

Lab 1.

This introductory lab is composed of three tasks. Your final objective is to run your first Hadoop application. For this goal, you must learn how to compile the source code and produce a jar, connect to a remote machine, transfer files using FTP and submit an application on a cluster.

At LABINF, you have all the software you need already installed in **Linux**.

1. Compiling using an IDE (Eclipse)

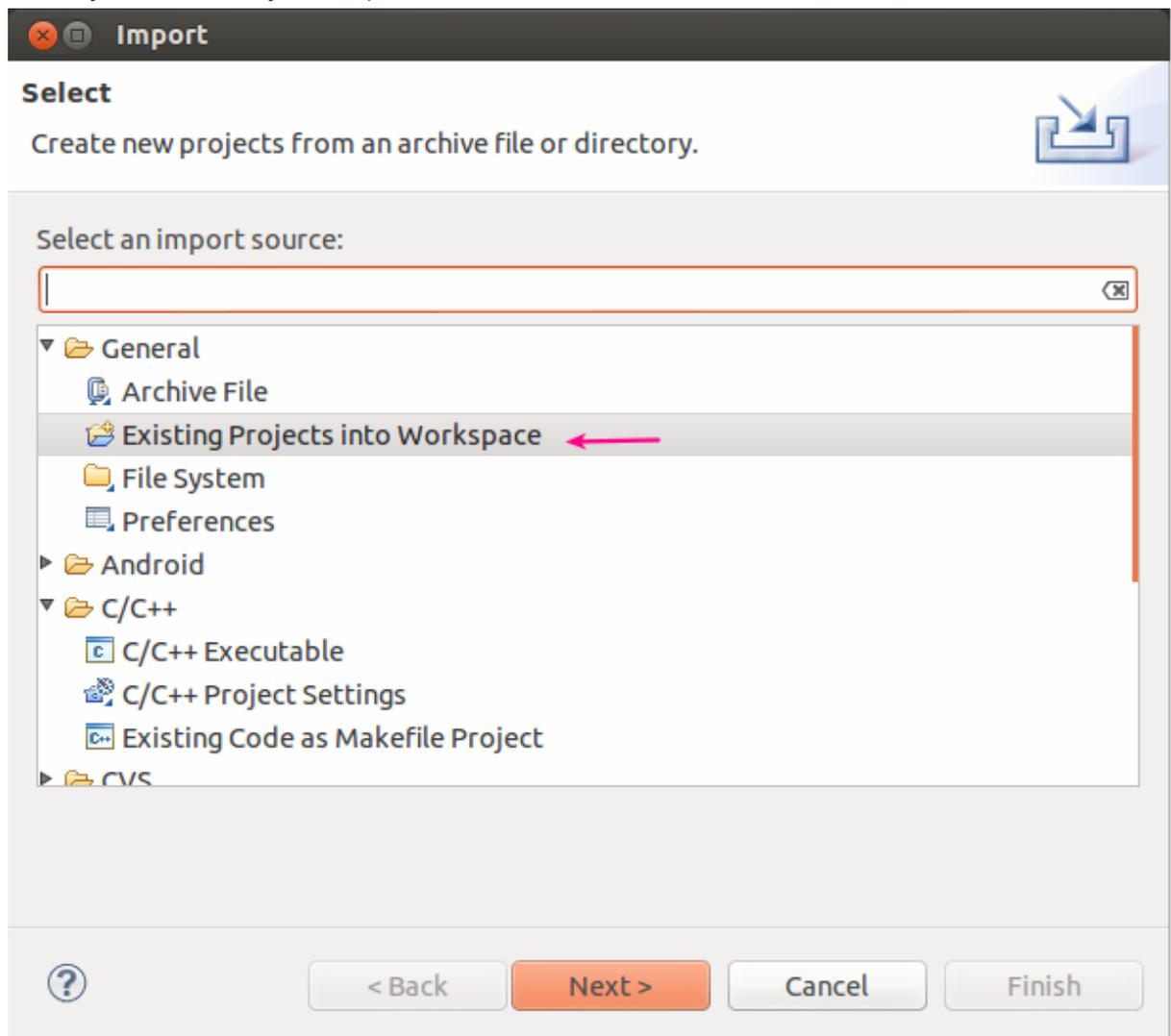
In this task, we will compile the source code of a simple Hadoop application. The easiest way would be using Maven, but here we will guide you through the manual creation of a jar instead. Remember that, if you don't use Maven, you will need to manually include the libraries every time you create a new project.

These modified instructions explain how to import manually the libraries (that we provide to you inside the lib/ folder of the zip). If you do not have enough free space in your home, unzip the libraries to a folder inside /tmp/, but remind that these files will be deleted at your logoff.

This first application is a simple word count. You can use the solution available on the web site of the course: [Lab1_BigData_with_libraries.zip](http://dbdmg.polito.it/wordpress/wp-content/uploads/2016/03/Lab1_BigData_with_libraries.zip)
(http://dbdmg.polito.it/wordpress/wp-content/uploads/2016/03/Lab1_BigData_with_libraries.zip)

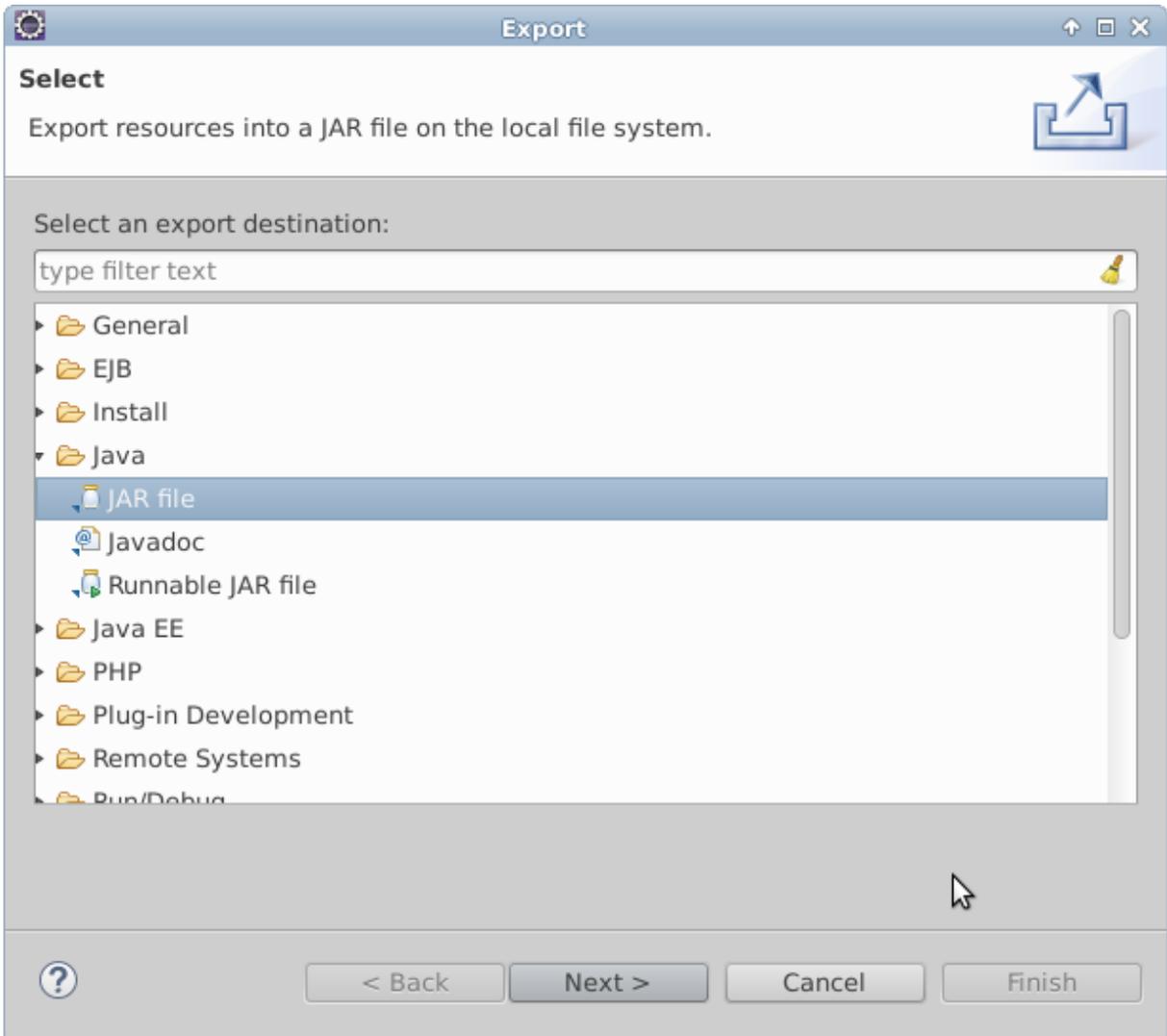
1. Open eclipse luna

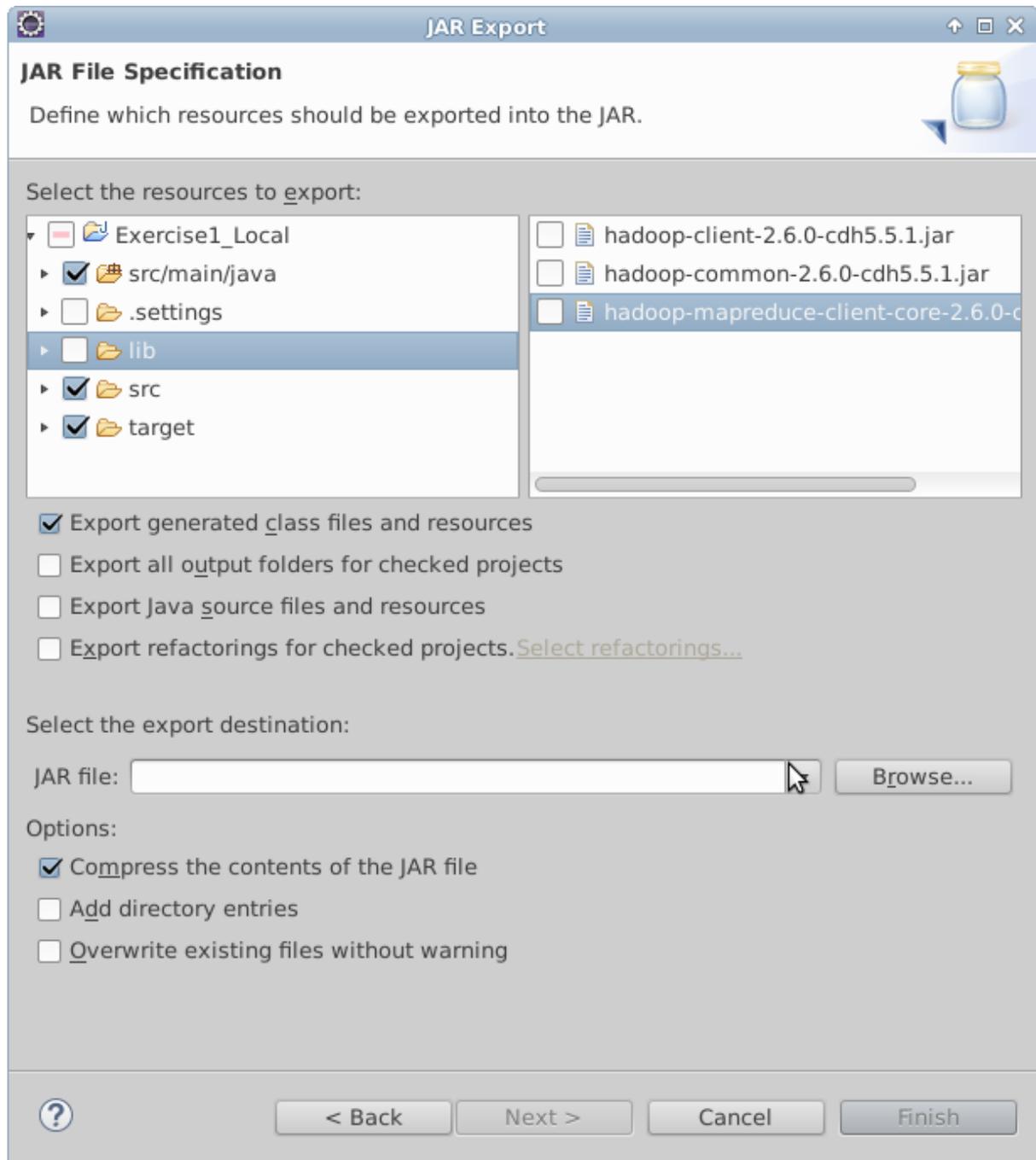
2. Import the project (File -> Import... | General -> Existing project... | choose the folder where you extracted your file)



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 avoid using maven, we cannot count on it to generate the .jar, as we would normally do. You can instead build a jar manually using the File -> Export command, as in the following 2 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 this 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.



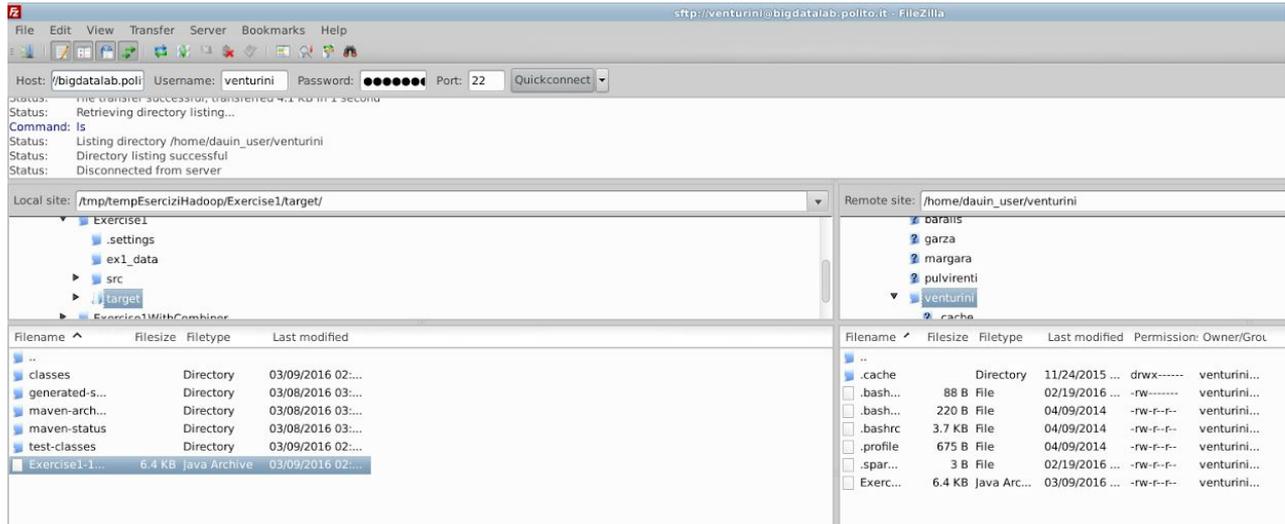


The final output of compilation is a *.jar. Try to look for Exercise1-1.0.0.jar inside the target/ folder.

2. Transfer files using FTP (Filezilla)

The objective of this task is to transfer some files from the local machine (i.e. the one you're working on right now) to a remote machine (e.g. the BigDataLab gateway). For this task, we are going to use Filezilla, but if you feel more comfortable with a command line feel free to use scp or rsync.

1. Connect to bigdatalab.polito.it [host] on port 22, using the username and password you have been given.
2. The Filezilla interface is divided in two section: on the left side you have the file system of your local machine, on the right side the file system of the remote machine.



Locate on the local site your *.jar (output of task 1), and a proper folder on the right (your home or a subfolder is fine). Right click on a file to find the upload command.

3. Check that the file was correctly transferred to the remote. You can either check the logs (above) or if the remote folder lists your file (on the right). In the figure, you see that a 6.4KB file, named Exercise1-1.0.0.jar, was copied to my home folder.

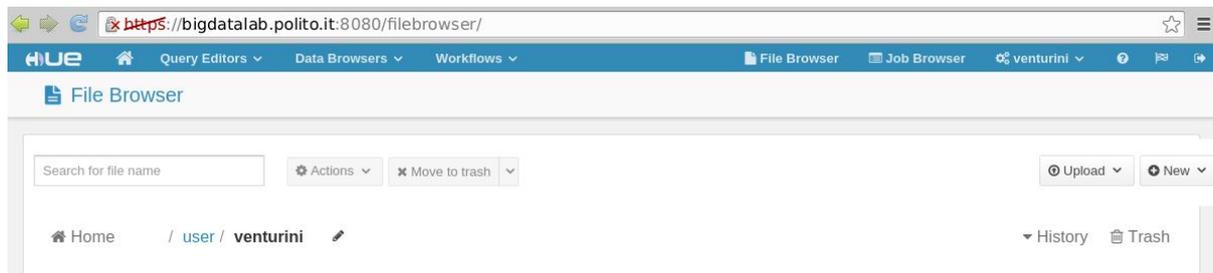
3. Manage HDFS (HUE).

In this task, you'll learn how to do basic management of the HDFS file system. To this goal, we will use a web interface called HUE. Again, you can do all the basic operations of this task on a command line; but this time, we advise you against this approach¹.

1. Go to <https://bigdatalab.polito.it:8080> and login with your usual BigDataLab credentials.
2. Go to the “file browser” tab. You should find your HDFS home, as shown below. Note that this is not the same file system as in task 2, so you will find probably an empty folder now.²

¹ You would need further configuration to access HDFS from a client on your local machine.

² 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. Upload the sample file you have in `ex1_data/document.txt`, and check everything worked fine by opening it inside HUE. You should find two-three lines of sample text inside.

4. 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 command line. Using `ssh`, we will connect to the gateway we have used in task 2, where we should find the jar we copied at that time. From this gateway machine, we can (finally!) submit a job³.

1. Connect, using SSH, on `bigdatalab`, with your usual `BigDataLab` credentials.

```
>> ssh sXXXXXX@bigdatalab.polito.it
```

2. Now that you have a terminal on the gateway, launch a job using this command:

```
>> hadoop jar Exercise1-1.0.0.jar  
it.polito.bigdata.hadoop.exercise1.DriverBigData 1  
your_input_file.txt ex1_out4
```

3. Find your job on HUE interface (JobBrowser), check its status (submitted, running, failed or succeeded) and find its `stderr` and `stdout`. Find also all the running jobs at the present time (i.e. not only yours).
4. Find the output file on HDFS (from HUE Filebrowser) and see the results for the wordcount.
5. Try to re-run the same job. Does it succeed this time? What's the problem?

³ This gateway machine is simply the only machine in the cluster that can be accessed from outside, and the only one where users are allowed to launch jobs and manage HDFS.

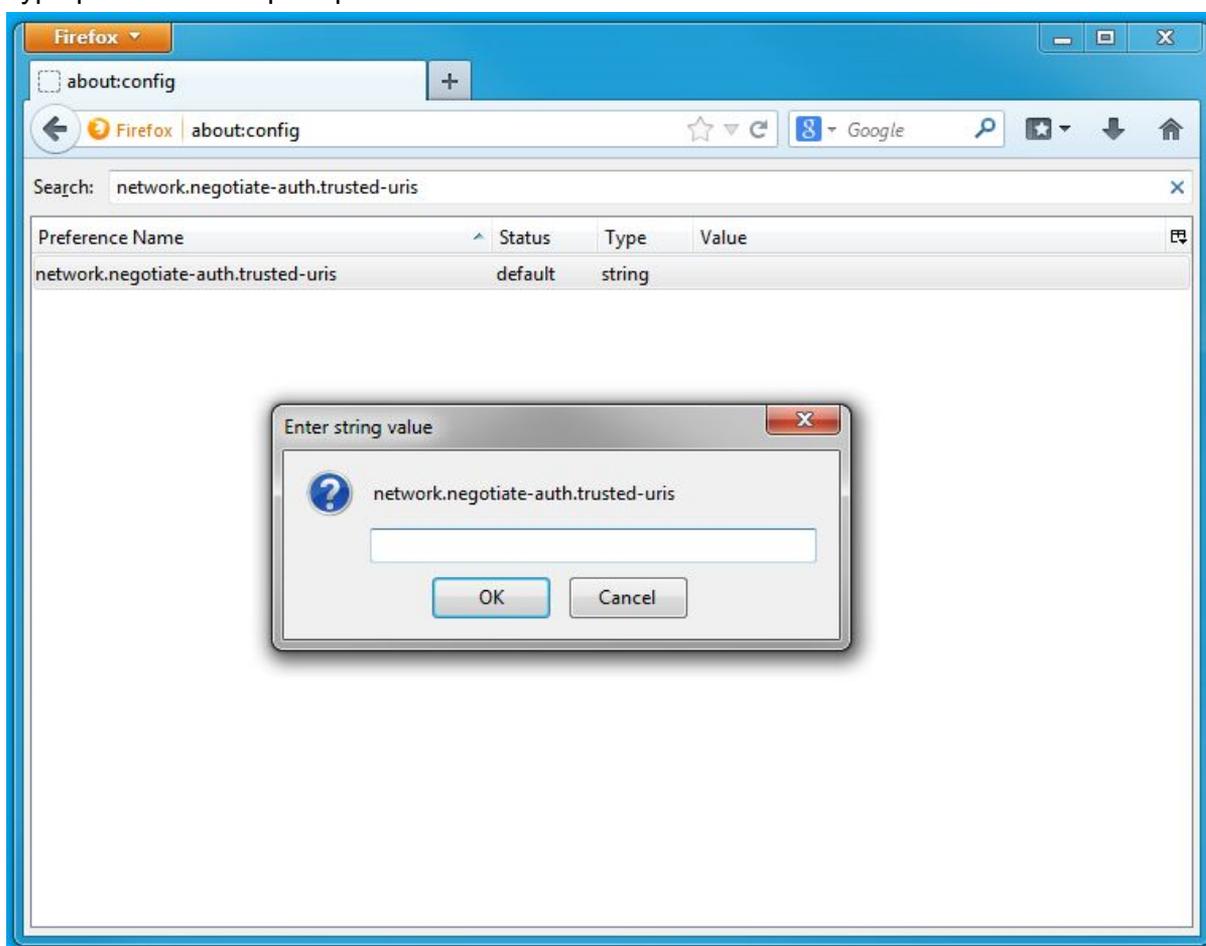
⁴ Try to understand carefully all the parameters in this command. You can read the sources you have compiled in task 1 to see what it's going on.

5. Analyze the performances of your job

First, we need to setup some security configuration to access protected web interfaces on our cluster. First, we obtain a kerberos ticket, that will grant us access to the system for a day; then, we will configure Firefox to use this ticket to access the protected web interfaces (HUE is the only web UI you can access without completing this procedure).⁵

Keep in mind this operations for our next labs.

1. On a command line, type `kinit sXXXXXX`. You will be asked for your BigDataLab password.
2. On Firefox, go to the `about:config` page and find `network.negotiate-auth.trusted-uris`. Type `polito.it` on the prompt.



3. Now you should have access to the full job history of YARN:
<https://ma1-bigdata.polito.it:19890/jobhistory/>
4. How many mappers did your job instantiate? How many reducers?
5. Relaunch your job on a bigger file, that you can find in `/data/students/bigdata-01QYD/Lab1/finfoods_text.txt` on HDFS. This a large collection of amazon reviews in the food category. Analyze the results. Can you

⁵ Details here: <http://bigdata.polito.it/content/access-instructions>

understand any interesting facts from your results? Do you see any space for improvements in your analysis?

6. The following figure was done on a small sample of your data (10000 reviews). Is it consistent with what you found on the complete dataset? Do you think a small sample is enough to represent the whole?



Bonus Task

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.

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