WORKING WITH GIT / GITHUB IN THE COMMAND LINE

Initialize the git repository in the Command Line ***you only need to do this ONCE for each repository*

`git init` - should respond saying "initialized empty git repo…)

TIP: Use **`git status`** to check if you already have a git repository in a given directory

Committing files to Git

`git add <FILENAME>` - do this to add a specific file
or
`git add .` - do this to add *all* files in that directory

`git commit -m "commit message here"` - you *must* include a commit message, it should be something short and meaningful (command line will respond with committed files)

or, if a file is already being tracked, you can combine the previous two commands into a single command:
`git commit -a –m "commit message here"`

Create a repository on Github ***you only need to do this ONCE for each repository*

Go to github.com and click the **+** button on the top right corner of the window - choose **New Repository**
Give the repository a name
Copy the commands in the second gray box, the one that says *"…or push an existing repository from the command line"* - It will look like this:

```
git remote add origin <URL TO GITHUB REPO>
git push -u origin master
```

Back in the command line, paste the two lines that you copied from Github, and press Enter
The command line will respond with a confirmation

Pushing Github

`git push origin master` - it will take few seconds, then will confirm

Go to your repository on **`github.com`** and confirm that the changes were committed

Helpful commands to use in the command line throughout this process

`ls` - will list all files
`git status` - will give you the status of your repo

If your files on Github are more up to date than what's on your local machine (for example, you create a README.md on Github) you need to ***pull*** these changes to your local machine
`git pull origin master`

WORKING WITH BRANCHES

Let's say we want to develop a new feature, the best way to do that is on a new branch.

Show all branches
`git branch`

Make a new branch
`git branch <branch name>`

Check what branch you're on
`git status`

Switch to a branch
`git checkout <branch name>`

To delete a branch (for example, if we don't like the feature we developed, or it's not working)
`git branch -D <branch name>`

Pushing to a new branch on Github
`git push origin <new branch name>`

Let's say you get your new functionality working, and you want to merge it and push this to Github, here's the best workflow to follow for merging branches.

```
git checkout master
git pull origin master
git merge <branch name>
git push origin master
```

USING GIT / GITHUB FOR COLLABORATION

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

REPO OWNER WORKFLOW

Create a repo on Github

Commit and push files to it like you normally would

You can accept a pull request on github.com

Keep your local repository up to date - after you merged a pull request and absorb changes made by another contributor (this means the code on github is NEWER than the code on your local machine), you need to pull those changes down to your local machine

```
git pull
```

Find out the state of your forked repository with the master one

```
git remote show origin
```

CONTRIBUTOR WORKFLOW

Fork the repo - on github.com, locate the repository that you want to contribute to. On the top right of the page, there is a button to **Fork**. This will copy the repository to your own github account.

Clone it to your local machine - there is a green button on your repository that says 'clone or download' click this and copy the url

In terminal, change directories to wherever you want this repository to live, then use the clone command:

```
git clone <url to your repo>
```

Change directories to go into the directory you just cloned - **`cd <directory name>`**

Contribute changes to the original repository - first, make changes to *your* repo and commit them to Github

```
git add -a -m "commit message"
git push origin master
```

If you want your edits to be accepted by the original project owner, you must submit a pull request - on github.com, go to YOUR repository (the one you forked), and click **New Pull Request**

- **-** base fork should be the repository that you are contributing to (this is not your repo)
- **-** head fork should be YOUR repository that contains whatever changes you made

Keep your fork up to date with the original repository -

```
git pull <url to original repo>
```

Configure the remote repository with your fork - type **`git remote -v`** and press Enter. You'll see the current configured remote repository for your fork - it should look like this:

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

Configure Git to sync your fork with the original repository (the one you forked) - on github.com, navigate to the original directory (the one *not* owned by you), and copy the link in the 'clone or download' button
`git remote add upstream <url to original repository>`

Check your configuration again, type **`git remote -v`** and press Enter. Now you should see:

```
$ 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)
```