:46260	ALIVE	2 (0 Used)	250.6 GB (0.0 B Used)
worker-20190106101406-10.148.0.217-37502	10.148.0.217:37502	ALIVE	2 (0 Used)	250.6 GB (0.0 B Used)
worker-20190106101408-10.148.0.217-33739	10.148.0.217:33739	ALIVE	2 (0 Used)	250.6 GB (0.0 B Used)
worker-20190106101411-10.148.0.217-43470	10.148.0.217:43470	ALIVE	2 (0 Used)	250.6 GB (0.0 B Used)
worker-20190106101414-10.148.0.217-40069	10.148.0.217:40069	ALIVE	2 (0 Used)	250.6 GB (0.0 B Used)
worker-20190106101416-10.148.0.217-34770	10.148.0.217:34770	ALIVE	2 (0 Used)	250.6 GB (0.0 B Used)
worker-20190106101419-10.148.0.217-34314	10.148.0.217:34314	ALIVE	2 (0 Used)	250.6 GB (0.0 B Used)
worker-20190106101422-10.148.0.217-45729	10.148.0.217:45729	ALIVE	2 (0 Used)	250.6 GB (0.0 B Used)
worker-20190106101424-10.148.0.217-38072	10.148.0.217:38072	ALIVE	2 (0 Used)	250.6 GB (0.0 B Used)
worker-20190106101427-10.148.0.217-40476	10.148.0.217:40476	ALIVE	2 (0 Used)	250.6 GB (0.0 B Used)

Example 3: We have asked for two nodes (in *start_interactive*), but we have configured the spark cluster (*start_spark*) to have a worker per node, using all cores available per worker (72).

URL: spark://r1i2n35:7077
 Alive Workers: 2
 Cores in use: 144 Total, 0 Used
 Memory in use: 501.1 GB Total, 0.0 B Used
 Applications: 0 Running, 0 Completed
 Drivers: 0 Running, 0 Completed
 Status: ALIVE

▼ Workers (2)

Worker Id	Address	State	Cores	Memory
worker-20190106100929-10.148.0.217-33696	10.148.0.217:33696	ALIVE	72 (0 Used)	250.6 GB (0.0 B Used)
worker-20190106100929-10.148.0.219-45349	10.148.0.219:45349	ALIVE	72 (0 Used)	250.6 GB (0.0 B Used)

▼ Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

▼ Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

- Every SparkContext launches a web UI (**Spark driver's web UI**), by default on port 4040, that displays useful information (jobs, stages, storage for RDD persistence, broadcasts, accumulators) about the application. Remember that in since we are using Pyspark, it automatically creates the spark context for us. After running a notebook, you could check the following web UI, by launching in another terminal.
 - ssh **USER@login.cirrus.ac.uk -L4040:MASTER NODE:4040**
 - web browser → localhost:4040

The Spark driver's web UI is to show the progress of your computations, while standalone Master's web UI is to let you know the current state of your "operating environment" (aka the Spark Standalone cluster).

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
9	runJob at PythonRDD.scala:153 runJob at PythonRDD.scala:153	2019/01/06 10:36:33	71 ms	1/1	4/4
8	runJob at PythonRDD.scala:153 runJob at PythonRDD.scala:153	2019/01/06 10:36:33	61 ms	1/1	1/1
7	runJob at PythonRDD.scala:153 runJob at PythonRDD.scala:153	2019/01/06 10:34:31	64 ms	1/1	4/4
6	runJob at PythonRDD.scala:153 runJob at PythonRDD.scala:153	2019/01/06 10:34:31	60 ms	1/1	1/1
5	count at <ipython-input-7-bcf6b48ba43a>:1 count at <ipython-input-7-bcf6b48ba43a>:1	2019/01/06 10:34:28	0.3 s	1/1	72/72
4	runJob at PythonRDD.scala:153 runJob at PythonRDD.scala:153	2019/01/06 10:34:23	24 ms	1/1	1/1
3	count at <ipython-input-4-e13515b0683a>:4 count at <ipython-input-4-e13515b0683a>:4	2019/01/06 10:34:19	0.9 s	1/1	1/1
2	runJob at PythonRDD.scala:153 runJob at PythonRDD.scala:153	2019/01/06 10:34:19	31 ms	1/1	1/1

Note: You can use a single command to bridge all the ports for the Spark UI

```
ssh -L8888: MASTER NODE:8888 -L8080: MASTER NODE:8080 -L4040: MASTER
NODE:4040 USER@login.cirrus.ac.uk
```