Submit Apache Spark Applications Lab

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:

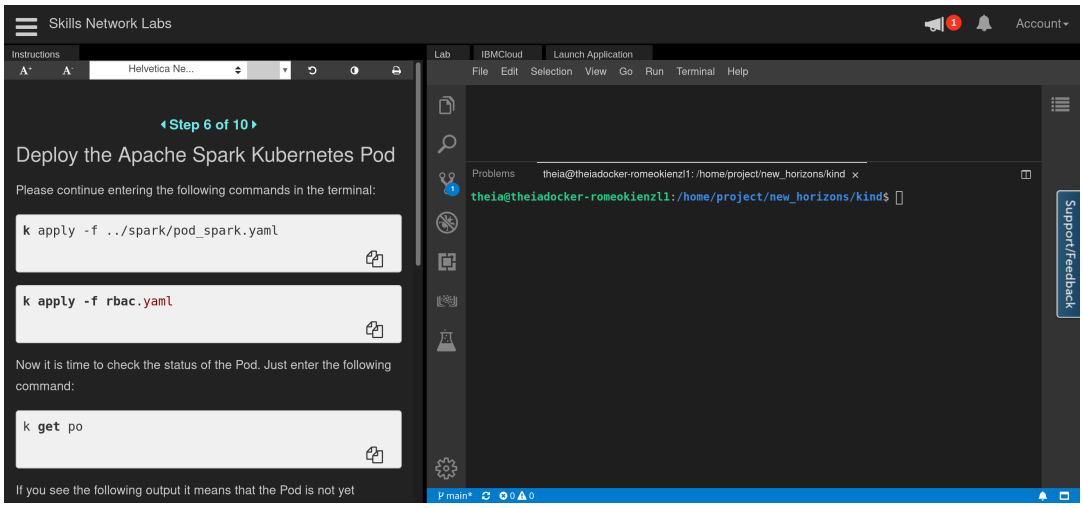
- A working *docker* installation
- Docker Compose
- The *git* command line tool
- A python development environment

Project Overview

Welcome to the lab on how to submit Apache Spark applications from a python script. This exercise is straightforward thanks to Docker Compose.

Install a Apache Spark cluster using Docker Compose

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



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

```
pip3 install pyspark
```

Get the latest code:

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

Change the directory to the downloaded code:

```
cd docker-spark
```

Start the cluster

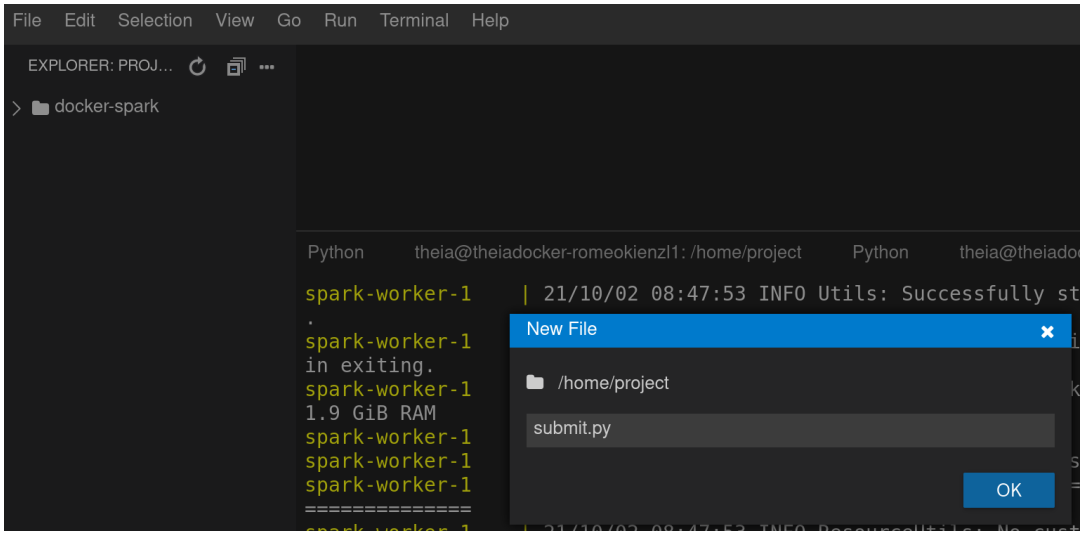
```
docker-compose up
```

After quite some time you should see the following message: *Successfully registered with master spark://: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*

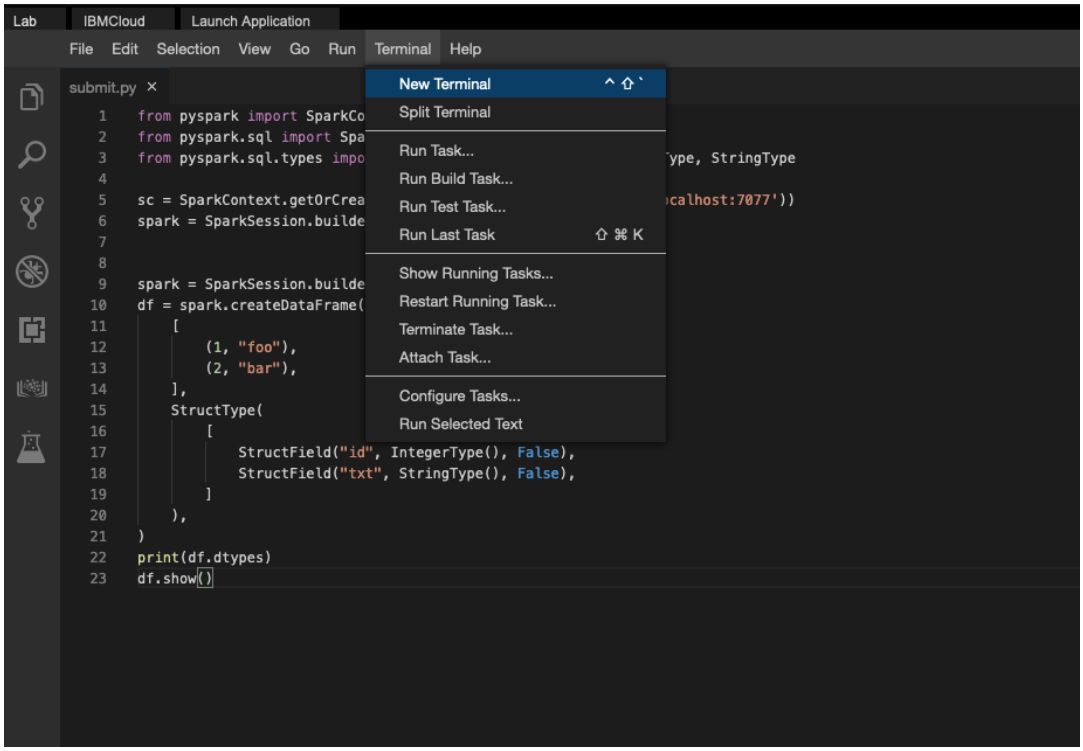


Paste the following code to the file and save.

```
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()
```

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.



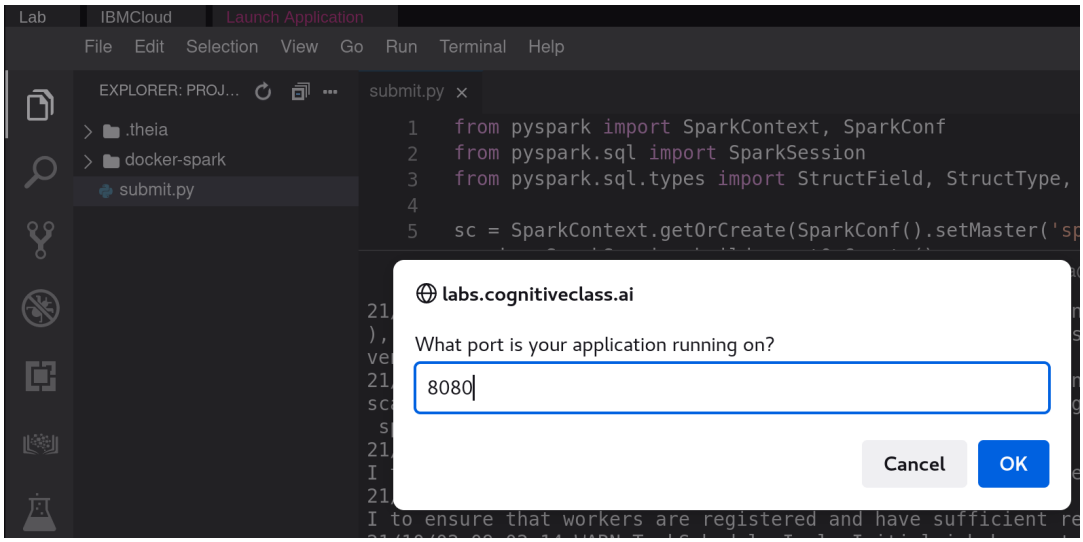
Once the terminal opens up at the bottom of the window, please type in the following command in the terminal to execute the Python script:

```
python3 submit.py
```

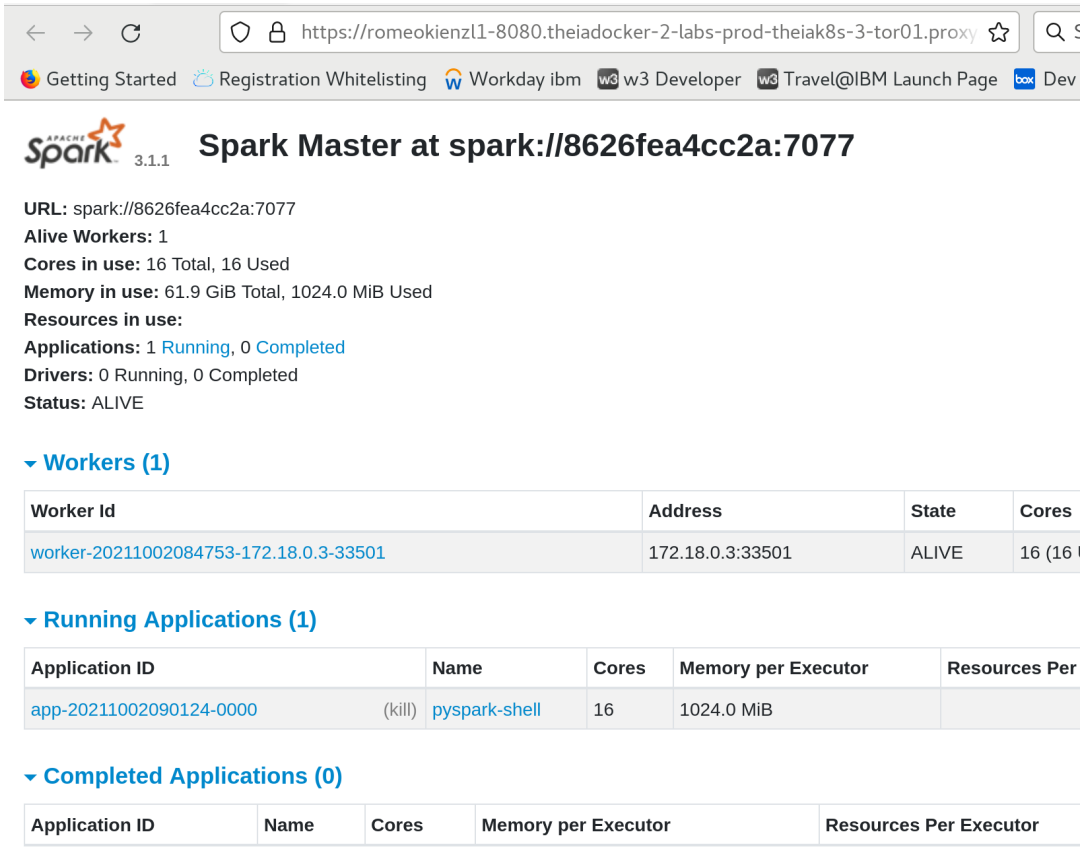
Experiment yourself

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)



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



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 using the *launch application* button. Just use port 8081 instead of 8080 this time.

You should fine 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.

Change Log

Romeo Kienzler, Initial version, October 2021