Lab 1: Hadoop Tutorial

February 19th, 2014

Objectives

The objectives of this lab session are the following:

- Familiarize yourself with our lab infrastructure and the Hadoop framework
- Prepare the graph of Wikipedia articles for the second lab

Deliverables

As this is a first introductory lab, there are exceptionally no deliverables.

1 General Instructions

Our labs are designed to be performed from any machine in room BC07-08, **running Ubuntu**[1]. However, you can optionally do the labs from your personal machine. To do so, you will need the following:

- Java JDK 1.6 or later[2]
- Eclipse 3.8 or later[3]
- An SSH client (`ssh` on Unix platforms, PuTTY on Windows[4])
- Access to your `myfiles` directory[5]

2 Initial Setup

2.1 Extracting the Files

For each lab, we will usually provide you with some code and data. These will be available from Moodle. Download the archive for this lab and extract it anywhere in your `myfiles` directory:

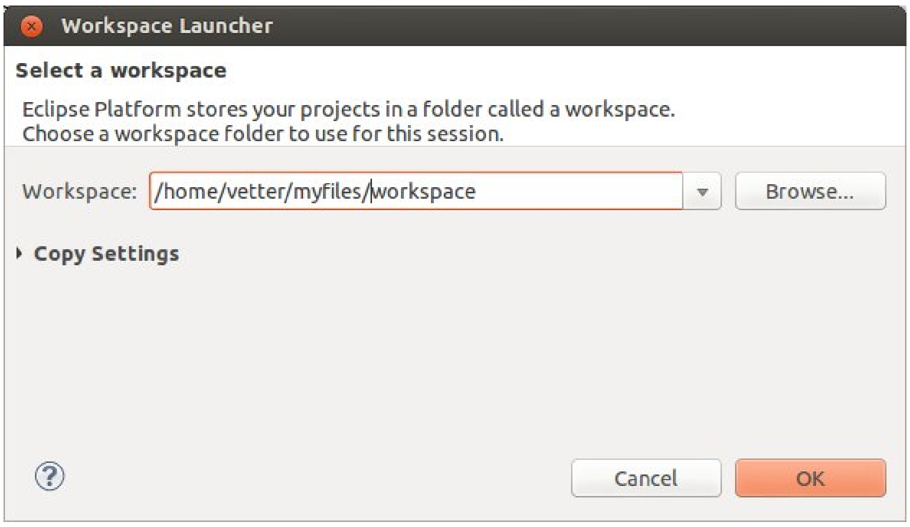
```
$ tar -C ~/myfiles/ -xzvf lab01.tar.gz
```

2.2 Importing the Project

Then, to import the project into Eclipse, follow these instructions:

1. Launch Eclipse (type `eclipse &` in a Terminal).
2. If prompted for setting up the Android SDK, simply click *Cancel*.
3. When asked to select a workspace, choose a location in your `myfiles` directory, e.g. `myfiles/workspace` and click *OK* (Figure 1).
4. Go to *File* → *Import*.
5. Select *General* → *Existing Projects into Workspace* and click *Next* (Figure 2).
6. In *Select root directory*, choose the directory you just extracted, e.g. `~/myfiles/ix-lab01` (Figure 3).
7. Click *Finish*.

[1]If the machine you are sitting at is running Windows, simply restart it and choose Ubuntu in the boot menu.
[2]`http://www.oracle.com/technetwork/java/javase/downloads/index.html`
[3]`http://www.eclipse.org/`
[4]`http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`
[5]For instructions on how to set that up on your personal computer, see `http://mynas.epfl.ch/`.



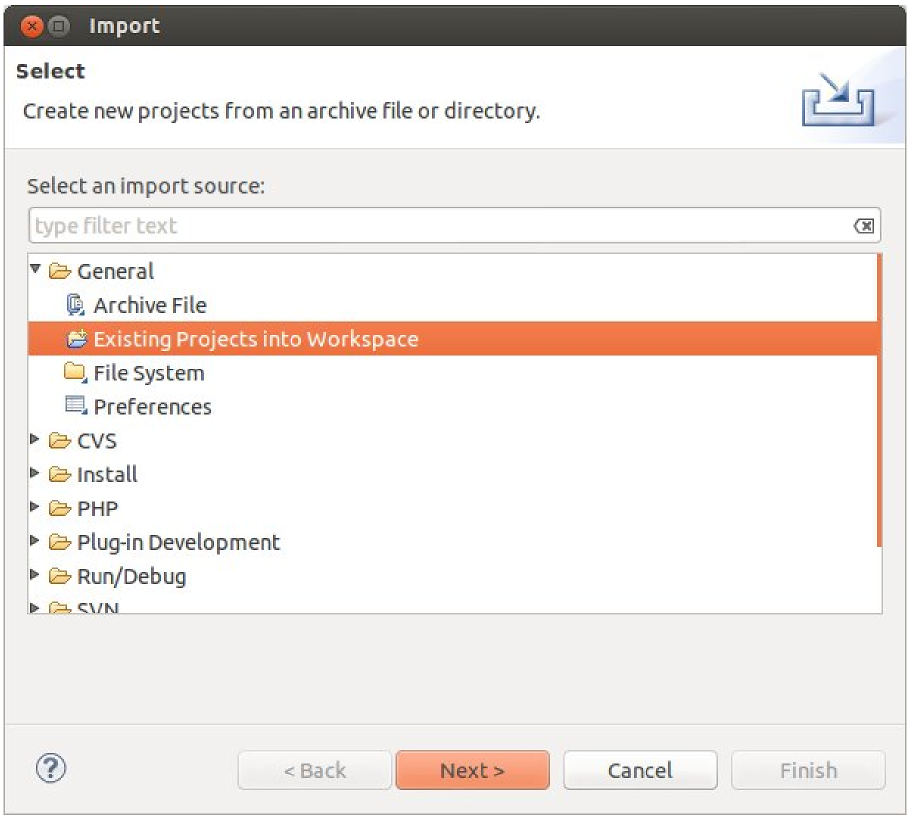
Select a workspace

Eclipse Platform stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: /home/vetter/myfiles/workspace

Copy Settings

Figure 1: Choose a workspace inside your `myfiles` directory



Select

Create new projects from an archive file or directory.

Select an import source:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
- CVS
- Install
- PHP
- Plug-in Development
- Run/Debug
- SVN

Figure 2: Importing the provided project into Eclipse

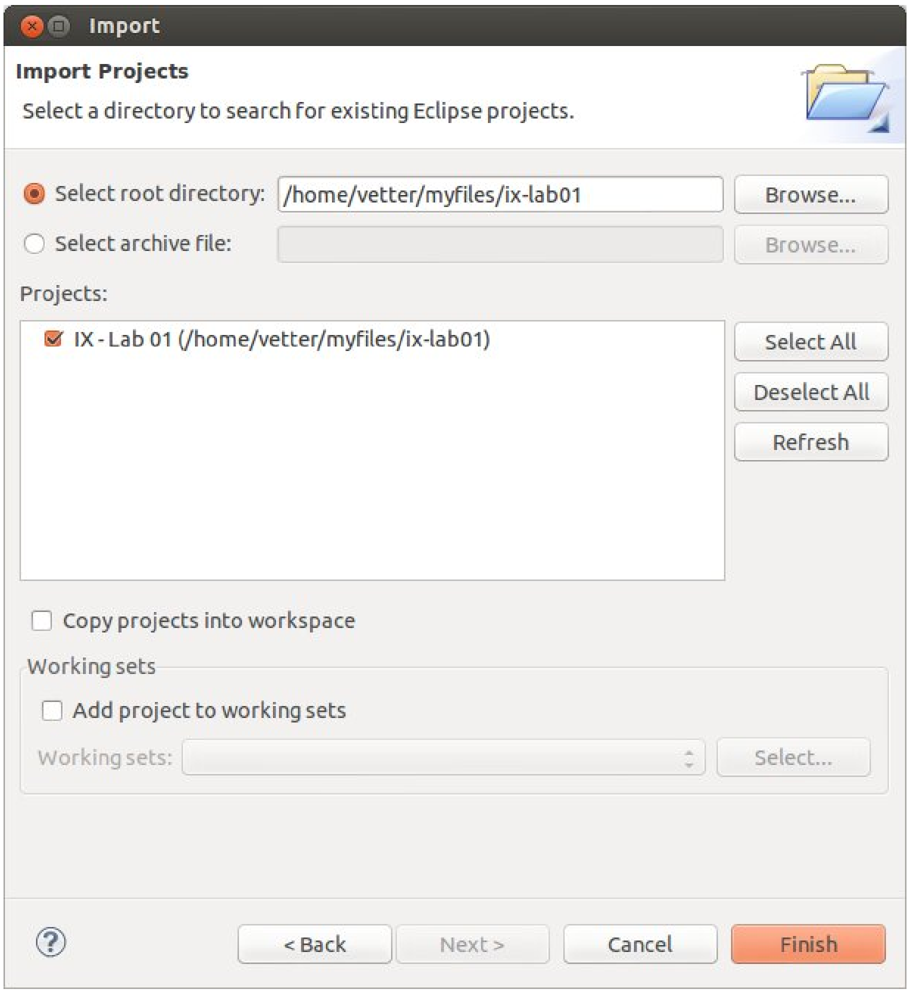

Import
Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory: /home/vetter/myfiles/ix-lab01
Browse...
Select archive file:
Browse...
Projects:
IX - Lab 01 (/home/vetter/myfiles/ix-lab01)
Select All
Deselect All
Refresh
Copy projects into workspace
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Cancel
Finish

Figure 3: Choose the project directory to import

2.3 Running WordCount

2.3.1 Standalone vs. Distributed Runs

When developing Hadoop applications, it is more convenient to first test them locally, before running them on a cluster. Luckily, Hadoop provides a way of launching any Hadoop job in a *standalone* mode, that runs into a single JVM and does not require to connect to a cluster. It allows to debug it more easily, if required. Standalone jobs are usually run on a sample of the dataset, so that one can quickly test them and obtain known results. Because of the limited environment, only one mapper and one reducer are used.

On the contrary, once you have verified that your job works correctly, you can deploy it to a Hadoop cluster, composed of multiple machines, to be run on the full dataset. As you will see, there are some intermediate steps required to package a job and send it to the cluster, and the deployment of the job will add some overhead. It is therefore not practical to test and debug applications directly on the cluster.

To summarize, the two different running environments that you will use are:

- **Standalone**: allows to test a job locally on a subset of the data. Only 1 mapper and 1 reducer are used.
- **Distributed**: allows to run jobs on the whole dataset using multiple machines. The number of mappers and reducers respectively depend on the input file size and cluster configuration.

2.3.2 Standalone WordCount

As said above, it is recommended to launch standalone jobs on a subset of the data, to allow for quick runs and convenient debugging. To test the WordCount job, we provide you with a sample of the Tweets dataset in the file `data/tweets-sample.txt`.

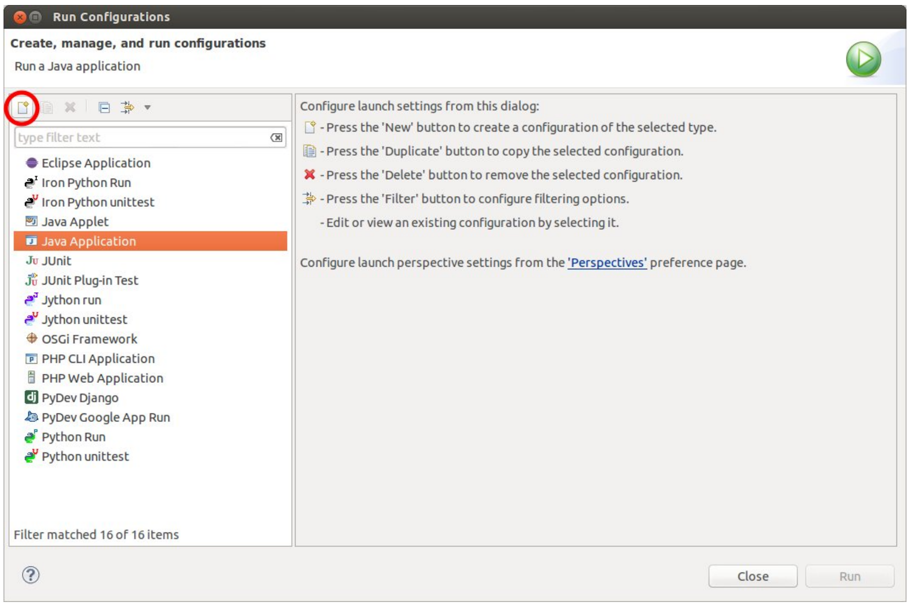
The format and the first few lines of this file are represented below:

```
# id | userId | screenName | utcOffset | createdAt | placeId | latitude | longitude |
    inReplyTo | sourceName | text
291856746568351745 | 583547325 | Norman__G | 28800 | 2013-01-17 10:37:25 |
    e0bf196c2d3e4397 | 47.181 | 8.52642 | 291856056668274688 | TweetDeck | @ZainiNazmi
    Swiss National Bank central bank. Takde bank nama Swiss Bank sini. yg ada Credit
    Suisse, UBS. yg org sorok duit private bank.
291856748745203712 | 86024102 | franci_de | 3600 | 2013-01-17 10:37:25 | 086752
    cb03de1d5d | 40.7564 | -73.9631 | ? | Twitter for iPhone | @Krystenritter: Vegas
    baby http://t.co/Q6VVtxCT I love it when actors are similar to their TV character in
     real life
291856772652732417 | 115085362 | sumemer | 3600 | 2013-01-17 10:37:31 | 88a9c00fe554a26e
     | ? | ? | 291851399271178240 | web | @xyuutsu41 no me preguntes cmo me ha dado
    porque no lo s. Pero ha sido win.
291856782945550337 | 387274691 | AnneBenjadid | 3600 | 2013-01-17 10:37:34 | 8
    fc3c9b3e656452c | 47.4016 | 6.4967 | 291856696047968256 | Twitter for iPhone |
    @CheeksDrogba graaaaave
```

To launch the `WordCount` application on the sample files, do the following:

1. In the Package Explorer of Eclipse, select the file `WordCount.java` from the package `ix.lab01.wordcount` in the `src` folder.
2. Go to *Run* → *Run configurations...*
3. In the new window, select *Java Application* in the list on the left, and click the first button, *New launch configuration*, in the upper left corner (Figure 4).
4. Go to the *Arguments* tab of the created configuration and add the following to the *Program arguments* textbox (Figure 5):

   ```
   data/tweets-sample.txt output/wordcount
   ```


Run Configurations

Create, manage, and run configurations

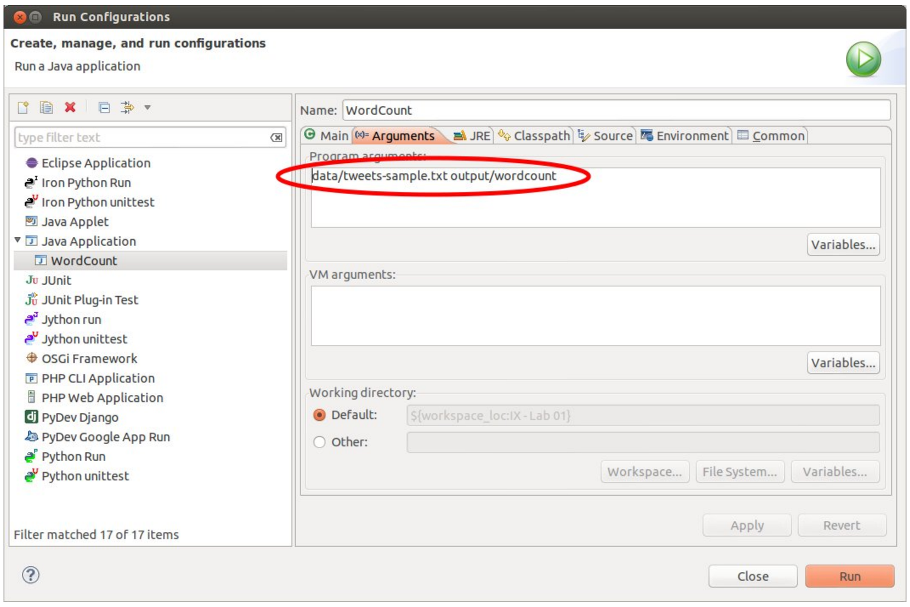
Run a Java application

Configure launch settings from this dialog:
- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the 'Perspectives' preference page.

Filter matched 16 of 16 items

Figure 4: Create a new run configuration by selecting "Java Application" and clicking on the first button in the upper left corner.


Run Configurations

Create, manage, and run configurations

Run a Java application

Name: WordCount

Program arguments:
data/tweets-sample.txt output/wordcount

VM arguments:

Working directory:
Default: ${workspace_loc:IX - Lab 01}
Other:

Filter matched 17 of 17 items

Figure 5: Add the input file and output folder to the arguments of the `WordCount` job.

These are the input file and output folder of the `WordCount` job.

5. Click *Run*

If everything went as planned, you should have a file named `part-r-00000` in the directory `output/wordcount`:

```
#       4
#2013   1
#6am    1
#acakfilm       21
#adwords        1
#afc    1
#agents 1
#algrie 1
```

It contains the output of your reducer, *i.e.*, key/value pairs where keys are words and values are the corresponding number of occurrences.

To sort this file by descending number of occurrences and print the top 20 words, you can use the following command:

```
$ cat output/wordcount/part-r-00000 | sort -n -r -k 2 | head -n 20
```

The top 3 words in the sample data are `//t`, `http` and `i`. The first two are artifacts of Twitter short URLs, of the form `http://t.co/asdf1234`, that our mapper splits into three words: `http`, `//t` and `co/asdf1234`.

Remark: To prevent accidental overwriting of job outputs, Hadoop will throw an exception if the output directory already exists. Simply remove it or specify another output directory when launching a job several times.

2.3.3 Exporting Your Code to a JAR File

To run any job on a Hadoop cluster, it first has to be packaged into a JAR file. To export your WordCount to a JAR file, follow these steps:

1. Right-click on the `src` folder in the Package Explorer
2. Select *Export...*
3. In the new window, select *Java* → *JAR file* and click *Next* (Figure 6)
4. In *Select the export destination*, click *Browse*. Navigate to your `myfiles` directory and type a file name (e.g. `ix-lab01.jar`) in the *Name* field and click *OK* (Figure 7)
5. Click *Finish* to export your code to the JAR file you picked (Figure 8)

You now have a JAR file containing your code in your `myfiles` directory: you are ready to run it on the cluster!

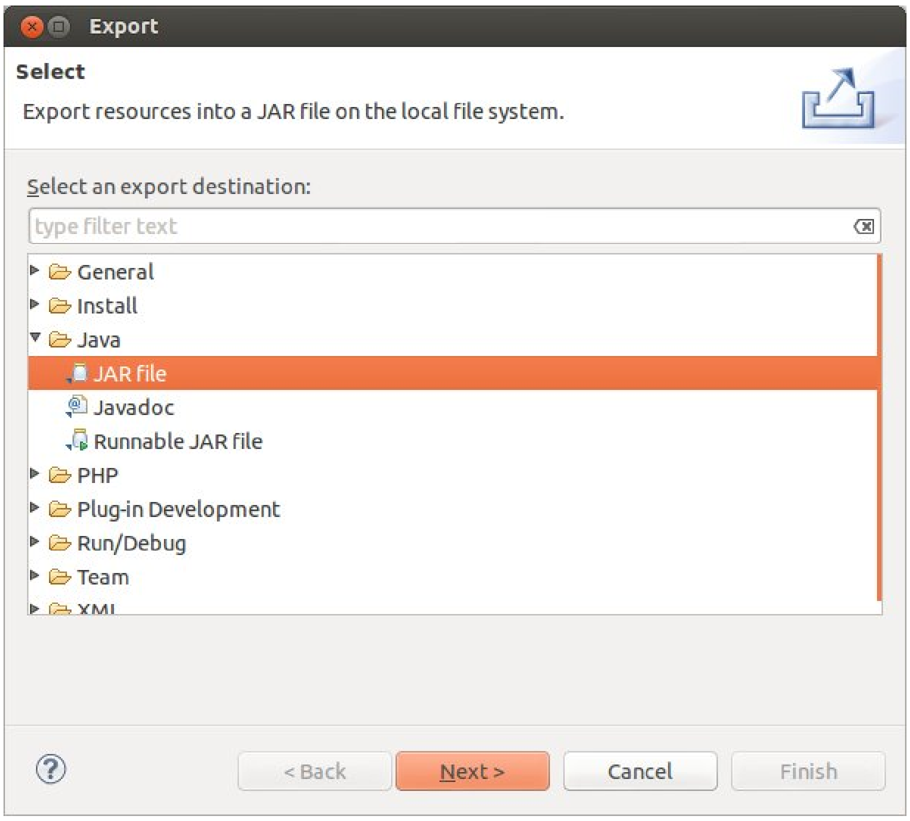


Figure 6: Exporting the `src` folder to a JAR file

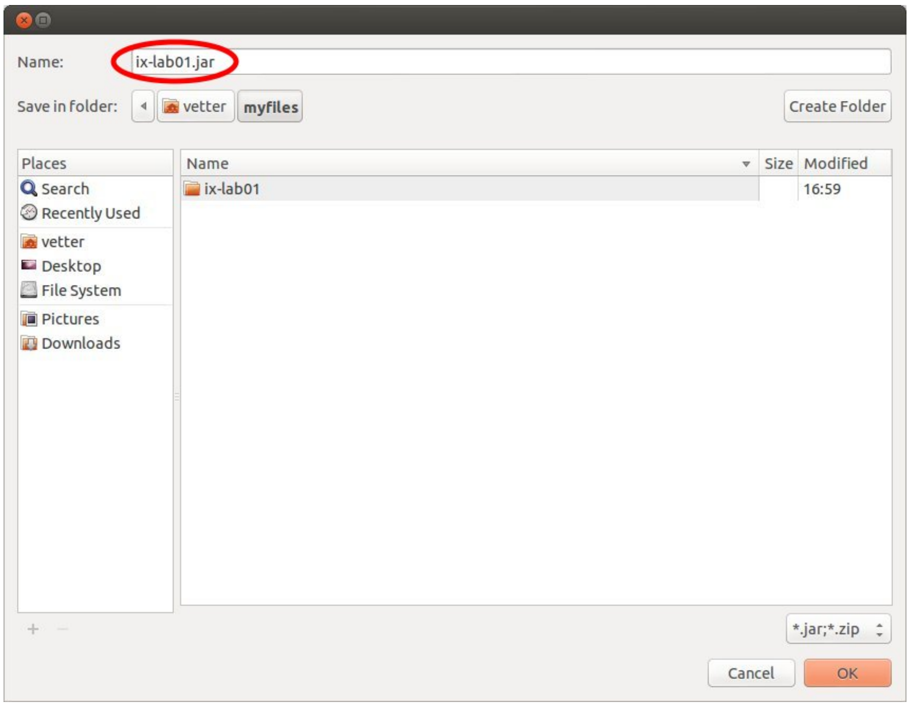
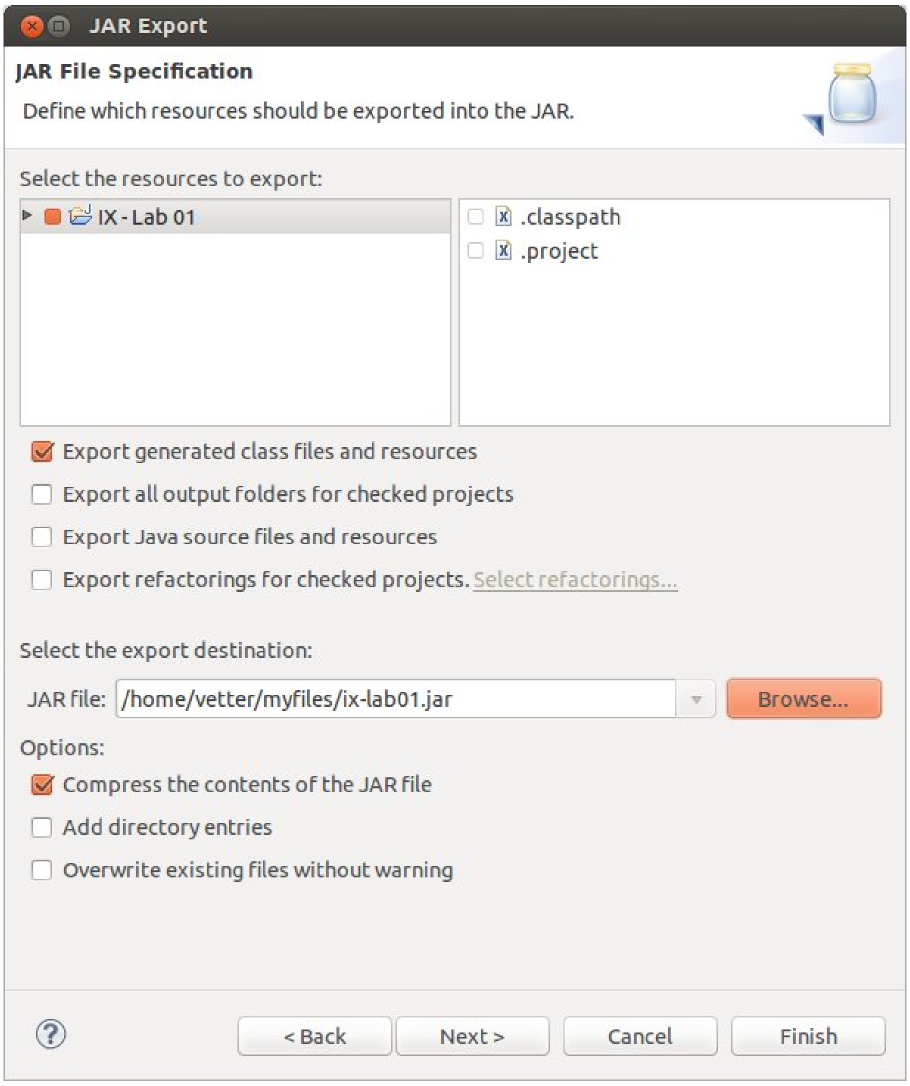


Figure 7: Browse to your `myfiles` directory and type the name of the JAR file to create, e.g. `ix-lab01.jar`


JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

IX - Lab 01

☐ .classpath
☐ .project

☑ Export generated class files and resources
☐ Export all output folders for checked projects
☐ Export Java source files and resources
☐ Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: /home/vetter/myfiles/ix-lab01.jar

Options:

☑ Compress the contents of the JAR file
☐ Add directory entries
☐ Overwrite existing files without warning

Figure 8: Select export destination for the JAR file

2.3.4 Connecting to the Cluster Entry Point

To launch a job on the Hadoop cluster, you first have to connect via SSH to one of its entry points. There are two servers[6] you can use: `ix-student1.epfl.ch` and `ix-student2.epfl.ch`. To balance the load, please connect to `ix-student1.epfl.ch` if your CAMIPRO number is odd, and `ix-student2.epfl.ch` if it is even.

When connecting, you should see a similar output:

```
$ ssh USERNAME@ix-student1.epfl.ch

        MyNAS is not mounted
        ********************

        run: mount  $HOME/myfiles       to mount your MyNAS account under $HOME/myfiles
        run: umount $HOME/myfiles       to unmount your MyNAS account
```

This message indicates whether your `myfiles` directory is mounted and indicates the commands to run to mount/unmount it. To have access to your myfiles directory from `ix-studentX.epfl.ch`, type the following command:

```
$ mount $HOME/myfiles
```

You will be prompted for your EPFL password. After typing it, you will have access to your `myfiles` directory and should see the JAR file you just created:

```
$ ls ~/myfiles
ix-lab01                ix-lab01.jar
```

2.3.5 Interacting With the Hadoop Distributed File System (HDFS)

As explained in the Hadoop tutorial, it provides a distributed file system, that allows to store big data files across multiple machines in a fault-tolerant way. This file system can be browsed using commands similar to the standard bash commands.

To list the files in your HDFS home directory, type the following command:

```
$ hadoop fs -ls
```

There is nothing in your HDFS home directory for the moment, but all Hadoop job outputs will be stored there. All datasets you will use can be found in the `/ix` directory:

```
$ hadoop fs -ls /ix
Found 4 items
-rw-r--r--   3 vetter ix       198821 2013-02-05 15:28 /ix/tweets-sample.txt
-rw-r--r--   3 vetter ix  19732044015 2013-01-18 17:24 /ix/tweets.txt
-rw-r--r--   3 vetter ix      3110080 2013-02-05 15:28 /ix/wikipedia-edits-sample.txt
-rw-r--r--   3 vetter ix 207706604046 2013-01-17 08:37 /ix/wikipedia-edits.txt
```

You can see that, for now, four files are available. If you want to have a look at what it contains, you can use the `cat` command:

```
$ hadoop fs -cat /ix/tweets.txt | head
# id | userId | screenName | utcOffset | createdAt | placeId | latitude | longitude |
    inReplyTo | sourceName | text
3241 | 221 | h0mee | -28800 | 2006-05-01 09:18:48 | ? | ? | ? | ? | web | just setting
    up my twttr
4429 | 280 | glen | 28800 | 2006-05-15 19:04:24 | ? | ? | ? | ? | web | just setting up
    my twttr
4446 | 280 | glen | 28800 | 2006-05-15 21:16:02 | ? | ? | ? | ? | web | I am at work now
    , and will go home early today my body is tired.
6091 | 322 | twang | -28800 | 2006-06-05 21:52:55 | ? | ? | ? | ? | web | just setting
    up my twttr
6093 | 322 | twang | -28800 | 2006-06-05 22:00:25 | ? | ? | ? | ? | txt | Gettin
    expensiv tek support
```

[6]These machines are only accessible from EPFL's network. Use the VPN if you want to do the labs from home.

```
7282 | 357 | wubbahed | -36000 | 2006-06-21 22:32:47 | ? | ? | ? | ? | txt | just
    setting up my twttr
7284 | 357 | wubbahed | -36000 | 2006-06-21 22:42:39 | ? | ? | ? | ? | web | working
    late in the office
7287 | 357 | wubbahed | -36000 | 2006-06-21 23:02:43 | ? | ? | ? | ? | web | writing
    long emails to clients
7308 | 357 | wubbahed | -36000 | 2006-06-22 01:30:53 | ? | ? | ? | ? | web | finally
    leaving work
```

Because these files are usually huge, always use tools such as `head` or `less` to view them.
To copy a file to HDFS, use the following command:

```
$ hadoop fs -copyFromLocal <localFile> <hdfsPath>
```

Similarly, you can fetch any file back from HDFS to the local file system with this command:

```
$ hadoop fs -copyToLocal <hdfsFile> <localPath>
```

Finally, to get more details about each command and an exhaustive list of the available options, use the `help`:

```
$ hadoop fs -help
```

2.3.6 Distributed WordCount

You are finally ready to run your `WordCount` job on the cluster. The input file you will use for this job is `/ix/tweets.txt`.
To launch your job using the JAR file in your `myfiles` directory, use the following command:

```
$ hadoop jar ~/myfiles/ix-lab01.jar ix.lab01.wordcount.WordCount /ix/tweets.txt output-
    wordcount
```

Remark: If you get a `java.lang.UnsupportedClassVersionError` exception with a message similar to `Unsupported major.minor version`, it means that you have not compiled your code with the correct Java version. Our servers use Java 1.6. To set the correct Java version in Eclipse, right-click on your project, go to *Properties* → *Java Compiler*, check *Enable project specific settings* and select `1.6` as the *Compiler compliance level.*

Remark: You may see a few `java.lang.Throwable: Child Error` messages when running jobs on the cluster. This means that some task failed because the node did not have enough memory (we are still tuning the cluster). However, thanks to Hadoop's fault-tolerance, the task will simply be rerun and your job will still complete.

Once the job is finished, you can find its output in the HDFS output directory you chose:

```
$ hadoop fs -ls output-wordcount
Found 12 items
-rw-r--r--   3 vetter ix          0 2013-02-05 08:28 /user/vetter/output-wordcount/
    _SUCCESS
drwx------   - vetter ix          0 2013-02-05 08:26 /user/vetter/output-wordcount/_logs
-rw-r--r--   3 vetter ix   67532546 2013-02-05 08:28 /user/vetter/output-wordcount/part-
    r-00000
-rw-r--r--   3 vetter ix   67628131 2013-02-05 08:28 /user/vetter/output-wordcount/part-
    r-00001
-rw-r--r--   3 vetter ix   67544156 2013-02-05 08:28 /user/vetter/output-wordcount/part-
    r-00002
-rw-r--r--   3 vetter ix   67596988 2013-02-05 08:28 /user/vetter/output-wordcount/part-
    r-00003
-rw-r--r--   3 vetter ix   67518189 2013-02-05 08:28 /user/vetter/output-wordcount/part-
    r-00004
```

```
-rw-r--r--   3 vetter ix   67533142 2013-02-05 08:28 /user/vetter/output-wordcount/part-
    r-00005
-rw-r--r--   3 vetter ix   67594588 2013-02-05 08:28 /user/vetter/output-wordcount/part-
    r-00006
-rw-r--r--   3 vetter ix   67611378 2013-02-05 08:28 /user/vetter/output-wordcount/part-
    r-00007
-rw-r--r--   3 vetter ix   67560991 2013-02-05 08:28 /user/vetter/output-wordcount/part-
    r-00008
-rw-r--r--   3 vetter ix   67639538 2013-02-05 08:28 /user/vetter/output-wordcount/part-
    r-00009
```

The first two entries are metadata from Hadoop, informing us that the job successfully ran and containing its execution history. More important are the `part-r-XXXXX` files, that contains the output of the different reducers (one file per reducer).

Looking at one of these files, we find the expected output:

```
$ hadoop fs -cat output-wordcount/part-r-00000 | head
####### 24
##############virginia  1
########stops   1
#######able     1
#######abusing#####     1
######middleman 1
######s#w       1
```

Hadoop provides us with a command to fetch all of these files at once, merge them on a single file and copy this file to your local directory (outside of HDFS):

```
$ hadoop fs -getmerge output-wordcount output-wordcount.txt
```

2.3.7 Job Status and History

An important information found in the output of a job launch is the job ID, usually shown on the 6th line of the job output:

```
13/02/05 08:26:45 INFO mapred.JobClient: Running job: job_201301251011_0089
```

This job ID (here `job_201301251011_0089`) allows you to keep control of your job after is has been launched. Because Hadoop is a distributed system, interrupting the output of the `hadoop job` command you launched will not actually stop the job: it merely kills the Hadoop *client* that is monitoring the job progress for you.

To see running jobs at any moment, you can use the following command:

```
$ hadoop job -list
```

To see the status of a job, use:

```
$ hadoop job -status job_201301251011_0089
```

Web interface: For a more visual and interactive way of checking your job status, Hadoop provides a web interface. You can see the current status of the cluster at `http://icsil1-s12-srv46.epfl.ch:50030` and get the details of a job at `http://icsil1-s12-srv46.epfl.ch:50030/jobdetails.jsp?jobid=JOBID`.

To kill a job (if it had a problem or you don't need its output anymore), use:

```
$ hadoop job -kill job_201301251011_0089
```

Finally, to see details about the execution of a job (number of mappers/reducers, etc.), use:

```
$ hadoop job -history wordcount-output
```

The `history` command takes the HDFS output directory as argument, not the job ID, because it parses the logs available in the job output folder.

2.3.8 Deleting Your Job Output

Once you are done with the job output, you can remove it from HDFS:

```
$ hadoop fs -rmr output-wordcount
```

Remark: Because the dataset we will handle in this course are pretty large, try not to keep multiple job outputs. Once you obtained the result you wanted, delete both local and HDFS files, to save space. You have a quota of 1GB of local data and 5GB of HDFS data that you can store.

3 Twitter Hashtags

Now that you are familiar with running Hadoop jobs both locally and on the server, your first task, as a data analyst, is to study trending topics on Twitter. To do so, you will focus on hashtags, *i.e.*, words that start with a `#`.

Question 1 Starting from the `WordCount` job, create a TopicCount job that counts the number of occurrences of hashtags only. Can you reuse the mapper/reducer of `WordCount`?

Test your new job locally on the sample data, to verify that it only counts hashtags, and then run it on the whole dataset on the cluster. What are the most frequent hashtags?

Question 2 By restricting your analysis to a given date, and looking at hashtags, can you find out what major event happened on September 6th, 2012?

4 Wikipedia Articles Graph

In this second part, we will prepare the data for the next lab session. The dataset we will use is the list of Wikipedia edits, described below:

```
# REVISION articleId revisionId articleName timestamp userName userId | CATEGORY cat1
    cat2 [...] | MAIN linkedArticle1 linkedArticle2 [...]
REVISION 10 233192 AccessibleComputing 2001-01-21T02:12:21Z RoseParks 99 | CATEGORY |
    MAIN
REVISION 10 133180268 AccessibleComputing 2007-05-24T14:41:58Z Ngaiklin 4477979 |
    CATEGORY | MAIN
REVISION 12 18201 Anarchism 2002-02-25T15:00:22Z ip:Conversion_script ip:
    Conversion_script | CATEGORY | MAIN Syndicalism Nihilism Gustave_de_Molinari
    Benjamin_Tucker Benjamin_Tucker Minarchism Noam_Chomsky Punk_rock [..]
```

We provide you with a class (`ix.utils.WikiEdit`) that allows you to parse each of these lines into a Java object. You can find more details about the dataset format in the documentation of this class. We will focus here on the article name and the `MAIN` part, that contains the list of links to other articles found in the text of each edited article.

The goal is to build the graph of articles: in this graph, each node is a Wikipedia article, and there is a link from article A to article B if article A contains a link to article B. This graph is thus directed.

4.1 Graph Example

To understand better the task at hand, here is a basic example, that lists each edit in a simplified way:

```
Edit1 Article1 10:00 | MAIN Article2 Article3
Edit2 Article2 11:00 | MAIN
Edit3 Article3 12:00 | MAIN Article2
Edit4 Article1 13:00 | MAIN Article2
```

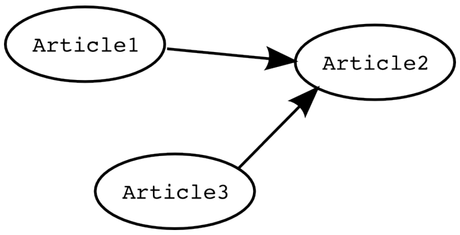


Figure 9: Wikipedia articles graph extracted from the example data

From the above data, we can build the articles graph shown in Figure 9. Even though article 1 first had links to both articles 2 and 3, we are interested in the final state of the graph, and thus *only* keep the links that are found in the last edit of article 1 (edit 4). We therefore have only two edges, article 2 having no link to other articles.

The output of the graph extraction process would thus be:

```
Article1|Article2       1
Article3|Article2       1
```

This graph will be studied extensively in the next lab, we focus here on its extraction only. Note that in this case, all edges have weight 1. We could omit these weights, but choose to keep a more general format to support later extensions.

4.2 Find the Last Edit

Extracting the articles graph from the edits dataset using Hadoop has to be done in two steps. The first thing we need to do is to find the last edit of each article.

Question 3 Using the skeletons provided, implement a first `LastEdit` job that extracts, for each article, its last edit:

- The **mapper** `LastEditMapper` will take each line of the raw data (that corresponds to one edit) and output a key/value pair, where the key is the name of the article and the value is a concatenation of the timestamp of the edit and the list of linked articles.
- The **reducer** `LastEditReducer` will then assemble all values with the same key, *i.e.*, all edits of the same article, and simply keep the most recent one. The final output of the reducer will be a key/value pair, where the key is the name of the article, and the value a concatenation of the timestamp of the most recent edit and the list of linked articles

Note that because we want to find links between articles, you can skip edits that have no outgoing links.

With the sample data `data/wikipedia-edits-sample.txt`, you should find just one article with outgoing links, with the following last edit:

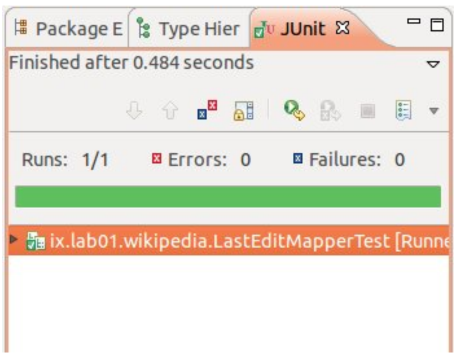
```
Anarchism       2004-05-03 23:56:28 | 16th_century, 1789, 1793, 17th_century, [...]
```

4.2.1 Debugging Your Mapper/Reducer

Debugging a map/reduce job can be difficult, even when run locally. To help you testing the mapper and reducer separately, we provide you with *unit tests*. These can be found in the `test` folder.

For example, to test your `LastEditMapper`, right-click on the class `ix.lab01.wikipedia.LastEditMapperTest` in the `test` folder and select *Run As* → *JUnit Test*.

This will run the tests in Eclipse and should display a window similar to Figure 10. If the bar is green, everything went well! If it is red, you can expand each of the tests in the list below to see what went wrong.



```
Finished after 0.484 seconds
Runs: 1/1    Errors: 0    Failures: 0
ix.lab01.wikipedia.LastEditMapperTest [Runne
```

Figure 10: Eclipse panel displaying the result of the unit tests.

4.3 Article Links

The second part of the graph extraction simply consists of enumerating the outgoing links of each article.

Question 4 Using the skeleton provided, implement the `ArticleLinks` job that enumerates the outgoing links of each article, given its last edit.

- The **mapper** `ArticleLinksMapper` receives an article name and a concatenation of the timestamp and list of outgoing links. For each linked page, it outputs a key/value pair where the key is the edge identifier (e.g. `article1|article2`), and the value is 1.
- In this job, there is actually no need of a reducer!

To run this second part, you need to use the output folder of the first part (last edit extraction) as input. Using the result shown at the end of the previous section, one would obtain the following output:

```
Anarchism|16th_century  1
Anarchism|1789  1
Anarchism|1793  1
Anarchism|17th_century  1
[...]
```

Again, you can use the unit tests in ArticleLinksMapperTest to check that your mapper does the correct job.

4.4 Full Graph Extraction

Once you are confident that both intermediate jobs work correctly, you can use the `ArticlesGraphExtraction` job, that combines both operations in one call.

Warning: running the full graph extraction on the whole wikipedia dataset generates a large output (more than 2 GB). Make sure you don't fill up your HDFS quota!