Workshop – Git

The goal of this Workshop is to discover Git, how it interacts with GitHub or other repository managers and to practice using it mainly through the command line.

Required Tools/Preparation :

- You must have git installed on your machine.
- You must have a GitHub account.

Methodology :

- The workshop is not a manual or a list of commands to execute, it merely states a set of goals for each section. You will have to find out how to achieve them.
- Should a problem arrise you are encouraged to take initiative and troubleshoot it yourself. If the scope of the problem seems to big or confusing you can of course request help.
- Teamwork is encouraged. This workshop can be done on your own, or in groups of 2/3 people.

Ressources :

- https://git-scm.com/ The official git website.
- https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf A summary of the git commands proposed by GitHub.

Part 1 : Creating a repository and accessing it

- Create a repository on your GitHub account.
- Clone this repository on your machine with the git command.
- Create some files in your repository and commit them.
- Push your changes on your GitHub account.

Part 1.2 : Good Practices and tips

- Modify multiple files and commit them one at a time.
- Change your git apparent user name and email address.
- Store your git credentials so you don’t have to enter them each time.

Part 2 : Working with multiple people

- Invite someone else to colaborate on your repository and have them clone it. (If you are doing the workshop alone, clone your repository elsewhere to simulate another coworker.)
- Let the other coworker commit and push some files to your repository.
- Get the changes from your coworker on your machine (without re-cloning the repo).

Part 2.1 : Working with branches

- Create a **dev** branch on your project and add some commits to it. Push the branch to the repo.
- Merge the **dev** branch into the **master** branch.
- Add a few commits to the **master** branch.
- Have a coworker add some files on a **feature** branch, rebase this branch on **dev** then merge **dev** on master.

Part 2.2 : Resolving conflicts

- From the same commit, create two branches in which the same file is modified but not in the same way. Try to merge one branch into the other.
- Resolve the conflicts by picking the version you prefer.
- Optional : Try this part again with a git gui tool of your choice.

Part 2.3 : Proposing changes

- Fork the repository of a coworker on your GitHub account.
- Clone the forked repo, create a new branch, add some commits to it.
- Create a Pull Request on the original repository and have your coworker approve it.

Part 2.4 : Tags

- Add a tag on a commit and push it.
- Add a few commit, then go back to the tagged version.

Part 3 : Going Further

- Create a new empty repository on GitHub and set it as the new remote of your current project.
- Make some changes, *stash* them to have a clean work copy. Restore them.
- Make some changes, cancel them before commiting them.
- Add some commits to a branch. Push them. Delete them, localy and remotely.