Git Cheatsheet

Table of Contents

Working with local repository

- ### git init
  - creates new git repository
- ### git clone <url>
  - clones an existing repository
  - *`git clone https://github.com/bujdeabogdan/git-cheatsheet.git`*
- ### git add
  - adds file to staging area(prepare for commit)
  - git only commits files that are in the staging area, this way you can change many files but you can select which files to commit and which ones to not
  - *`git add file.txt`*
  - this one adds all the files in the staging area

```
git add .
```

- ### git commit
  - creates a snapshot of the repository
  - saves the state of the files at a certain moment

        - *git commit –m "commit message"*
        - *git commit –a –m "this commits all the files in the repository even if they are in the staging area or not"*

- git commit –amend
    - useful when you made a commit but you want to include other changes in that commit
    - let's say you forgot to change the db ip from local to production, and you don't want to make another commit, or you made a mistake in the commit message
    - the flow is like this:
        - you make the bad commit
        - `git commit -m "bad commit"`
        - make your changes(create/update/delete files) and stage them
            - `git add .`
        - amend the commit
            - `git commit -amend - m "new message"`
        - or if you don't want to change the message
            - `git commit -amend -no-edit`
- ignore files
    - when you want to ignore certain files or type of files, you can use .gitignore
    - this is a file in the root of the repository, that contains the name/path/type of file(s) that you want to ignore
    - here are some examples:
        - `*.exe`
            - will ignore all exe files from all folders
        - `file.tmp`
            - will ignore the files named "file.tmp"
        - `/bin/*.txt`
            - will ignore all the .txt files in the folder bin
- stash files
    - when you have to switch to another branch, instead of doing a commit to save the current state of the project, you can just stash the changed files

```
git stash
```

- after this, the files reset to the last commit and you can switch between branches
- when you go back to the last branch, use

`git stash apply`

- you can create more than one stash, but remember that "apply" will use the latest one
- you can view the list with
    - `git stash list`
- you can remove the latest stash with
    - `git stash drop`

Pull requests & Forks

- https://help.github.com/articles/fork-a-repo/
- https://help.github.com/articles/using-pull-requests/