

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 irreparable 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

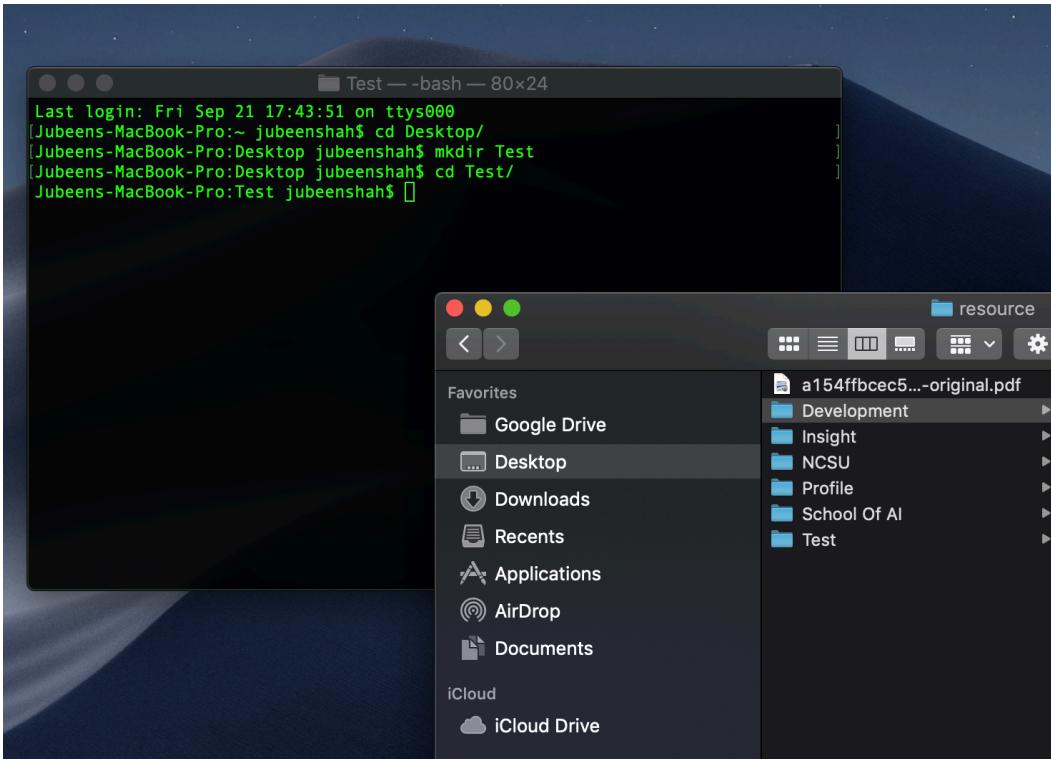
To keep this tutorial much concise, I will not get into the details of installing git. If however, you need help to install `git` onto your pc I would recommend going through [this link](#)

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 on `Windows` (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 manually 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.

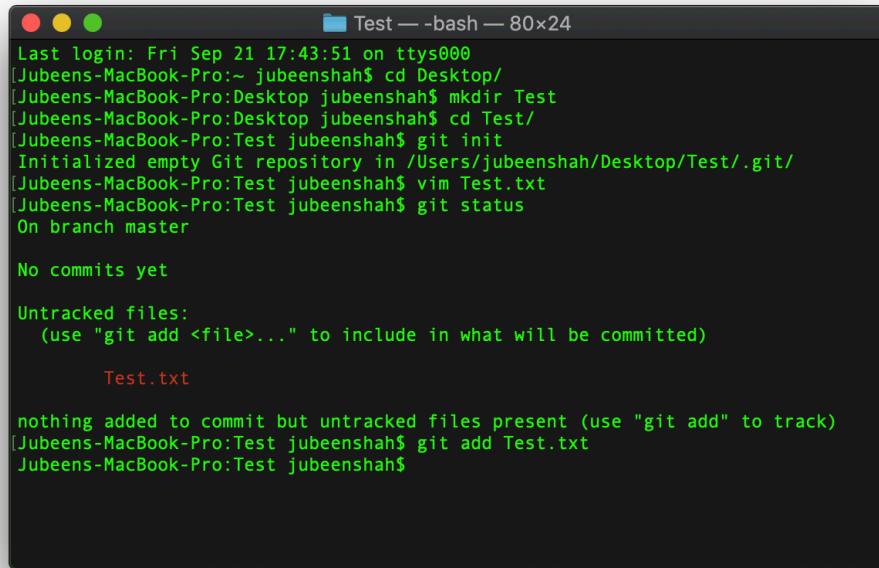
```
vim Test.txt
```

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 some **untracked** files in the repository.

```
git add Test.txt
```



```
Last login: Fri Sep 21 17:43:51 on ttys000
[Jubeens-MacBook-Pro:~ jubeenshah$ cd Desktop/
[Jubeens-MacBook-Pro:Desktop jubeenshah$ mkdir Test
[Jubeens-MacBook-Pro:Desktop jubeenshah$ cd Test/
[Jubeens-MacBook-Pro:Test jubeenshah$ git init
Initialized empty Git repository in /Users/jubeenshah/Desktop/Test/.git/
[Jubeens-MacBook-Pro:Test jubeenshah$ vim Test.txt
[Jubeens-MacBook-Pro:Test jubeenshah$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    Test.txt

nothing added to commit but untracked files present (use "git add" to track)
[Jubeens-MacBook-Pro:Test jubeenshah$ git add Test.txt
[Jubeens-MacBook-Pro:Test jubeenshah$
```

Terminal 2

```
git status
```

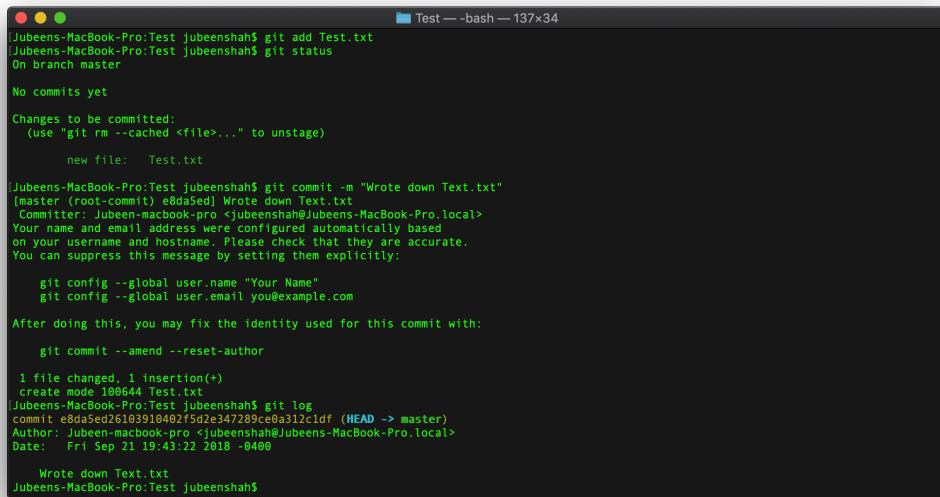
Now it will show you that a new file is added to the branch. Now you commit the changes

```
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.



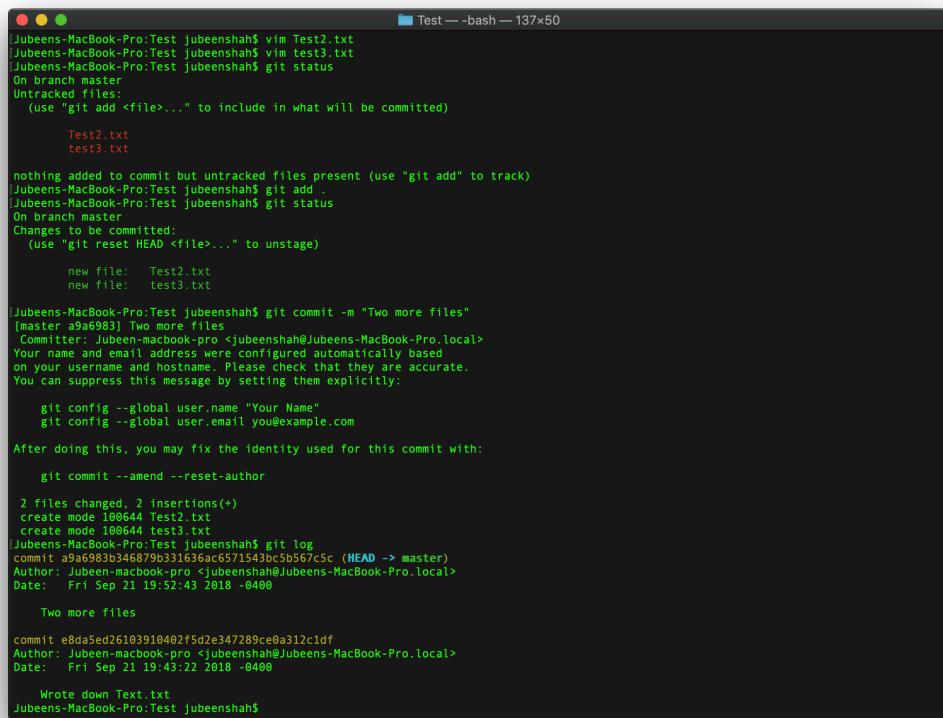
```
Jubeens-MacBook-Pro:Test jubeenshah$ git add Test.txt
[Jubeens-MacBook-Pro:Test jubeenshah$ git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:  Test.txt
[Jubeens-MacBook-Pro:Test jubeenshah$ git commit -m "Wrote down Text.txt"
[master (root-commit) e8da5ed] Wrote down Text.txt
  Committer: Jubeen-macbook-pro <jubeenshah@Jubeens-MacBook-Pro.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:
  git config --global user.name "Your Name"
  git config --global user.email you@example.com
After doing this, you may fix the identity used for this commit with:
  git commit --amend --reset-author
1 file changed, 1 insertion(+)
create mode 100644 Test.txt
[Jubeens-MacBook-Pro:Test jubeenshah$ git log
commit e8da5ed26103910402f5d2e347289ce0a312c1df (HEAD -> master)
Author: Jubeen-macbook-pro <jubeenshah@Jubeens-MacBook-Pro.local>
Date:   Fri Sep 21 19:43:22 2018 -0400
      Wrote down Text.txt
[Jubeens-MacBook-Pro:Test jubeenshah$
```

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 in your `commitment` using the hash code.

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

```
git add .
```



```
Jubeens-MacBook-Pro:Test jubeenshah$ vim Test2.txt
[Jubeens-MacBook-Pro:Test jubeenshah$ vim test3.txt
[Jubeens-MacBook-Pro:Test jubeenshah$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    Test2.txt
    test3.txt

nothing added to commit but untracked files present (use "git add" to track)
[Jubeens-MacBook-Pro:Test jubeenshah$ git add .
[Jubeens-MacBook-Pro:Test jubeenshah$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   Test2.txt
    new file:   test3.txt

[Jubeens-MacBook-Pro:Test jubeenshah$ git commit -m "Two more files"
[master a9a6983] Two more files
Committer: Jubeen-macbook-pro <jubeenshah@Jubeens-MacBook-Pro.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:
  git config --global user.name "Your Name"
  git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:
  git commit --amend --reset-author

  2 files changed, 2 insertions(+)
  create mode 100644 Test2.txt
  create mode 100644 test3.txt
[Jubeens-MacBook-Pro:Test jubeenshah$ git log
commit a9a6983b346879b331636ac65715430c5b567c5c (HEAD -> master)
Author: Jubeen-macbook-pro <jubeenshah@Jubeens-MacBook-Pro.local>
Date:   Fri Sep 21 19:52:43 2018 -0400

  Two more files

commit e8da5ed26103910402f5d2e347289ce0a312c1df
Author: Jubeen-macbook-pro <jubeenshah@Jubeens-MacBook-Pro.local>
Date:   Fri Sep 21 19:43:22 2018 -0400

  Wrote down Text.txt
[Jubeens-MacBook-Pro:Test jubeenshah$
```

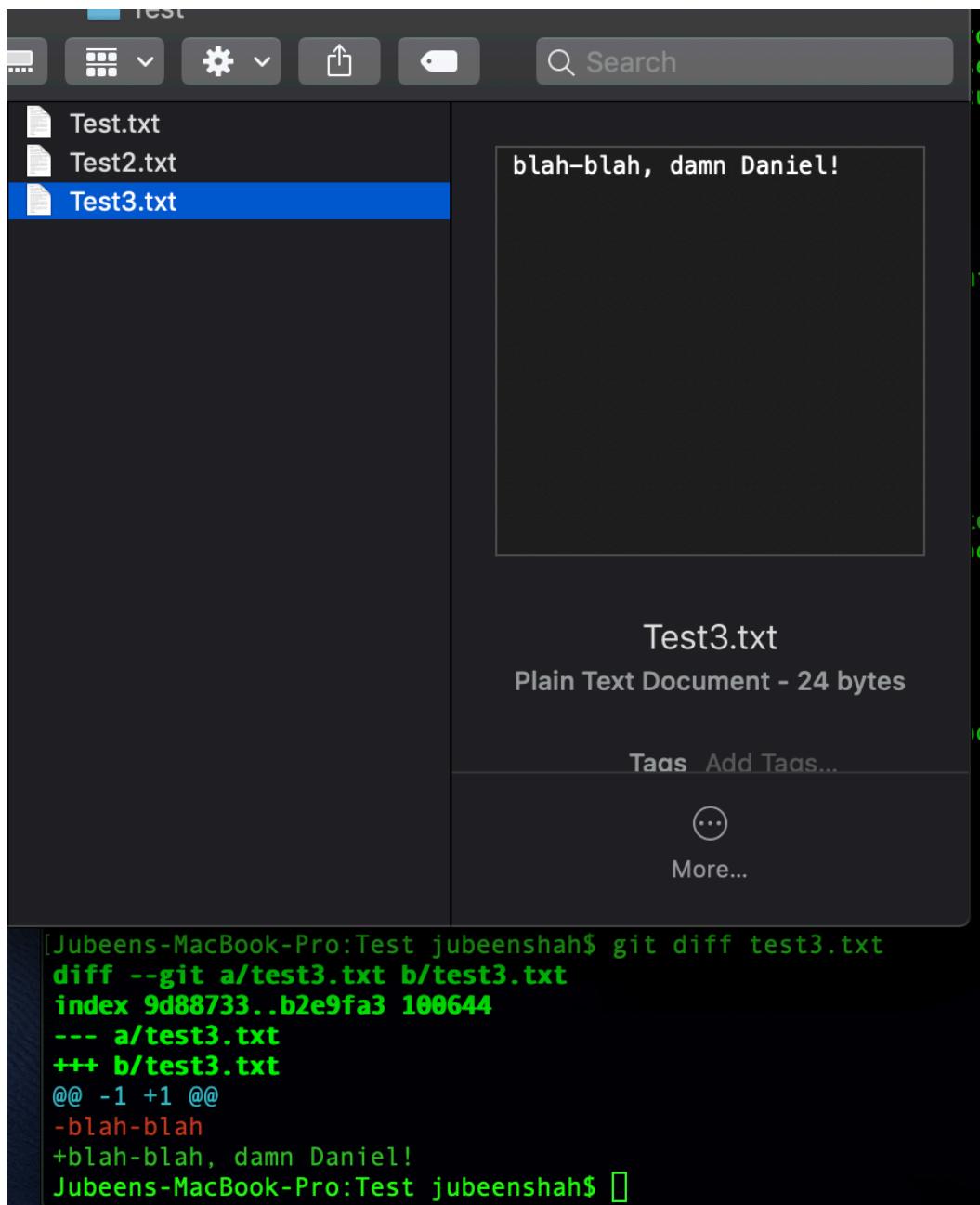
Terminal 4

To summarize :

- So the working directory is the one in which you have created the `.*txt` files.
- Then we push the files to the staging area using the `git add .` command.
- Then we push 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 :

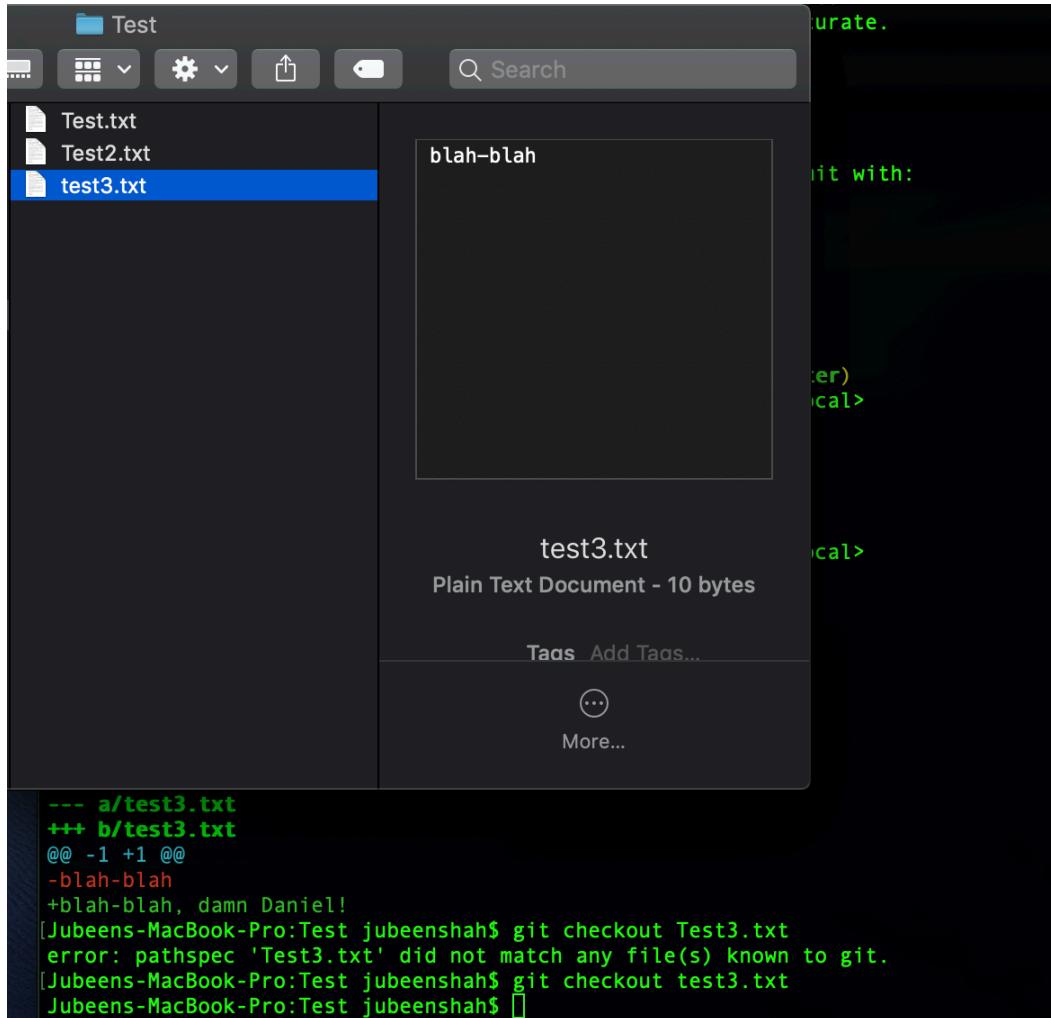
```
git diff Test3.txt
```



Terminal 5

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

```
git checkout Test3.txt
```



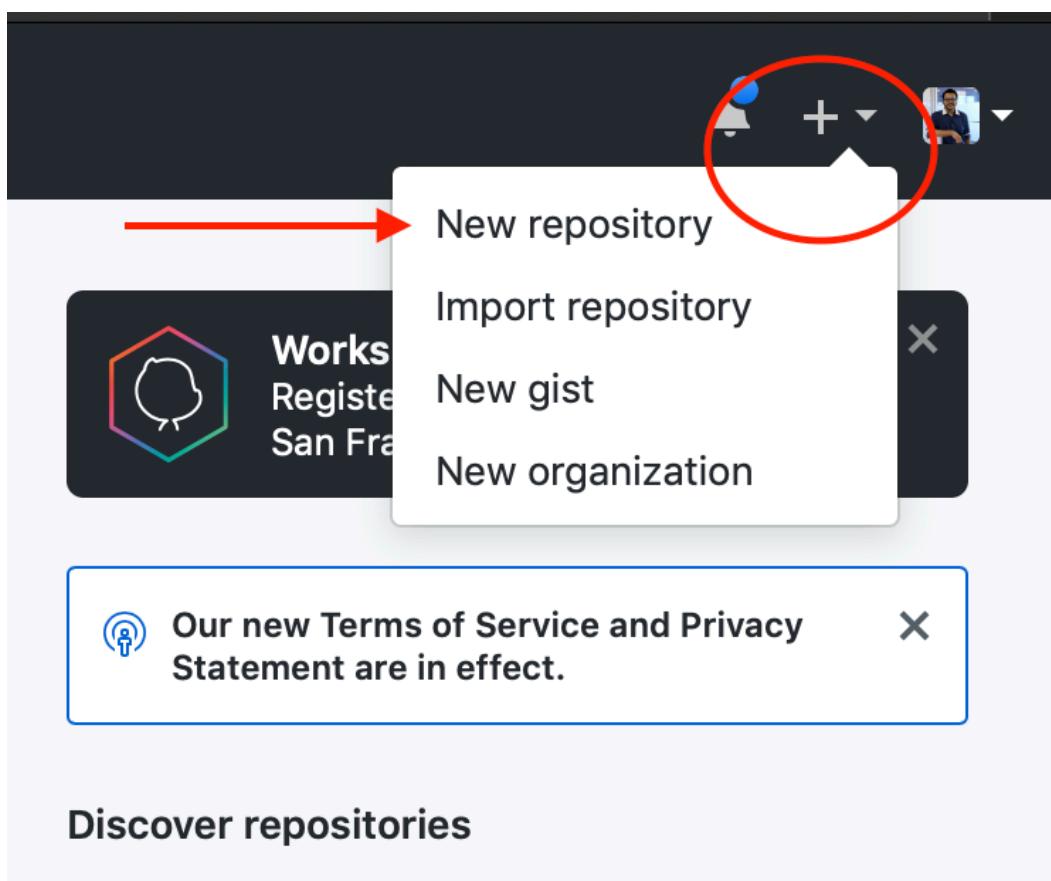
Terminal 6

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](https://github.com) and use that pretty form on the right to set up an account on GitHub. All you need is your email address.

Sign in to your account, and the top right hand side, near your identicon, there should be a + symbol. Click there, followed by the [New repository](#) option.



Create a New repository - 1

- Add a [repository](#) name
- Add a [description](#)

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

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner  jubeenshah 	Repository name <input style="border: 1px solid #ccc; padding: 2px; width: 150px; height: 20px; margin-bottom: 5px;" type="text" value="Test"/> 
--	---

Great repository names are short and memorable. Need inspiration? How about `scaling-waffle`.

Description (optional)

 **Public**
 Anyone can see this repository. You choose who can commit.

 **Private**
 You choose who can see and commit to this repository.

Initialize this repository with a README
 This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

[Add .gitignore: None](#) |
 [Add a license: None](#) |
 

[Create 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.

The screenshot shows a GitHub repository page for 'jubeenshah / Test'. At the top, there are navigation links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. Below these are buttons for Unwatch (1), Star (0), Fork (0), and a copy icon. A section titled 'Quick setup — if you've done this kind of thing before' provides instructions for setting up a local repository. It includes a 'Set up in Desktop' button, an 'HTTPS' button (selected), and an 'SSH' button, along with the URL 'https://github.com/jubeenshah/Test.git'. It also recommends including a README, LICENSE, and .gitignore. Another section, '...or create a new repository on the command line', contains a code block with the following commands:

```
echo "# Test" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/jubeenshah/Test.git
git push -u origin master
```

A third section, '...or push an existing repository from the command line', shows the command:

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

A fourth section, '...or import code from another repository', has a 'Import code' button. At the bottom of the page is a 'ProTip!' note: 'Use the URL for this page when adding GitHub as a remote.'

Quick setup

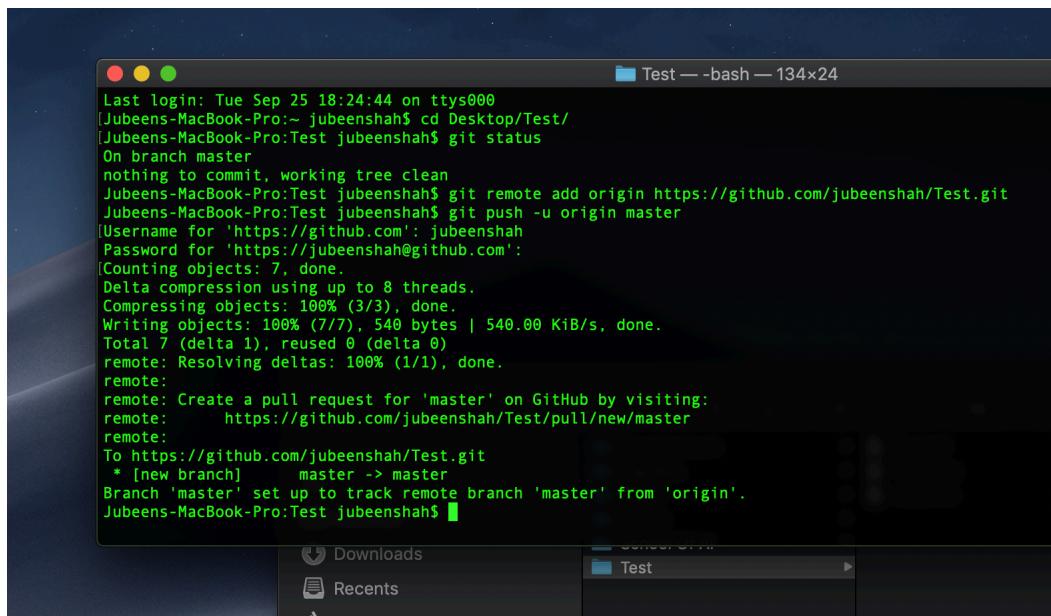
If you have followed this chapter diligently, you already have a `local repository` setup. So you can use the CLI commands from `push an existing repository from the command line`

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 lines 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



```
Last login: Tue Sep 25 18:24:44 on ttys000
[Jubeens-MacBook-Pro:~ jubeenshah$ cd Desktop/Test/
[Jubeens-MacBook-Pro:Test jubeenshah$ git status
On branch master
nothing to commit, working tree clean
[Jubeens-MacBook-Pro:Test jubeenshah$ git remote add origin https://github.com/jubeenshah/Test.git
[Jubeens-MacBook-Pro:Test jubeenshah$ git push -u origin master
[Username for 'https://github.com': jubeenshah
>Password for 'https://jubeenshah@github.com':
[Counting objects: 7, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (7/7), 540 bytes | 540.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:     https://github.com/jubeenshah/Test/pull/new/master
remote:
To https://github.com/jubeenshah/Test.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
Jubeens-MacBook-Pro:Test jubeenshah$ ]
```

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.

No description, website, or topics provided. [Edit](#)

Manage topics

2 commits 1 branch 0 releases 0 contributors

Branch: master [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download](#)

Jubeen-macbook-pro and Jubeen-macbook-pro Two more files Latest commit a9a6983 4 days ago

Test.txt Wrote down Text.txt 4 days ago

Test2.txt Two more files 4 days ago

test3.txt Two more files 4 days ago

Help people interested in this repository understand your project by adding a README. [Add a README](#)

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.

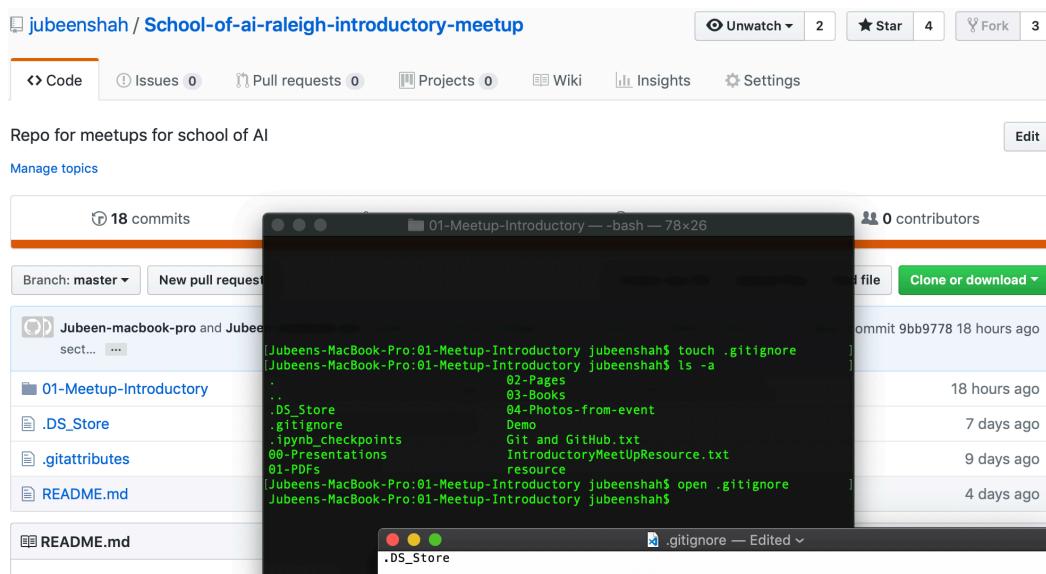
Why do we need to do this? you might ask. Imagine this scenario. You have an Amazon AWS based application, making use of certain API keys. What if you push these API keys, to the the `remote repository` for everyone to see? Just keep imagining the possibilities.

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 unnecessary 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-written .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.