# Tutorial Git Usage

Source: Rob Janssen – software engineer at StudyPortals

Created by: Bilger Yahov

Date: 21.11.2015

Git usage philosophy – Gitflow.

<http://nvie.com/posts/a-successful-git-branching-model/>

Recommended reading: <https://git-scm.com/doc>

In the Powershell console, Git uses several colors to denote the state of things:

* Light blue: locally, everything’s OK. This is no guarantee whatsoever that everything is up to date, though!
* Light blue with things like “+0 ~1 -1” in dark red. This means you have files that have not been added yet. The “+” number is for files added, the “~” number is for files modified, the “-“ number is for files deleted.
* Green: changes are committed and ready to be pushed.
* Yellow: conflict: you are trying to commit things while someone else has made changes as well.
* Bright red: your repository is behind the origin-repository on Github; you first need to do a git pull --ff-only

Since Git is decentralized (distributed), there are a number of things you can do locally to the point where the entire repository is just an empty husk of its former self where you do not affect anyone else working on it. Lots of actions can be done repeatedly and safely, and as long as you don’t do:

git push

So, one thing you always can do is this:

git status

It will tell you what the current state of your local repository is. However, it does not magically update itself; in the meantime, your fellow students might’ve made several commits, and git status will tell you “everything is up to date”.

The first thing you generally start with in the morning – that is, if you do not have any unstaged commits or commits not yet pushed – is

git pull --ff-only

“git pull” is a combination of git fetch (get the latest commits) and git merge (apply them to your local repository).

Then, you start working on changing some files, adding them, deleting others, whatever; it’s all good.

Let’s assume things go well and nobody gave you any nasty surprises:

git pull --ff-only

git add .

git commit -m “Change file x, delete file y, add file z”

git push

Commit early and commit often.

Try to keep your commits as logical blocks.

Agile development means no big ideas up front; get the basics right first.

Branches cost nothing; use them!

You can create branches from branches to avoid clashes when working on the same functionality/piece of code.