Git & GitHub

In this paper I'm gonna address the need and importance of an VCS(version control system) for a developer. And in the end I'm gonna present the Git tool and one of the most popular Hubs for it, GitHub.

What is Git ?
Git is a version control system that runs on most platforms (Linux, Windows, OS X etc...).It's a distributed system (it does not have a central database, everybody is free to make their own Hub, or store there version there own data locally). It is released under GNU GPL license that allows everybody to use the software in commercial and private use with only requirements being that you need to disclose the source, include a copy of the license in your code, and inform about the majore changes you made (if you did).

It was made by Linus Torvalds (the guy who made the Linux kernel) for the Linux Project .Since then it was used for a lot of projects, like : Android, Arch Linux, Debian, Eclipse, Fedora, Perl, Ruby on Rail, Mint, Gnome, Gimp, and many more.

Why use it ?
There are some versions control system, but the advance of Git is that it's free and open source, being made by the guys who believes in software freedom, that was an obvious choice. Some competitors are Mercurial, CSV SVN, and may more.

Having a piece of software that keeps track of the changes somebody made in a project is crucial if the size of the project is increasing. With this tracking mechanism we get a variety of tools. Git having a database that mainly stores data, not deleting, once you committed something (in other words decided that you should keep it) you know that you will find it. In fact it's very hard to hide something that you committed. Let's take this scenarios, you have a piece of software that is watched by git with 3 version, let's name them **version1, version2, and version3** if for some reason you wanna go back to **version2** the graph stricture will look something like this :

version1 → version2 → version3 → version2

It will make a symbolic copy of the versions you wanna revert to. This helps with tracking the people who made the change, why, and still allowing to go back to past versions.

Is Git only for programmers ?
The answer is no. People use this types of version control system for many thing, writing documentations, keeping track of changes in a book, or people who design logos and other things use it to keep track of the changes.

It is used in many fields of study.

What do I need to know to use Git?
Usualy, not much, but in this paper I'm going to talk about using the Command Line to interact with

Git, but there are many graphical tools.

Why use the command line if there are graphical tools ?
This is a very debated topic. There are two large categories of interfaces. CLI (Command line interface) and GUI (graphical user interface) . A lot of GUI are just graphical wrappers of an CLI. This is because a developer works better with commands for several reasons :

1. It's easy to test, when you work with commands you know that if you have a special command and a given input you can check if the output it's okay. You can have big chains with commands and input and check the output automatically;
2. It gives more flexibility, when you need to configure something you just write 2-3 more letters, in a CLI mode, but in a GUI, you need to hunt down panels, and check-boxes to get the desired outcome ;
3. It offers the ability of pipelining, you can chain commands and make the output of one be the input of the other, this allows for modularity, instead of having a big program that does everything you have more little programs that work together to obtain the desired behavior. We can take as an example 3 simple linux commands
   - **top** – shows all the process running
   - **grep** – searches for a given pattern
   - **pkill** – kills a process

   We can chain this commands, to get all the processes, search the one we want to stop, and stop it.

But of course not everybody is willing to learn a CLI because even if they have a pattern and if you worked with some the others will be familiar to a center degree, it's a steep learning curve.

Command line prerequisites and quick overview

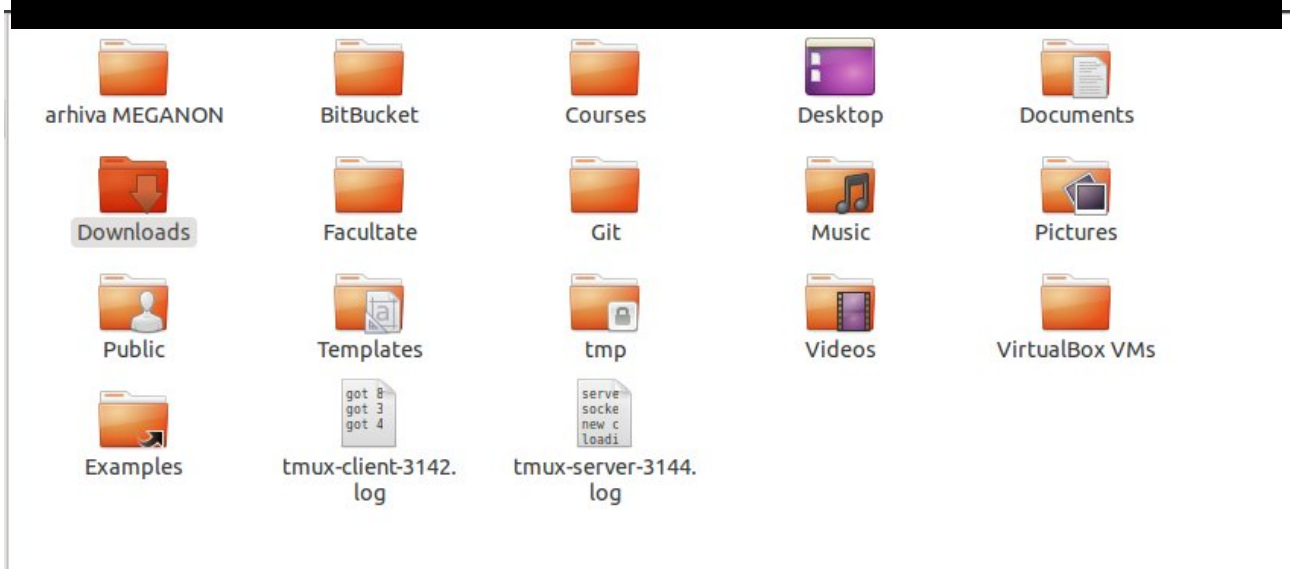
I'm going to make a brief walk through the commands we need to know .
Command that we are going to use :

- **ls** – lists all the folders
- **mv <arg1> <arg2>** - moves the file or directory specified in <arg1> in the path specified by <arg2>. Example : **mv myfile.txt dir1/myfile.txt** moves **myfile.txt** in the directory **dir1** and keeps it with the same name
- **cp <arg1> <arg2>** - copies the file specified in <arg1> in another file with name specified in <arg1>. Exemple : **cp file1.txt file2.txt**
- **touch <arg>** - nakes an emty file with the name specified in <arg>
- **mkdir <name>** - makes a directory with the name specified in <name>
- **rm** <arg> - remove the file with the name specified in <arg>

In the following pictures I'm gonna show the difference between GUI and CLI file handling .:

v

```
matei@matei-laptop:~$ ls -lh
total 92K
drwxrwxr-x 10 matei matei 4,0K dec 23 21:20 arhiva MEGANON
drwxrwxr-x  2 matei matei 4,0K dec 22 23:24 BitBucket
drwxrwxr-x  3 matei matei 4,0K dec 23 13:54 Courses
drwxr-xr-x  3 matei matei 4,0K dec 24 21:42 Desktop
drwxr-xr-x  2 matei matei 4,0K dec 22 16:39 Documents
drwxr-xr-x  4 matei matei 4,0K dec 24 22:10 Downloads
-rw-r--r--  1 matei matei 8,8K dec 22 16:27 examples.desktop
drwxrwxr-x  3 matei matei 4,0K dec 23 19:01 Facultate
drwxrwxr-x  3 matei matei 4,0K dec 22 23:25 Git
drwxr-xr-x  2 matei matei 4,0K dec 22 16:39 Music
drwxr-xr-x  2 matei matei 4,0K dec 22 16:39 Pictures
drwxr-xr-x  2 matei matei 4,0K dec 22 16:39 Public
drwxr-xr-x  2 matei matei 4,0K dec 22 16:39 Templates
drwxr-xr-x  2 root  root  4,0K dec 23 13:53 tmp
-rw-rw-r--  1 matei matei   54 dec 23 16:45 tmux-client-3142.log
-rw-rw-r--  1 matei matei  14K dec 23 16:45 tmux-server-3144.log
drwxr-xr-x  2 matei matei 4,0K dec 22 16:39 Videos
drwxrwxr-x  4 matei matei 4,0K dec 23 21:41 VirtualBox VMs
matei@matei-laptop:~$
```



This are the basic commands to handle file, for more information you could check this website http://www.comptechdoc.org/os/linux/usersguide/linux_ugbasics.html .

GIT

Instalation
To install Git you need to donwload the compiled file or donwload the source code and copile it yourself.
Here is the website : https://git-scm.com/downloads .

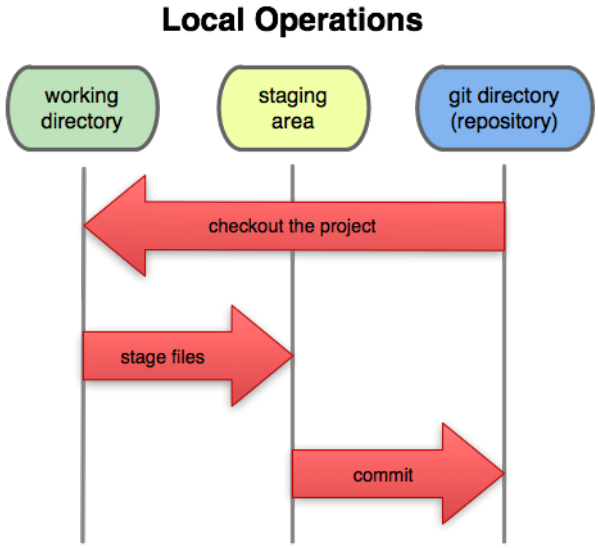
Setting up the name and email
Before we use Git we need to set up our name and email. This information will be used to sign our commits.
We do this by opening up a terminal (or cmd on windows) and run the followring commands :

git config –global user.name “Your Name”
git config –global user.email “yourEmail@example.com”

What concepts we need to understand
In order to use Git we need to know some concepts, in other words how the software is going to handle our data.

States
Our file can be in three basic states

- **Working Directory** in this state our file is not watched bi git, any change that we make will not be saved, or noticed
- **Staging area** in this state our file is ready to be packaged in a commit, in a version
- **Git repository** if our file is here it means that it's saved and we know exactly what changed



When a file is added to the Staging area we actually add a copy of it, if we change the file after we added in the Staging area the changes will not be recorded! This is something to be careful about. This is a part of the work flow, if you made a change to the file and you want that change to persist you can add the file again and a new copy of it will be ready to be committed.

Commands
Every command will be prefixed with the keyword **git**.

To create a local repository you use :
git init

To add a file to the staging area you type :
git add <fileName>

If you wanna add all the files from the folder you are in, you use
git add *

To commit (make a new version) with the changes that are kept in the Staging area you

use :

git commit

After this command you will be prompted to write a commit message, the purpose of this message is to inform the other developers of the changes you made, they should be able to understand what you tried to accomplish and what you added new with this version by only reading your commit message .

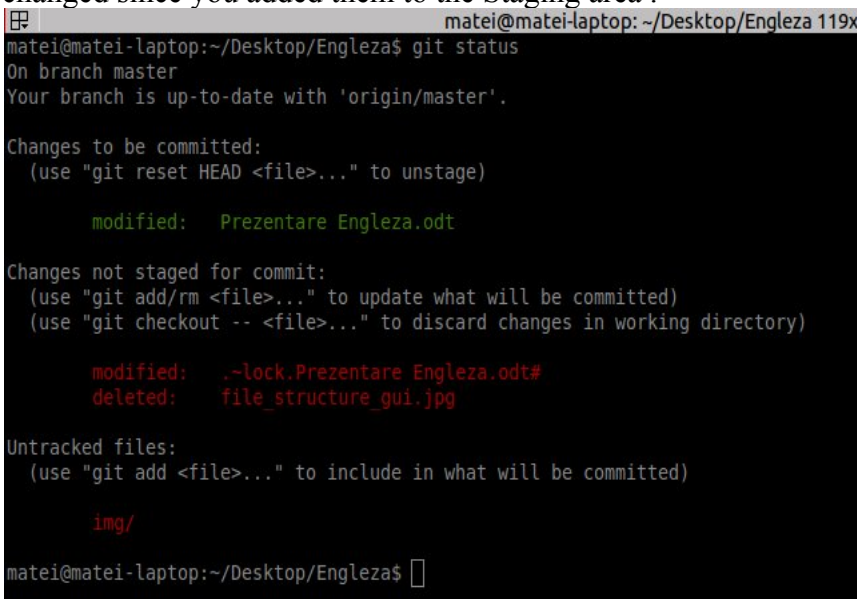
If your message is short you can use the shorthand :

git commit -m “your message goes here “

If you want to see the status of your repository (your git instance) you type :

git status

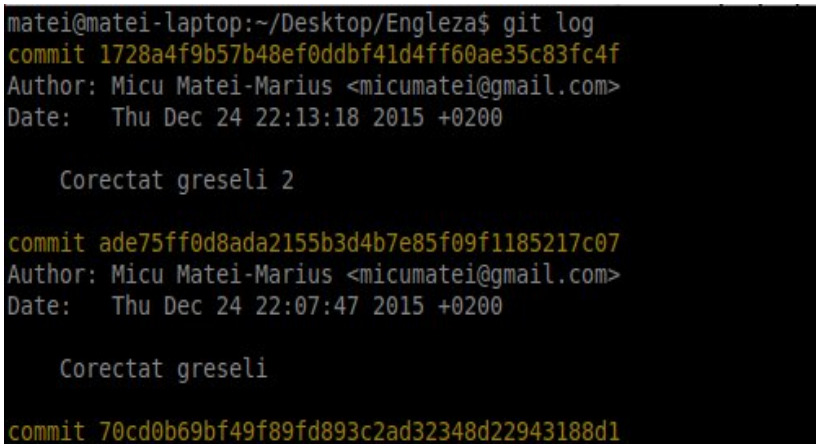
This command will show you the files that are not staged, the files that are staged, and the files that changed since you added them to the Staging area .



If you wanna log out all the commit you do :

git log

This will show you the commit chain, the author, the date and the message of the commit



GitHub

What GitHub does is let you store this repository in the cloud so other people can access it and commit to it, this is where Git and GitHub can help manage large projects with tens of developers that work from differer countries .

In order to use this feature we need to make an account on there website https://github.com/ .
They allow you to store unlimited repositories as long as they are public (everybody can see them) if you need private repositories you need to pay a monthly subscription. They offer student discounts and goodies if you can prove that you are a student. For some reasons my student email address was not confirmed. Here is the education discount : https://education.github.com/

After you made an account you create a repository in the cloud, for a easy guide look at this tutorial offered by GitHub support forum https://help.github.com/articles/create-a-repo/ .

Now we need to link our local repository with the one in the cloud .
We need to run the following command :

git remote add origin https://thisIsTheLinkToYourRepository.git

After we did this we have access to new commands that will sync what we have locally to what we have in the cloud, after we committed something we need to push it up in the cloud, and we do this by running :

git push origin master

If we wanna sync what other pushed from other places we just pull everything by runing :

git pull origin master

Fore more advance uses here are some great resources :

- https://www.atlassian.com/git/tutorials/
- http://git-scm.com/docs/gittutorial

To keep the spirit of Collaborating this paper will be stored on GitHub at this url :

https://github.com/micumatei/Proiect-Engleza-Git-and-GitHUb

Biography :

- http://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/
- https://sfconservancy.org/supporter/
- https://git-scm.com/book/en/v2/Getting-Started-Git-Basics
- https://ro.wikipedia.org/wiki/Git
- http://choosealicense.com/
- http://www.comptechdoc.org/os/linux/usersguide/linux_ugbasics.html
- https://git-scm.com/
- https://help.github.com/articles/adding-a-remote/
- https://www.atlassian.com/git/tutorials/
- http://git-scm.com/docs/gittutorial