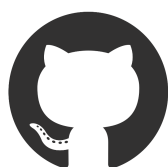


# Introduction to Git & GitHub

*ECON499R/RW: Undergraduate Research in Economics*

Belicia Rodriguez

09.10.2020



## Contents

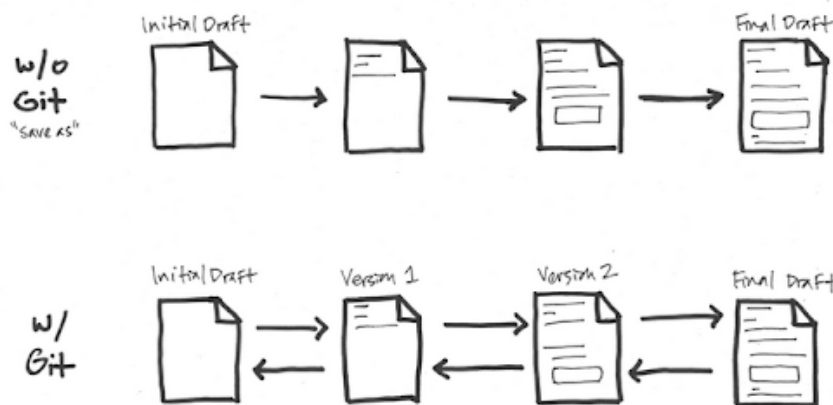
<b>1</b>	<b>Introduction to Git and GitHub</b>	<b>2</b>
1.1	Version Control and Git . . . . .	2
1.2	GitHub & Git Commands (Add, Push, Pull) . . . . .	3
1.3	Getting Started . . . . .	4
<b>2</b>	<b>Installing Git</b>	<b>5</b>
<b>3</b>	<b>Setting Up GitHub Account</b>	<b>6</b>
<b>4</b>	<b>Creating a Repository</b>	<b>9</b>
<b>5</b>	<b>Using Git Commands to Update Repository</b>	<b>14</b>
5.1	git status, git add, git commit, git push . . . . .	14
5.2	git pull . . . . .	17
<b>6</b>	<b>Concluding Message</b>	<b>19</b>
<b>7</b>	<b>Extra Resources</b>	<b>20</b>
7.1	GitHub Desktop . . . . .	20
7.2	Git Cheat Sheet . . . . .	20
7.3	GitHub Student Developer Pack . . . . .	20
7.4	Collaboration with Git . . . . .	20
7.5	Create A Website Using GitHub . . . . .	21

# 1 Introduction to Git and GitHub

## 1.1 Version Control and Git

*What is version control?*

- Version control is a system that records changes you make to a file or set of files over time; this way, if you want to use a previous version of a file, you can recall that specific version.
- Normally, once you save a version of a document, you cannot revert back to its previous version. You can only undo minor changes before saving the document, but once you save and exit the document, any previous versions are gone.
- With version control, you can have access to multiple versions of a document, and you can have a record of all the changes that has been done to a document. Your file's version are stored in a version database, also called a repository. If you don't like a current version of a document and want to revert back to an older version, you have that option.

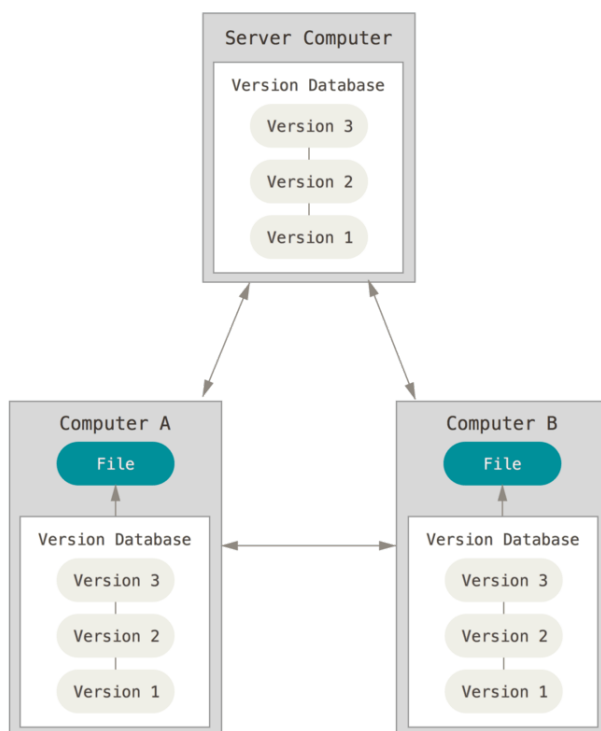


*What is git?*

- Git is a type of version control system called *distributed version control system* <sup>1</sup>
- Pretend that Computer A is your “local computer”, where all of your file's versions are stored in a version database, also called a repository. Git can also store a copy of your repository in a server computer (i.e GitHub). This server will store a copy of your local repository; that way, if anything happens to your local repository, you can backup your local repository using your server repository.
- If you want to have a copy of your server repository on another computer (i.e work computer), then you can have the server send a copy of its repository to your work computer. Now, there are three copies of your repository located in three different locations.

---

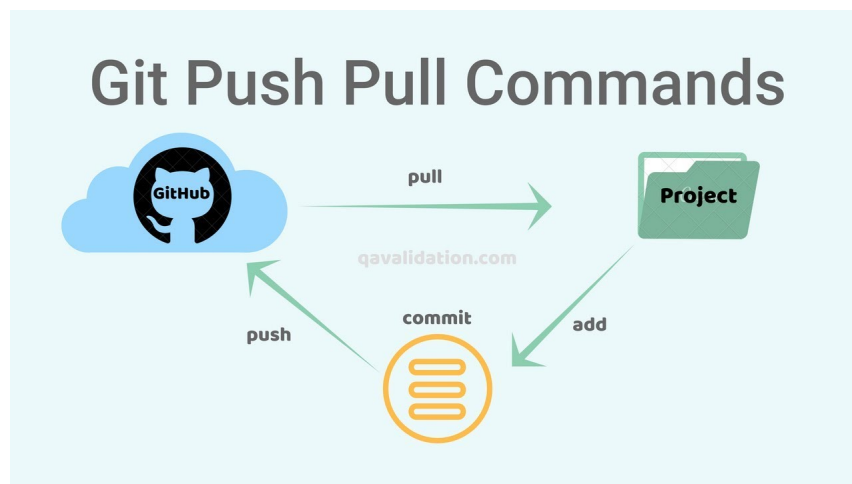
<sup>1</sup>Git, “1.1 Getting Started - About Version Control”, [Link to Website](#)



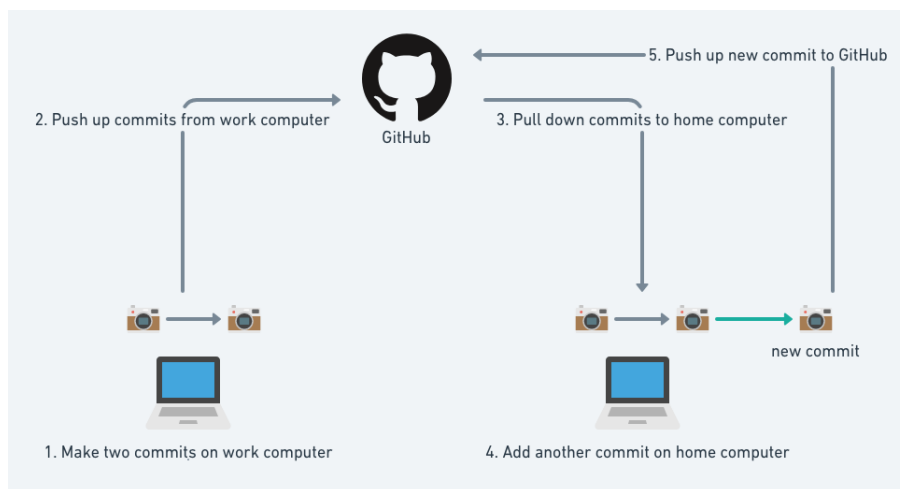
## 1.2 GitHub & Git Commands (Add, Push, Pull)

*What is GitHub and how do I use git commands with GitHub?*

- GitHub is a centralized location for hosting Git repositories (think about it: Git HUB).
- It is where we can store a copy of our local repository, previously referred to as the server repository.
- GitHub also has many other features that can help in keeping track of a project with many collaborators (i.e Issues & Projects tabs for a repository)
- Git has four fundamental commands that create versions of documents and updates them to GitHub or your local/remote repositories: `git add`, `git commit`, `git push`, and `git pull`
- *git add*: tells git you **want to add** a version of a file to your local repository
- *git commit*: commit **saves the versions** of the files you added previously to your local repository
- *git push*: push sends the updated local repository to your server repository in GitHub
- *git pull*: if you made changes to your server repository through another remote computer, then you use `git pull` when you want to update your local repository with those changes
- Side Note: Git repositories can store and update almost any file type. I've stored RMarkdowns, jupyter notebooks, python and R scripts, PDFs, and pictures.



## *The Cycle of GitHub*



## 1.3 Getting Started

*How do I get started?*

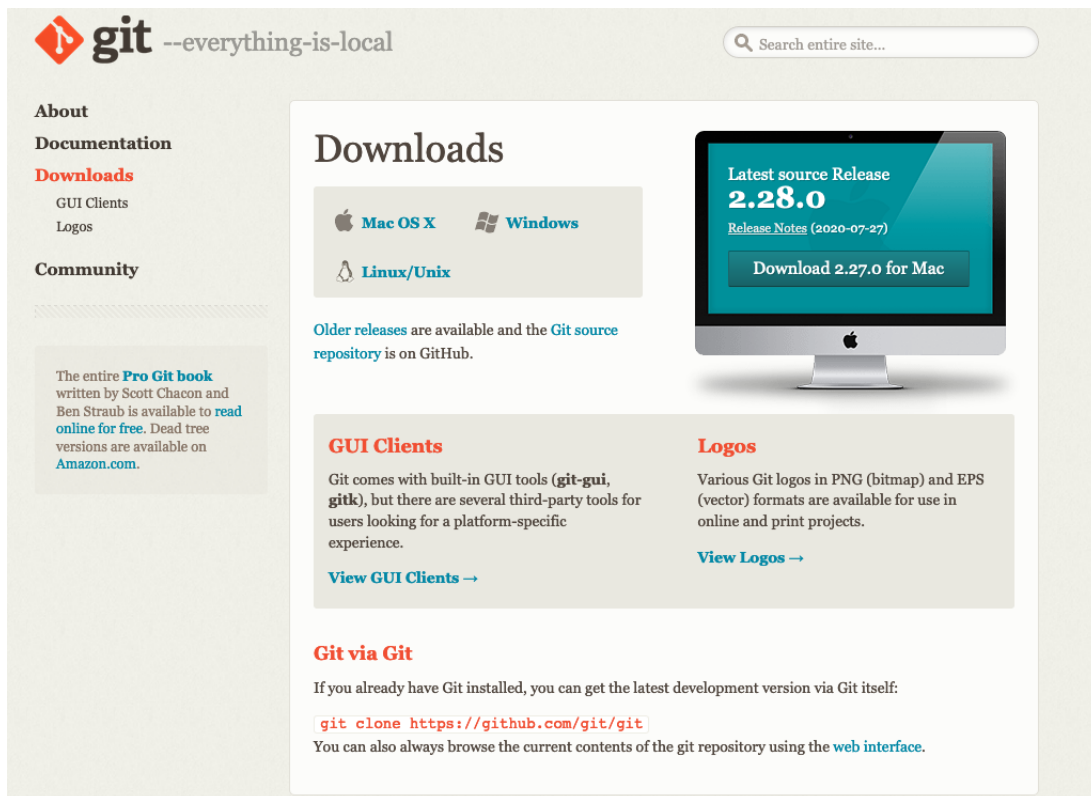
The rest of this document will show how to complete the following steps:

1. Download Git onto your local computer
2. Create a GitHub account
3. Create a repository on GitHub
4. Create a copy of a repository on your local computer
5. Add a file to your local repository
6. Push your file to GitHub
7. Make changes to a file on GitHub
8. Pull changes to your local computer

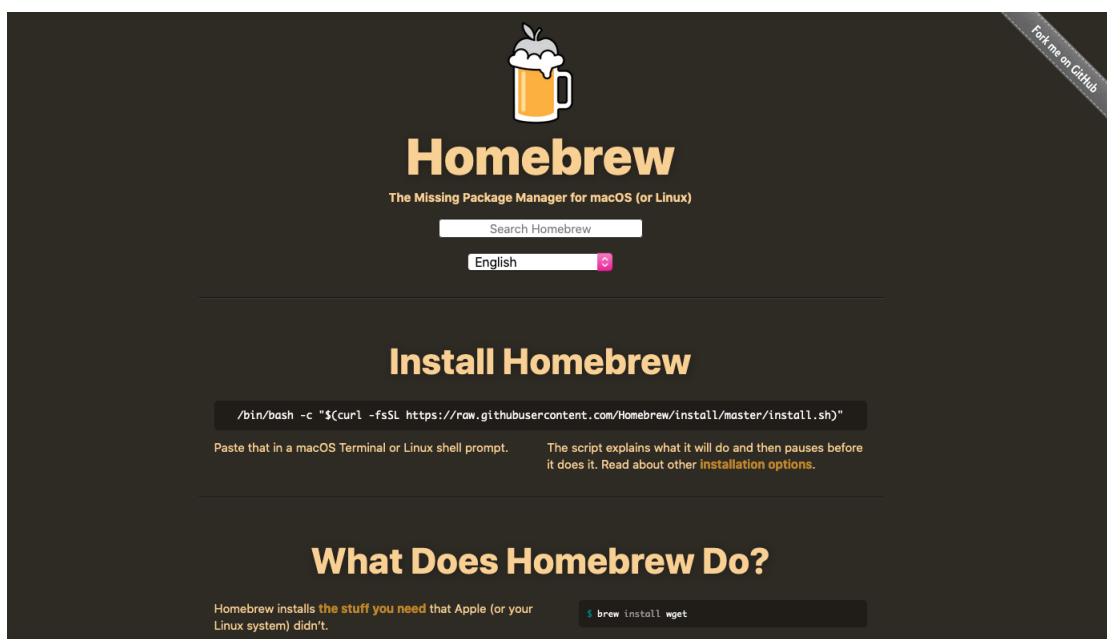


## 2 Installing Git

- Navigate to git's website: <https://git-scm.com/download/>

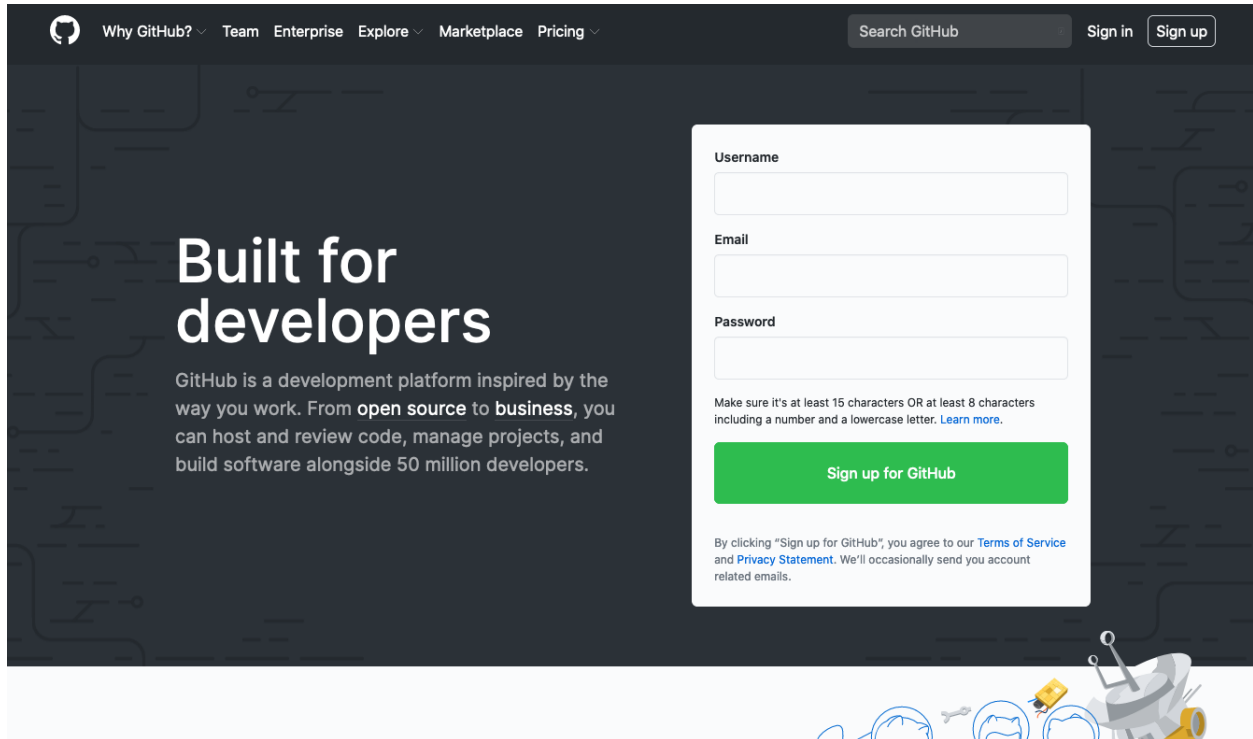


- Follow the instructions to download git depending on your operating system.
- For Windows: The download should start once you click on the Windows option.
- For Linux: Use your terminal and whichever package manager to download git.
- For Mac: Use your terminal to first install Homebrew (To check if Homebrew is install, type “brew help” into your terminal. If your terminal returns “command not found” then you need to install it) and then install git using the command “brew install git.”

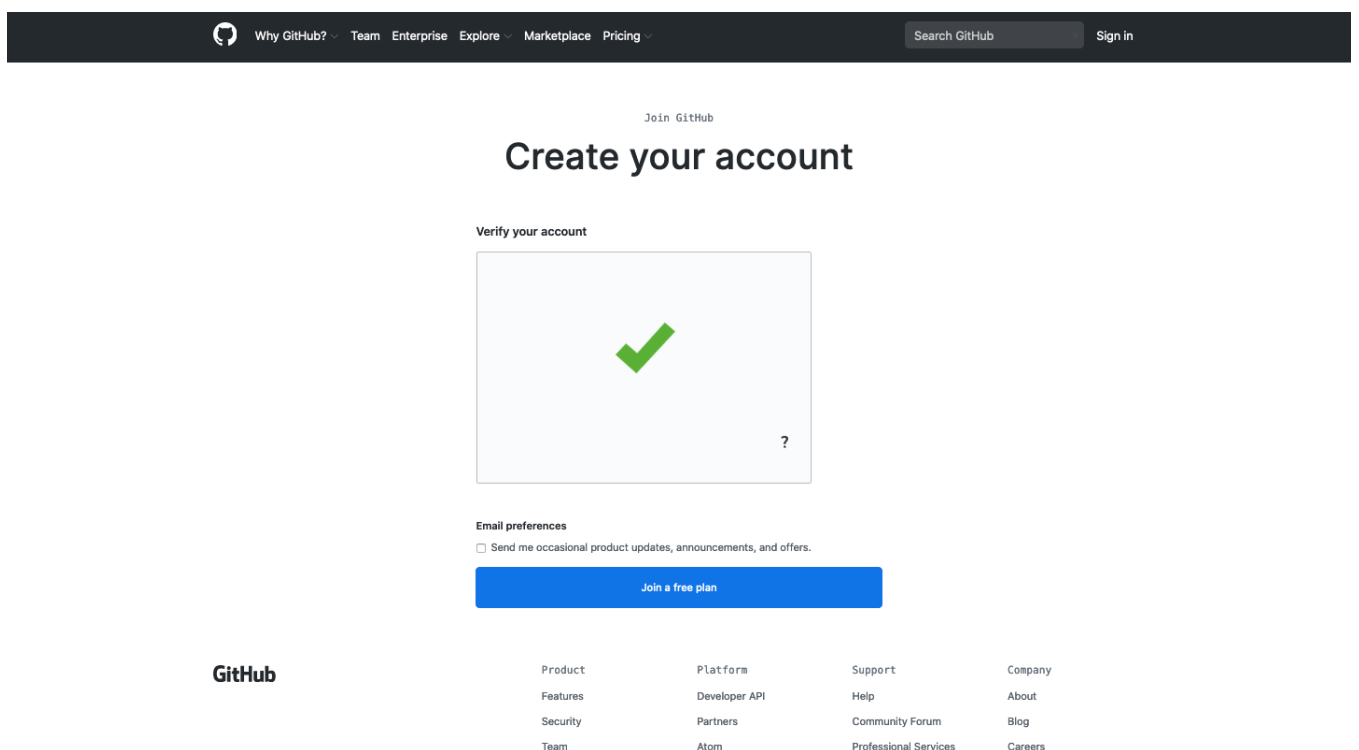


### 3 Setting Up GitHub Account

- Navigate to GitHub’s website: <https://github.com/>.
- The front page provides the first step in signing up for an account. Fill in the information needed.



- Verify your account and click “join a free plan.”
- Note on Free Plan: GitHub has a Pro account that students can upgrade to. See the extra section at the end of this worksheet.
- Note on Email Preferences: I unchecked the email preferences box because I just don’t need more emails cluttering my inbox. Maybe you feel the same way.



- Fill out the information page for your GitHub account.

- Once you've finished and moved on from the information page, verify your email.

Selected plan: Free

## Welcome to GitHub

Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there.

**What kind of work do you do, mainly?**

- Software Engineer (I write code)
- Student (I go to school)
- Product Manager (I write specs)
- UX & Design (I draw interfaces)
- Data & Analytics (I write queries)
- Marketing & Sales (I look at charts)
- Teacher (I educate people)
- Other (I do my own thing)

**How much programming experience do you have?**

- None (I don't program at all)
- A little (I'm new to programming)
- A moderate amount (I'm somewhat experienced)
- A lot (I'm very experienced)

**Please verify your email address**

Before you can contribute on GitHub, we need you to verify your email address. An email containing verification instructions was sent to [redacted].

[Resend verification email](#) [Change your email settings](#)

---

**GitHub**

Subscribe to our newsletter  
Get product updates, company news, and more.

[Subscribe](#)

- Product: Features, Security, Team, Enterprise, Customer stories, The ReadME Project, Pricing, Resources, Roadmap
- Platform: Developer API, Partners, Atom, Electron, GitHub Desktop
- Support: Help, Community Forum, Professional Services, Learning Lab, Status, Contact GitHub
- Company: About, Blog, Careers, Press, Social Impact, Shop

© 2020 GitHub, Inc. [Terms](#) [Privacy](#) [Site Map](#) [What is Git?](#) [Twitter](#) [Facebook](#) [YouTube](#) [LinkedIn](#) [GitHub](#)

Almost done, **@beliciarodriguez!** To complete your GitHub sign up, we just need to verify your email address:

[Verify email address](#)

Once verified, you can start using all of GitHub's features to explore, build, and share projects.

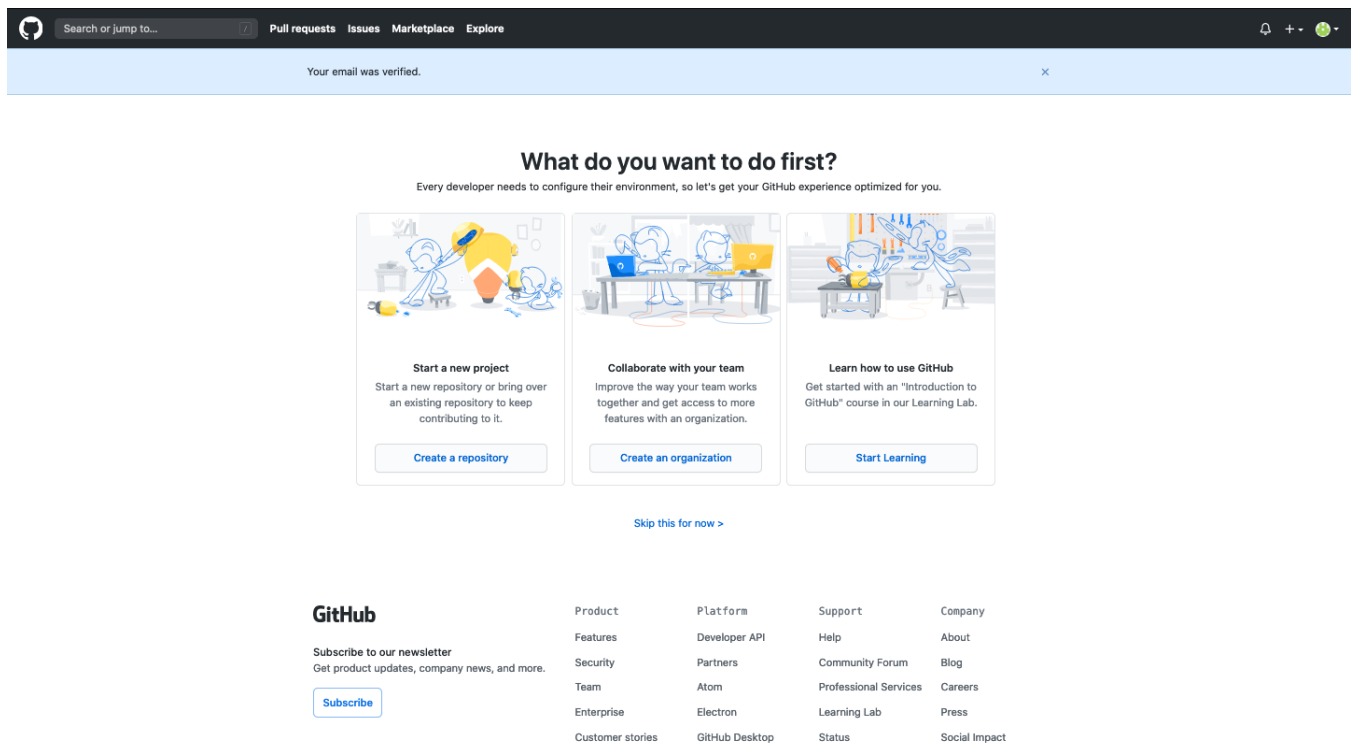
Button not working? Paste the following link into your browser: [redacted]

You're receiving this email because you recently created a new GitHub account or added a new email address. If this wasn't you, please ignore this email.

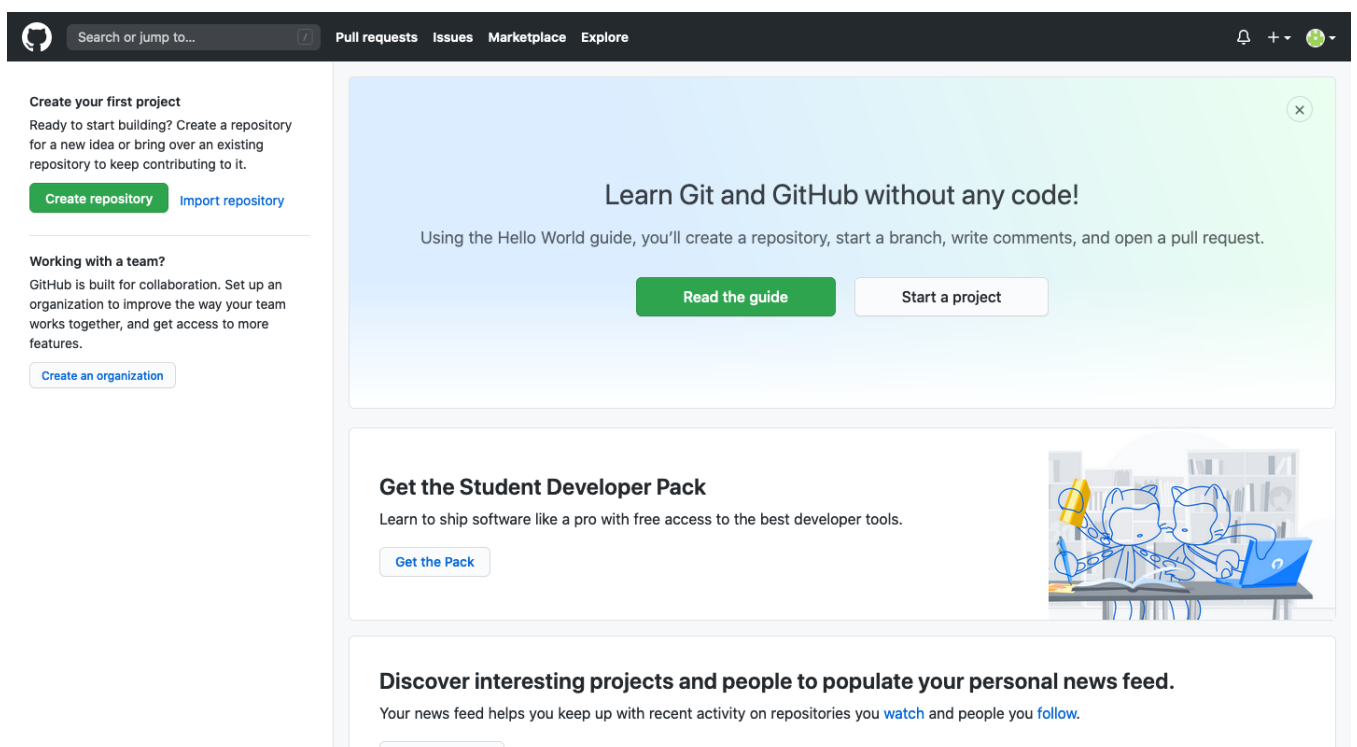
[Email preferences](#) [Terms](#) [Privacy](#) [Sign into GitHub](#)

**GitHub**

- Once you've verified your email, you will be sent to the following page; for this demonstration, click "skip this for now" under the options. I'd rather you learn how to create a new repository from the home page rather than these short cuts.



- And there you have it! You've created your GitHub account.
- Note on HelloWorld Guide on Screen: The guide explains how to create a repository, branch, commit, and pull request, and I find the document to also be a nice beginner tutorial to go through; however, it's not realistic while using GitHub to not have to use a little bit of coding when doing these git actions. Link to the guide: <https://guides.github.com/activities/hello-world/>.
- Note on Student Developer Pack: I will be going more in depth on this in the extras section of this document.



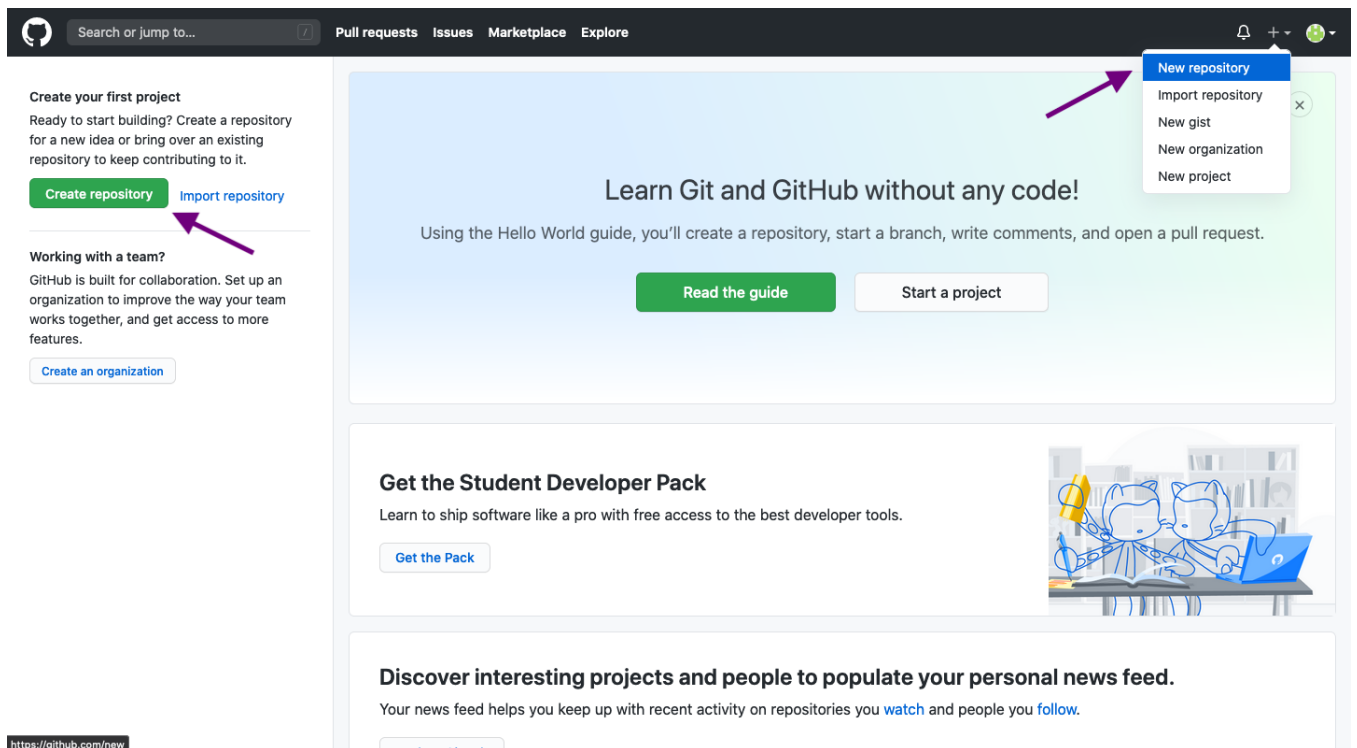


## 4 Creating a Repository

This section will explain how to create a repository that will be found both on your account on github and on your local computer. There are two methods; the first starts on github.com and clones the repository onto your local computer. The second method takes a folder on your local computer, makes it a repository, and pushes the repository to your github account. We will only be showing how to do the first method because it is the most common, but it is good to be aware that there are different ways of creating repositories.

Later in this section, we will begin using the terminal. The only command that I use besides the git commands is “cd”, meaning “change directories.” “cd” changes the directory your terminal is currently looking into. If you want to check what files can be found in the location your terminal is currently viewing, type “ls”. These commands may only be native to Mac computers, so if you have Windows or a Linux computer you may need to do some Google searching on what your commands are for your terminal.

- Return to your GitHub homepage: <https://github.com/>.
- To initiate the creation of a new repository, click either the green “Create repository” button on the left side of your screen, or click the plus button next to your icon and “New repository” button.



- There are several components to a repository that you need to set-up in this stage.

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---

**Owner \***  / **Repository name \***

Great repository names are short and memorable. Need inspiration? How about [upgraded-rotary-phone?](#)

**Description (optional)**

---

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

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

---

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

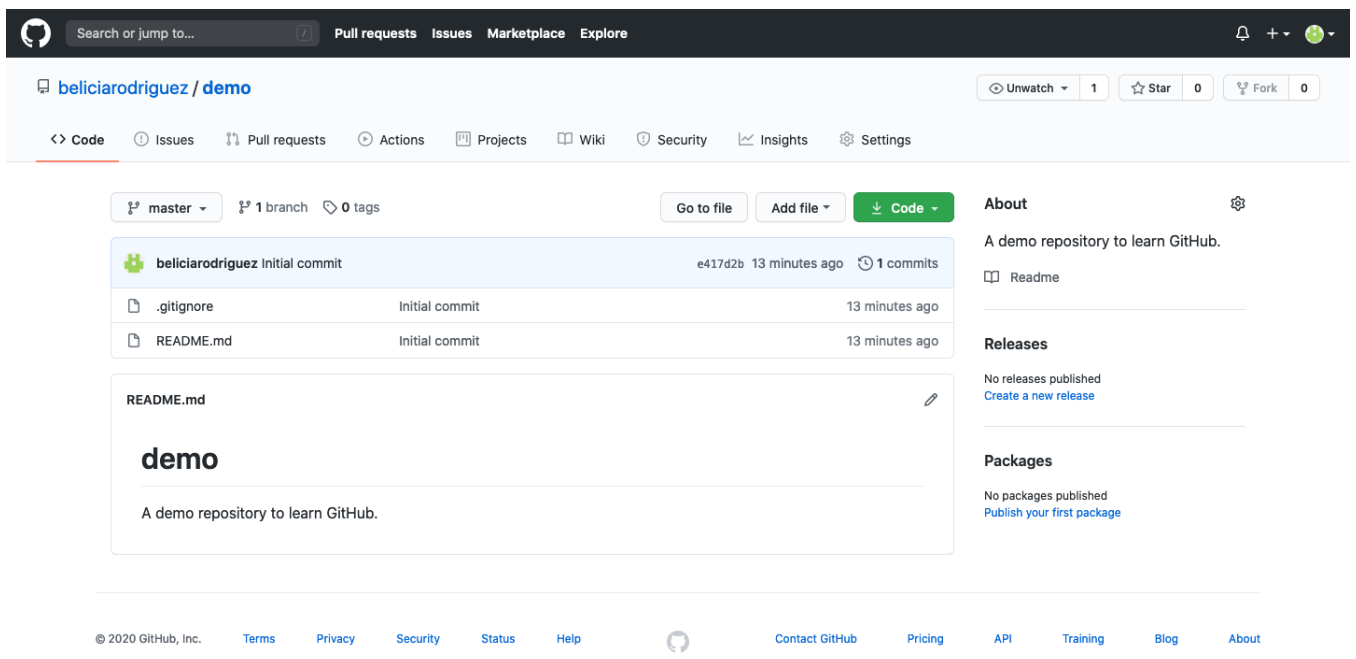
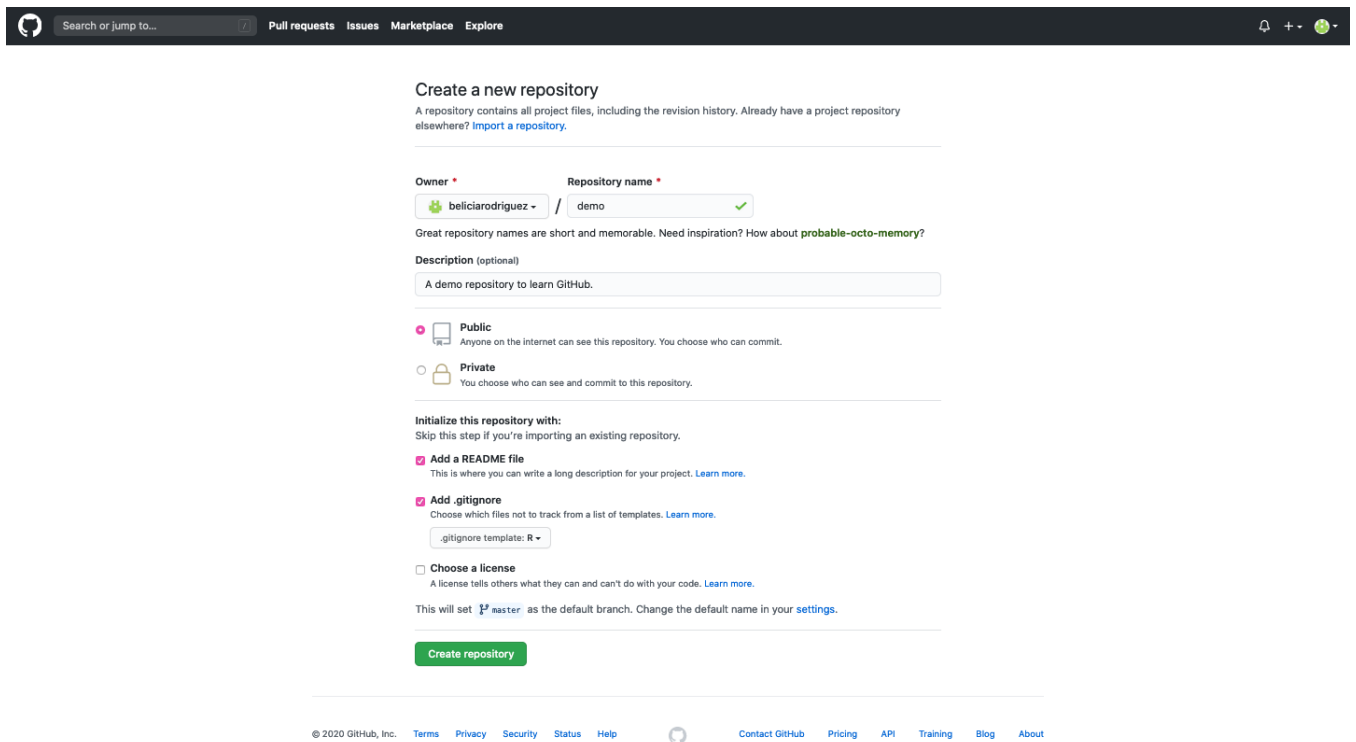
**Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

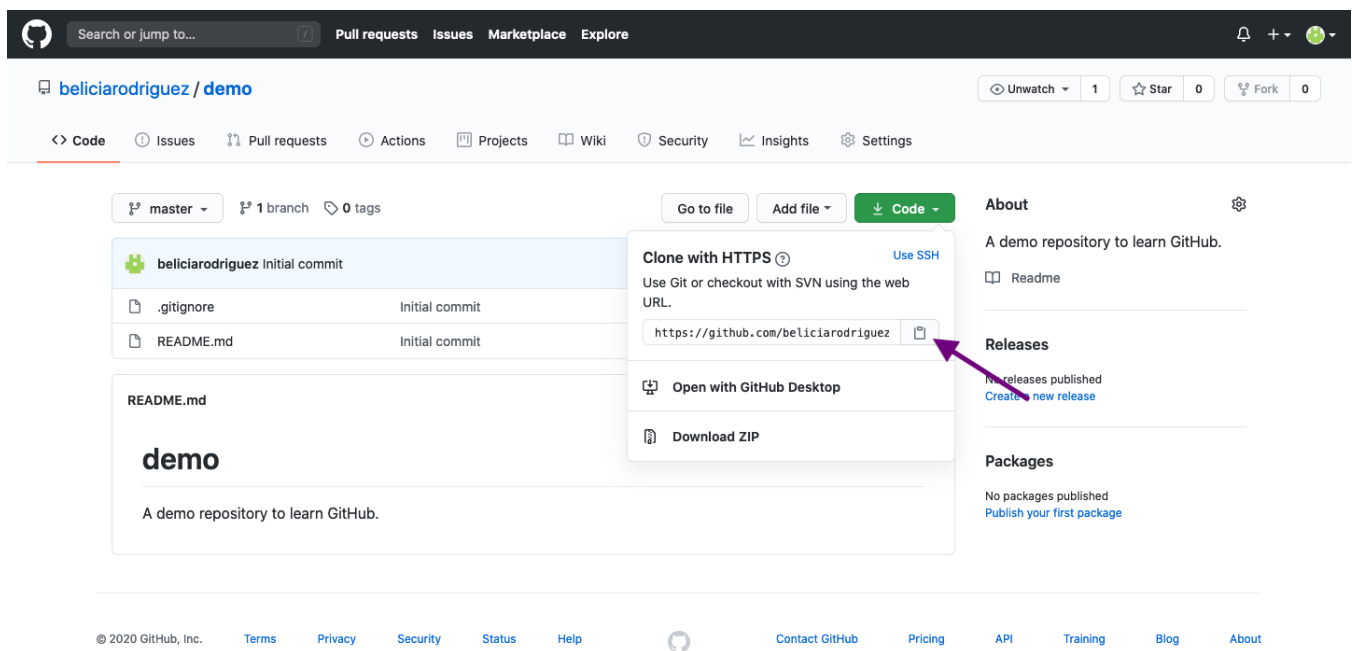
**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

---

- *Repository Name:* Name of your repository. Try to keep it short and succinct. For our purposes, we can name our repository “demo”.
- *Description (optional):* A sentence describing the purpose and contents of your repository, such as, “A demo repository to learn GitHub.” (You can probably think of a better description.)
- *Public or Private:* You can allow your repository to be seen by anybody (public) or by only people you give access to (private). For now, we will make our repositories public.
- *Add a README file:* Repositories are customarily given a README file, which is a markdown file where you can write a longer description about what is included in your repository. I would check this box off, and we can later write into the README file.
- *Add .gitignore:* A .gitignore file is a list of folders and subdirectories that you do not want git to push or pull from your local or GitHub repository. I would check this box off and select the programming language you will primarily using in your repository as your template; in our case, type “R” into the “Filter ignore” search box.
- *Choose a license:* This doesn’t apply to us right now; this is mostly for repositories that are meant to be shared with others, and the license writes out what can or cannot be done with the code in the repository. I would not check it off.
- These settings are not permanent, and if you decide later on that, for example, you want to make your repository private, you can change that setting later on.
- Once you’re satisfied with your initial repository settings, click “Create Repository”.



- You’ve made your repository! Note that our repository has a .gitignore file and a README.md file. The name of the repository is next to our username (EX. beliciarodriguez/demo) and our description is in the right-side column under “About”.
- Also note that your README file is displayed underneath the files in your repository. This helps whoever sees your repository can immediately read its description.
- Currently, your repository is only available on github.com, but we want our repository to also be on our local computer so we can start creating and editing files.
- Click the green “Code” button, and a small box will appear with three different options.



- *Clone with HTTPS*: Every repository has its own “git URL”. For example, my demo repository’s git URL is the following: `https://github.com/beliciarodriguez/demo.git`. Using this git URL, I can “clone” my repository as a folder to my local computer, and my local computer will recognize the folder as a git repository, and I can use git commands inside of this folder.
- *Open with GitHub Desktop*: GitHub Desktop is an application by GitHub that helps simplify pushing and pulling commits from your remote (i.e GitHub website) repository to your local computer (and vice versa). What’s nice about the application is that you do not need to use the terminal, which we will use later on. I’ve shared a link to GitHub Desktop in the extra resources portion of this document.
- *Download ZIP*: You can also download all of the files of your repository as a folder to your local computer, but your local computer would not recognize it as a repository. This option is useful if you just want the files of a public repository (that is usually not your own) and do not plan on interacting with the repository.
- We will be cloning with HTTPS; copy your repository URL by clicking the clipboard button next to your unique URL.
- Open up your operating system’s terminal. We will use our first git command: “git clone”. git clone retrieves your repository from your repository URL and clones it to your local computer. To use this command, type “git clone [paste your unique repository URL here]”.

```
Last login: Tue Sep 1 14:08:07 on ttvs000
(base) b ~ % cd github_talk
(base) b ~ % git clone https://github.com/beliciarodriguez/demo.git
Cloning into 'demo'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 1004 bytes | 251.00 KiB/s, done.
(base) b ~ % cd github_talk %
```

Name	Date Modified	Size	Kind
demo	Today at 2:15 PM	--	Folder
README.md	Today at 2:15 PM	42 bytes	md

- *Note on the First Line* (cd github\_talk): as I mentioned in the beginning, cd is a command in the terminal that stands for “change directory”. I used it to move into a folder I previously created called “github\_talk”. You may also want to create a designated folder for all your GitHub repositories and then use the “cd” command to change your terminal directory to that folder; therefore, when you clone your repository, your repository will appear in that directory.
- *Note on .gitignore file*: You may have noticed our .gitignore file is not visible in our repository folder. This is something I’ve seen happen on the Mac, but, in my experience, this is not an issue on Linux. For Windows I’m not certain if this is a problem. Either way, files that begin with a “.” may not be visible in the folder. That does not mean that the file is not there. You may need to use a text editor such as Atom (<https://atom.io/>) to see and edit these files. For Mac, you may also see a .DS\_store, which are files that preserve the order of your files in Finder.
- The below image is a screenshot of my Atom editor viewing the repository we’ve just cloned, and the .gitignore file is being displayed. As you can see, my demo repository is ready to be used on my local computer!

```

1 # History files
2 .Rhistory
3 .Rapp.history
4
5 # Session Data files
6 .RData
7
8 # User-specific files
9 .Ruserdata
10
11 # Example code in package build process
12 *-Ex.R
13
14 # Output files from R CMD build
15 /*.tar.gz
16
17 # Output files from R CMD check
18 /*.Rcheck/
19
20 # RStudio files
21 .Rproj.user/
22
23 # produced vignettes
24 vignettes/*.html
25 vignettes/*.pdf
26
27 # OAuth2 token, see https://github.com/hadley/htrr/releases/tag/v0.3
28 .htrr-oauth
29
30 # knitr and R markdown default cache directories
31 *_cache/
32 /cache/
33
34 # Temporary files created by R markdown
35 *.utf8.md
36 *.knit.md

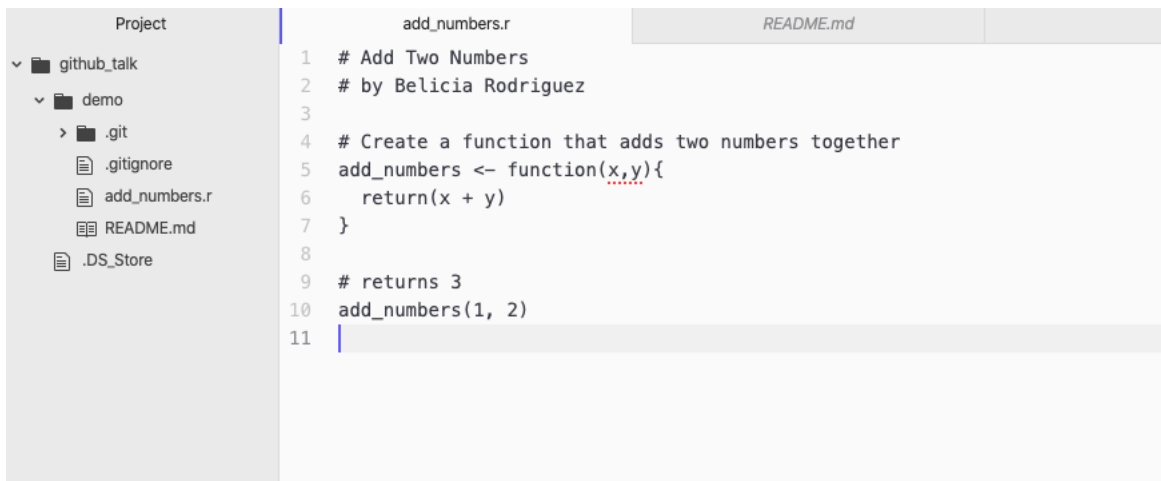
```

## 5 Using Git Commands to Update Repository

This section will demonstrate how to “push” and “pull” from your local repository (i.e on your local computer) to the repository on your remote repository (i.e GitHub website). We will go over the process for both directions and review other useful commands.

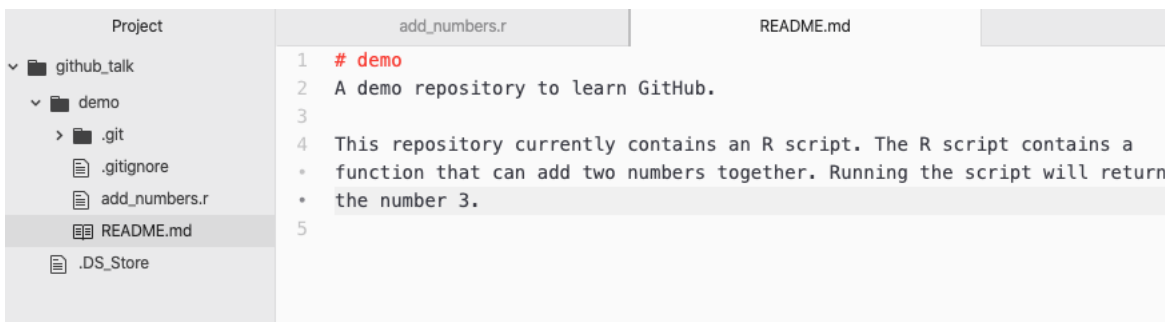
### 5.1 git status, git add, git commit, git push

- First, we need to create a new file for our repository. In this demonstration, I made an R script that contains a function that adds two numbers together. The file is saved to my repository.
- I will also update my README file so that anyone who visits my repository knows that there is an R script that contains an “add\_numbers” function.



The screenshot shows a code editor with a file explorer on the left. The file explorer shows a project named 'github\_talk' with a subfolder 'demo' containing files '.gitignore', 'add\_numbers.r', 'README.md', and '.DS\_Store'. The main editor area shows the content of 'add\_numbers.r' with the following code:

```
1 # Add Two Numbers
2 # by Belicia Rodriguez
3
4 # Create a function that adds two numbers together
5 add_numbers <- function(x,y){
6   return(x + y)
7 }
8
9 # returns 3
10 add_numbers(1, 2)
11
```



The screenshot shows the same IDE with the 'README.md' file selected in the file explorer. The main editor area shows the content of 'README.md' with the following text:

```
1 # demo
2 A demo repository to learn GitHub.
3
4 This repository currently contains an R script. The R script contains a
5 • function that can add two numbers together. Running the script will return
6 • the number 3.
```

- Open your terminal and make sure you “change directories” (cd) so that you are working in your repository folder.
- *An Error on Changing Directories:* I believe it’s just as important to highlight errors as it is to show the correct way. When I changed directories, I only changed to my “github\_talk” folder and did not additionally go to my “demo” repository folder. When I used a git command, my terminal gave me an error; “not a git repository.” Git will give you this error if you are trying to use git commands in a folder that is not a repository. To fix this error, make sure you are in your repository folder. For me, that just means changing directories one more time.

```

Last login: Wed Sep  2 15:29:50 on ttys000
(base) [REDACTED] ~ % cd github_talk
(base) [REDACTED] github_talk % git status
fatal: not a git repository (or any of the parent directories): .git
(base) [REDACTED] github_talk % █

```

- First, type “git status” into your terminal. git status shows all the files you have changed and gives recommendations on what commands you should use next. It is a great way to see the “status” of your files before you move on to doing any other commands.
- The first line tells us what branch we are working on, which is the “master” branch. The following line tells us our repository is up to date (meaning our local and remote repositories match).
- Our “README.md” file is under the section “changes not staged for commit,” which means git has recognized this as a file in your repository. Git has noted that you have made changes to this file that are not on the remote repository nor have you “saved” or, in GitHub lingo, “committed” this version of the file.
- Our “add\_numbers.r” is under the section “Untracked files,” meaning git has not been tracking changes on this file, usually because it is a new file that you’ve recently added to the repository.

```

Last login: Wed Sep  2 15:30:01 on ttys000
(base) [REDACTED] ~ % cd github_talk/demo
(base) [REDACTED] demo % git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

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

no changes added to commit (use "git add" and/or "git commit -a")
(base) [REDACTED] demo % █

```

- Now we want to add our changes to our “commit.” Our “commit,” which we will do later, will contain all of the additional changes we want to make to our remote repository. Before we commit, we have to first “add” our changes. We can add each individual file by typing “git add README.md” first then “git add add\_numbers.r”. However, I always appreciate a shortcut, and we will be typing “git add .” to add all the changes we’ve made to our commit.

```
(base) [redacted] demo % git add .
(base) [redacted] demo % git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
       modified:   README.md
       new file:   add_numbers.r

(base) [redacted] demo %
```

- Now both of our files are ready to be committed.
- To commit our files, we will type the following: `git commit -m "adding files"`
- The “adding files” is a message associated with that commit, and each commit must have a message associated with it.

```
(base) [redacted] demo % git commit -m "adding files"
[master c28ab45] adding files
 2 files changed, 12 insertions(+)
 create mode 100644 add_numbers.r
(base) [redacted] demo % git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

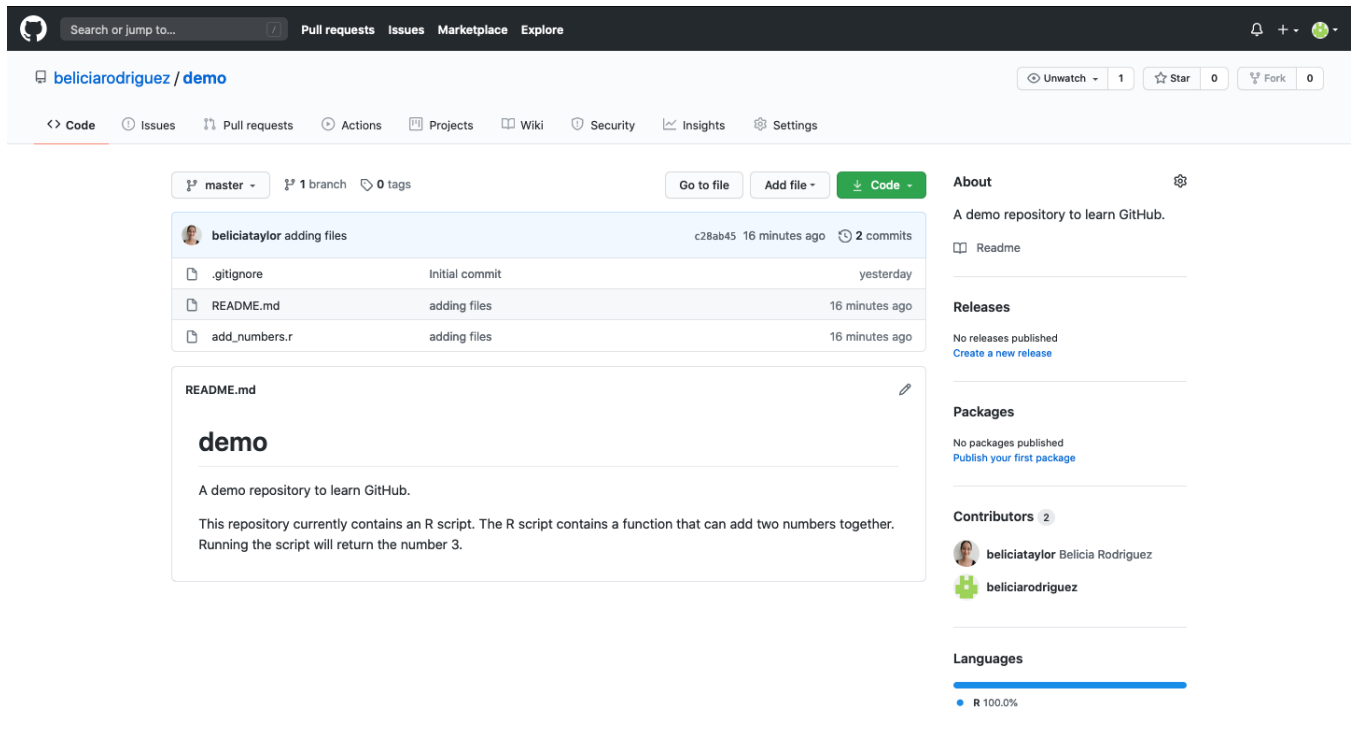
nothing to commit, working tree clean
(base) [redacted] demo %
```

- Note that when we enter `git status`, our files have disappeared, but we get an updated message that our branch is ahead of “origin/master” by 1 commit. This means we still have yet to update our remote repository with our changes.
- Now, type “`git push`” to publish our changes to our remote repository.

```
(base) [redacted] demo % git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 590 bytes | 295.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/beliciarodriguez/demo.git
 e417d2b..c28ab45  master -> master
(base) [redacted] demo %
```

- There you have it! You’ve updated your remote repository. If you go back to <https://github.com/> and see your repository, you will notice a couple of updates.
- *Note on Picture:* Because I have registered my local computer under a different github account (and I didn’t want to go through the process of connecting and disconnecting my demo and original accounts), I pushed my updated using my real account. You will see “beliciataylor” as a contributor and the one creating the commits. The process was the same. Your screen will solely have your account.

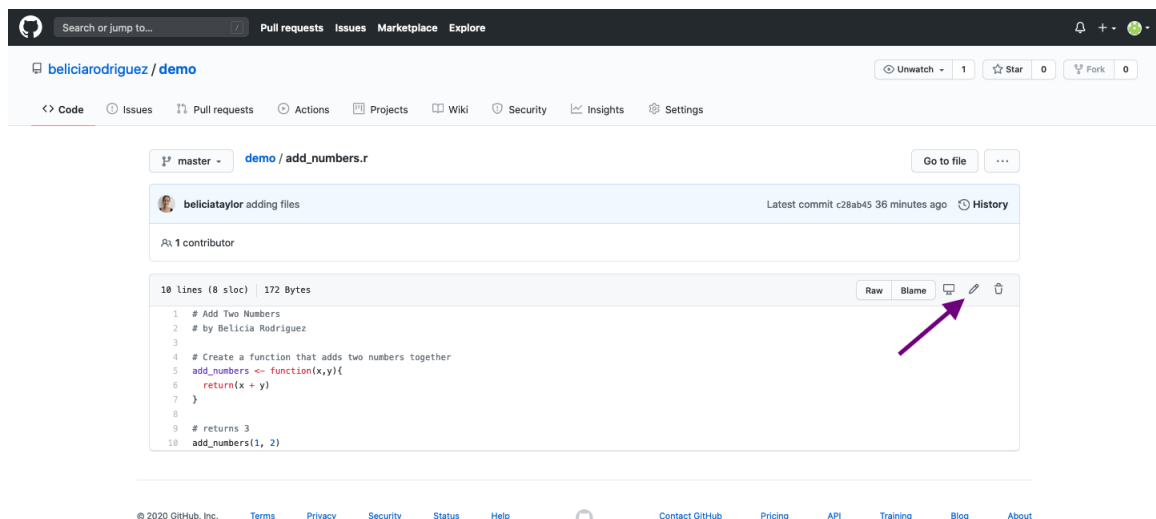




- Our add\_numbers.r file has been added and our README file has been updated.
- Notice how our README file underneath our files has been updated with the sentence we added. This is a nice feature GitHub has to make the first thing a visitor of this repository sees is your README file describing the repository.

## 5.2 git pull

- If you have collaborators on your repository or you decide to update your repository from a different computer (i.e a school computer or a virtual machine), you may have updates on your remote repository that are not on your local computer
- We can also create changes to our repository directly on github.com. In our example, we will make changes to our “add\_numbers.r” file.
- Go to your “add\_numbers.r” file and click on the pencil icon in the corner.



- We will add another function to the file titled “add\_three\_numbers,” which I think is pretty self-explanatory.
- Below our text box, GitHub has already set up a commit message. Once we’ve clicked “commit changes”, our repository is updated.

The screenshot shows a GitHub web editor interface. At the top, there's a breadcrumb "demo / add\_numbers.r" and a "Cancel" button. Below that is a code editor with a "Preview changes" tab. The code is as follows:

```

1 # Add Two Numbers
2 # by Belicia Rodriguez
3
4 # Create a function that adds two numbers together
5 add_numbers <- function(x,y){
6   return(x + y)
7 }
8
9 # returns 3
10 add_numbers(1, 2)
11
12 # Create a function that adds three numbers together
13 add_three_numbers <- function(x,y,z){
14   return(x + y + z)
15 }

```

Below the code editor is a "Commit changes" dialog. It has a text input field containing "Update add\_numbers.r" and a larger text area for an optional extended description. There are two radio buttons: the first is selected and labeled "Commit directly to the master branch.", and the second is labeled "Create a new branch for this commit and start a pull request." At the bottom of the dialog are "Commit changes" and "Cancel" buttons.

- Now head back over to your terminal, change directories into your local repository folder, and type “git pull”

```

(base) [redacted] demo % git pull
warning: Pulling without specifying how to reconcile divergent branches is
discouraged. You can squelch this message by running one of the following
commands sometime before your next pull:

  git config pull.rebase false # merge (the default strategy)
  git config pull.rebase true  # rebase
  git config pull.ff only      # fast-forward only

You can replace "git config" with "git config --global" to set a default
preference for all repositories. You can also pass --rebase, --no-rebase,
or --ff-only on the command line to override the configured default per
invocation.

remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 729 bytes | 145.00 KiB/s, done.
From https://github.com/beliciarodriguez/demo
   c28ab45..c8f5cc1 master    -> origin/master
Updating c28ab45..c8f5cc1
Fast-forward
 add_numbers.r | 5 +++++
 1 file changed, 5 insertions(+)
(base) [redacted] demo %

```

- The warning in the beginning may appear, but it is only a warning and not a problem with the actual pull request. You will not normally be getting this warning, but if you do, don't worry. Follow the prompts to prevent it from happening again, or just ignore it.
- Now if you check your R script file on your local computer, you will see our additional function!

```

Project
├── github_talk
│   └── demo
│       ├── .git
│       ├── .gitignore
│       ├── add_numbers.r
│       ├── README.md
│       └── .DS_Store
└── add_numbers.r
1 # Add Two Numbers
2 # by Belicia Rodriguez
3
4 # Create a function that adds two numbers together
5 add_numbers <- function(x,y){
6   return(x + y)
7 }
8
9 # returns 3
10 add_numbers(1, 2)
11
12 # Create a function that adds three numbers together
13 add_three_numbers <- function(x,y,z){
14   return(x + y + z)
15 }
16
demo/add_numbers.r 16:1
LF UTF-8 Plain Text master Fetch GitHub Git (0)

```

## 6 Concluding Message

Now you know how to update your local and remote repositories on GitHub! Hopefully you're now curious about what more you can do with GitHub and are ready to start making more repositories. If you have any questions about how to use GitHub or have any comments about this document, feel free to email me (btrodr2@emory.edu). Thank you!

## 7 Extra Resources

### 7.1 GitHub Desktop

*Website:* <https://desktop.github.com/>

As fun as it is to use the terminal, GitHub Desktop is a tool that allows you to push, pull, commit, and run other git commands without the terminal. This application can help quicken the workflow between your local computer and GitHub.

### 7.2 Git Cheat Sheet

*PDF Link:* <https://education.github.com/git-cheat-sheet-education.pdf>

This is a really handy cheat sheet that GitHub made which categorizes many of the most useful commands git into their functions.

### 7.3 GitHub Student Developer Pack

*Website:* <https://education.github.com/pack>)

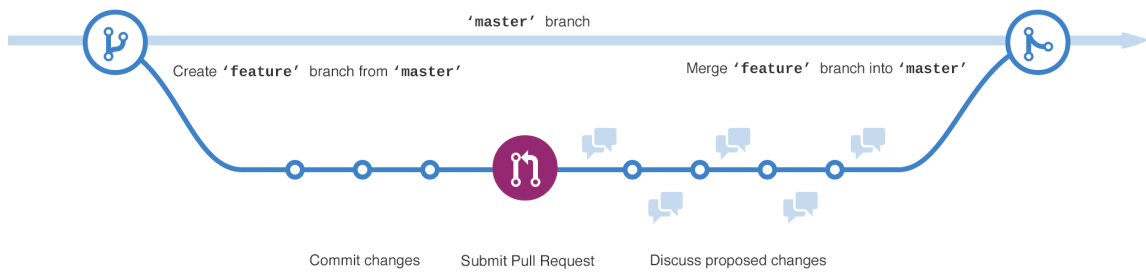
Register your GitHub account with your Emory email (if you didn't originally create your account with your Emory email, don't worry, you can add a secondary email to your account in your account settings) and you can have access to many cool deals that GitHub has for students. A couple of my most noteworthy: If you want to update your account to Pro, you can do so through the pack. If your interested in cloud computing, the student pack gives you \$100 of AWS Educate credit and \$100 of Microsoft Azure. If you want to register a domain name, the pack gives you one free domain name using name.com.

### 7.4 Collaboration with Git

Git allows several people to collaborate on one repository ; this is done by using features called branches and merging. EX. Let's say you're working on a feature for a website. You can create a new "branch" that does not affect the main branch or "live" version of the website. You can work on versions of the website with your feature on your separate branch. When you're finished your work, you can merge your branch with the master branch and launch your new and improved version of the website<sup>2</sup>. You can also discuss proposed changes to your project with other collaborators before merging your branch with the master branch. In this way, everyone is aware of all the changes happening to the project.

---

<sup>2</sup>Example adapted from following post: <https://www.nobledesktop.com>



## 7.5 Create A Website Using GitHub

*GitHub Pages:* <https://pages.github.com/>

*Hugo Academic Theme:* <https://themes.gohugo.io/academic/>

*My Website Using GitHub:* <https://beliciataylor.github.io/>

*GitHub Repository Hosting My Website:* <https://github.com/beliciataylor/academic-kickstart>

You can create a website using GitHub repositories! GitHub actually creates a unique domain name for you, which is in the format “[your username].github.io”. For example, you can create a personal website using Hugo Academic. For example (and yes, a shameless plug), I’ve created a website using my GitHub domain name and Hugo Academic. If you’re interested, then I would suggest reading the documentation for the website theme and looking at an example website. The repository that launches my website is linked above.