

# Lab 1

This introductory lab is composed of three tasks. Your final objective is to run your first Hadoop application locally on your PC and then on the BigData@Polito cluster. For this goal, you must learn how to import a maven project, compile the source code, produce a jar file, copy your application on a remote machine and submit your application on the BigData@Polito cluster.

Please note:

- to run your application locally, inside Eclipse, Linux or macOS are suggested (Windows users can use a virtual machine running Linux or try to set up their windows environment by following the instructions available on the course web page: [ConfigureWindowsEnvironment.pdf](#))
- to import a maven-based project in Eclipse, write your code, compile your application, produce a jar file and submit/execute your application on the BigData@Polito cluster also Windows can be used

## 1. Import a maven project in Eclipse and run your application locally with Eclipse

In this task, we will import a maven-based Java project into Eclipse, compile it and run it locally on your PC to analyze the content of the local folder `example_data`. The imported project is the MapReduce implementation of the word count application.

1. Download from the course web page the zip file `Lab1.zip`, which contains the MapReduce-based implementation of the word count application and the data folder `example_data`
  - a. Direct link:  
<https://dbdmg.polito.it/wordpress/wp-content/uploads/2020/03/Lab1.zip>
  - b. Use the file `Lab1Windows.zip` if you plan to run your application locally by using Windows:  
<https://dbdmg.polito.it/wordpress/wp-content/uploads/2020/03/Lab1Windows.zip>
2. Watch the following video to learn how to import the content of `Lab1` in Eclipse and run the `Lab1` application locally on your PC:
  - a. Direct link:  
[https://www.dropbox.com/s/niilmhv6k1130dt/01\\_ImportProject\\_LocalRun.mp4?dl=0](https://www.dropbox.com/s/niilmhv6k1130dt/01_ImportProject_LocalRun.mp4?dl=0)
3. Have a look at the source files and the structure of the project. Run the application by changing the number of reducers (first parameter of the application) and **analyze the number of output files with respect to the number of reducers.**

## 2. Run your application on BigData@Polito

The objective of this task is to run your application on the cluster BigData@Polito (that is an Hadoop cluster) on the same data you used in the previous task.

First, you must produce a jar file containing your application and copy it on the local file system of the gateway (a server that is connected with the Hadoop cluster and can be used to submit MapReduce jobs on the Hadoop cluster). Then, you must upload the data you want to analyze in the distributed HDFS file system of Hadoop. Finally, you can run your application on the cluster to analyze the data you stored on the distributed file system.

1. Watch the following video to learn how to create a jar file containing your application by using Eclipse + maven, how to copy data on HDFS and how to submit your application on the cluster BigData@Polito:
  - a. Direct link:  
[https://www.dropbox.com/s/65xy3hu9qvqp2oc/02\\_Jar\\_ClusterExecution.mp4?dl=0](https://www.dropbox.com/s/65xy3hu9qvqp2oc/02_Jar_ClusterExecution.mp4?dl=0)
2. **Check the number of mappers** (i.e., the number of instances of the mapper class)
  - a. 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).
3. Run 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.**

To remove a folder from HDFS, by using the hdfs command line tool, you can use the following command:

```
hdfs dfs -rm -r <path of the folder you want to delete>
```

If you need to access the log files associated with the execution of your application by using the command line, use the following commands:



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
  - 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?**



### 3. 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.