Git Hands on session N°1

Git Introduction

Git is a distributed version-control system dedicated for tracking changes in source code in software development created by the IT-rockstar Linus Torvalds, the creator of Linux.

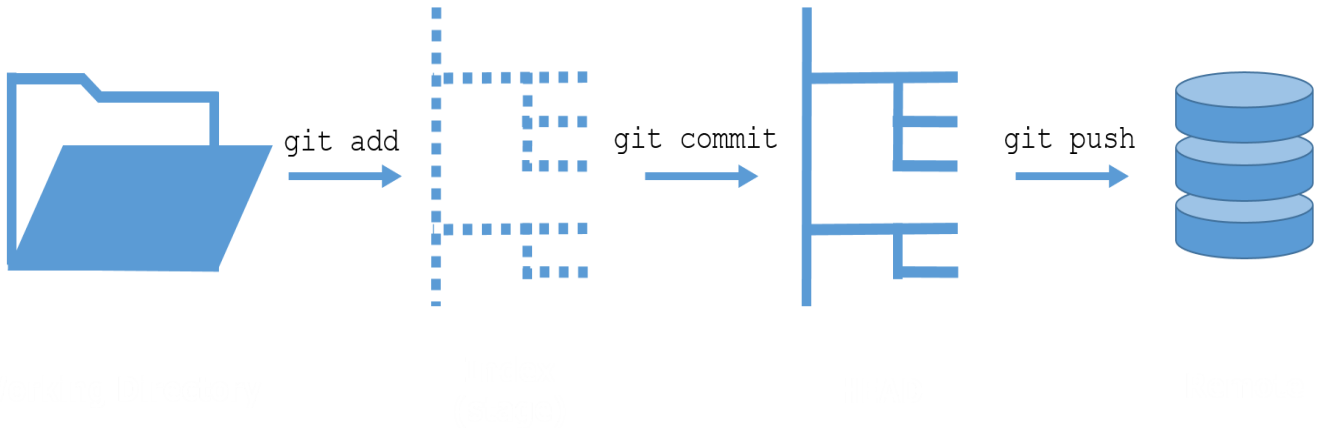
Fun fact: *Torvalds sarcastically used the name git (which means unpleasant person in British English slang): "I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'git'. The man page describes Git as "the stupid content tracker".*

The way git works is pretty simple and gravitates around two key concepts:

- Using staging to keep track of code updates on your local machine and on a remote one.
- Using a Branching policy to enable developments with multiple developers at the same time and avoid conflicts.

Today we will cover the first point, and the second one will be done in another session.

The staging policy is composed of 4 steps that you can see in the following drawing:



Of course each command could be reversed, but some of them are harder to reverse than other. We will cover them in this tutorial.

Let's go for some git

You will work on a project that is using git.

In this project you will have some *character1.txt* files and some *character2.txt* files.

They are played by a python script that will display the questions and the answers.

Your mission will be to edit thoses questions and answers files to correct mistakes and forgotten stuff in order to be the same as they were in the well-known movies they are from.

- There are 9 exercices and for each one you will have to answer to some questions. For this you will have to write the line of code that anwser to the question in the dedicated cell (starting with `#Your answer`.)

- All the actions that you will have to do will always be written in this color

- You will find a great documentation about git on the cheat-sheet that will be given during the learning session.

Exercise 1: Clonning

In this exercise we want you to clone a repository, this repository is available at: [REPLACETHISBYREPOSURL]

If you can't access to it or if the repository's name is wrong please let us know.

It's your turn, let's go clone this repository!

```
# Your answer
```

If everything is good you should see a directory named `[REPLACEBYREPONAME]` in the following output:

```
# Run this
ls
```

Is it good? Perfect, now we will just go inside this dowloaded directory for the following exercices.

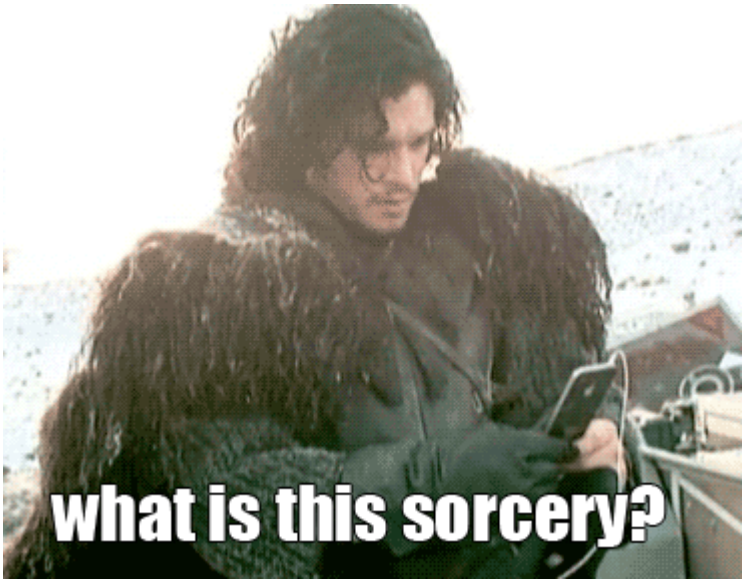
Please replace the path with the name of your git repository:

```
# Run this
cd [REPLACEBYREPONAME]
```

Exercise 2: Commit

In this exercise you will do your firsts commits.

To do this you will work on the dialogs from your first TV serie, Game of Thrones



Let's see how to run the scene.

```

```

```
# Run this
python display_dialog.py --char1 character1/Ygritte.txt --char2
character2/Jon.txt
```

Your job will be to replace the last line of the dialog, don't worry I wrote a script for that you will just have to specify by what you want to replace it.

Go ahead replace the Ygritte [???] by what she should say. (replace the [???] in the following cell by your text).

```
# Your answer
sh replace_in_file character1/Ygritte.txt "[???]"
```

Perfect now that you have done a modification on a file, you can commit it on git.

Use `git commit -m` to commit your changes The `-m` option is used to specify a git commit message, it is mandatory so watch out!

```
# Your answer
```

Did not work? that's normal you can't commit files that you did not add to your local changes. To see the files that have changes we usually use `git status`

Try again but this time add the files where you have done some changes and commit them

```
# Run this
git status
```

```
# Your answer
```

```
# Your answer
```

Bravo! You have done you're first commit

Exercise 3: Push

We will continue to work on this Game Of Thrones dialog for this exercise. Your goal is now to push your changes to the remote server that is on GitHub.

Go ahead push your commit to the remote server!

```
# Your answer
```

Easy right? Now you can check that everything is ok on the github page.

You just pushed to the master branch your commit. You can also commit multiple times before pushing to the master branch. It will push all your local commits to the remote server at the same time and you could still keep track of every commits you've done.

> ***Pro-tips:***
>
> - The `git status` command is really useful, use it every time you have a doubt on something, the provided informations are gold and it can't break anything!
> - The `git add` command can lead to some traps. you can use `git add .` or `git add *` to add all edited and new files that you currently have, but using this can lead to adding unnecessary files to the git remote. (just imagine that you accidently push a 1Tb Dataset that you just copied...). **Our advice is to add each file one by one whenever you are not familiar with git.**
> - In fact you can create and add a `.gitignore` to git. In this file you can list all things that should not be tracked by git. For example you can add `*.csv` in the gitignore file to avoid tracking CSV files.
> - Keep in mind that if you don't specify a branch, you will always push to the branch you are currently on. You can check which branch you are on using `git status`.
> - If you want to be sure to push to the branch that you want, to avoid any error for example, you can use `git push origin BRANCHNAME`

Exercise 4: Repeat

Because repetition is a learning enabler, now you will have to repeat Exercise 2 and Exercise 3 for the following dialogs:

> - Dr. No (*Sylvia Trench* and *James Bond*)
> - Fight Club (*Tyler*)
> - Appollo 13 (*Jack Swigert*, *CAPCOM* and *Jim Lovell*)
> - Back to the future part II (*Marty McFly*, *Doc Emmet Brown* and *Biff Tannen*)

4.1 Dr. No

```
# Run this
python display_dialog.py --char1 character1/James.txt --char2
character2/Sylvia.txt
```

Let's edit this

```
# Your answer
sh replace_in_file character1/James.txt "[???]"
```

And now update changes on remote server

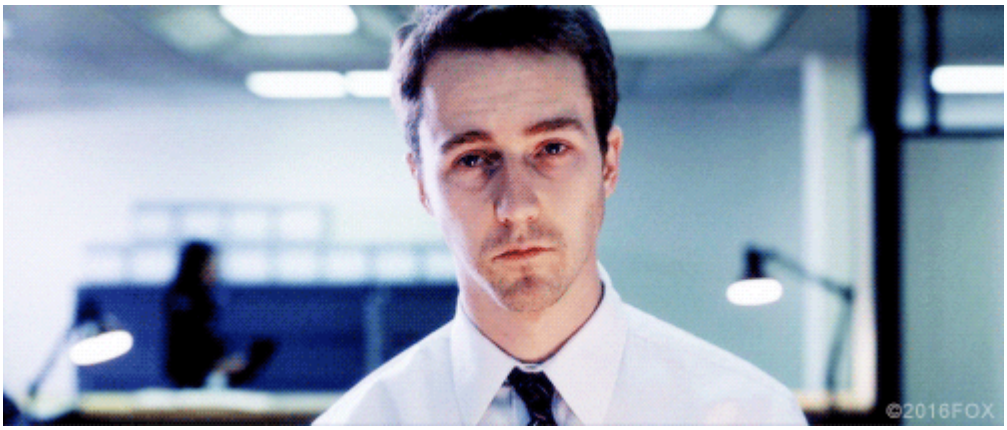
```
# Your answer
```

```
# Your answer
```

```
# Your answer
```

```
# Your answer
```

4.2 Fight Club




```
# Run this
python display_dialog.py --char1 character1/voice_fc.txt --char2
character2/Tyler.txt
```

Let's edit this

```
# Your answer
sh replace_in_file character2/Tyler.txt "[???]"
```

And now update changes on remote server

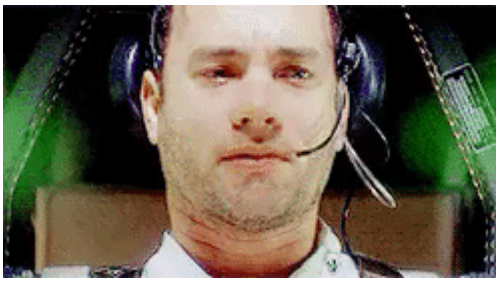
```
# Your answer
```

```
# Your answer
```

```
# Your answer
```

```
# Your answer
```

4.3 Apollo 13



```
# Run this
python display_dialog.py --char1 character1/appollo13.txt --char2
character2/Jim.txt
```

Let's edit this

```
# Your answer
sh replace_in_file character2/Jim.txt "[???]"
```

And now update changes on remote server

```
# Your answer
```

```
# Your answer
```

```
# Your answer
```

```
# Your answer
```

4.4 Back to the future part II

```
# Run this
python display_dialog.py --char1 character1/Marty.txt --char2
character2/Emmet.txt
```

Let's edit this

```
# Your anwser
sh replace_in_file character2/Emmet.txt "[???]"
```

And now update changes on remote server

```
# Your answer
```

```
# Your answer
```

```
# Your answer
```

```
# Your answer
```

Exercise 5: Revert a local commit

For this exercise we will work on Star Wars: Episode V - The Empire Strikes Back. (watch out for Mandela effects)

The goal will be to do multiple commits and to revert them. Keep in mind that you can commit as much as you want before using push to update the remote server.

```
# Run this
python display_dialog.py --char1 character1/DarthVader.txt --char2
character2/Luke.txt
```

For this time I'll give you the answer, Luke said *"NOOOOOOOOOOOOO NOOOOOOOO"* twice, just like this.

Let's edit this with the answer I gave you

```
# Your answer
sh replace_in_file character2/Luke.txt "[???]"
```

Now commit your changes (but don't push)

```
# Your answer
```

```
# Your answer
```

```
# Your answer
```

Ok, in fact I was wrong Luke didn't said it twice, it think it was just one simple *"NOOOOO"*.

Please manually replace your last updates with the new value

Now commit (again) your changes (but don't push)

```
# Your answer
```

```
# Your answer
```

```
# Your anwser
```

Well after reviewing the scene (that you can find just here: https://www.youtube.com/watch?v=bv20ZoBcdO8 we were right in the first place.

Reset the last commit to come back to the status we have with the two *"NOOOOOO!"*

```
# Your answer
```

```
# Your answer
```

```
# Your answer
```

Push your changes to the remote server.

```
# Your answer
```

```
# Your answer
```

You just reverted a local commit and pushed, well done!

> ***Pro-tips:***
>
> - The `git reset HEAD~1` is pretty simple if you want to go back in time for 5 commits you just have to replace the 1 by a 5.
> - Another way of doing this can be to find the commit id you want to go back and use: `git reset --soft commit_id`

Exercise 6: Revert a commit on Remote server.

Sometimes it can happen that you code a whole feature and that after pushing to the remote server you realize that a lot of what you have done was wrong. If so, you have 2 options:

- Do a new commit that will delete your changes

- Reset the commit on the remote server

For this exercise we will do the last option, because it can be complicated to delete all changes, you can forget something.

The movie dialog for this exercise will be from Forrest Gump.

```
# Run this
python display_dialog.py --char1 character1/Forrest.txt --char2
character2/Lady.txt
```

Let's edit the file and push changes to the remote server.

```
# Your answer
sh replace_in_file character1/Forrest.txt "[???]"
```

```
# Your answer
```

```
# Your answer
```

```
# Your answer
```

```
# Your answer
```

Perfect now let's imagine that we want to go back to the status we had before this exercise.

Find the commit ID and revert it on the remote server.

```
# Your answer
```

```
# Your answer
```

```
# Your answer
```

```
# Your answer
```

Exercise 7: Creating a branch

Questions

- Create a new branch and go on it.
- Do the Forrest Gump correction (like in the begining of the previous exercise) and push it on the new branch.

Exercise 8: Changing Branch

We will work with our last movie Titanic let's see the dialog:

- Go back to master branch
- edit answer/Jack.txt file with the replace_in_file script
- Push changes to the master branch

Exercise 9: Merging two branches

Now we will merge the branch you created into the master branch.

Questions

- Merge the branch you created in Exercise 7 in the master branch

Pro-tips

> - Merging branches is not so trivial it could leads to conflicts. Normally you did not have any in the previous exercise and we will conver how to handle them in the next git learning session.
> - Working with branches is the best way to use git, and we don't recommend to push stuff on the master branch, even if it's just your project. The master branch should always remain a pure and free-of-bugs zone.

If you came this far, you have finished the first part!! well done!

To go further:

- **Do level 1 & 2 here: https://learngitbranching.js.org/**
- **Find two friends and:** Try to put one by one the names of all the attendees of the session, all at the same time in a file. This will obviously generate conflicts that you'll have to manage.