Git quick guide

Git is a collaborative tool that we use for Systems Engineering, for managing files (instead of Google Drive). This guide is about how to use it for normal SE meetings.

*Important note: this guide only works if people compartment their work: DO NOT WORK ON THE SAME FILE SEPARATELY or else that will generate conflicts. For text formats this can be solved quite easily, but for other weird formats (.mpp, .zip, everything that has a file a structure not corresponding to raw lines), it may not be solvable.*

**Creating a repository:** it is very simple. It can be easily done via GitHub. Facultative options include adding a .gitignore (Git ignores automatically untracked changes referenced in the .gitignore), a README.md (to check; it is the simplest way), and a license (none by default; switch the relevant licensing type if you want to protect the intellectual property of your work. *tolosat\_systems\_engineering* is under a GPL 3.0 license).

**Cloning a repository:** go to the command prompt, and go to any directory where you want the repository to be cloned to (*cd <folder path>* for changing the current directory; *dir [opt: <folder path>]* to list all the elements of the current directory if no option is selected, or to list those of the targeted folder if the folder path is set for instance *dir myfolder*). Then type:

*git clone https://github.com/<proprietary>/<repository>*

For instance, for *tolosat\_systems\_engineering* owned by Cédric (belmant) it is:

*git clone https://github.com/belmant/tolosat\_systems\_engineering*.

**Pull, push and committing changes:** go to the git repository directory inside the command prompt. Then to pull and erase the actual folder with changes from the *remote* branch (i.e. the server branch), type:

*git pull*

To see the list of commits, type:

*git log*

Once you have worked on one version of the repository, you need add your changes to the stage so that they are referenced as changes to be committed:

*git add <file/folder path>*

Where the <file/folder path> can represent a folder (therefore all the contents of the folder are staged), of a file (therefore only the selected file is staged). *git reset* allows you to reset the stage if you added one file to the stage by error.

*Note: . and .. are respectively the paths of the current directory and the parent directory.*

Then, make sure you have all the files you want to commit marked as staged by typing:

*git status*

And by reading the output of the command you can check it’s all correct. Then, before committing, to check whether anyone pushed in the meantime on the *master* branch:

*git remote update*

That will tell Git whether the original version you pulled from is still the same as the one on the server by reading the outcome of *git log*.

Then, two options:

1) the remote origin is the same as yours; then you can commit and push:

*git commit* (then “I” to insert text into the vim text editor, esc, :wq to write changes and quit)

*git push*

*git log* (to make sure you did everything correctly and that you commit is listed).

2) the remote origin is different. That means someone committed its changes to the server by pushing before you. In that case, you need to save (stash) your changes, pull the latest version, apply your saved (stashed) changes, commit and push again. The pipeline is the following:

*git stash*

*git pull*

*git stash pop*

*git commit*

*git push*

*git log* (the latest commit should appear just before your own commit).

For more information of Git commands, their actions and how to use them, just google it. Git is very complex and this guide is very simplistic.