A step-by-step guide to Git

Don't be nervous. This beginner's guide will quickly and easily get you started using Git.

By Kedar Vijay Kulkarni

January 25, 2018 | 10 Comments | 6 min read

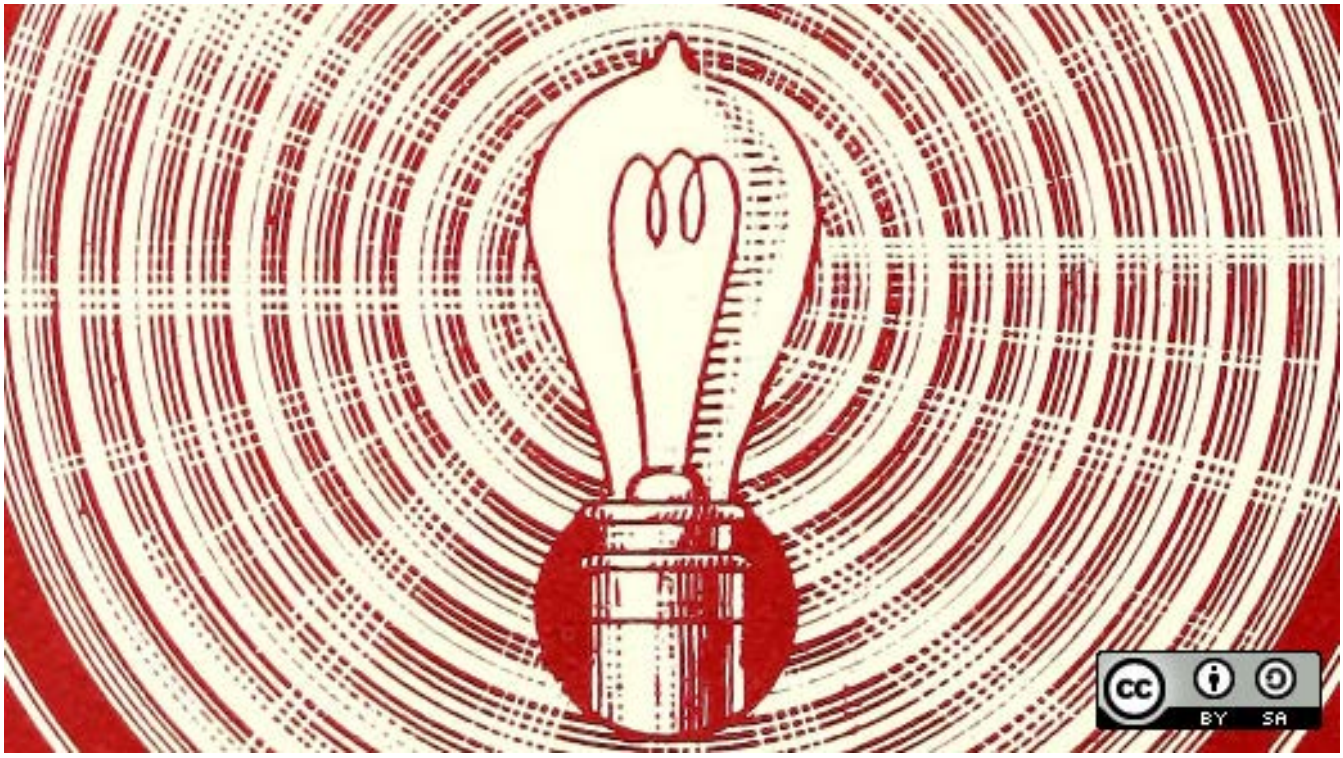


Image by:

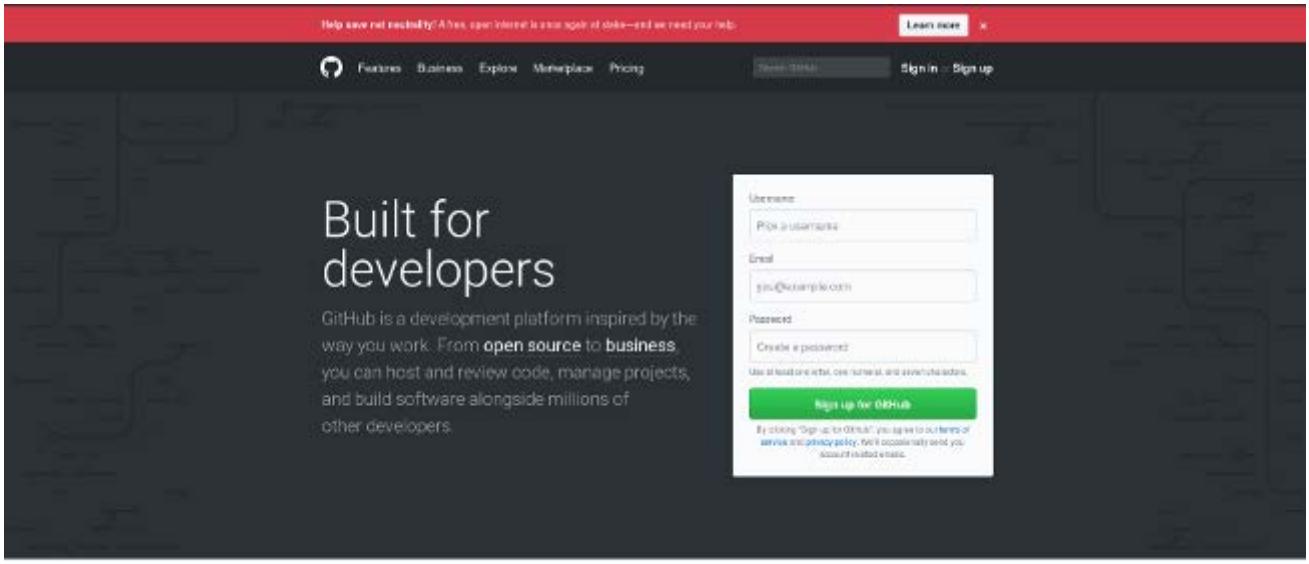
Internet Archive Book Images. Modified by Opensource.com. CC BY-SA 4.0

If you've never used Git, you may be nervous about it. There's nothing to worry about—just follow along with this step-by-step getting-started guide, and you will soon have a new Git repository hosted on GitHub.

them over time. It also allows you to collaborate with your peers on a program, code, or file. GitHub and similar services (including GitLab and BitBucket) are websites that host a Git server program to hold your code.

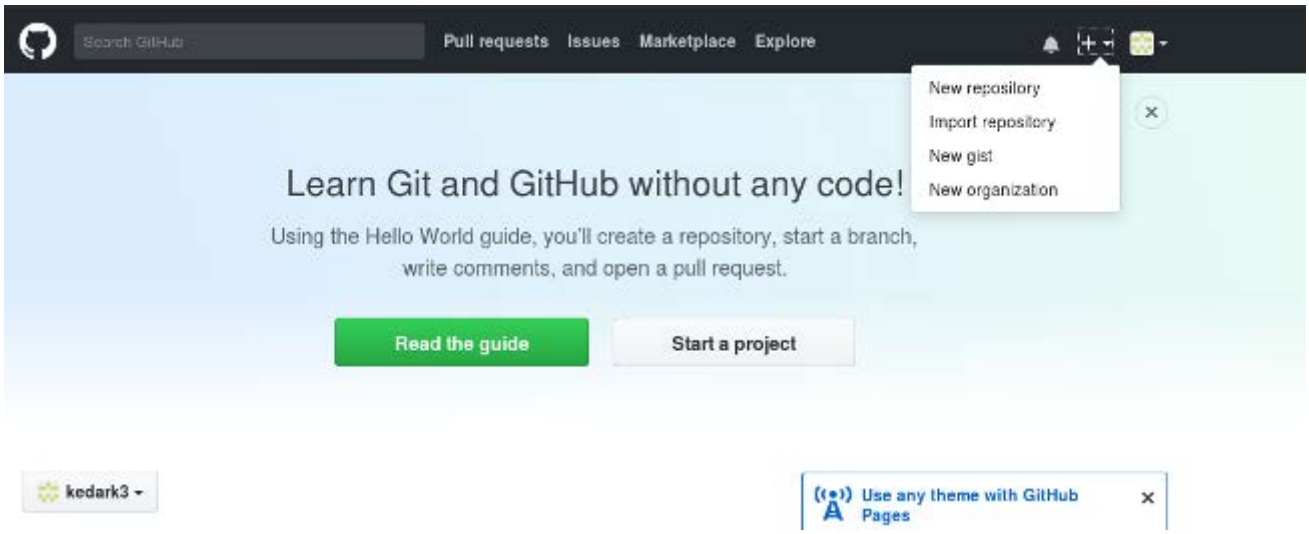
Step 1: Create a GitHub account

The easiest way to get started is to create an account on GitHub.com (it's free).



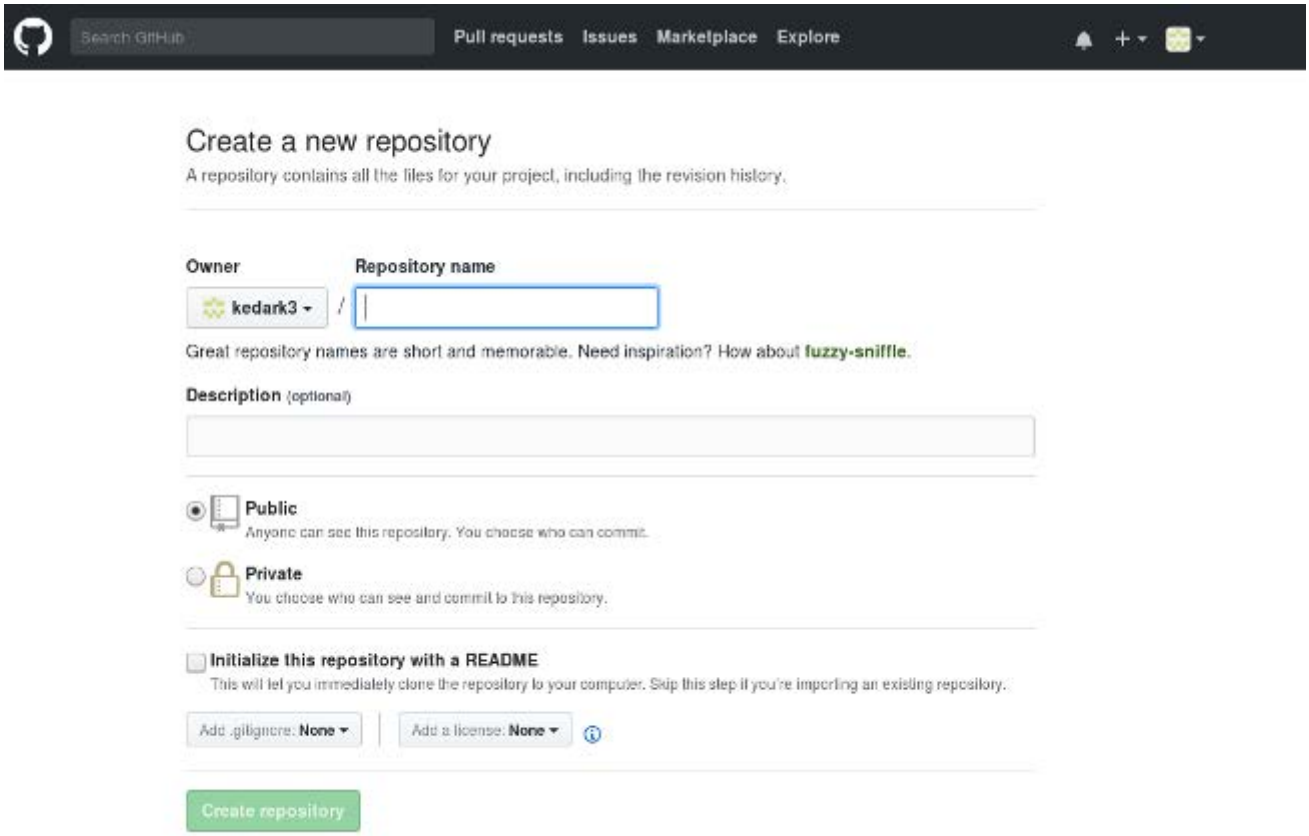
opensource.com

Pick a username (e.g., octocat123), enter your email address and a password, and click **Sign up for GitHub**. Once you are in, it will look something like this:



opensourc

we're creating a Git repository to store code. To create a new repository, select **New Repository** from the + sign dropdown menu (you can see I've selected it in the upper-right corner in the image above).



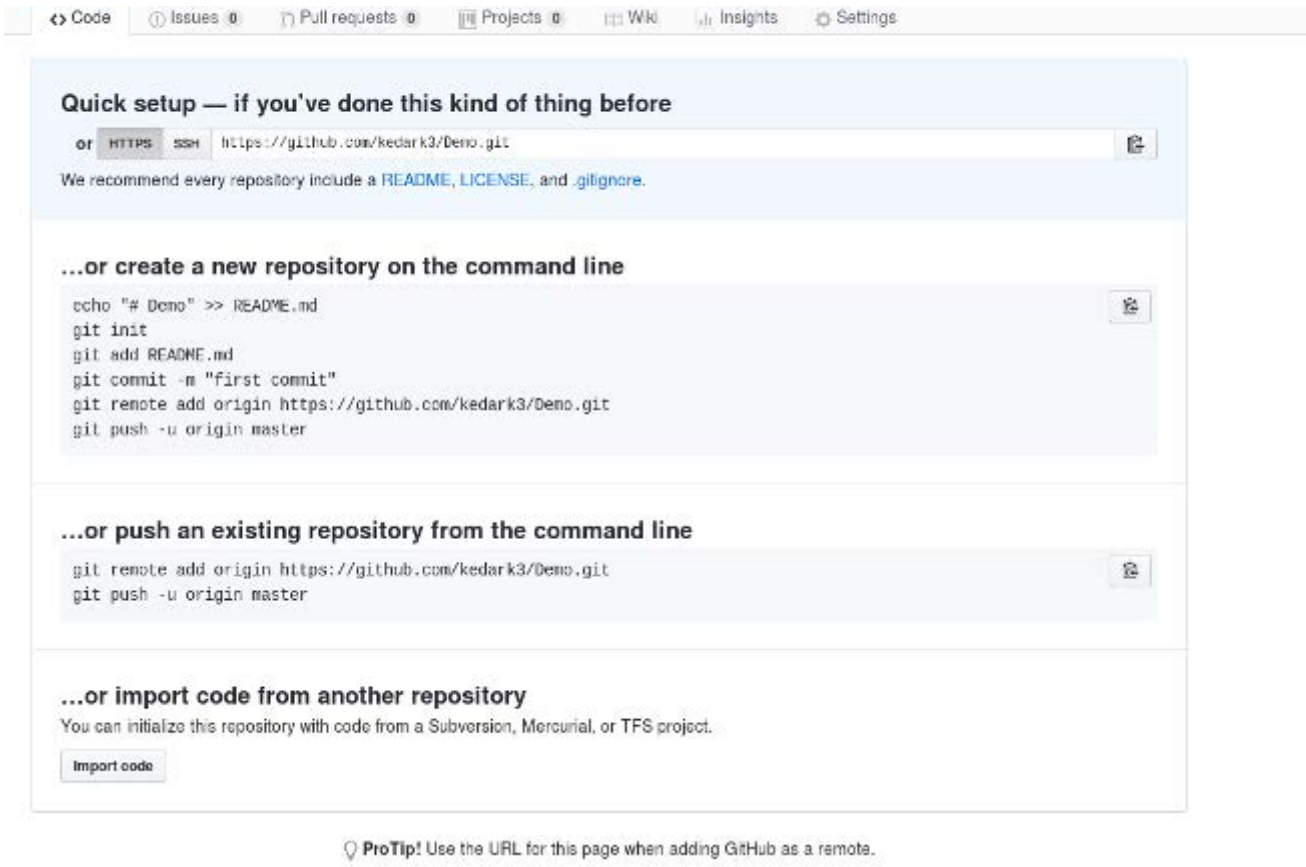
opensource.com

Enter a name for your repository (e.g, "Demo") and click **Create Repository**. Don't worry about changing any other options on this page.

Congratulations! You have set up your first repo on GitHub.com.

Step 3: Create a file

Once your repo is created, it will look like this:



opensource.com

Don't panic, it's simpler than it looks. Stay with me. Look at the section that starts "...or create a new repository on the command line," and ignore the rest for now.

Open the *Terminal* program on your computer.

opensource.com

Type `git` and hit **Enter**. If it says command `bash: git: command not found`, then install Git with the command for your Linux operating system or distribution. Check the installation by typing `git` and hitting **Enter**; if it's installed, you should see a bunch of information about how you can use the command.

In the terminal, type:

```
mkdir Demo
```

This command will create a directory (or folder) named *Demo*.

Change your terminal to the *Demo* directory with the command:

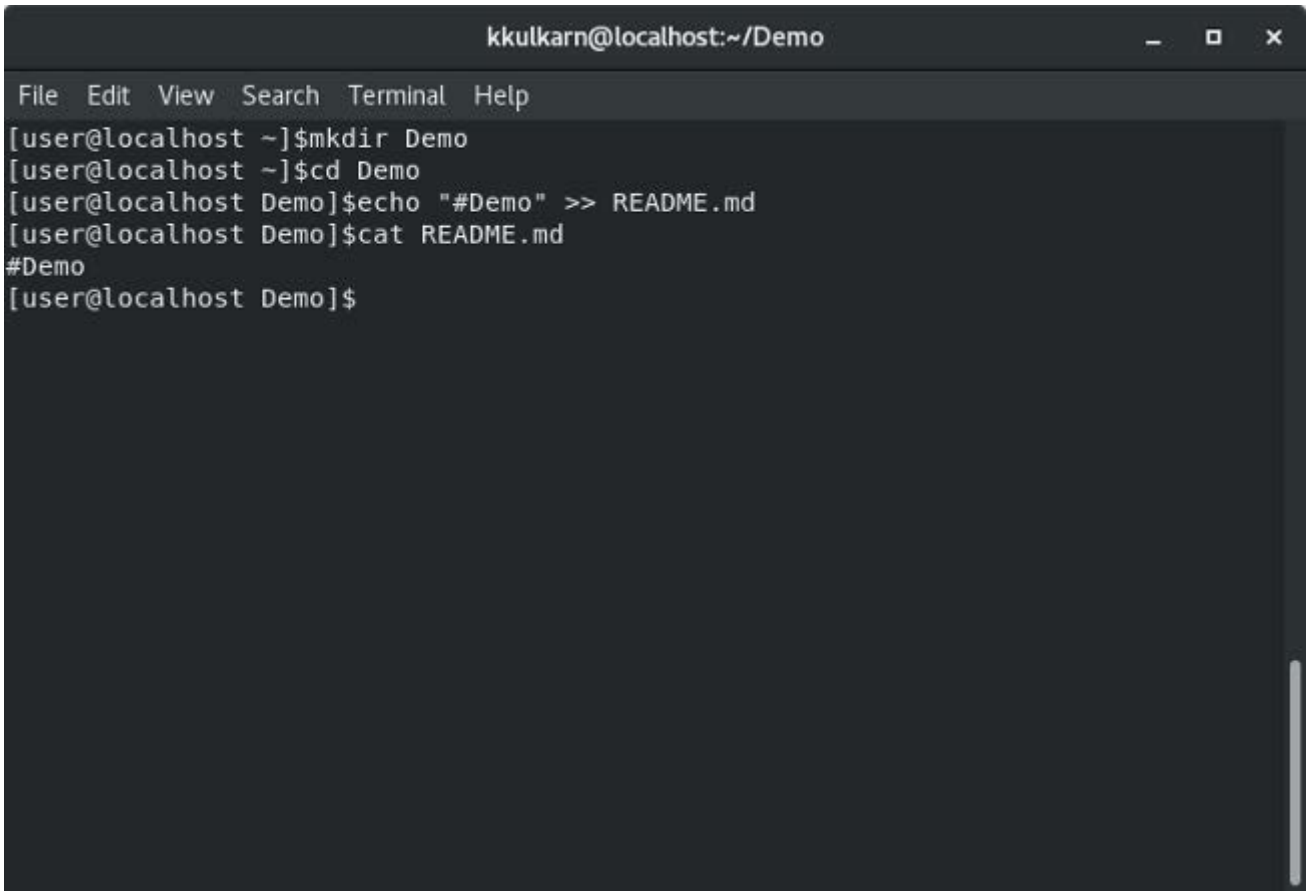
```
cd Demo
```

Then enter:

```
echo "#Demo" >> README.md
```

```
cat README.md
```

This will show you what is inside the `README.md` file, if the file was created correctly. Your terminal will look like this:



opensource.com

To tell your computer that *Demo* is a directory managed by the Git program, enter:

```
git init
```

Then, to tell the Git program you care about this file and want to track any changes from this point forward, enter:

```
git add README.md
```

Step 4: Make a commit

[Red Hat Developers Blog]

[Programming cheat sheets]

[Try for free: Red Hat Learning Subscription]

[eBook: An introduction to programming with Bash]

[Bash Shell Scripting Cheat Sheet]

[eBook: Modernizing Enterprise Java]

So far you've created a file and told Git about it, and now it's time to create a *commit*. Commit can be thought of as a milestone. Every time you accomplish some work, you can write a Git commit to store that version of your file, so you can go back later and see what it looked like at that point in time. Whenever you make a change to your file, you create a new version of that file, different from the previous one.

To make a commit, enter:

```
git commit -m "first commit"
```

That's it! You just created a Git commit and included a message that says *first commit*. You must always write a message in commit; it not only helps you identify a commit, but it also enables you to understand what you did with the file at that point. So tomorrow, if you add a new piece of code in your file, you can write a commit message that says, *Added new code*, and when you come back in a month to look at your commit history or Git log (the list of commits), you will know what you changed in the files.

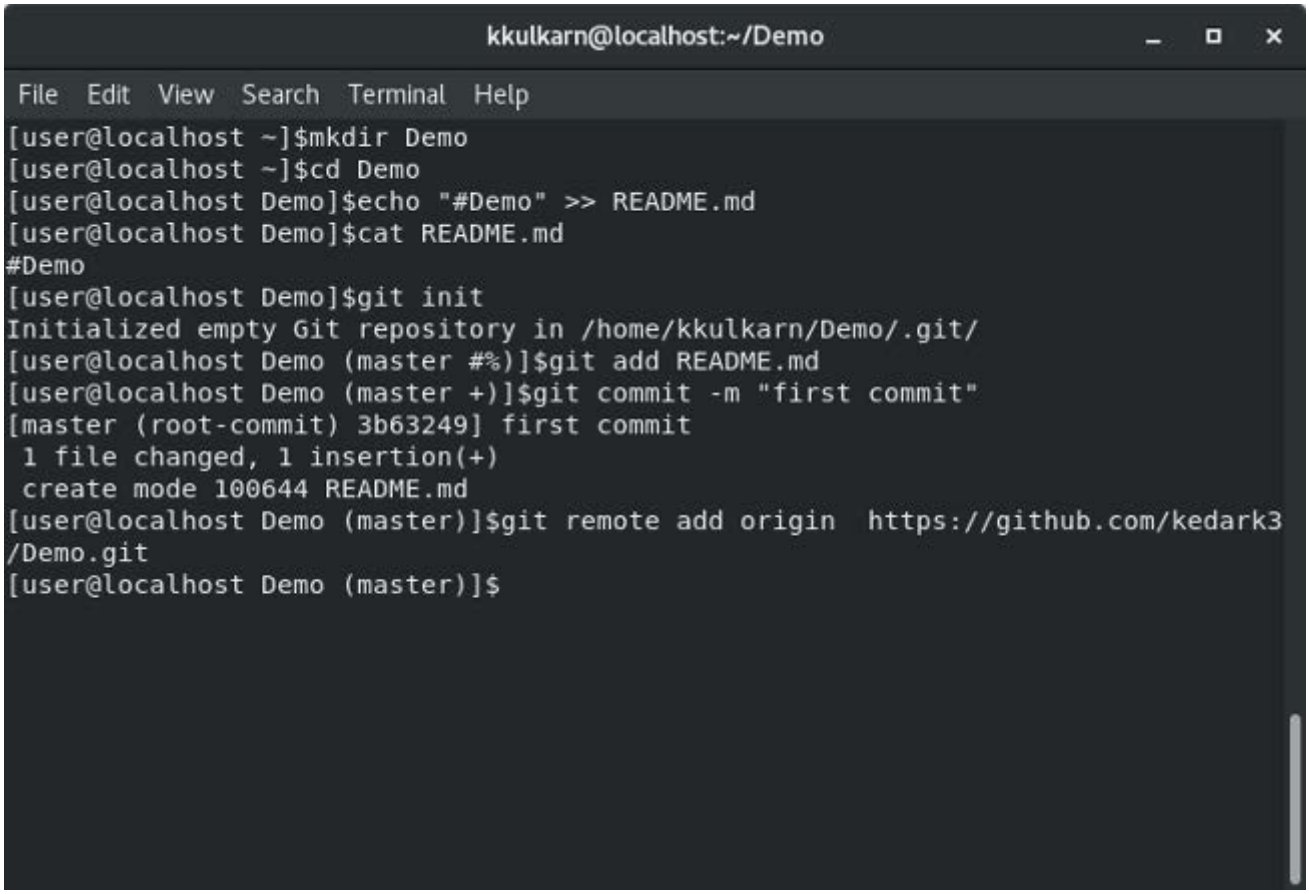
Step 5: Connect your GitHub repo with your computer

Now, it's time to connect your computer to GitHub with the command:

```
git remote add origin https://github.com/<your_username>/Demo.git
```

repository on GitHub.com by typing `origin` instead of the full URL and Git will know where to send your code. Why `origin`? Well, you can name it anything else if you'd like.

Now we have connected our local copy of the *Demo* repository to its remote counterpart on GitHub.com. Your terminal looks like this:



opensource.com

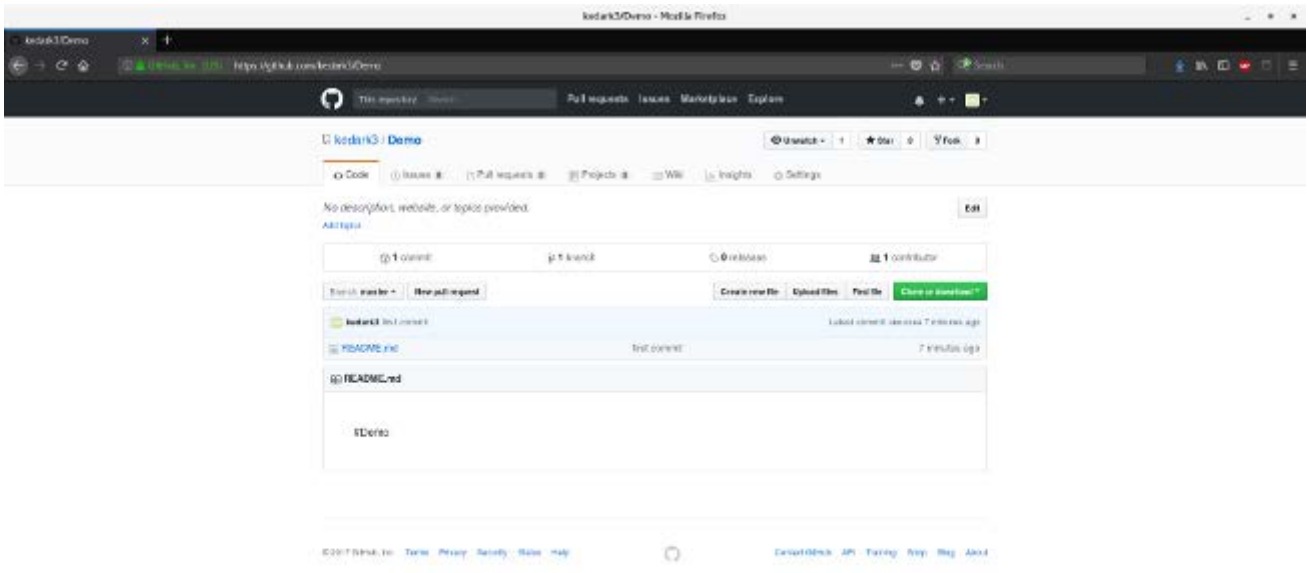
Now that we have added the remote, we can push our code (i.e., upload our `README.md` file) to GitHub.com.

Once you are done, your terminal will look like this:

```
#Demo
[user@localhost Demo]$git init
Initialized empty Git repository in /home/kkulkarn/Demo/.git/
[user@localhost Demo (master #%)]$git add README.md
[user@localhost Demo (master +)]$git commit -m "first commit"
[master (root-commit) 3b63249] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
[user@localhost Demo (master)]$git remote add origin  https://github.com/kedark3
/Demo.git
[user@localhost Demo (master)]$git push -u origin master
Username for 'https://github.com': kedark3
Password for 'https://kedark3@github.com':
Counting objects: 3, done.
Writing objects: 100% (3/3), 608 bytes | 608.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/kedark3/Demo.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
[user@localhost Demo (master)]$
```

opensource.com

And if you go to `https://github.com/<your_username>/Demo` you will see something like this:



opensource.com

That's it! You have created your first GitHub repo, connected it to your computer, and pushed (or uploaded) a file from your computer to your repository called *Demo* on GitHub.com. Next time, I will write about (