Git repository documentation (group 1)

Nicolò La Cara

Abstract

This document provides a documentation of the git repository and some best practices in order to get the best out of git. Every group will be assigned to its own working directory (i.e. where that group should push its changes).

Contents

1 Setup

To clone the server's repository in your machine use the command:
`git clone <username>`[1]`@atelier.inf.usi.ch:/home/lacarn/Group_1`, don't show this address with anyone outside of this group. Before cloning and start working on the git repository, make sure to perform these three commands in case you didn't:

1. `git config --global user.name "your name"`
2. `git config --global user.email "your@email.com"`
3. `git config --global pull.rebase false`

2 Groups working directories

The repository `/home/lacarn/Group_1` is so composed: Where, each main directory (and so also its

```
Group_1
 ├── docs
 │    ├── how-to
 │    ├── general
 │    └── Summaries
 ├── styles
 │    ├── sg1
 │    ├── sg2
 │    ├── sg3
 │    ├── sg4
 │    ├── sg5
 │    └── template
 ├── html
 │    ├── sg1
 │    ├── sg2
 │    ├── sg3
 │    ├── sg4
 │    ├── sg5
 │    └── template
 └── resources
      ├── sg1
      ├── sg2
      ├── sg3
      ├── sg4
      └── sg5
```

subdirectories) is so assigned between every member:

***/template**	**docs**	**docs/Summaries**	***/sg1**	***/sg2**	***/sg3**	***/sg4**	***/sg5**
Sebastiano **Simone** **Yhoshua**	**Alessandra** **Nicolò** **Noah**	*All members*	Adrián Robert Valentyn Vasyl	Andrea Artur Nadia Reina	Frantisek Mateusz Theodoros Vlad-Stefan	Christian Klaudio Xirong Nicky	Daryna Gaia Xualin

Every member should follow this table and only push their changes in their correspondent directory, except for the **leaders** that may also push in other directories.

[1]your username from the email

3 Directories content

styles contains all the css sheets for the website general theme

docs contains all the documentation of the project

- **how-to** contains the css and git guidelines
- **general** contains the general documentation, like the inception document and the final report
- **Summaries** contains all the summaries provided by the group members after the researches about the project main subject (i.e. history of Informatics after 1968)

sg* contains all the files of a specific group

- **html** contains all the html files for the specific web pages of the groups
- **css** contains all the css sheets for the specific web pages of the groups
- **resources** contains all the resources for the specific web pages of the groups (photos).

4 Guidelines

These are some guidelines and best practices that you may follow in order to make things easier for all the group members and better organise the work.

4.1 Workflow

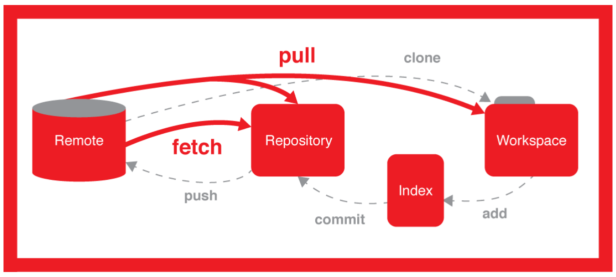
4.1.1 Pull operation

Always keep your local repository up to date by performing the **pull** operation before starting your work.

Before performing a pull operation and so get the changes in your code right away, you could also do:

1. `git fetch` to put your local repository up to date **without** actually merging the changes in your workstation;
2. `git diff main origin/main` to look at the differences between your local repository and the remote repository;
3. `git pull` **or** `merge` to actually merge the changes in your workstation.

This allows you to eventually avoid solving big conflicts directly in your code and get confused in case there are multiple.



A picture to better understand git functioning

4.1.2 Commit operation

After you are done with your task, make sure to `git add` and `git commit -m` your changes.

Don't commit every line of code that you change, but keep the commit as small as possible and focused on very specific, but meaningful, features, in order to reduce the risk of conflicts and make it easier to understand what changed in the project and eventually revert it.
Don't commit something that includes multiple changes in different sections of the codebase, since a bug in one of the areas would translate in a loss of work in case a revert is needed.
Always include a message in your commit that permits others to understand:

- What you changed in the code;
- Why you've done it;
- The context of your changes.

When writing commit messages make sure to be clear and concise, use verbs like *fix, refactor, add, remove* as prefix and include the directory in which the changes happened. Try to fit the message in maximum 50 characters, this helps other members to easily understand the exact location and meaning of your work.

Examples of good commit messages:

- `git commit -m 'Update styling for mobile responsiveness (styles)'`
- `git commit -m 'Fix bug with footer not showing (sg1)'`
- `git commit -m 'Add documentation for git repo (docs/how-to)'`
- `git commit -m 'Refactor pages structure for better clarity (sg3)'`
- `git commit -m 'Remove (sg3)'`

4.1.3 Push operation

As well as the pull, you should also regularly perform the **push** operation after every coding or refactoring session that you believe added an important feature that needs to be communicated with the other members (e.g an improvement).

If possible, a push operation should only contain commits that refers to the same component or area that you applied changes to (e.g. a page of the website). This, in order to better track every "big change" of every area of the project.
Remember to always pull (or fetch and then merge) before any push you make.