**Git clone** -> download a git repository

Downloading a git folder from the website results in no presence of the (hidden) folder .git. If ‘git clone’ is used from bash the hidden folder is present.

**Git log** -> show all the changes for a certain project. To exit the log lo press ‘q’

**git log –oneline** -> show a more concise log, only with a short description of the main changes

**git log –oneline –all** -> show the whole list of changes, including also those that occurred after the current version (one could be active on a version that has been superseeded)

**git log --oneline --all --graph** -> show the list of changes with some graphical representation of the changes (forks, braches, etc.)

**git checkout *fd7ec2a*** -> activate a previous version (commit) of the project (identified by its hash). It is also used to swith from one folder to another, for example from the master folder to a branch

**git checkout -b *branch\_name*** -> creates and activates *branch\_name* in one go. It combines git branch *branch\_name* + git checkout *branch\_name*.

**git diff *fd7ec2a 055e29b*** -> show the differences between 2 commits

**git diff --staged** -> show differences of the latest staged versions

**git init** -> initialise a repository

**git status** -> get info about the status of the files in a certain repository

**git add *file\_to\_add*** -> put the file “file\_to\_add” to the staging area

**git add .** -> adds all files (excluding those specified to be ignored)

**git commit** -> commit all files in the working folder. TO BE NOTED: if the files that are being committed are closely related they can be committed together (same commit) otherwise there should be separate commits (and readme) for every file.

|  |
| --- |
| Working directory |
| Staging area |
| Repository |

**git commit -m *“message\_to\_type”***-> commit with the possibility to directly type a message

**git commit –amend -m “*message\_to\_type*”** -> instead of creating a new commit

**git add --patch** -> add only a certain part of a script. The reason is because one could want to keep separate commits for different meaningful additions. Once run the command Git will ask for each of the additional parts of the modified file if they have to be added or not. The user will choose.

**git branch *branch\_name*** -> create a branch for the existing project. The branch can be made with the scope of keeping separate a certain area of development, that could be a code section

**git branch -d *branch\_name*** -> delete branch named *branch\_name*

**git checkout *branch\_name*** -> activate and go into a branch specified by branch\_name

to let git ignore files or folders a txt file called **.gitignore** has to be created, where the name of ignored items can be listed. If git status is called then it will not include in the list the files/folders that should be ignored

**git merge *branch\_name*** -> merging a branch into the main folder. This way the code in the branch is also included in the main code

**git push *origin master*** -> push the commit from the local repository *master* to github repository *origin*

**git fetch *origin*** -> send modification from remote repo to local repo

**git merge *origin/master*** -> merge the remote with the local repo

git pull would have done the same as git fetch origin + git merge origin/master