school of ai

Git and GitHub

In this chapter, I want to introduce you to GitHub, to things such as version control, Cloning repositories, merge repositories, fork, pull requests, and whole bunch of other interesting things.

Version Control

Version control in the simplest form. Let's say that I create a new code file and I write a few lines in it. Now I decide to put it under version control using `git`. And let's say that I call this save point as number one. Now this is my `first version` . Later on as I progress. I write maybe a few more lines of code and at this point I decide to make another save point and I call this my `second version`.

So further down the line I accidentally screw up my entire code file and it's irrepairable and I get to the point where I would rather **burn** my entire code file rather than having to try and fix it. You do get into these situations because very often your code is interlinked and each class depends on another and sometimes you can screw up in a way where you know all hope is lost and I simply just want to roll back to the last save point. I can do that using git. I can do that using other tools as well. But the most popular tool and the one that we're going to be talking about is get.

Installing Git

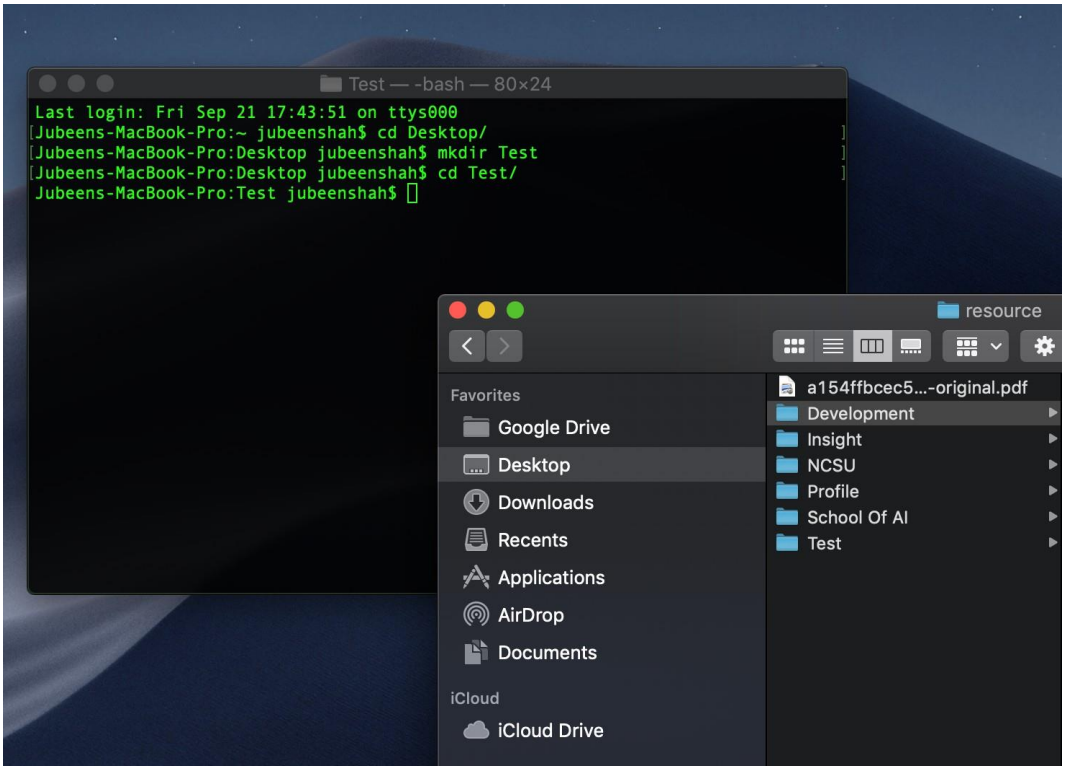
To keep this tutorial much concise, I will not get into the details of installing git. If however, you need help to `git` onto you pc I install would recommend going through this link

Version Control using Git

Assuming, that you have `git` installed, let us move ahead. Again, to keep this chapter concise I will not be talking about how to use git `Windows` on (Sorry, Windows users). Just `MacOS`. However, if you use windows, you can go through this video, which is like a crash course to using `git`. For other users please follow along.

Open your terminal.

```
cd Desktop/
mkdir Test
cd Test/
```



Terminal 1

```
git init
```

This is to initialize the `git` repository in the `Test` directory. Then I would ask you to manuall create a text file in the `Test` folder. Just write some random text in it and save it. Alternatively, you can use the `vim` command for doing the same.

```
vim Test.txt
```

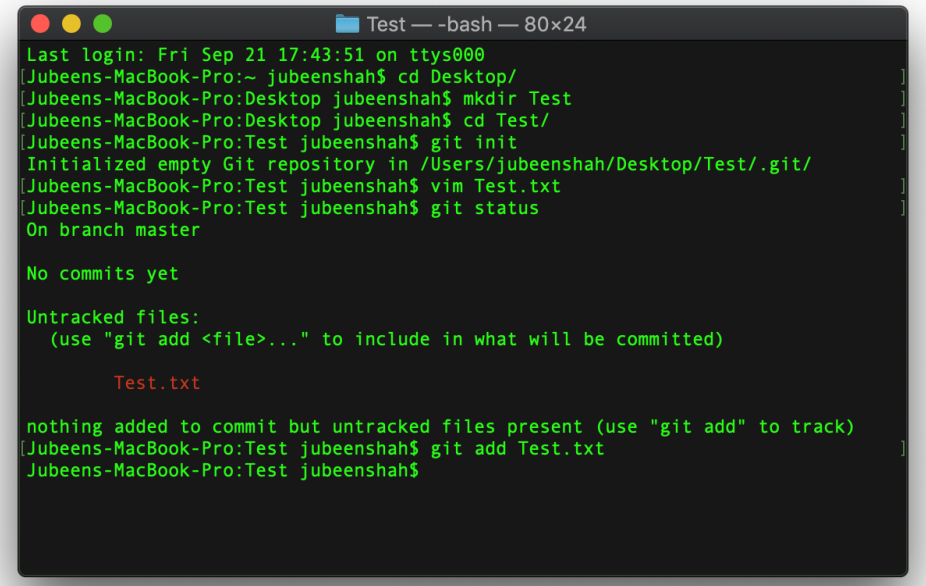
Then type in the text, once done; hit `esc` type in `:wq` and then hit

`return` / `enter`

```
git status
```

This command will tell you that, there `untracked` files in the some repository.

```
git add Test.txt
```



Terminal 2

```
git status
```

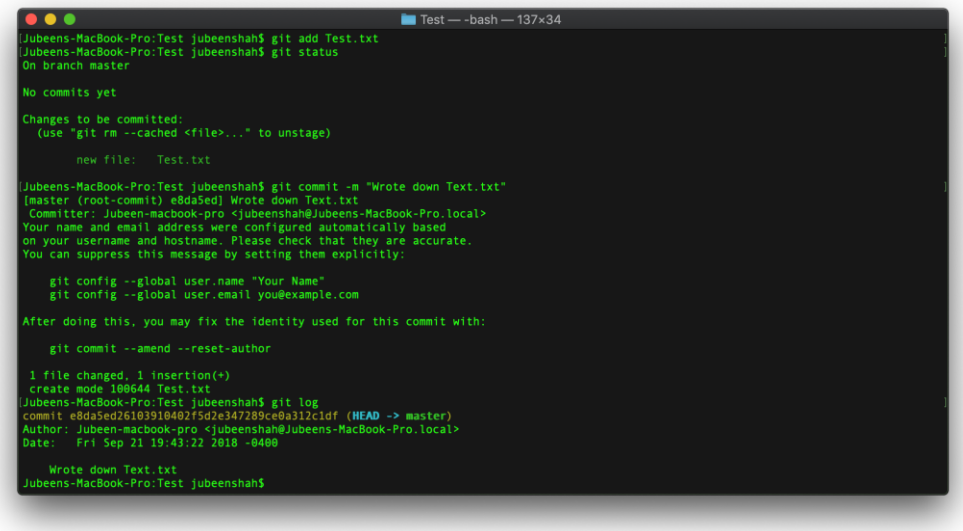
Now it will show you that a new file is added to the branch. Now you commit the changes

```
git commit -m "Wrote down Text.txt"
git log
```

The `git commit` would help you to commit the changes to your repository, just like you do in a "relationship", you're bound to him/her.

Unless you're like my Ex who has "commitment" issues.

You **Should** use the option `-m` to write commit messages. These messages would help you in the future to recognize the changes that you've made to the file. Then you can use the `git log` command to show you the entire log of all the `commits` that you have ever made.

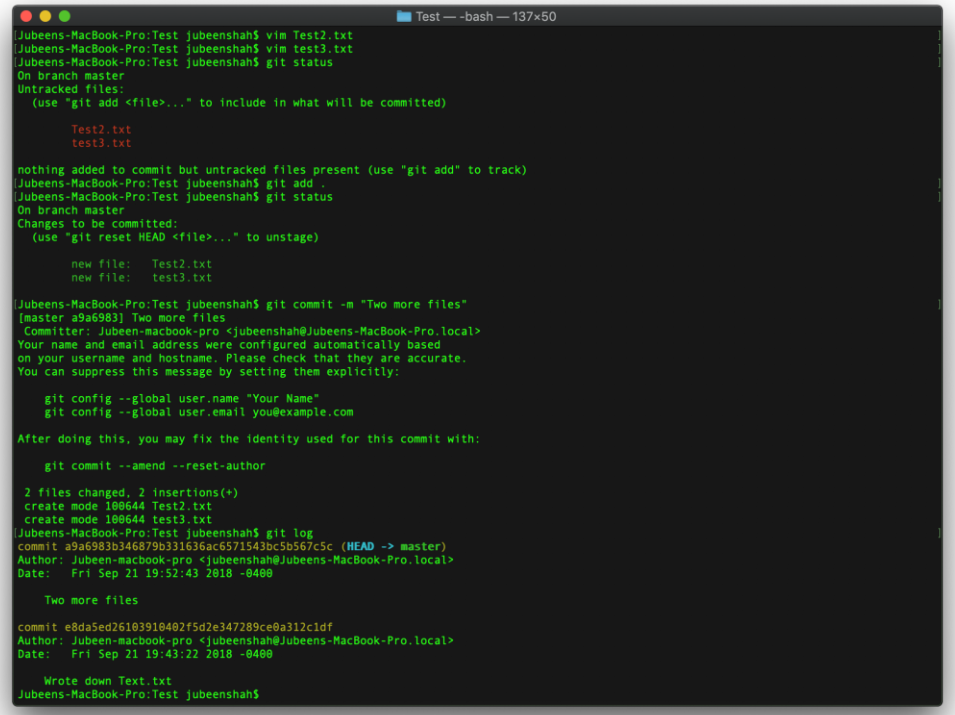


Terminal 3

As you can see there is a `hash` code, also you see who the Author is, and additionally you get the information such as the timestamp. So, unlike a real relationship, you can actually rollback to a previous stage `commitment`
in your using the hash code.

I repeated the same steps again, just using one different command

```
git add .
```



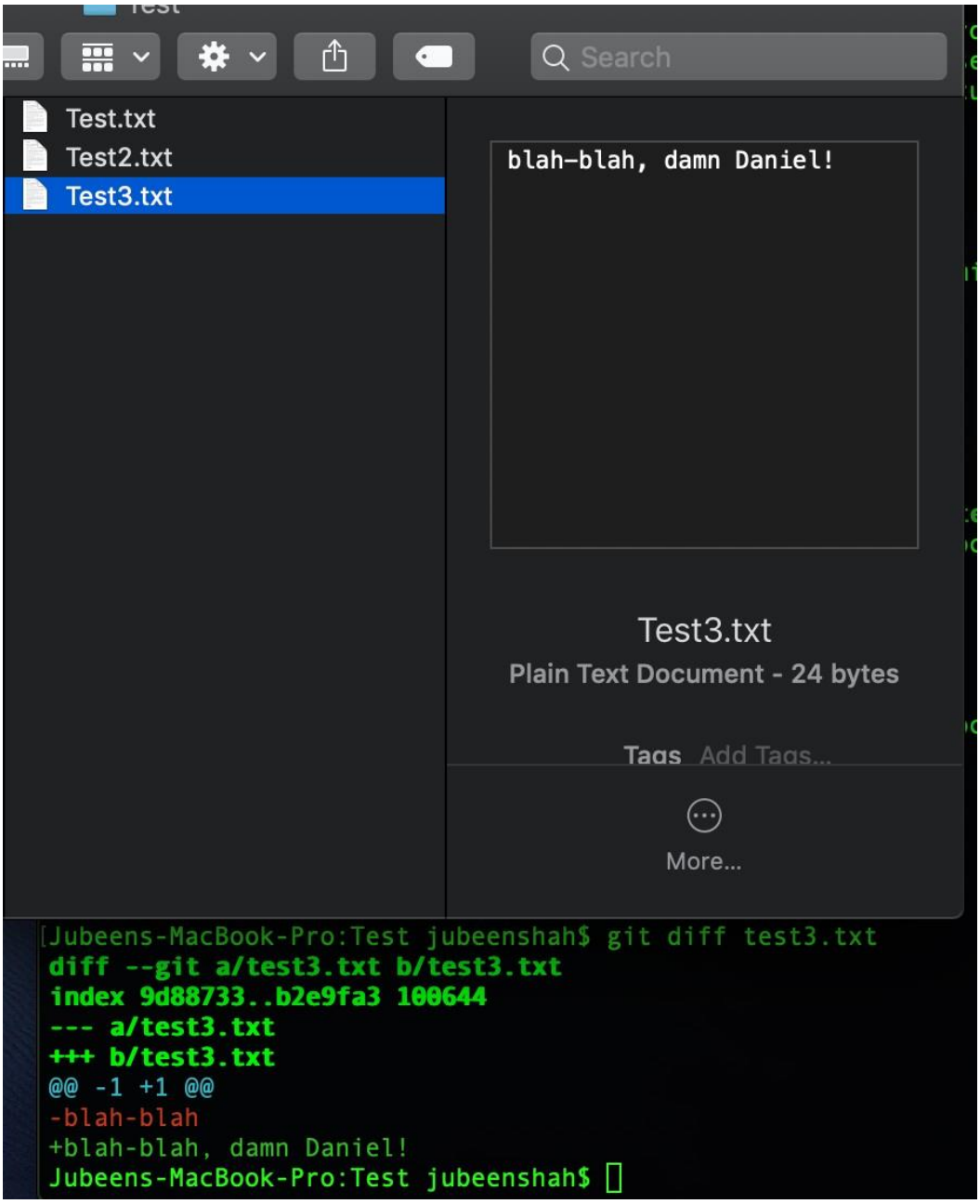
Terminal 4

To summarize :

- So the working directory is the one in which you have created `.*txt` the files.
- Then we push the files to the staging area using `git add .` the command.
- Then we push the it to the `local repository` using the `git commit -m ""` command

Now I have made some changes in the Test3.txt file but it is something that I do not want to commit to the local repo. To find the difference we can use the command :

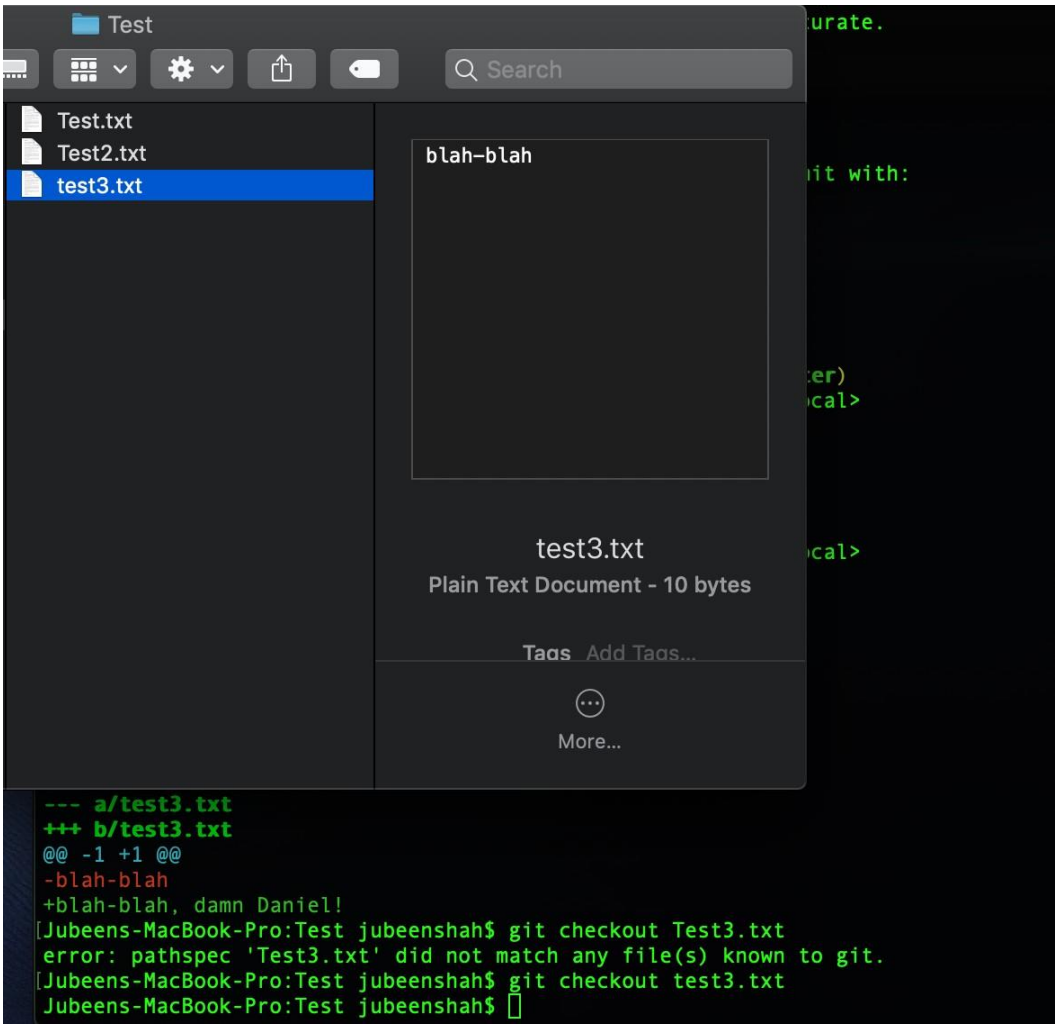
```
git diff Test3.txt
```



Terminal 5

These are some changes if you don't want to commit. You can simply use the command :

```
git checkout Test3.txt
```



Terminal 6

You can see that the roll back happens as soon as you hit enter.

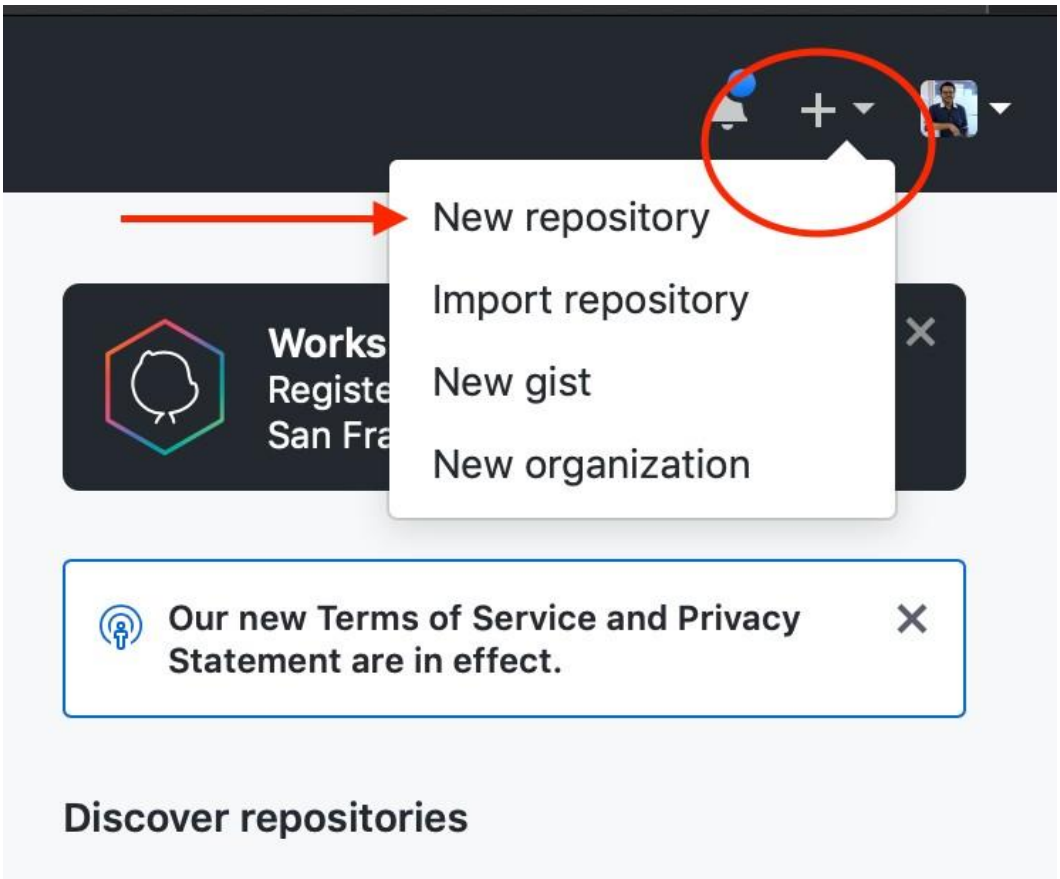
Now don't you wish there was something like this in your relationship.

However, it should be noted that this repository is locally available. A better way to use `GitHub` is to use its `remote repositories` functionality.

GitHub and Remote Repositories

Now, let's talk about using remote repositories using github. For using this awesome site, you would have to go to Github.com and use that pretty form on the right to set up an account on GitHub. All you need is you email address.

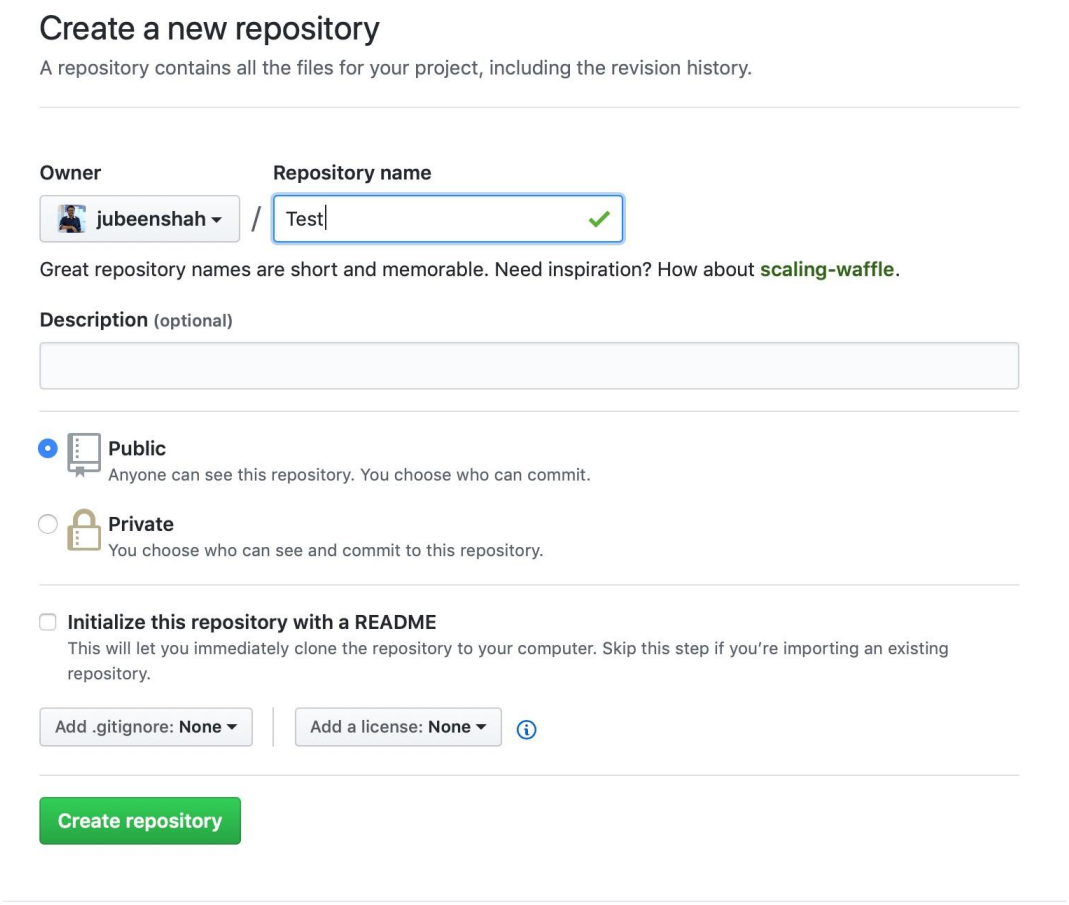
Sign in to you account, and the top right hand side, near your identicon, there should be `+` symbol. Click there, followed by the `New repository` option.



Create a New repository - 1

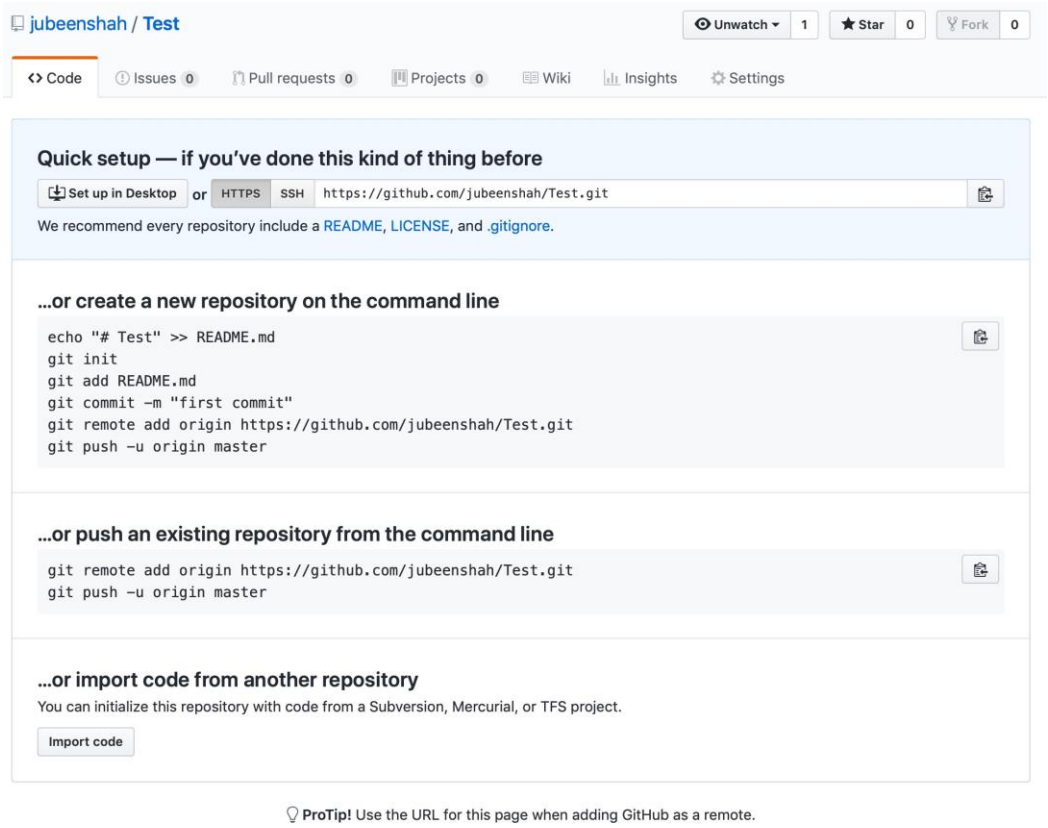
- Add a `repoistory` name
- Add a `description`

- By default the deployment option is `Public`
- Click on `Create my repository`



Create a New repository - 2

Now you will have a `Quick setup` page shown to you. If you are a big GUI fan, please by all means download the App for Mac or Windows, or you can carry on with the Command Line Interface (CLI) we have been using till now.



Quick setup

If you have followed this chapter diligently, you already have a `local repository` setup. So you can using the CLI commands from

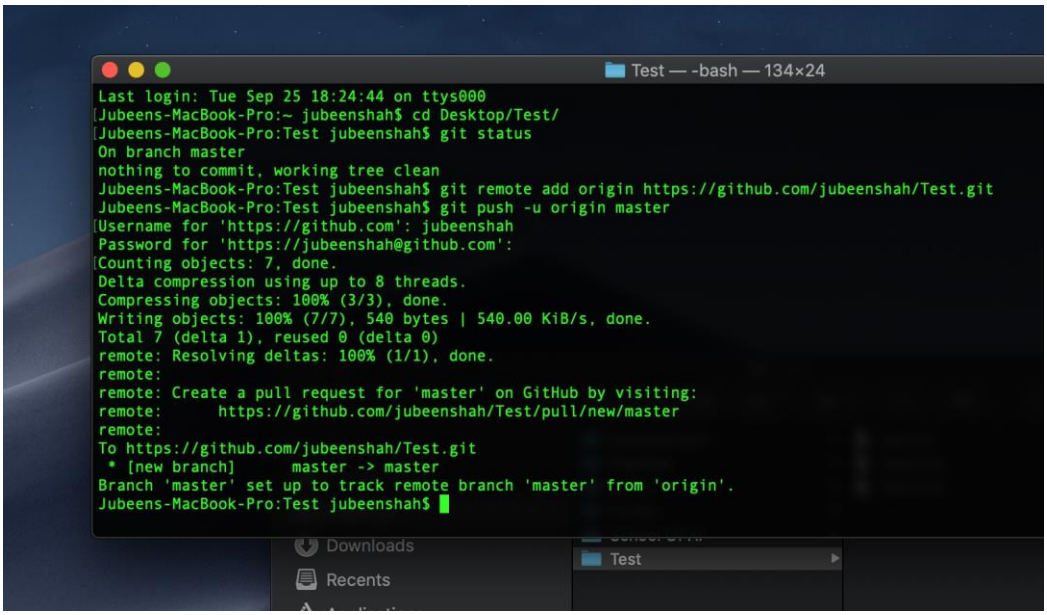
`push an existing repository from the command line`

For me it is as follows

```
git remote add origin https://github.com/jubeenshah/Test.git
git push -u origin master
```

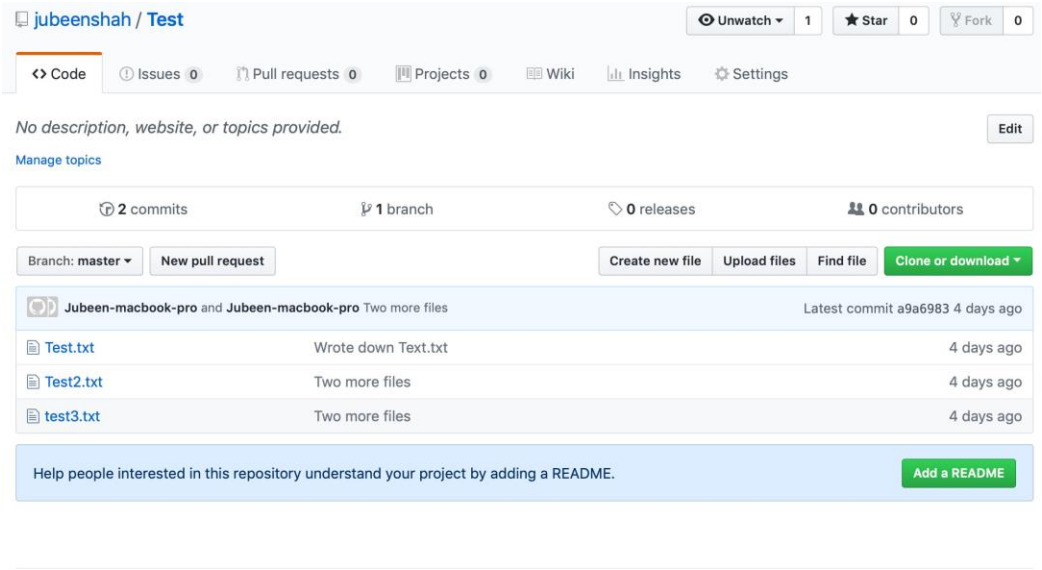
So you can copy and paste both the line into your terminal and hit enter. It will prompt you for the username and password. Enter it and

you should be greeted with something like this



Terminal after pushing to GitHub

That is it. You've successfully pushed a local repository onto GitHub. If you reload the page, on your browser you should see all the files that you added.



Files on GitHub

Gitignore

In this section we'll learn about how to set up the repository, in a way that certain files are ignored.

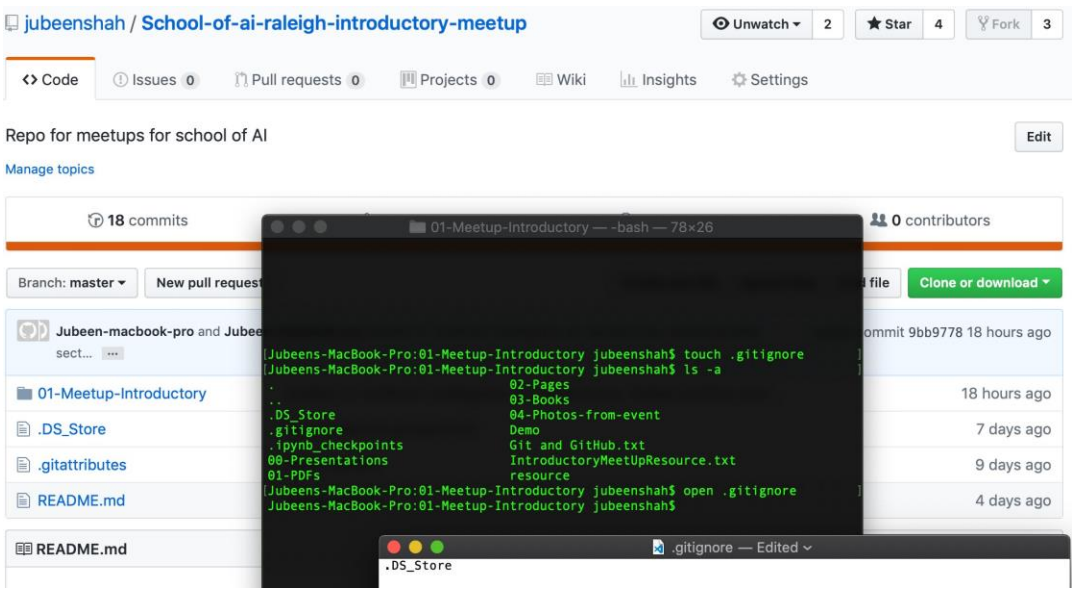
Why do we need to do this? you might ask. Imagine this scenario. You have an Amazon AWS based application, making use of certain API keys. What if you push these API keys, to the the `remote repository` for everyone to see? Just keep imagining the possibilities.

Let's see an example. Currently, the school-of-ai-introductory-meetup repository has a .DS_Store file. To give it to you quickly. That file is unecessary for everyone else except me.

So what I would do is, go to the folder of the repository on my Mac.

```
cd Desktop/Development/School-of-ai-raleigh/01-Meetup-
Introductory
touch .gitignore
open .gitignore
```

Type in the file names, you do not want to be committed to the remote repository.



.gitignore

To have a look at the complete list of pre-wrtitten `.gitignore` that you can make use of, visit this repository by GitHub

Git Clone

This is simple to do. Very simple in-fact. Using the URL of the

repository. go back to the terminal on you mac. Then you can use the command `git clone` + `URL`

```
git clone `URL`
```

That is it.