**Git & GitHub**

|  |  |
| --- | --- |
| **Command** | **Function** |
| git init | Initializes a new Git repository |
| git clone <url> | Clones a remote repository to your local machine |
| git add <file> | git add \* | Adds changes to the staging area | Add all the files |
| git commit -m "message" | Commits the changes in the staging area to the repository |
| git status | Shows the status of the working directory |
| git log | Displays the commit history |
| git pull | Fetches and merges changes from a remote repository |
| git push | Pushes local commits to a remote repository |
| git branch | Lists all branches in the repository |
| git checkout <branch> | Switches to a specific branch |
| git merge <branch> | Merges changes from one branch into the current branch |
| git remote -v | Lists all remote repositories and their URLs |
| git fetch | Downloads objects and refs from another repository |
| git reset <file> | Unstages changes for a file |
| git reset --hard <commit> | Resets the repository to a specific commit |
| git stash | Stashes changes in a dirty working directory |
| git diff | Shows the changes between the working directory and staging |
| git remote add <name> <url> | Adds a new remote repository |
| git rm <file> | Removes a file from the working directory and staging |
| git config <option> <value> | Sets configuration options for Git |

# **1) Configuration**

* Set your name:

**>> git config --global user.name "Your Name"**

* Set your email:

**>> git config --global user.email** [**youremail@example.com**](mailto:youremail@example.com)

* Set your preferred text editor (e.g., for commit messages):

**>> git config --global core.editor "vim"**

* List all your current Git configurations:

**>> git config –list**

* Set an alias (e.g., 'lg' for a pretty log output):

**>> git config –global alias.<abrev> ‘word that you want to abbreviate’**

**Example:**

make an alias 'g' to 'git'

>> git config --global alias.g 'git'

# **2) Public Key**

To set a public Key write the following command

**>> ssh-keygen -t rsa**

Set password🡪 view the public key

**>> cat ~/.ssh/id\_rsa.pub**

Copy it and add it to your account (setting🡪security🡪SSH🡪Create New SSH key)

- test your SSH connection to GitHub using:

**>> ssh -T git@github.com**

# **4) Branching**

* Create a new branch:

**>> git branch <branch\_name>**

* Switch to a branch

**>> git checkout <branch\_name>**

* Creating and switching to a new branch in one step

**>> git checkout -b <branch\_name>**

* Listing all branches in the repository

**>> git branch**

**>> git branch -a** #list all branches regardless of the relation between them.

* Merging changes from one branch into another:

**>> git merge <branch\_name>**

* Deleting a branch (after merging or if no longer needed):

**>> git branch -d <branch\_name>**

# **5) stash**

* Stash the changes

**>> git stash**

* List stashed changes

**>> git stash list**

* Stash changes with a custom name instead of (stach@{#})

**>> git stash save "Custom Name"**

* pop the most recent stash and remove it from the stash list

**>> git stash pop**

**>> git stash pop stash@{1}** # pop a specific stash

* Apply🡪 same as pop but without removing it from the stash list

**>> git apply**

**>> git apply stash@{1}** # Apply a specific stash

* Show changes in the most recent stash

**>> git stash show**

**>> git stash show stash@{1}** # show changes in a specific stash

* Clear all stashed changes

**>> git stash clear**

# **Create respo from existing project.**

* Initialize your existing folder, add it ,and commit it

**>> git init**

**>> git add \***

**>> git commit \_m “Message”**

* Go to git hub create new repo, then copy the repo link (http or SSH)

**>> git remote add origin** [**git@github.com(user/repo)**](mailto:git@github.com(user/repo))

* Pull then push your commits to the repo

Note: by adding “-u” you will pull then push

**>> git push -u origin <branch name>**

# **Reset the Head**

The idea is to change the pointer head and make it point on older commit hence delete. Hence any commit you have did after the one that you have currently point add will be deleted.

* Show all your previous logs

**>> git log**

A screen shot of a computer

Description automatically generated

* Copy the commit code that you want to be at

**>> git reset –hard <commit code >**

**>> git push origin <branch Name> --force**

# **Ignore File or Directory**

* Create a “.gitignore”
* **>> touch .gitignore**
* Open the ignore file to write In it

|  |
| --- |
| .gitignorefile |
| \*.txt #ignore any .txt file.  ! file\_name.txt # except this file don’t ignore it.  folder\_name \ # ignore this folder.  file\_name.py # ignore this file |

Note: the only way to push a file that exist in the “gitignore” file is to force it

**>> git add -f <filename.py>**