What's Git?

Git is a DevOps tool used for source code management. It is a free and open-source version control system used to handle small to very large projects efficiently. Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development. Linus Torvalds created Git in 2005 for the development of the Linux kernel.

Install & set Git

Go to Git/downloads or copy this link: (https://git-scm.com/downloads).

Using Command Prompt (terminal):
Linux: `sudo apt install git` ➔ Install git
W/L: `git --version` ➔ Check git version on your system, e.g. git version 2.17.1 or version 2.19.1.windows.1
L: `git config --global user.name "name"` ➔ Define git's user name
L: `git config --global user.email "email"` ➔ Define git's user email

Windows: `git config --global user.name name` ➔ Define git's user name
W: `git config --global user.email email` ➔ Define git's user email
W: `git config --list` ➔ View all added data

How Git works?

Git allows users to track code changes and manage their project using simple commands. The heart of Git is a repository used to contain a project. A repository can be stored locally or on a website, such as GitHub. Throughout development, the project has several save points, called **commits**. The commit history contains all the commits, i.e., changes implemented in the project during development.

- Git project sections: Working Directory ⇨ git add ⇨ Staged Area ⇨ git commit⇨ Git Repository
- Git file states: Modified ⇨ Staged ⇨ Committed

Initializing Git Repository

A Git repository is the .git/ folder inside a project. This repository tracks all changes made to files in your project, building a history over time. Meaning, if you delete the .git/ folder, then you delete your project's history.

L: Select your project ⇨ Right click ⇨ open in terminal ⇨ `git init` ➔ repository (.git file) is created
W: Select your project ⇨ Right click ⇨ Git bash here ⇨ `git init` ➔ repository (.git file) is created

Git's Ordinary Commands

W/L: `ls` ➔ get all files in this route
W/L: `clear` ➔ clear all terminal's instructions
W/L: `q` ➔ get out from terminal's (End) message
W/L: `cd directory name` ➔ go to specific route
W/L: `cd` or `cd ~` ➔ navigate home directory
W/L: `cd /` ➔ navigate root directory
W/L: `cd ..` ➔ navigate up one directory level
W/L: `cd -` ➔ navigate back one directory level
W/L: `pwd` ➔ print the current terminal rout
W/L: `mkdir file name` ➔ make a new file

W/L: `touch file_name` ➔ create new file
W/L: `code file_name` ➔ open file in coding App
W/L: `nano file_name` ➔ open file in terminal
W/L: `git init` ➔ start local repository

Git's Basics Commands

Files in git repository are tracked and untracked, untracked files have to be tracked:
W/L: `git add file_name` ➔ add a specific file to the repository
W/L: `git add .` ➔ add all files to the repository
W/L: `git status` ➔ track project's files in the repository
W/L: `git diff file_name` ➔ illustrates accurate details rather than git status for untagged changes (before using `git add`)
W/L: `git commit -m "message"` ➔ insert project's file/s to the repository
W/L: `git commit -a -m "message"` ➔ track and insert files to repository directly (without using `git add`)
W/L: `git commit -amend` ➔ edit previous commit message
W/L: `git log` ➔ view all commits history by the repository
W/L: `git log -p` ➔ view the differences by commits
W/L: `git log -p -2` ➔ view the differences by commits 1 and 2
W/L: `git log --stat` ➔ show statistics for all done commits
W/L: `git log --pretty=oneline` ➔ show only commit's messages
W/L: `git log --pretty=format:"%ae"` ➔ show only commit's messages entry owner (email)
W/L: `git log --pretty=format:"%an"` ➔ show only commit's messages entry owner (name)
W/L: `git log --pretty=format:"%an %ae"` ➔ show only commit's messages entry owner (name email)
W/L: `git log --since=2.days` ➔ show only commits by previous 2 days
W/L: `git log --since=1.months` ➔ show only commits by previous 1 months
W/L: `git log --since=2.weeks` ➔ show only commits by previous 2 weeks
W/L: `git log --untill="2018-11-16"` ➔ show only commits until desired date
W/L: `git log --before="2018-11-16"` ➔ show only commits before desired date
W/L: `git log --author="author_name"` ➔ show only commits by this author
W/L: `git reset file_name` ➔ undo all commits in this file without deleting edits
W/L: `git reset --hard file_name` ➔ undo all commits in this file and delete edits temporally
W/L: `git checkout -- file_name file_name` ➔ undo deleting files, back them again to the repository
W/L: `git checkout -- file_name` ➔ undo edits in this file

Files in git repository can be untracked or seen, by:
W/L: `touch .gitignore` ➔ make a special git file
Open `.gitignore` file ⇨ type in the file `*file extension (e.g. *.txt)` or `file name/`

W/L: `git mv file_name file_new_name` ➔ rename file
W/L: `git mv file_name container_name/file_name` ➔ move file to another container
W/L: `git rm file_name` ➔ delete file
W/L: `git config --global alias.alias_name command_name` ➔ make a special git alias

Git's Branching & Merging Commands

W/L: `git branch` ➔ get all repository branches, the active one in green color
W/L: `git branch branch_name` ➔ create a new branch for the repository

W/L: `git checkout branch_name` ➔ move to the desired branch
W/L: `git checkout -b branch_name` ➔ make a new branch and switch to it directly
W/L: `git merge branch_name` ➔ merge a branches with master
W/L: `git stash save "message"` ➔ save edits temporally and report a problem
W/L: `git stash list` ➔ view all stashed saved
W/L: `git stash list` ⇨ copy stash name e.g. stash@{0} ⇨ `git stash apply stash_name_` ➔ apply edit in our files
W/L: `git stash drop` ➔ delete last stash
W/L: `git stash pop` ➔ redo edits and delete from stash list
W/L: `git tag version_name` ➔ make low tag (version) for the repository
W/L: `git tag` or `git tag -l` ➔ all tags history
W/L: `git tag -a version_name -m "message"` ➔ make high tag (version) for the repository
W/L: `git show version_name` ➔ show the version
W/L: `git log` ⇨ copy commit id ⇨ `git tag -a version_name commit_id`➔ apply version to late commit ⇨ ctrl + x ⇨ Yes ⇨ Ok
W/L: `git log` ⇨ copy commit id ⇨ `git checkout -b branch_name tag_name`➔ apply branch from desired version

Git Sharing and Updating Projects Commands

W/L: `git clone URL project_name` ➔ clone abroad repository
W/L: `git remote` ➔ all remote repositories list
W/L: `git remote -v` ➔ all remote repositories list with links
W/L: `git remote add URL_nickname Remote_URL` ➔ add remote repository
W/L: `git push -u URL_nickname branch_name` ➔ push (add) branch's file to remote repository
W/L: `git remote rm URL_nickname` ➔ delete remote repository
W/L: `git remote rename URL_nickname new_URL_nickname` ➔ rename remote repository
W/L: `git fetch URL_nickname remote_branch_name` ➔ get edits (commits) from remote repository
W/L: `git branch -a` ➔ view all new edits
W/L: `git log URL_nickname/remote_branch_name` ➔ get in commits in remote repository
W/L: `git merge URL_nickname/remote_branch_name` ➔ add new edit in remote repository to local file
W/L: `git pull URL_nickname remote_branch_name` ➔ add new edit in remote repository to local file directly

Huthaifa Altiti, Electrical Engineer
Reach me on GitHub or Twitter: **@huthaifaaltiti**