Git Commands

* **clone** - bring a repository that is hosted somewhere like GitHub into a folder on your local machine;
* **add** – track your files and changes in Git
* **commit** – save your files in Git
* **push** – upload Git commits to a remote repo, like GitHub
* **pull** – download changes from remote repo to your local machine, the opposite of push

usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]

[--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]

[-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]

[--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]

<command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)

clone Clone a repository into a new directory (git clone git@github......./.....)

init Create an empty Git repository or reinitialize an existing one (git init)

1. Work on the current change (see also: git help everyday):

add Add file contents to the index (git add nnnn.dwe)

mv Move or rename a file, a directory, or a symlink

restore Restore working tree files

rm Remove files from the working tree and from the index

sparse-checkout Initialize and modify the sparse-checkout

2. Examine the history and state (see also: git help revisions):

bisect Use binary search to find the commit that introduced a bug

diff Show changes between commits, commit and working tree, etc

grep Print lines matching a pattern

log Show commit logs

show Show various types of objects

status Show the working tree status

3. Grow, mark and tweak your common history:

branch List, create, or delete branches

commit Record changes to the repository

merge Join two or more development histories together

rebase Reapply commits on top of another base tip

reset Reset current HEAD to the specified state

switch Switch branches

tag Create, list, delete or verify a tag object signed with GPG

4. Collaborate (see also: git help workflows):

fetch Download objects and refs from another repository

pull Fetch from and integrate with another repository or a local branch

push Update remote refs along with associated objects (for example git push origin main)

'git help -a' and 'git help -g' list available subcommands and some concept guides. See 'git help <command>' or 'git help <concept>' to read about a specific subcommand or concept.

See 'git help git' for an overview of the system.

git:(main) ls –la - list everything in a directory including hidden files and folders(in blue is a folder).

Saving changes in Git in Terminal:

git status -> a command that shows all the files and folders updated, created or deleted but haven’t been saved in a commit yet.

git add . ->if the result is green - all changes are tracked and they are ready to be committed

git commit –m “here is a message to your commit” -> saving a commit localy

(after a message you can also add a new –m in order to add a description to your commit)

git push -> pushing a saved commit live to a remote repo where my project is hosted

GIT PUSH

**git push origin main**

origin - means a location to a git repository

main - is a branch we wanna push to

git remote add origin [git@github......./](mailto:git@github......./)..... -> to add a reference to the remote repository on Github

checking on that:

git remote –v

GIT BRANCHING

Main branch

Feature branch

git branch -> shows a list of all branches created, \* indicates the branch that you are in at the moment

git checkout -> is used to switch between branches

git checkout –b name\_of\_your\_branch -> creating a new branch with a name name\_of\_your\_branch

git checkout main -> switching to the branch main

git diff -> shows what changes have been made and compares two versions of the code and shows all the lines that have been changed

In order to merge you should:

1) Push all the commits to GitHub (git push )

2) copy the given line and set it to your command line

3) Pull your commits to the local main:

git pull (origin – if you haven’t set an upstream yet)

4) git merge main (if you are not on the main)

5) resolve any popping up conflicts

6) git status – shows that all files have been modified ( you need to make another commit to fix the merge conflicts)

7) git commit –am “Your message about merging process”

Deleting branch – git branch –d name\_of\_your\_branch

git commit –am “Your message” -> adds and commits at the same time, but it only works for modified files not for the new ones.

UNDOING IN GIT

Undoing a change:

git reset - to undo the change with a mistake (or git reset nameofthe.file )

Undoing a commit:

git reset HEAD~1 -> HEAD is a pointer to the last commit – so it undoes the commit that has been just made

git log -> to see the log of all your latest commits

git reset \*hash of the commit from the log\* -> unstages all the commits made after the indicated commit

git reset –hard \*hash of the commit from the log\* -> completely removes all the commits made after the indicated commit

FORKING

Forking is good for the repos that are not your own