***Introduction***

***Git is an example of VCs (version control systems)***

***VCs is repository of files (like a database), for every change and new submitted version of the project, you can know who submitted the version, when, what are the changes***

***Git is a tool helps teams working on a project such that:***

***Each person has a copy of the work, but there’s a copy saved in a safe place***

***They may work in parallel on the same file***

***After each person finishes his work, he submits the new code to the safe place and merges it with the original code after resolving conflicts***

***History of submissions and original code versions are kept***

***Some uses of VCs:***

***Reversion: going back to and older version in case the new version has problems***

***Tracking: if there’s a something you don’t remember who submitted it or why or when***

***Branches: adding new feature, but away from the original code so as not to affect it***

***Merging branches: adding the branches to the original code, git takes a great part in this process***

***Examples of VCs: CVS, subversion, git***

***Why git?***

***Fast***

***Well tested (used by a lot of companies)***

***Distributed (each person has a copy, no need to access the server)***

***Popular (has many tools)***

***Open source (created by the creator of Linux)***

***Basics with Git***

***Commands:***

***1) initialize git repository:***

***(git init)->, when you write this command, a subfolder git is created that carry the history***

***2) To know the status of the repository:***

***(git status)->show untracked files, and on which branch you on***

***Note that: if there’s a file not tracked (not in staging area ) will appear in red color, and after adding it to the staging area u need committing it to the server, all of this you may know from the git status command***

***3) To add/remove file to the staged area:***

***(git add/rm + filename/filepath)->add file to the staging area, not yet committed***

***Additions to add command: git add’\*.txt’, all .txt files will be added, including those in subfolders***

***4) to commit a file to the repository:***

***(git commit –m “commenting on the submission “)->submitting to server***

***5) to know the history of server submissions (commits)***

***(git log)->include submission date, author, and commit ID &message of commit***

***NOW, we’ve done all the local work (local repository), we want to submit our work to and internet server (remote repository) (Github/bitbucket)***

***6) To create remote repository***

***(git remote add+name+URL)->add new repository in the URL with the name***

***7) Add files/changes to remote repository***

***(git push –u+repositoryName+branchName)->files can be accessed remotely to other members***

***Note that: the first time you write the command with –u, after that you just write (git push), repository name is saved***

***8) Get files from remote repository***

***(git pull++repositoryName+branchName)->will show you new files added by other members***

***9) To know what changes happened since your last commit***

***(git diff HEAD)->will show u which file changed &the changes inside it***

***10) Changes happened is the staging area***

***(git diff --staged)->files added to staged area and changes in them***

***11) Remove a file back from staging area***

***(git reset+filepath)-> File will be out of staging area as if not added***

***12) To reset file data to data at last commit***

***(git checkout-- +filepath)->overwrite the file data with the data of last submission***

***13) Add new branch (doesn’t affect the repository)***

***(git branch+name)***

***14) to know the existing branches***

***(git branch)-> list all branches , note that a\* will inserted before the branch you are accessing***

***15)to switch to a branch***

***(git checkout +branchName)***

***Note that: after doing changes in a branch you must commit it so as you can do the next command***

***16) to add changes from another branch to the current branch***

***(git merge+branchName)->take data and changes from the branch add it to the current branch***

***17) to delete branch after getting data from it***

***(git branch –d+branchName)***