Submit Apache Spark Applications Lab

Skills Network

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

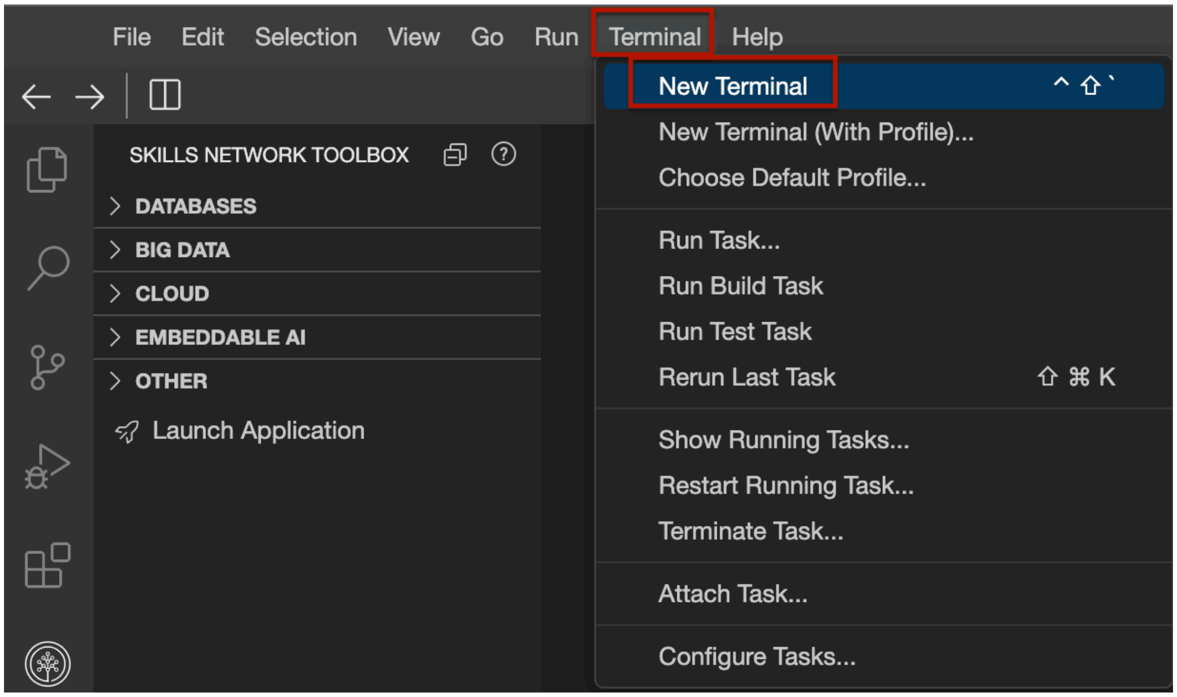
Note: If you are running this lab within the Skillsnetwort Lab environment, all prerequisites are already installed for you

The only pre-requisites to this lab are:

- The *wget* command line tool
- A python development environment

Start the Spark Master

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*.



2. Please enter the following commands in the terminal to download the spark environment.

```
wget https://archive.apache.org/dist/spark/spark-3.3.3/spark-3.3.3-bin-hadoop3.tgz && tar xf spark-3.3.3-bin-hadoop3.tgz && rm -rf spark-3.3.3-bin-ha
```

   This takes a while. This downloads the spark as a zipped archive and unzips it in the current directory.

3. Run the following commands to set up the `JAVA_HOME` which is preinstalled in the environment and `SPARK_HOME` which you just downloaded.

```
export JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64
export SPARK_HOME=/home/project/spark-3.3.3-bin-hadoop3
```

4. Run the following command to create a config file for the master.

```
touch /home/project/spark-3.3.3-bin-hadoop3/conf/spark-defaults.conf
```

5. Click the button below to set up the configuration of your spark master.

Open **spark-defaults.conf** in IDE

6. Paste the following content in the `spark-defaults.conf`. This will set the cores and the memory that the master will have to allocate to the workers.

```
spark.executor.memory 4g
spark.executor.cores 2
```

7. Change to the `SPARK_HOME` directory.

```
cd $SPARK_HOME
```

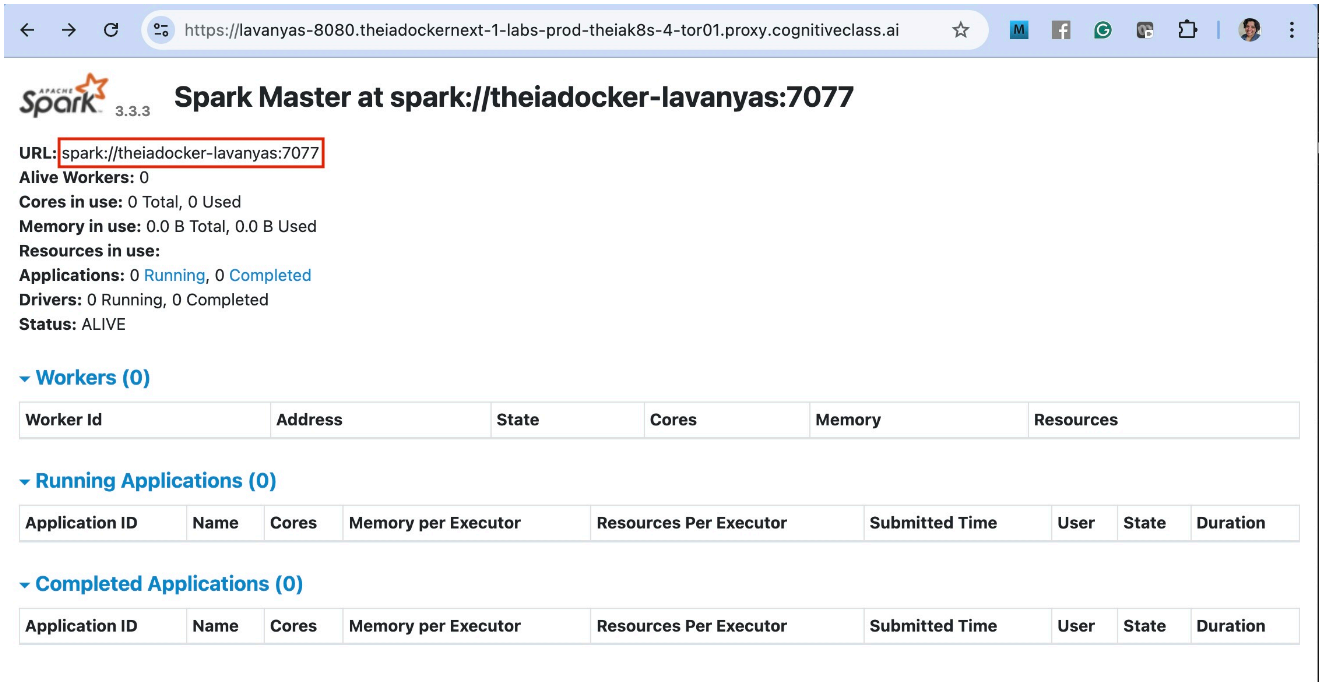
8. Run the spark master by executing the following command.

```
./sbin/start-master.sh
```

9. Once it successfully starts up, click the button below to verify that the master is running as exepcted.

Spark Master

If the application has started up successfully, you will see a page as given below.



https://lavanyas-8080.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai

Spark Master at spark://theiadocker-lavanyas:7077

URL: spark://theiadocker-lavanyas:7077
Alive Workers: 0
Cores in use: 0 Total, 0 Used
Memory in use: 0.0 B Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (0)

Worker Id	Address	State	Cores	Memory	Resources

Running Applications (0)

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

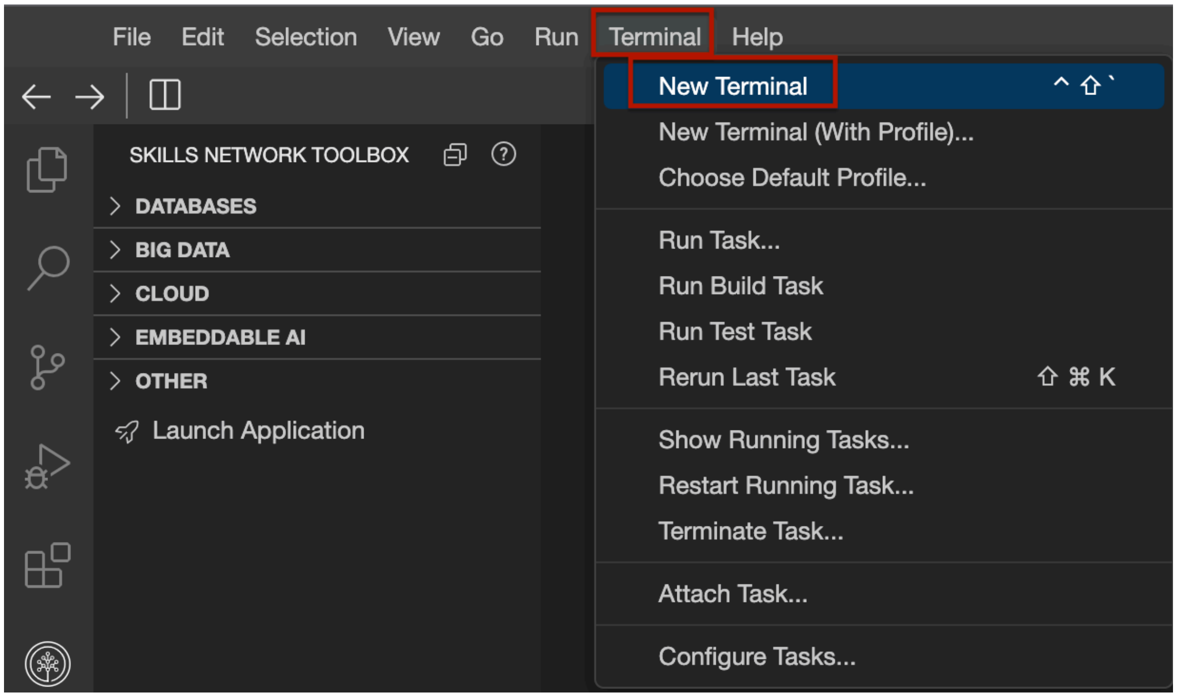
Completed Applications (0)

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

10. Copy the URL of the master as show in the image and note is down in a text editor such as a notepad on your computer.

Start the worker

1. Click `Terminal` from the menu, and click `New Terminal` to open a new terminal window.



2. Once the terminal opens up at the bottom of the window, run the following commands to set up the `JAVA_HOME` and `SPARK_HOME`.

```
export JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64
export SPARK_HOME=/home/project/spark-3.3.3-bin-hadoop3
```

3. Change to the `SPARK_HOME` directory.

```
cd $SPARK_HOME
```

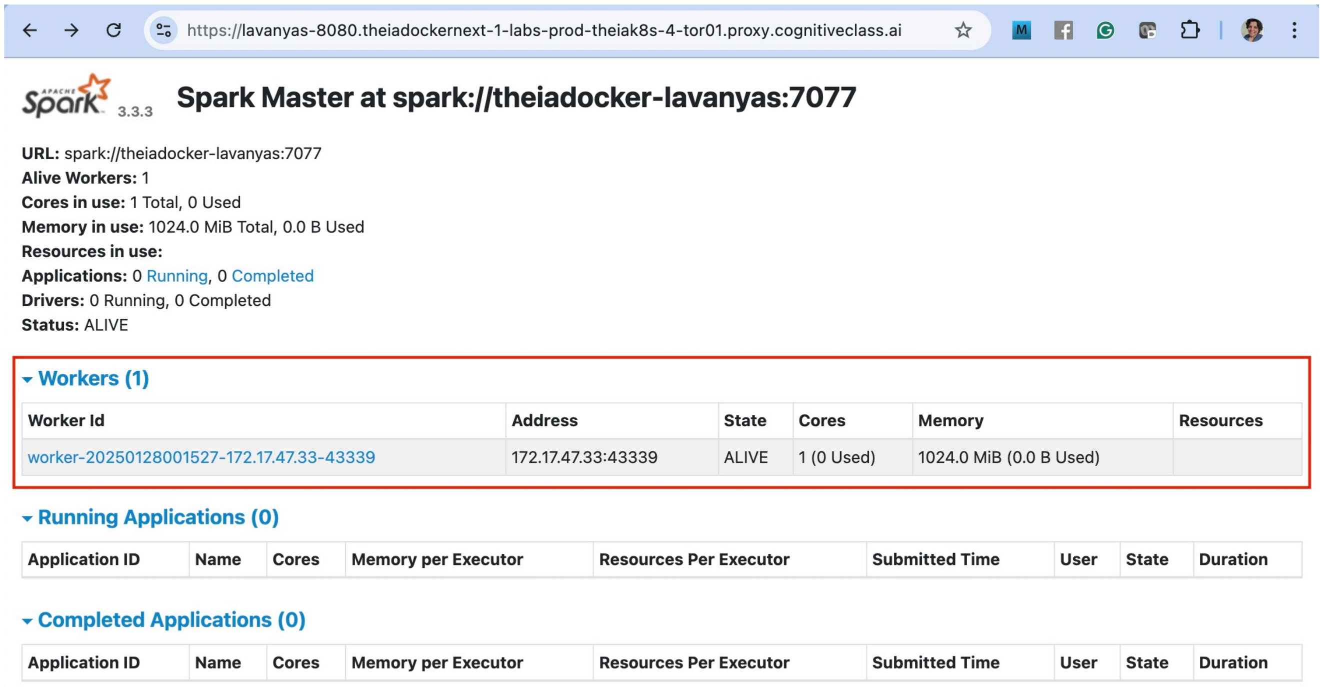
4. Run the spark worker by executing the following command. Remember to replace the placeholder `yourname` in the command below with your name as given in the spark master URL that you noted down in the previous step.

```
./sbin/start-worker.sh spark://theiadocker-yourname:7077 --cores 1 --memory 1g
```

5. Once it successfully starts up, click the button below to verify that the worker is running as exepcted.

Spark Master

You should see that the worker is now registed with the master.


https://lavanyas-8080.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai
3.3.3

Spark Master at spark://theiadocker-lavanyas:7077

URL: spark://theiadocker-lavanyas:7077
Alive Workers: 1
Cores in use: 1 Total, 0 Used
Memory in use: 1024.0 MiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20250128001527-172.17.47.33-43339	172.17.47.33:43339	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	

Running Applications (0)

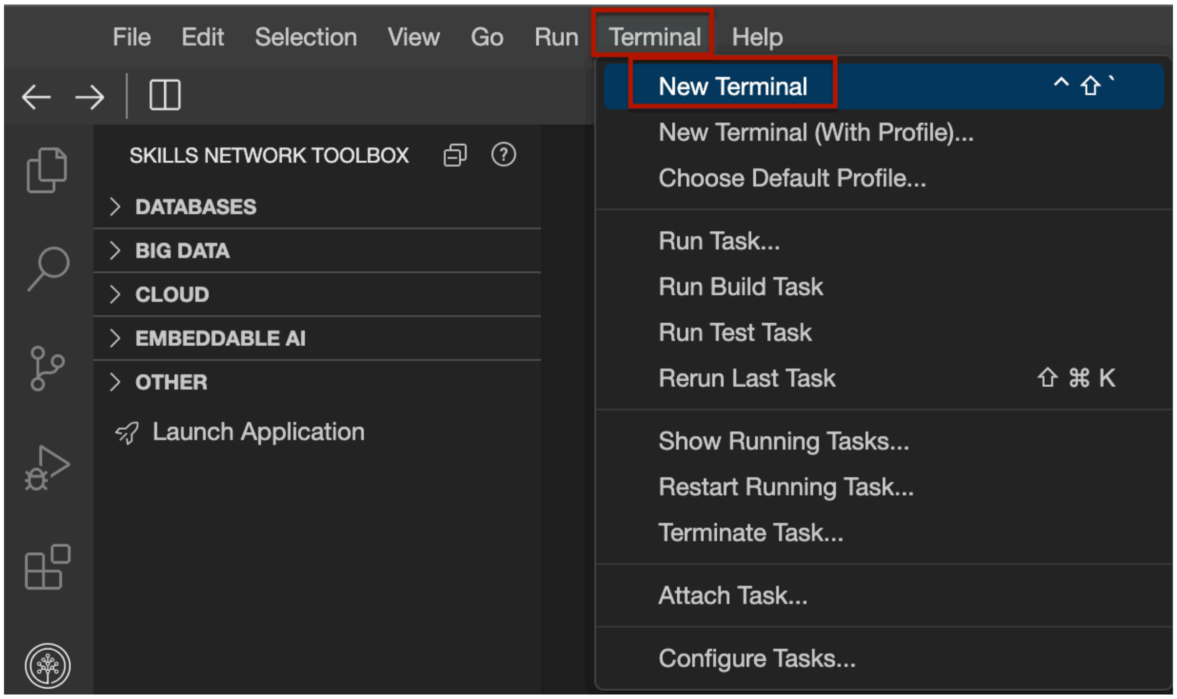
Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (0)

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

Create code and submit

1. Click `Terminal` from the menu, and click `New Terminal` to open a new terminal window.



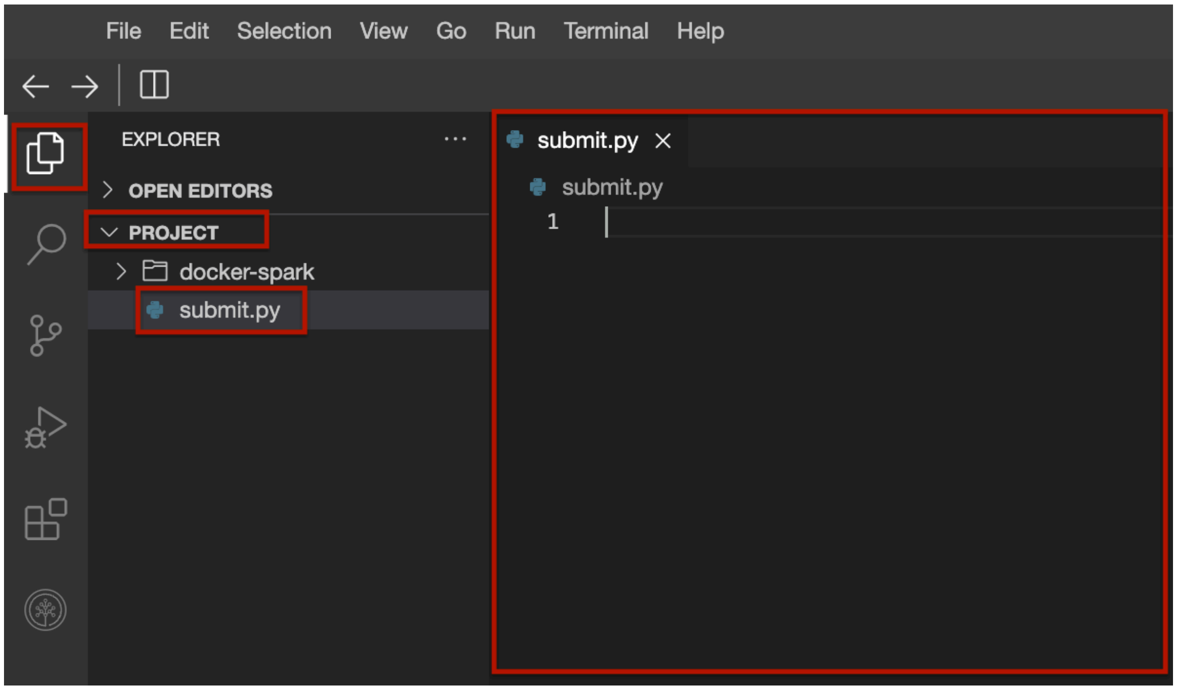
2. Once the terminal opens up at the bottom of the window,run the following command to create the pyton file.

```
touch submit.py
```

A new python file called `submit.py` is created.

3. Open the file in the file editor by click the button below or following the visual guidance in the image.

Open **submit.py** in IDE



4. Paste the following code to the file and save. Remember to replace the placeholder `yourname` in the code below with your name as in the spark master URL.

```
import findspark
findspark.init()
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.types import StructField, StructType, IntegerType, StringType
spark = SparkSession.builder \
    .master('spark://theiadocker-yourname:7077') \
    .config('spark.executor.cores', '1') \
    .config('spark.executor.memory', '512m') \
    .getOrCreate()
df = spark.createDataFrame(
    [
        (1, "foo"),
        (2, "bar"),
    ],
    StructType(
        [
            StructField("id", IntegerType(), False),
            StructField("txt", StringType(), False),
        ]
    ),
)
print(df.dtypes)
print("\nDataFrame:")
df.show()
```

3. Run the following commands to set up the `JAVA_HOME` and `SPARK_HOME`.

```
export JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64
export SPARK_HOME=/home/project/spark-3.3.3-bin-hadoop3
```

4. Install the required packages to set up the spark environment.

```
pip3 install findspark
```

5. Type in the following command in the terminal to execute the Python script.

```
python3 submit.py
```

You will see output as below:

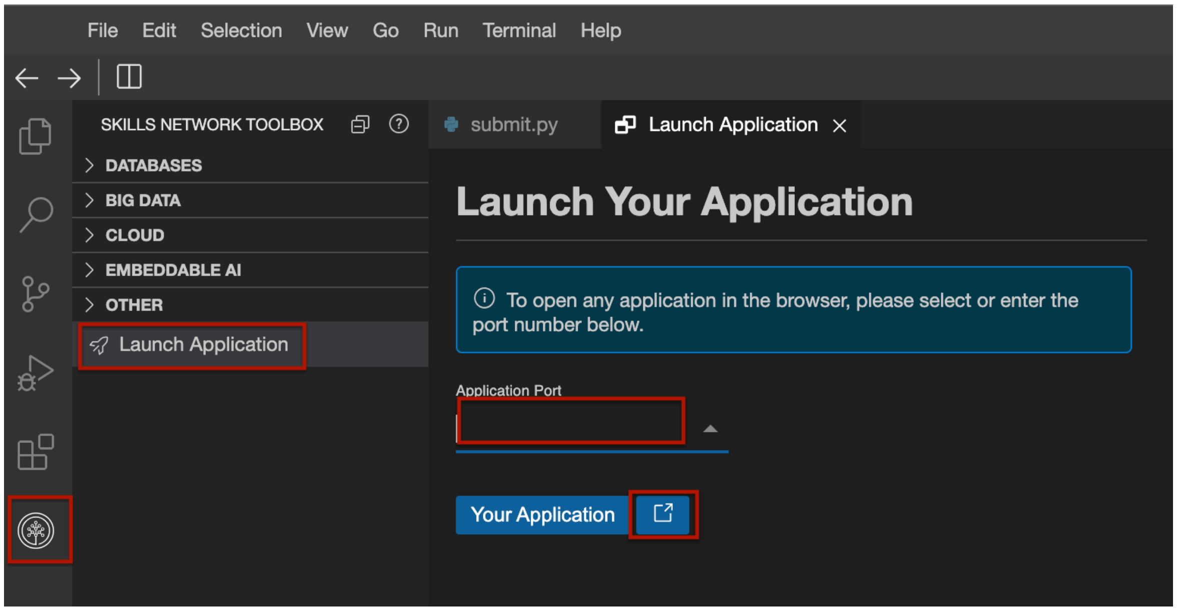
```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/01/27 23:50:53 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[('id', 'int'), ('txt', 'string')]
+---+---+
| id|txt|
+---+---+
|  1|foo|
|  2|bar|
+---+---+
```

Experiment yourself

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

1. Click on the button below to launch the `Spark Master`. Alternatively, 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.

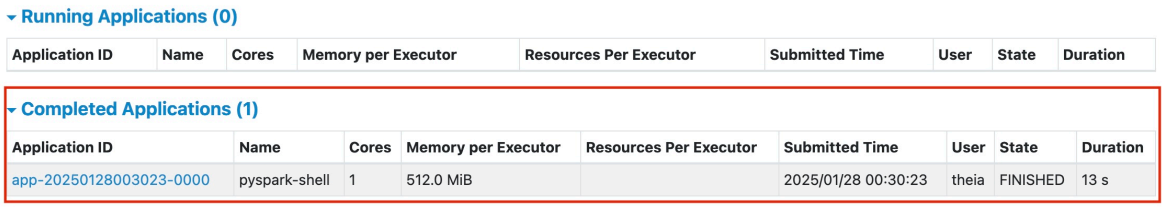
Spark Master



Launch Your Application

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

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



Running Applications (0)

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

Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20250128003023-0000	pyspark-shell	1	512.0 MiB		2025/01/28 00:30:23	theia	FINISHED	13 s

3. Please notice that you can see all registered workers (one in this case) and running/completed 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.

4. Click the button below to open the `Spark Worker` on 8081. Alternatively, 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 `8081` and launch.

Spark Worker

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.

Author(s)

Romeo Kienzler
Lavanya T S