github.com to Git and vice versa using Terminal

First, you need to install and enable Git on your local machine. See https://git-scm.com/book/en/v2/Getting-Started-Installing-Git to learn how to get started.

Getting terminal into zsh instead of bash (no idea why, but…)

At the Terminal command prompt, enter:

```
zsh
```

Hit enter or return.

The shell being used in the current Terminal session will change to the Zsh and the command prompt will change to a percent sign (%).

You can return to the bash shell by either quitting Terminal, or at the prompt enter:

```
bash
```

Hit enter or return.

To get a repository from github.com using Terminal:

Go to the repository on github. Click on Copy and then get the url listed there.

In terminal (change to zsh), go to the folder you want and type:

```
git clone <paste url>
```

This will create the files in the folder you're in locally.

Make changes as necessary.

To move files back to github.com from local copies using Terminal:

In terminal (change to zsh), go to the folder you want.

The rest of this document is from the class. Pasted here for ease of use:

initialize an empty Git repository by typing this command:

```
git init
```

You'll get a message to let you know it worked

```
Initialized empty Git repository in <<your/filepath/here>>
```

Now check the status of the repo by typing

```
git status
```

You should get a message that says something like

```
On branch master No commits yet
```

Let's create a new file with the touch command. We'll call it index.html.

```
touch index.html
```

Check to see if your repository status has changed by typing

git status

You will see that your new file displays in red. This means the file is *staged*, but you have yet to add it and commit it to your repository.

```
On branch master
No commits yet
Untracked files:
    (use "git add <file>..." to include in what will be committed)
  index.html
```

This is where things get a little tricky because just because you created a file, it doesn't mean you've placed it in your Git repository yet. To do that, you have to do two things: Add the file to your repository and then commit it. The first part you have to do only once and then committing the file comes in handy later on as you make more and more changes to the file.

To add the file, type:

```
git add index.html
```

Let's check the Git status to see what changes.

```
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
new file:   index.html
```

Perfect! As you can see, the file name has changed to green. This means it has been added to the repository. Now, you must commit it with a message stating what the particular commit changes in our code.

**A note about commits

It may seem strange that after you added a file, you have to carry out an extra step to save it permanently. This is where the magic of version control really shines. Each time you make changes to your code and want to save it, you add the file and execute the commit command. So, instead of just saving the file and overwriting your previous work, this creates a trackable series of changes, made through separate commits, which you can refer to and even revert to should you need to.

Every commit to Git needs to have a message included. The message should state, in the present tense, what exactly the changes you made do. This step is what keeps track of all your changes, and the commit messages make it easier for you to know what each version does. For this particular commit, we created an empty HTML file, so the commit message will be:

```
git commit -m "adds empty index.html file"
```

You should get a message similar to this:

```
[master (root-commit) 60a0120] adds empty index.html file 1 file changed, 0 insertions(+), 0 deletions(-) create mode 100644 index.html
```

And that's it! You have successfully created a repo and made your first commit!

To make further commits, all you have to do is stage your file, and commit it, with the commands:

```
git add <filename> git commit -m "what this commit does"
```