**Please note that this document is non-exhaustif.**

## What is GIT?

Main principle is to separate the code in pieces of advances, the “commits”. We will have an history of each piece, and each one is containing a message explaining and the code it changed

## GIT GUI

Git works well on command-line, but I strongly advise you to use a GUI. Android Studio uses its own GUI, so we will use this one. However, if you need to use a spare one (if you need to add a PDF for example) use **GitKraken**, so easy of use.

## GIT Definitions

repo: a repo is our git for the code, for example http://www.github.com/nanvix/nanvix is the git repo for application nanvix

.gitignore: file which is saying which files must be ignored on the repo (binaries mainly). I will create if needed

## GIT Functions

git clone: Will initialize a local repo (on your computer) based on a distant one (on github). First function to do to work on a repo

git pull: Will synchronize your local version with distant version. **Always do that before working**, in order to be up to date. Best way is to to it as force pull if there is a problem, that way it will erase your local modifications which are not on the server for better synchronization. Don’t do if you are currently working on files ofc, it’s before working

git add *; git commit -m “message”: on Android studio you won’t have to do the git add (in fact you’ll have to do this but with a GUI). When you modify files, and you want to put them to the server, you first do a git add to say “ok I want to add these files to my commit” (here in the example I added all modified files), then git commit -m “message” to say “I ant my commit number *automatically given number* to contain the modifications of this files *added files* and the message is *message*

git push: When you do a git commit, the commit is on **your local repo**. You then have to “push” your commits to the server. **This command has always to be done after a git commit** (otherwise it’s useless)

## How to erase a commit or several commits ???

If there are still local it’s easy, however if it’s on the distant server **use GitKraken to erase it/them**

If you have to erase only one, “git revert” will erase last commit **on your local repo** (always), then “git push --force”
If you want to erase more “git rebase master *commit number*” On GitKraken, you just have to right click on the commit you want to restart with “rebase master to this commit” then git push, force push

## <u>GIT Branchs</u>

You may be wondering what is “master”. Git is including **branches**. Branches are different versions of the code. **You have to create a branch if you’re going to work for a long time on files on which other may works** (Example: I want to work for at least 2 days on “main”).

To select the branch you want to work on, the command is “git checkout *branch*”. Then all your other commands will be done on this branch

That way, you will do your own spare version of the code, and when you’ll finish, we will **merge** your branch with the common branch (master).

git merge is the hardest command on git. It’s used when 2 people did modifications on the same file at the same moment (ex: 10h: A pull the code, 11h: B pull also, same version as A, 12h: A finished, he push is modifications on main.c, 13h B finished, he tries to push his modifications on main.c → problem because the version he pulled is not the version online anymore, a merge is required, and I will kill B). Merge is used to add the two modifications without erasing any. **We will try to avoid using it as much as possible**. For this, we will:

- Push frequently, and try to say to the group when we’re working on files other may also be working on
- If we have to work for a long time on a file, **create a branch**. That way we will have to merge only once, at the end and it will be easy