Git & Github Cheat Sheet by 3SN

3sn.cheatsheet
All ALIASES are stored in a local file, named **.gitconfig**.
On a Windows system, this file is located in the folder **C:\\.gitconfig**

Youtube resources

The very basics
A bit more advanced, though still very understandable

```
// Defining your name and email

/* These settings will be used in ALL your repo's. */
/* Has to be done only once                        */
git config --global user.name  "Your Name"
git config --global user.email "Your email"
```

Initialising your LOCAL repo

```
/* Has to be done only once PER PROJECT
=== Remark ====
I strongly recommend to:
1. Create your project on Github
2. and clone it locally
3. Merge the .git folder and the readme file with your local
        files.. */

// Other solution:
git init // In local Project Folder.

// CREATE / DEFINE Aliases
git config --global alias.<alias_name> "<git_command>"

// Example
git config --global alias.co "checkout"
```

COMMON aliases

```
// Status .:. Current situation
git stat // -- git status

// Log in 1 line
git lo1 // -- log --oneline
```

```
// Tree show log in tree format
git tree // -- log --oneline --graph --decorate --all
```

BRANCHES aliases

```
// 1. CREATE a new branch with the specified name.
git brc <branch_name> // -- git branch <branch-name>

// 2. CHECK OUT / SWITCH TO a specified branch.
git bro <branch_name> // -- git checkout <branch-name>

// 1 & 2 CREATE & CHECK OUT / SWITCH TO a specified branch.
git brco <new_branch_name> //   -- git checkout -b <new-branch-
        name>

// MERGE the specified branch into the CURRENT branch.
git brm  <specific_branch_name> //   -- git merge --nof
        <specific-branch-name>

// DELETE the specified branch.
git brd <branch_name> //   -- git branch -d <branch-name>

// 3. LIST all LOCAL branches in the repository
git brll //    -- git branch

// 4. LIST all REMOTE branches.
git brlr //    -- git branch -r

// 3 & 4. LIST all REMOTE branches.
git brla //    -- git branch -a
```

STAGING Related

```
// 0. Add One File to staging
git staf <filename>  //   --  git add <filename>

// 1. Add All Files to staging
git staa    // --  git add . OR git add -all

// 2. Commit with message
git com “<msg>” //   --  git commit -m “<msg>” OR git commit –
        message “<msg>”
```

```
// 1 & 2. Add all Files to staging and Commit with message....
        All in One .:. BETTER .:.
git coma “<msg>” //   -- git commit -am “<msg>” OR git commit –
        all –message “<msg>”

// Remove One File from staging (but keep changes)
$ git stad <filename> //   --  git reset HEAD <filename>

// Remove ALL Files from staging (but keep changes)
$ git stada  //   --  git reset HEAD
```

REMOTE Related

```
// PULL / GET all files from Github
git pull //   --  git pull

// PUSH / SEND all Changes TO Github
git push //   --  git push
```