


# A small git exercise

Nicolas Gartner

11-12 January 2021

The objective of this exercise is to make you discover the  **git** tool, and have your first experience with it. It will be a collaborative development of a small repository, where each person in the class will have to write one phrase in a text document.

## 1 Initial setup

The first part of the work will be to set up properly the working directory, in which we will write the common text document. Please go to an appropriate directory, where you would want to find that folder. An appropriate folder would be, for example, Documents. Open you terminal, go that folder and execute the following commands:

```
mkdir git-exercise
cd git-exercise
git init
```

This should create a *.git* repository inside git-exercise.

Add the common remote repository and pull existing files from the master branch:

```
git remote add origin https://github.com/ngartner/Git-exercise
git pull origin master
```

You could replace the name *origin* by any name that you would like, it is the name that you are associating to the adress <https://github.com/ngartner/Git-exercise>. You should now see a file called *MainFile.txt* in your folder. Using **gitk** you should see the same window as in figure 1.

```
gitk
```

As you can see, one person already wrote one line in that file, and this has to be continued. Another method to obtain that first repository would be to write the command in the appropriate folder you desire:

```
git clone https://github.com/ngartner/Git-exercise
```

This would do exactly the same as all the previous commands: creating a new file, defining the origin and pulling the data, but it does not allow you to define the name of the created folder and the name *origin*.

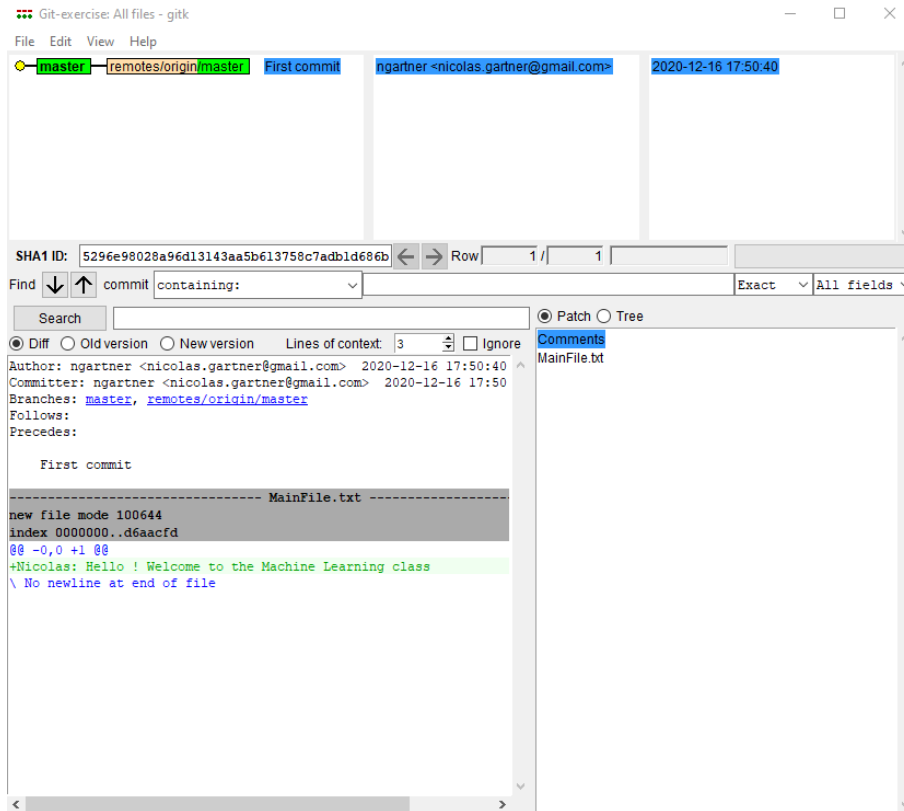


Figure 1: The first commit of the project

## 2 Your first commit

This second part consists in writing your commit inside a new branch that you would create. The next task will be to add you own sentence in *MainFile.txt*. Please write it **on a new line** according to the following syntax:

Your name: The sentence you would like to write.

That sentence should be a personal message from you to every other student. Then, save your file into a new branch:

```
git branch your_name
git checkout your_name
git add .
git commit -m "The commit message you want to write"
```

Please, replace *your\_name* and "*The commit message*" with your own name and a personal message. Observe the evolution of the command *git status* and *gitk* between each of these command. You can use *F5* from your keyboard to refresh the window *gitk*. In the end, the *gitk* window layout should be similar to figure 2. The branch with the name you have chosen will be on the top left, with the commit that you have made. On the bottom left, the sentence that you have written will appear in green with a plus sign, because that line have been added. If you have modified the line that was present in *MainFile.txt*, it will appear twice. Once in red with a minus sign and once in green with a plus sign. If one line appears only in red with a minus sign, it has been deleted.

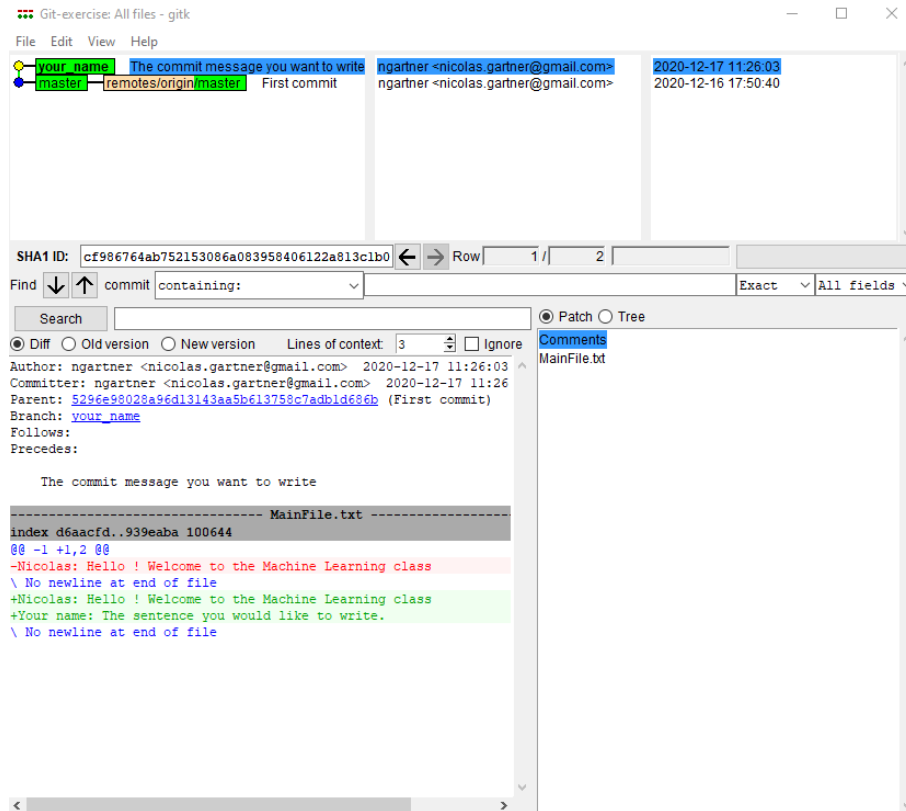


Figure 2: The display of gitk after your first commit

### 3 Sharing your work

After saving the commit, the data are saved, but only on your computer. This commit should be available for everyone, and so it will be uploaded on the git server (here it is Github). To do so we will push the branch that you have created to the server:

```
git push origin your_name
```

Observe what happened on gitk. You can now also go to the Github website and see the interface that you have there: <https://github.com/ngartner/Git-exercise>. As you can see, all students that have pushed their work have their branch on Github, which you can access. However, all the work has to be gathered on the same unique version. Therefore, one student will now take the role of team leader, and four others will manage one fourth of the class by alphabetical order. The team leader creates the final branch, that will be the combination of the four quarter. The four quarter-class manager, will have the task of gathering the work of the team into one branch. The branches will have specific names:

- The main branch name should be in the form 'Dayoftheweek-Period' and the period can be the morning or the afternoon, for example, 'Monday-Morning' or 'Thursday-Afternoon'.
- The name of each quarter-class branch should be 'TwoFirstLetterofDay - Am or Pm - FirstLetterofFirstPerson and FirstLetterofLastPerson' (Do not include spaces). For example, if it is Monday morning and the list of first letter of student names in the group starts with A and ends with E, then the branch name becomes: 'Mo-Am-AE'.

To combine the different branch, you will have to merge them. This can be done with:

```
git branch branch-name (If you have not done it before)
git checkout branch-name (If you have not done it before)
git pull origin other-branch-name
```

Normally, an automatic merging of the two branch should occur and will create a merging commit automatically. That commit is created on the branch you are currently working with and so you are not forced to switch to another branch. If not, this layout will appear in your file:

```
<<<<<<< HEAD:mergetest
This is my third line
=====
This is a fourth line I am adding
>>>>>>> 4e2b407f501b68f8588aa645acafffa0224b9b78:mergetest
```

- <<<<<<<: Indicates the start of the lines that had a merge conflict. The first set of lines are the lines from the file that you were trying to merge the changes into. For example, in the case you are currently on branch 'Monday-Morning' and pulling from branch 'Mo-Am-AE', the line below those symbols comes from the branch 'Monday-Morning'.
- =====: Indicates the break point used for comparison. Breaks up changes that user has committed (above) to changes coming from merge (below) to visually see the differences. For example, in the case you are currently on branch 'Monday-Morning' and pulling from branch 'Mo-Am-AE', the line below those symbols comes from the branch 'Mo-Am-AE'.
- >>>>>>>: Indicates the end of the lines that had a merge conflict.

To handle the conflict, you should replace that structure with the lines that you want to keep. After it is done, add a new commit.

```
git add .
git commit -m "Merging commit message"
```

Don't forget to push the result of the merging, so it can be available for everyone. This exercise is completed when all student sentences are contained in the main 'Dayoftheweek-Period' branch.

Have a look at <https://github.com/ngartner/Git-exercise/network>. It presents you the evolution of the work you have done and the different changes between branches. A similar evolution can be seen in the *gitk* window if you pull the final branch.

## 4 Final remarks

🔖 **git** might be useful for the team project, and you are encouraged to use it, however its use is not mandatory and will not be assessed. The fact that you would create a remote repository however, is something that you could share afterwards to your future employers to show your knowledge about machine learning. In the Github repository of the course, you will find a cheatsheet to help you out with the git commands.