Submit Apache Spark Applications Lab

Estimated time needed: 20 minutes

In this lab, you will learn how to submit Apache Spark applications from a python script. This exercise is straightforward thanks to Docker Compose.

Learning Objectives

In this lab, you will:

- Install a Spark Master and Worker using Docker Compose
- Create a python script containing a spark job
- Submit the job to the cluster directly from python (Note: you’ll learn how to submit a job from the command line in the Kubernetes Lab)

Prerequisites

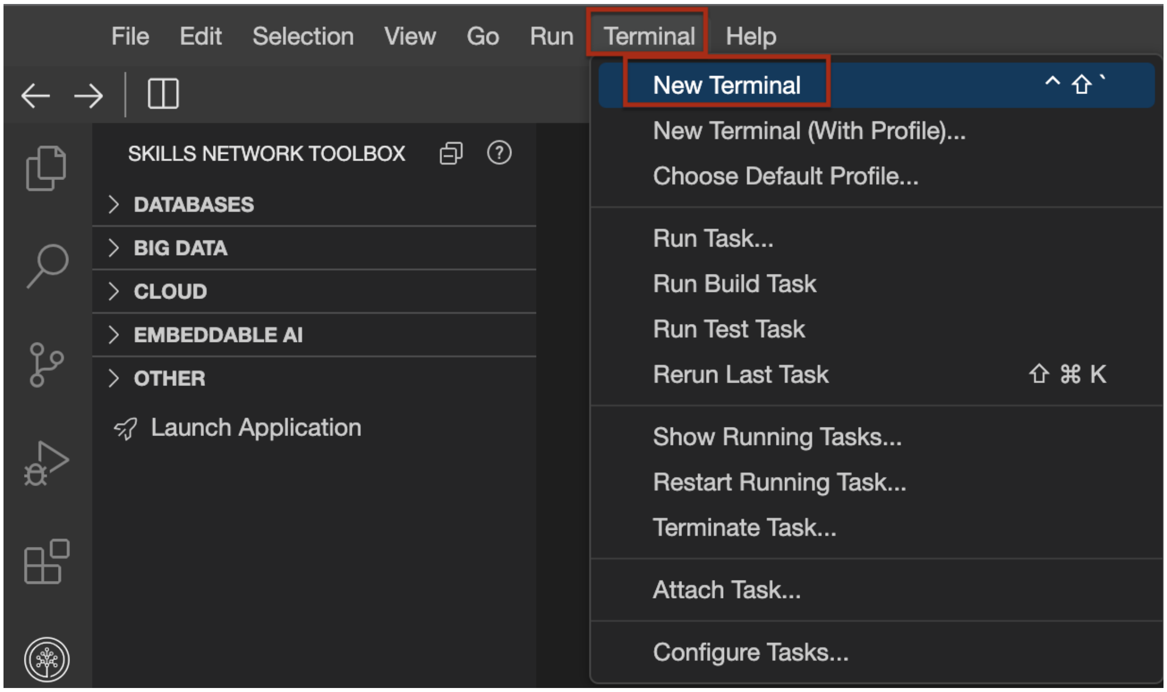
Note: If you are running this lab within the Skillsnetwort Lab environment, all prerequisites are already installed for you

The only pre-requisites to this lab are:

- A working *docker* installation
- Docker Compose
- The *git* command line tool
- A python development environment

Install a Apache Spark cluster using Docker Compose

On the right hand side to this instructions you’ll see the Cloud IDE. Select the *Lab* tab. On the menu bar select *Terminal>New Terminal*.



Get the latest code:

```
1. 1

1. git clone https://github.com/big-data-europe/docker-spark.git
```

Copied! Executed!

Change the directory to the downloaded code:

```
1. 1

1. cd docker-spark
```

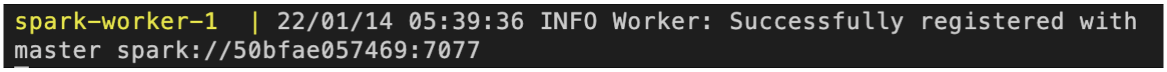
Copied! Executed!

Start the cluster

```
1. 1

1. docker-compose up
```

Copied! Executed!

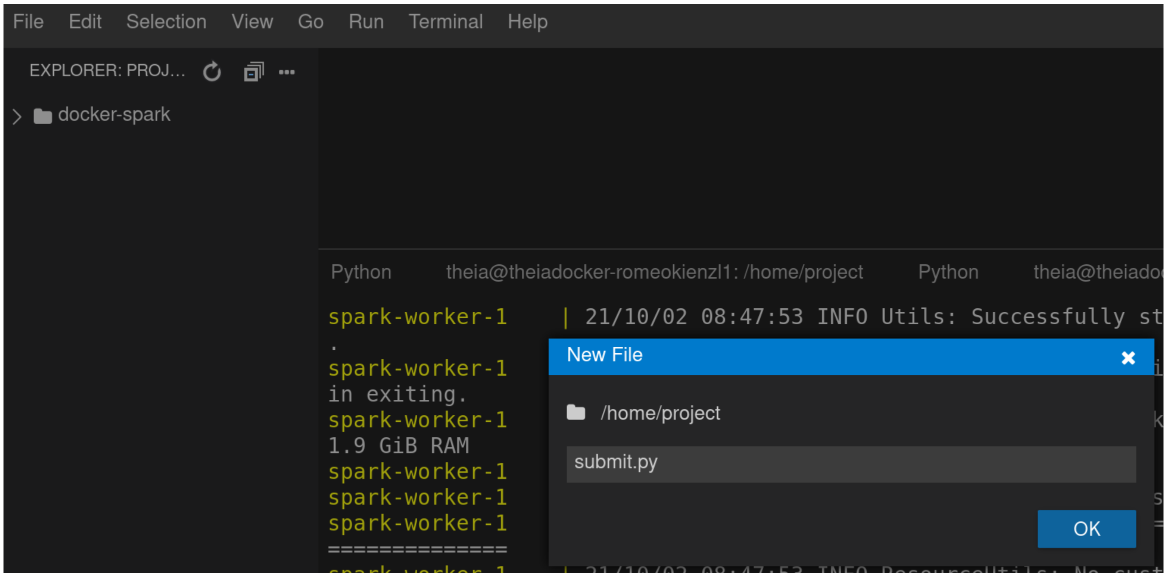
After quite some time you should see the following message:
Successfully registered with master spark://<server address>:7077



```
spark-worker-1  | 22/01/14 05:39:36 INFO Worker: Successfully registered with
master spark://50bfae057469:7077
```

Create code

Create a new python file called *submit.py*



```
spark-worker-1    | 21/10/02 08:47:53 INFO Utils: Successfully st
.
spark-worker-1
in exiting.
spark-worker-1
1.9 GiB RAM
spark-worker-1
spark-worker-1
spark-worker-1
==============
```

submit.py

Paste the following code to the file and save.

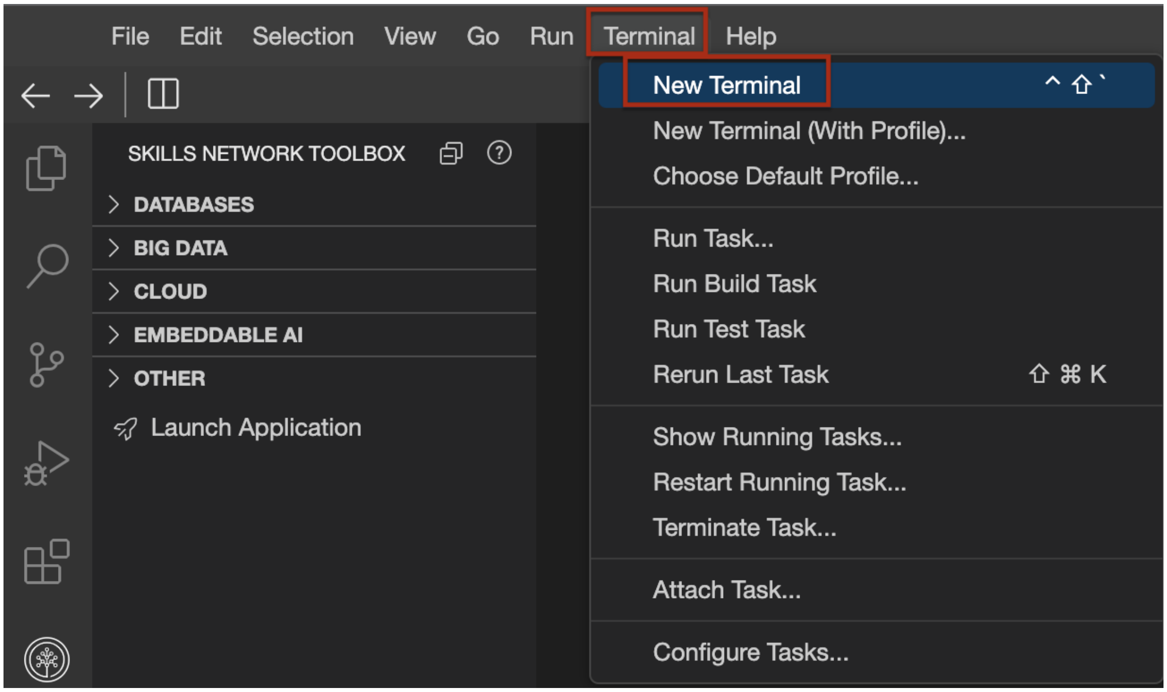
```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
```

```
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.types import StructField, StructType, IntegerType, StringType

sc = SparkContext.getOrCreate(SparkConf().setMaster('spark://localhost:7077'))
sc.setLogLevel("INFO")

spark = SparkSession.builder.getOrCreate()

spark = SparkSession.builder.getOrCreate()
df = spark.createDataFrame(
    [
        (1, "foo"),
        (2, "bar"),
    ],
    StructType(
        [
            StructField("id", IntegerType(), False),
            StructField("txt", StringType(), False),
        ]
    ),
)
print(df.dtypes)
df.show()
```

Copied!

Execute code / submit Spark job

Now we execute the python file we saved earlier. Click on “Terminal” on the bar on top and click on “New Terminal” as shown in the image below.



Once the terminal opens up at the bottom of the window, run the following commands to install these must-have packages and setup your environment to work on pySpark.

```
1
2
3
4
5
6
```

```
pip install --upgrade distro-info
pip3 install --upgrade pip==23.2.1

pip install virtualenv
virtualenv pysparkenv
source pysparkenv/bin/activate
```

Copied! Executed!

Please enter the following commands in the terminal:
Install *PySpark*:

```
1
```

```
1. pip install pyspark
```

Copied! Executed!

Once the terminal opens up at the bottom of the window, please type in the following command in the terminal to execute the Python script:

```
1. 1
```

```
1. python3 submit.py
```

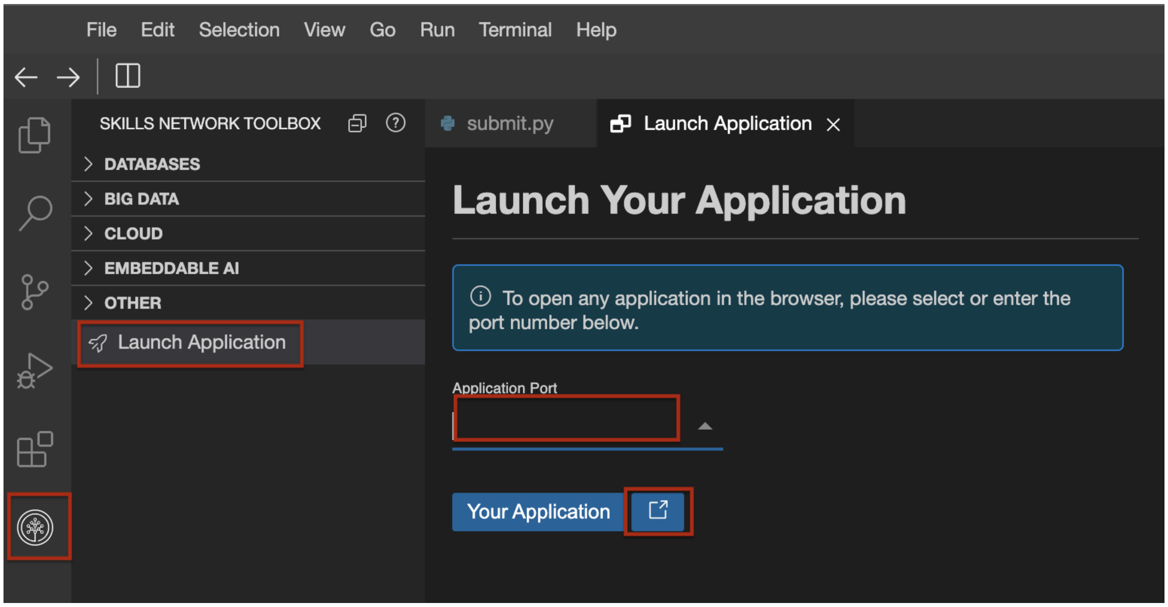
Copied! Executed!

Experiment yourself

Please have a look at the UI of the Apache Spark master and worker.

Using the *Launch Application* button you can gain access to it. Start with port 8080 first (Spark master)

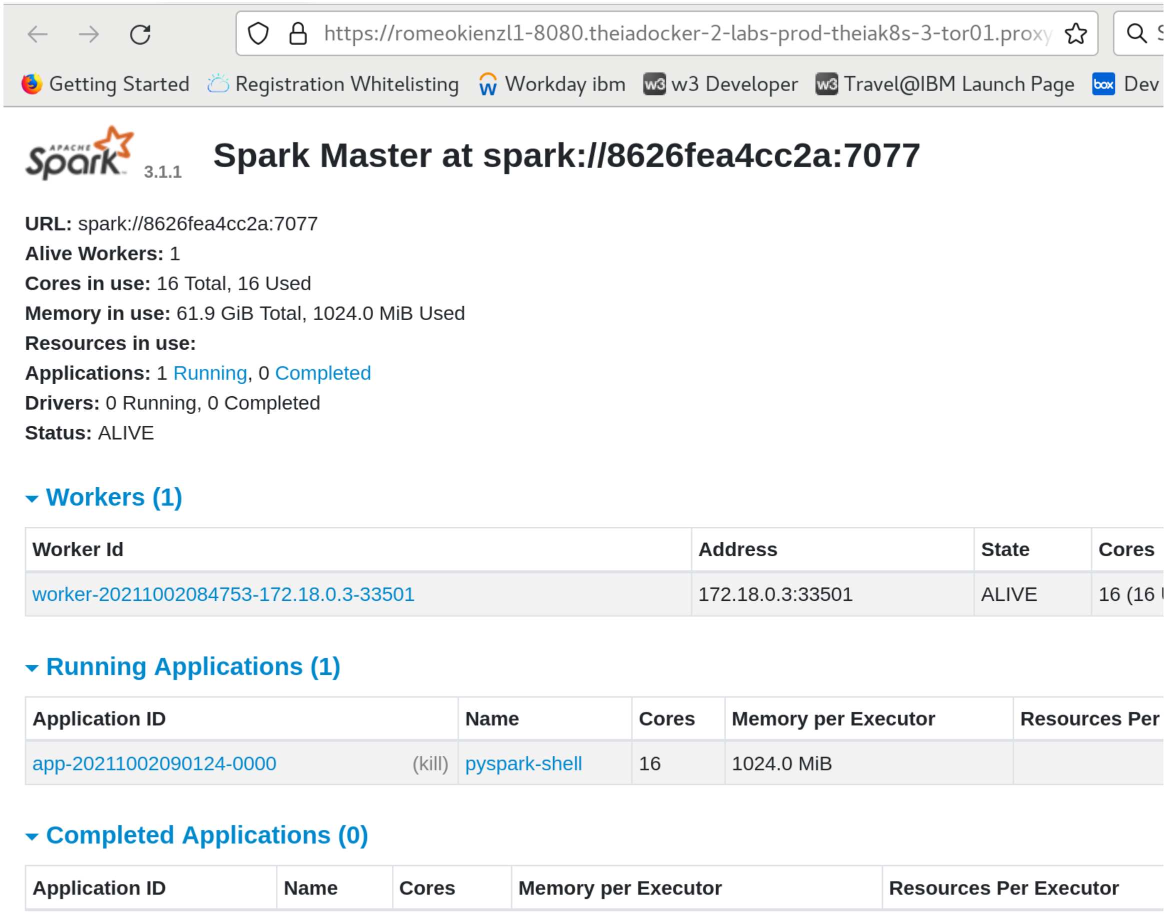
Click on the Skills Network button on the left, it will open the “Skills Network Toolbox”. Then click the Other then Launch Application. From there you should be able to enter the port number as `8080` and launch.



Launch Your Application

To open any application in the browser, please select or enter the port number below.

This will take you to the admin UI of the Spark master (if your popup blocker doesn’t prevent it):


https://romeokienzl1-8080.theiadocker-2-labs-prod-theiak8s-3-tor01.proxy
Getting Started
Registration Whitelisting
Workday ibm
w3 Developer
Travel@IBM Launch Page
Dev
3.1.1

Spark Master at spark://8626fea4cc2a:7077

URL: spark://8626fea4cc2a:7077
Alive Workers: 1
Cores in use: 16 Total, 16 Used
Memory in use: 61.9 GiB Total, 1024.0 MiB Used
Resources in use:
Applications: 1 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores
worker-20211002084753-172.18.0.3-33501	172.18.0.3:33501	ALIVE	16 (16

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per
app-20211002090124-0000 (kill)	pyspark-shell	16	1024.0 MiB	

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor

Please notice that you can see all registered workers (one in this case) and submitted jobs (also one in this case)

> Note: The way how the lab environment works you can't click on links in the UI - in a real installation, this of course is possible.

Please repeat the steps above, re-submit the job and open the Spark worker UI by launching the application. Just use port 8081 instead of 8080 this time.

You should find your currently running job here as well.

Summary

In this lab you've learned how to setup an experimental Apache Spark cluster on top of Docker Compose. You are now able to submit a Spark job directly from python code. In the Kubernetes lab you'll learn how to subit Spark jobs from command line as well.

Changelog

Date	Version	Changed by	Change Description
October 2021	1.0	Romeo Kienzler	Initial version
01-09-2022	1.1	K Sundararajan	Updated instructions for `Launch Application` as per new Theia IDE & the format for `Changelog`
30-12-2022	1.2	K Sundararajan	Updated `pyspark` installation command