

# Cloud Computing (INGI2145) - Lab Session 3

Nicolas Laurent, Marco Canini

## 1. Background

In this lab session, you will learn how to code a simple MapReduce job for Hadoop and how to run the job in the INGI2145-vm.

**To setup the VM for this exercise**, you will need to start by pulling from the course Git repository to obtain some skeleton code. The code is located inside the path `INGI2145-vm/lab3`. Be sure to check the `README.md` file to learn how to run the code and generate project files for IDEs (Eclipse, IDEA).

## 2. Programming exercise

In this lab session, we'll ask you to implement a word-counting job in Hadoop. You'll take as input a text file containing a text spread on multiples lines and you must produce as output a file containing a list of `<word, word_frequency>` pairs. Each pair should appear on its own line.

For example, if you have the following input:

```
bowties are cool
I own bowties
```

You should obtain an output similar to this (order doesn't matter):

```
are 1
bowties 2
cool 1
own 1
I 1
```

Then fill in the mapper and reducer classes to perform the job. A few hints:

- You should start by thinking about the types you are going to use for input/reduce/output keys/values. For the input, it is probably better to use the default types (`LongWritable` and `Text`).
- Consult the Hadoop Java API here: <http://hadoop.apache.org/docs/stable/api/>
- The Java class `StringTokenizer` from the standard library can help with your task.
- If you run into issues, be sure to consult the log file (`lab03.log`) or change the configuration to output more data to the console.

And now a few questions for you:

- Imagine your solution was distributed across multiple nodes. Which of the data flow blocks we saw in the lecture could you use to make it more efficient?
- The input is sorted by word, but we might want to sort it by decreasing frequency. Write a second job to sort the output of the first job. Is there a way to leverage one of the data flow blocks we saw to let the framework do the work for us?

### 3. Running your job locally in the INGI2145-vm

To make it convenient to build and run the word-counting example within the INGI2145-vm, we have already setup a build automation tool called gradle (<http://www.gradle.org/>).

Simply follow the instructions below:

- To compile the code:  
`./gradlew compileJava`
- To run the program without command line arguments:  
`./gradlew run`
- To run the program with command line arguments:  
`./run.sh <arg1> <arg2> <arg3> ...`
- To build a JAR file that can be uploaded to Amazon Elastic MapReduce:  
`./gradlew assemble`  
The jar will be put under ``build/libs``.

The homework assignment 1 contains a step-by-step guide for running a Hadoop job on Amazon's Elastic MapReduce.