Lab 1

This introductory lab is composed of two main tasks. Your first objective is to run your first MapReduce application on the BigData@Polito cluster. For this goal, you must learn how to import a predefined project, compile the source code, produce a jar file, copy your application on a remote machine (the gateway) and submit your application on the BigData@Polito cluster. The second objective is to write your first MapReduce application.

1. Compiling using the Ecplise IDE

In this task, we will "compile" the source code of a simple Hadoop application and we will create a jar file with the class files of the project. The shared Eclipse project contains the basic libraries that are needed to develop the application and create the class and jar files.
You cannot use this project to run locally on the PC of the Lab or on your own PC the MapReduce application (other libraries are needed to run locally this application).

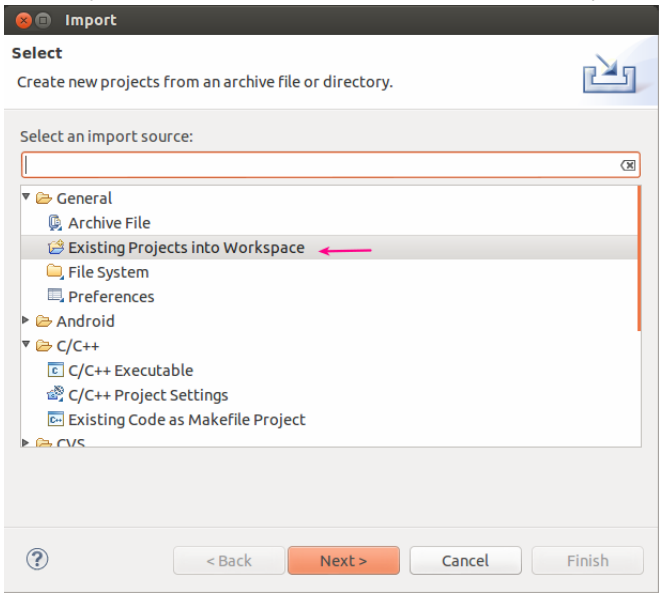
The imported project is the MapReduce implementation of the word count application.

Download from the course web page the zip file Lab1_BigData_with_libraries.zip, which contains the MapReduce-based implementation of the word count application and the example data folder example_data (direct link: http://dbdmg.polito.it/dbdmg_web/wp-content/uploads/2021/10/Lab1_BigData_with_libraries.zip).

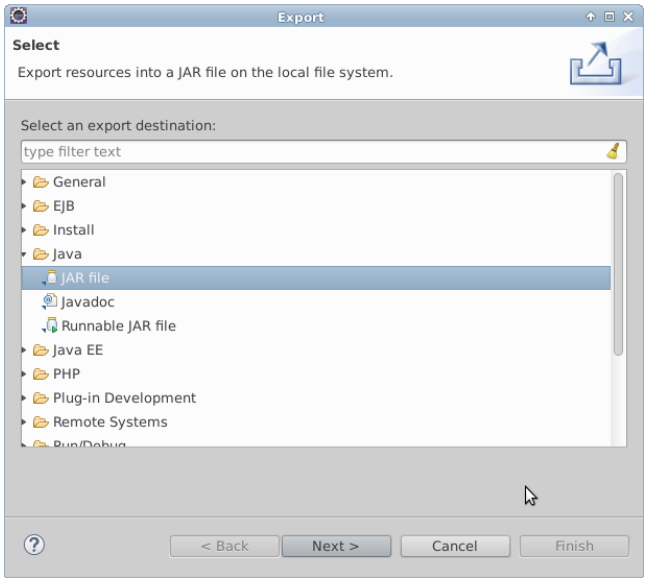
Decompress the zip file inside a local folder on the PC of the LAB or on your PC.

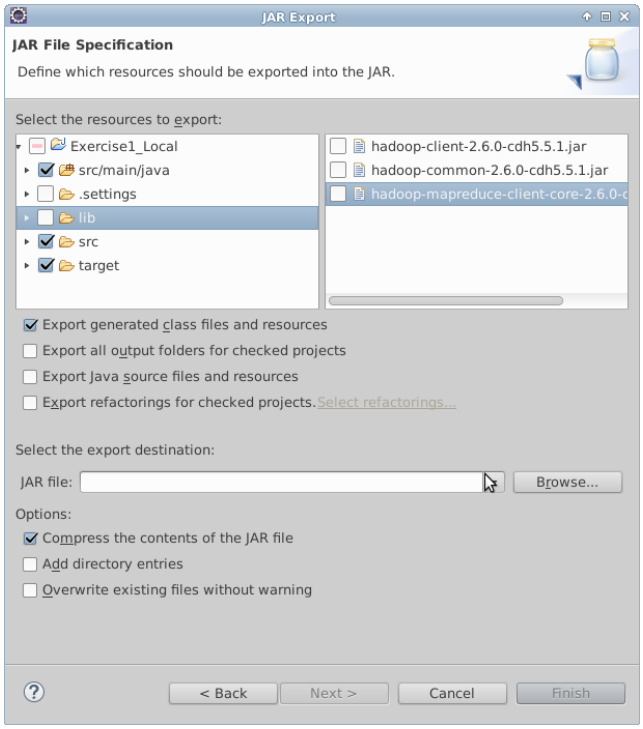
Import the project inside Eclipse and create the jar file as follows.

1. Open Eclipse
2. Import the project (File -> Import… | General -> Existing project... | choose the folder where you extracted the content of the zip file)



3. Have a look at the source files and the structure of the project. Where is the mapper? Where is the reducer?
4. Since we cannot use maven on the PC of the LAB, we cannot count on it to generate the jar file. You can instead build a jar manually using the “File -> Export” command, as in the following two figures. Remember to avoid inserting all the libraries in your jar, or you would end up producing a fat jar heavy to transfer. We need these libraries locally to compile, but they are already present in the classpath of the cluster and there is no need to include them in the jar again.



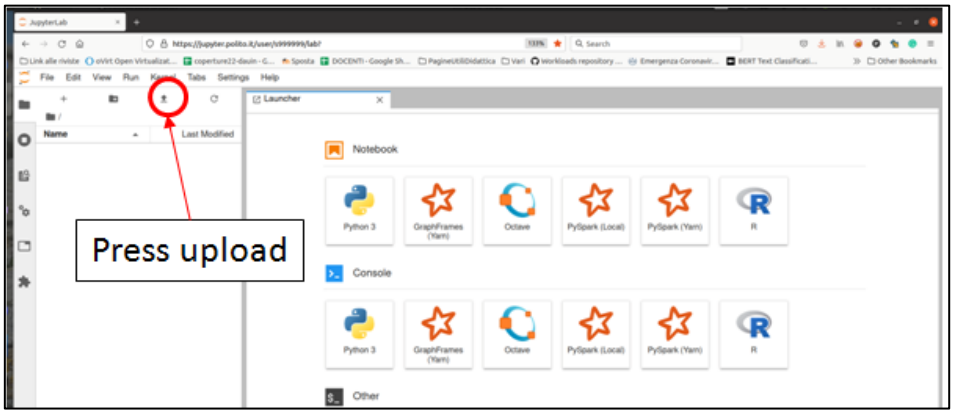


5. Specify the name of the output jar file (e.g., set the JAR file textbox to Exercise1-1.0.0.jar).

2. Upload your application on BigData@Polito

The objective of this task is to upload your application on the cluster BigData@Polito.

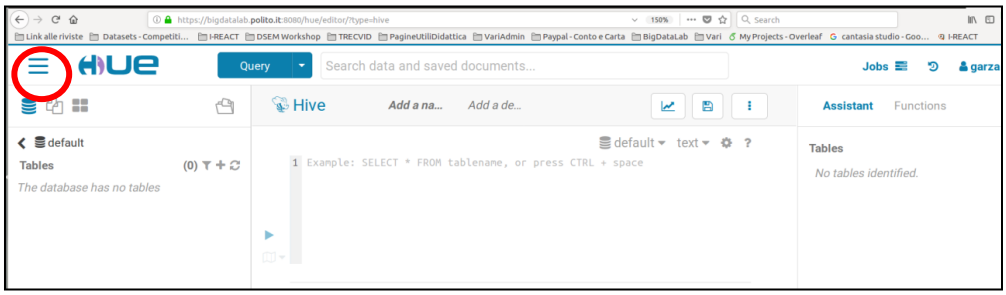
1. Connect to https://jupyter.polito.it
2. Upload the jar file containing your application on the local file system of the gateway



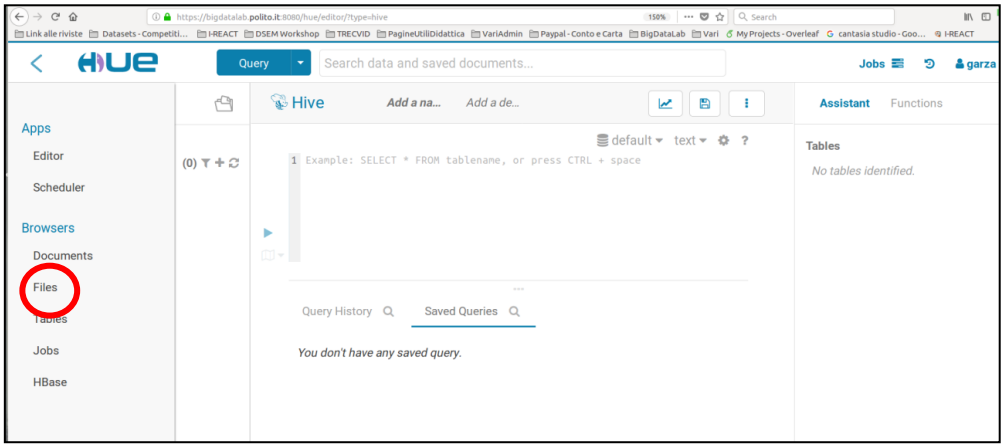
3. Manage HDFS through the HUE web interface

In this task, you will learn how to do basic management of the HDFS file system. To this goal, we will use a web interface called HUE.

1. Go to https://hue.polito.it/ and login with your usual BigData@Polito credentials.
2. Go to the "Browsers/Files" tab. You should find **your HDFS home**, as shown below. Note that this is not the same file system as in task 2 (i.e., it is not the local file system of the gateway), so you will find probably an empty folder now.[1]
   Your **HDFS home** is not located on the gateway, but is stored in the Hadoop cluster.



[1] If the difference between the two "homes" is not clear to you at this point, do not keep on. Spend some time to clarify your ideas.

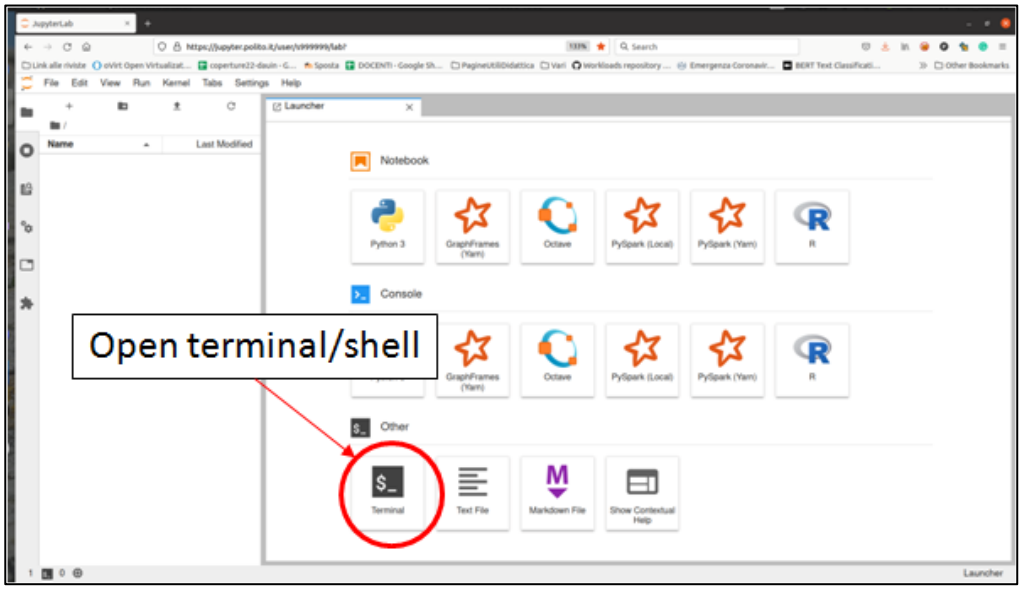


3. Create the folder example_data on HDFS
4. Upload the sample files from the local folder example_data available on your PC to the folder example_data on HDFS.
5. Find out on your own how to delete/move the files, or download them. It will be helpful in the next labs.

4. Submit a job

Now we have everything we need to submit our sample application. It is finally time to open a shell.

1. Connect to https://jupyter.polito.it
2. Open a terminal clicking on "Terminal" inside the Launcher area.



3. Now that you have a terminal on the gateway, launch a MapReduce job (i.e., run your application on the cluster) using the following command:

```
hadoop jar Exercise1-1.0.0.jar
it.polito.bigdata.hadoop.DriverBigData 2 example_data ex1_out
```

where:

- "**Exercise1-1.0.0.jar**" is the JAR file containing your application

- "**example_data**" is the input HDFS folder. A relative path starts in your home in HDFS. You can also use an absolute path in HDFS.
- "**ex1_out**" is the output folder in HDFS, not on the gateway local file system. You can see its content in HUE (a relative path starts in your home in HDFS)

4. Check the number of mappers (i.e., the number of instances of the mapper class)
   - You can retrieve the information about the number of mappers (map tasks) and other statistics in the information showed on the terminal during the execution of your application (last part of the showed information).
5. Find your job on HUE interface (JobBrowser tab), check its status (submitted, running, failed or succeeded).
6. Find the output file on HDFS (from HUE file browser) and see the results associated with the execution of your application.
7. Try to re-run the same job. Does it succeed this time? What's the problem?
   - To remove a folder from HDFS, you can use HUE or you can use the hdfs command line tool executing the following command in the terminal you opened on jupyter.polito.it:
     hdfs dfs -rm -r <path of the HDFS folder you want to delete>
8. Run again the application on the cluster by changing the number of reducers (first parameter of the application) and analyze the content of the output folder of the HDFS file system.

If you need to access the log files associated with the execution of your application by using the command line, use the following commands in the terminal of jupyter.polito.it:

1. To retrieve the log associated with the standard output
   - yarn logs -applicationId <id of your application> -log_files stdout
     - The "id of your application" is printed on the terminal at the beginning of the execution of your application
       - The format of "id of your application" is application_number_number
       - Example of "application id" application_1584304411500_0009
     - You can retrieve the application id also from the HUE interface
   - The returned result contains one stdout log section for each task (driver, map and reduce tasks)
     - One for the Driver
     - One for each Mapper
     - One for each Reducer

2. To retrieve the log associated with the standard error
   - yarn logs -applicationId <id of your application> -log_files stderr

Test on a bigger file

Now you will execute your application on a larger file that we already uploaded on the HDFS file system of BigData@Polito.
The absolute path of that file in HDFS is the following:
/data/students/bigdata-01QYD/Lab1/finefoods_text.txt
It is a large collection of Amazon reviews in the food category. Each line of finefoods_text.txt contains one review.

1. Launch your application on the Amazon review file:
    a. Set the second parameter of your application to /data/students/bigdata-01QYD/Lab1/finefoods_text.txt
2. Analyze the results.
    a. Can you understand any interesting facts from your results?
    b. Do you see any space for improvements in your analysis?
3. The following figure was done on a small sample of your data (10000 reviews).
    a. Is it consistent with what you found on the complete dataset?
    b. Do you think a small sample is enough to represent the whole?

5. Bonus track

A word count can be seen as a special case of an n-gram count, where n is equal to 1. n-grams, in our context, are sets of contiguous words of length n.

For example, in the sentence “She sells seashells by the seashore“, 2-grams are: “She sells”, “sells seashells”, “seashells by”, “by the”, “the seashore”.

Modify your word count program to count 2-grams frequencies. Consider each line of text as a separate document, as in the Amazon reviews file (so, do not count as contiguous words on separate lines).

1. What is the time/space complexity of your program?
2. How long would you expect it to run, compared to the simple word count?
3. Try to run it on the toy text and on the Amazon reviews.