

Spark-11

- Install & Setup [Docker]
- Code - walk throughs

Installation & Setup

Learning: Single box
↓

→ Docker containers

→ Install on Linux/Mac/Win

Production (Network & Ops teams)

① Cluster + Spark

② AWS EMR

③ Azure Databricks

④ Databricks

Docker explained in detail ↴

■ Case Study 13: Semantic Search Engine for Q&A [Design + Code]

▶ 73.1 Semantic Search for Q&A [Design + Code] --- Part 1 👁

▶ 73.2 Semantic Search for Q&A [Design + Code] --- Part 2 👁

▶ 73.3 Semantic Search for Q&A [Design + Code] --- Part 3 👁

▶ 73.4 Semantic Search for Q&A [Design + Code] --- Part 4 👁

Docker-based:

① Docker-desktop : <https://www.docker.com/products/docker-desktop>

② Spark on Docker: `$ docker pull jupyter/pyspark-notebook`
`$ docker run -it -p 8888:8888 jupyter/pyspark-notebook`

```
(base) Srikanths-MBP:home varma$ docker run -it -p 8888:8888 jupyter/pyspark-notebook
Executing the command: jupyter notebook
[I 07:17:05.000 NotebookApp] Writing notebook server cookie secret to /home/jovyan/.local/share/jupyter/runtime/notebo
e_secret
[I 07:17:05.612 NotebookApp] JupyterLab extension loaded from /opt/conda/lib/python3.8/site-packages/jupyterlab
[I 07:17:05.612 NotebookApp] JupyterLab application directory is /opt/conda/share/jupyter/lab
[I 07:17:05.615 NotebookApp] Serving notebooks from local directory: /home/jovyan
[I 07:17:05.615 NotebookApp] The Jupyter Notebook is running at:
[I 07:17:05.615 NotebookApp] http://34f08692a5ec:8888/?token=1b5188f89d2108ac5acdf0866eba14b626da7ec2faece1cb
[I 07:17:05.615 NotebookApp] or http://127.0.0.1:8888/?token=1b5188f89d2108ac5acdf0866eba14b626da7ec2faece1cb
[I 07:17:05.615 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 07:17:05.618 NotebookApp]
```

To access the notebook, open this file in a browser:

file:///home/jovyan/.local/share/jupyter/runtime/nbserver-7-open.html

Or copy and paste one of these URLs:

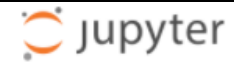
http://34f08692a5ec:8888/?token=1b5188f89d2108ac5acdf0866eba14b626da7ec2faece1cb

or http://127.0.0.1:8888/?token=1b5188f89d2108ac5acdf0866eba14b626da7ec2faece1cb

```
[I 07:17:11.861 NotebookApp] 302 GET / (172.17.0.1) 0.61ms
```

```
[I 07:17:11.866 NotebookApp] 302 GET /tree? (172.17.0.1) 0.81ms
```

<http://localhost:8888/>



Password or token:

Log in

Token authentication is enabled

If no password has been configured, you need to open the notebook server with its login token in the URL, or paste it above. This requirement will be lifted if you [enable a password](#).

The command:

```
jupyter notebook list
```

will show you the URLs of running servers with their tokens, which you can copy and paste into your browser. For example:

```
Currently running servers:
```

```
http://localhost:8888/?token=c8de56fa... :: /Users/you/notebooks
```

or you can paste just the token value into the password field on this page.

See [the documentation on how to enable a password](#) in place of token authentication, if you would like to avoid dealing with random tokens.

Cookies are required for authenticated access to notebooks.

pySpark API: <http://spark.apache.org/docs/latest/api/python/pyspark.html>

Compute value of PI using Monte Carlo Approach

Source: <http://spark.apache.org/examples.html>

```
import pyspark
import random
if not 'sc' in globals():
    sc = pyspark.SparkContext()
```

```
NUM_SAMPLES = 1000000
```

```
def inside(p):
    x, y = random.random(), random.random()
    return x*x + y*y < 1
```

```
# parallelize:http://spark.apache.org/docs/latest/api/python/pyspark.html?highlight=parallelize
```

```
# filter: http://spark.apache.org/docs/latest/api/python/pyspark.html
```

```
count = sc.parallelize(range(0, NUM_SAMPLES), 10) \
    .filter(inside).count()
```

```
print("Pi is roughly %f" % (4.0 * count / NUM_SAMPLES))
```

→ Jupyter

→ Terminal

```
$ spark-submit pi.py
```

Word Count using Spark

*text_file = sc.textFile("./sample.txt") #Can be a hdfs://.. path
also*

*counts = text_file.flatMap(lambda line: line.split(" ")) \
 .map(lambda word: (word, 1)) \
 .reduceByKey(lambda a, b: a + b)*

*counts.saveAsTextFile("./count_output") # can be a hdfs://...
directory/path also*

*# \$ cat count_output/**

\$ ls -lrt count_output : Multiple part files

Loads of examples:

<http://spark.apache.org/examples.html>

<https://spark.apache.org/docs/latest/sql-getting-started.html>

<https://spark.apache.org/docs/latest/mllib-guide.html>

<https://github.com/apache/spark/tree/master/examples/src/main/python>

