How to Use Git Commands

To work on a project just by yourself

1. Start a repository (or project) in github account
2. Keep it as public, name it, initialize the repository with a README file
3. Type into readme what you want to describe the project.
4. To work on files in a repository do the following:
5. Go to repository
6. Click clone or download (choose clone with HTTPS)
7. Click on clipboard image to copy
8. Go into terminal (gitbash). Navigate to the correct folder (may need to create a folder first)
9. Type in ls to see a list of all files in folder if have trouble finding folder and then navigate to it.
10. Type 'git clone' in command line and then past what you copied from github
11. Press enter
12. Open up the folder. To do from command line, type start . and then name of folder (make sure you are in correct directory)
13. Now navigate to the project that was created in github using the cd command followed by folder or file name. It will have the word main after the folder name once you have navigated to the correct place.
14. If you need to move files to the new repo folder, you will need to do that manually if you're on your work computer, trying to move from the command line makes them disappear – probably a security issue.
15. Now we need to sync things:
    a. Type in git status – this tells you if you have untracked files.
    b. Use the command git add . to add all the untracked files.
    c. Type in git status again – this shows you the new files to be committed.
    d. Type in git commit -m "comments in here"
    e. Type in git push – this lets github know we committed these changes. It may ask you for your password.
16. When working on a repository with others, use the command git pull to pull the latest approved versions of things from the repo.
17. If you want to change something in the repo – create a branch to do that, do NOT work on the main branch directly. Then you push your side branch and it's reviewed by team and if is ok, someone else will approve. If working on a team, is bad form to approve your own branch modifications (merge your own pull request) unless you are the ultimate leader.
18. How to create and work on a side branch:
    a. Type in git branch to see how many branches you have
    b. To add a branch and check it out at same time: git checkout -b branch name
    c. Type in git branch again and your new branch should be listed
    d. To move to this new branch from the main branch:
        i. Git checkout branch name
            1. After you have made changes to your side branch:
                a. Git status

    b. Git add .
    c. Git status
    d. Git commit -m “comments”
    e. Git push
    f. Create a pull request online and leave unmerged for others to review/test/approve. Don’t merge your own pull request when work on a team.
    g. If after a git push you get a fatal error, git hub will tell you what should go in command, so copy and paste that into command line.
2. You can have multiple commits to move from branch back to main: git checkout main
3. You can merge a branch with main – make sure you’re in branch: git merge main. If you get a merge conflict you can open up the page you were working on in Sublime and there will be the fllowing syntax in the document: head main. Head is where your branch is at, main refers to the main branch. Talk with your team to figure out which way you want to change it, do any changes and then remove the head and main syntax and save. Then:
    a. Git status
    b. Git add .
    c. Git commit -m “comments”
    d. Git status
    e. Git push if get error use git push origin name of branch
    f. Then go into github online and check for comments.

<u>Working on Open Source Projects</u>

1. Find the project you want to work on and fork it over to your github account
2. Once it is in your github account clone it
3. Go to git bash and type in git clone and past the link
4. Go to repository and read instructions.
5. CD into the folder/file
6. Open it in Sublime and follow instructions.
7. Create a new branch and check it out: git checkout -b branch name
8. Git status
9. Git add
10. Git commit -m “comments”
11. Git push origin name of file/folder
12. Go to github forked repo online and click on new pull request on your branch. In comments say what you did and leave it there for others to review and merge
13. Some other useful commands: Git diff checks for any changes, Git push origin name of file this pushes stuff to your branch.

How to Keep Fork Up to Date

easy way to **always make sure your fork has the most up to date version of the original project**. Here is how:
Once you are in your forked project directory in your command prompt....

1. Type git remote -v and press **Enter**. You'll see the current configured remote repository for your fork.

```
git remote -v
origin  https://github.com/YOUR_USERNAME/YOUR_FORK.git (fetch)
origin  https://github.com/YOUR_USERNAME/YOUR_FORK.git (push)
```

1. Type git remote add upstream, and then paste the URL you would copy from the original repository if you were to do a git clone. Press **Enter**. It will look like this:

```
git remote add upstream https://github.com/zero-to-mastery/PROJECT_NAME.git
```

1. To verify the new upstream repository you've specified for your fork, type git remote -v again. You should see the URL for your fork as origin, and the URL for the original repository as upstream.

```
git remote -v
origin    https://github.com/YOUR_USERNAME/YOUR_FORK.git (fetch)
origin    https://github.com/YOUR_USERNAME/YOUR_FORK.git (push)
upstream  https://github.com/ORIGINAL_OWNER/ORIGINAL_REPOSITORY.git (fetch)
upstream  https://github.com/ORIGINAL_OWNER/ORIGINAL_REPOSITORY.git (push)
```

Now, you can keep your fork synced with the upstream repository with a few Git commands.

One simple way is to do the below command from the master of your forked repository:

```
git pull upstream master
```