**Step 0: Install git and create a GitHub account**

The first two things you'll want to do are:

⇒ **install git** and **create a free GitHub account.**

**Git and GitHub*:*** *git and GitHub are****not****the same thing.*

*⇒Git is an open-source, for  tracking changes in any set of files.*

*⇒ GitHub is a company founded in 2008 that makes tools which integrate with git. You do not need GitHub to use git, but you cannot use GitHub without using git.*

**Step 1: Create a local git repository**

*⇒* To begin, open up a terminal and move to where you want to place the project on your local machine using the cd (change directory) command. For example, if you have a 'projects' folder on your desktop, you'd do something like:

mnelson:Desktop mnelson$ cd ~/Desktop

mnelson:Desktop mnelson$ mkdir myproject

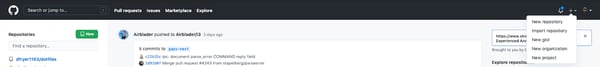
mnelson:Desktop mnelson$ cd myproject/

⇒ To initialize a git repository in the root of the folder, run the [git init](http://git-scm.com/docs/git-init) command:

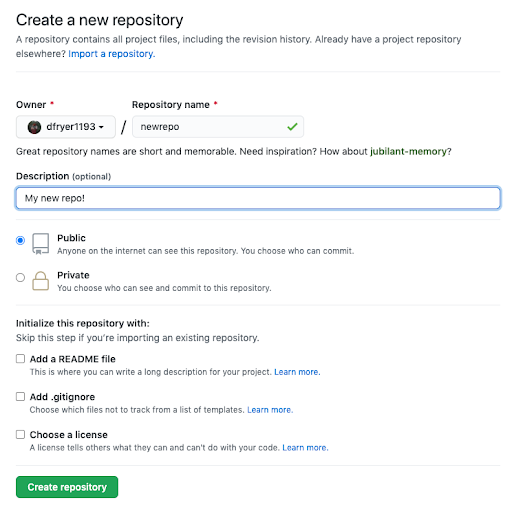
mnelson:myproject mnelson$ **git init**

*⇒* **Create a new repository on GitHub**

To create a new repo on GitHub, log in and go to the GitHub home page. You can find the “New repository” option under the “+” sign next to your profile picture, in the top right corner of the navbar:



 After clicking the button, GitHub will ask you to name your repo and provide a brief description:



When you're done filling out the information, press the 'Create repository' button to make your new repo.

GitHub will ask if you want to create a new repo from scratch or if you want to add a repo you have created locally. In this case, since we've already created a new repo locally, we want to push that onto GitHub so follow the '....or push an existing repository from the command line' section:

**Step 2: Add a new file to the repo**

⇒ Go ahead and add a new file to the project, using any text editor you like or running a [touch](http://linux.die.net/man/1/touch) command. `***touch newfile.txt***` just creates and saves a blank file named newfile.txt.

Once you've added or modified files in a folder containing a git repo, git will notice that  the file exists inside the repo. But, git won't track the file unless you explicitly tell it to. Git only saves/manages changes to files that it *tracks*, so we’ll need to send a command to confirm that yes, we want git to track our new file.

mnelson:myproject mnelson$ touch mnelson.txt

mnelson:myproject mnelson$ ls

mnelson.txtAfter creating the new file, you can use the [git status](http://git-scm.com/docs/git-status) command to see which files git knows exist.

mnelson:myproject mnelson$ ***git status***

On branch master

Initial commit

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

mnelson.txt

nothing added to commit but untracked files present (use "git add" to track)

What this basically says is, "Hey, we noticed you created a new file called mnelson.txt, but unless you use the 'git add' command we aren't going to do anything with it."

A [commit](https://docs.github.com/en/free-pro-team@latest/github/getting-started-with-github/github-glossary#:~:text=the%20repository%20owner.-,commit,who%20made%20them%20and%20when.) is a record of what changes you have made since the last time you made a commit. Essentially, you make changes to your repo (for example, adding a file or modifying one) and then tell git to put those changes into a commit.

Commits make up the essence of your project and allow you to jump to the state of a project at any other commit.

So, how do you tell git which files to put into a commit? This is where the [staging environment or index](https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository) come in. As seen in Step 2, when you make changes to your repo, git notices that a file has changed but won't do anything with it (like adding it in a commit).

To add a file to a commit, you first need to add it to the staging environment. To do this, you can use the [git add](http://git-scm.com/docs/git-add) <filename> command (see Step 3 below).

Once you've used the git add command to add all the files you want to the staging environment, you can then tell git to package them into a commit using the [git commit](http://git-scm.com/docs/git-commit) command.

Note: The staging environment, also called 'staging', is the new preferred term for this, but you can also see it referred to as the 'index'.

**Step 3: Add a file to the staging environment**

Add a file to the staging environment using the git add command.

If you rerun the git status command, you'll see that git has added the file to the staging environment (notice the "Changes to be committed" line).

mnelson:myproject mnelson$ git status

To reiterate, the file has **not**yet been added to a commit, but it's about to be.

**Step 4: Create a commit**

It's time to create your first commit!

Run the command git commit -m "Your message about the commit"

mnelson:myproject mnelson$ git commit -m "This is my first commit!"

[master (root-commit) b345d9a] This is my first commit!

1 file changed, 1 insertion(+)

create mode 100644 mnelson.txt

**Step 5: Create a new branch**

Now that you've made a new commit, let's try something a little more advanced.

Say you want to make a new feature but are worried about making changes to the main project while developing the feature. This is where [git branches](https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell) come in.

Branches allow you to move back and forth between 'states' of a project. Official git docs describe branches this way: ‘A branch in Git is simply a lightweight movable pointer to one of these commits.’ For instance, if you want to add a new page to your website you can create a new branch just for that page without affecting the main part of the project. Once you're done with the page, you can [merge](https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging) your changes from your branch into the primary branch. When you create a new branch, Git keeps track of which commit your branch 'branched' off of, so it knows the history behind all the files.

Let's say you are on the primary branch and want to create a new branch to develop your web page. Here's what you'll do: Run [git checkout -b <my branch name>](http://git-scm.com/docs/git-checkout). This command will automatically create a new branch and then 'check you out' on it, meaning git will move you to that branch, off of the primary branch.

After running the above command, you can use the [git branch](http://git-scm.com/docs/git-branch) command to confirm that your branch was created:

mnelson:myproject mnelson$ git branch

master

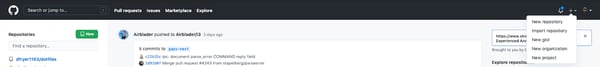
\* my-new-branch

The branch name with the asterisk next to it indicates which branch you're on at that given time.

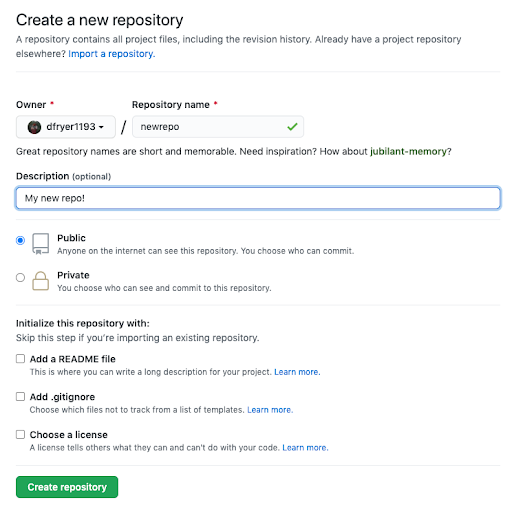
**Step 6: Create a new repository on GitHub**

If you only want to keep track of your code locally, you don't need to use GitHub. But if you want to work with a team, you can use GitHub to collaboratively modify the project's code.

To create a new repo on GitHub, log in and go to the GitHub home page. You can find the “New repository” option under the “+” sign next to your profile picture, in the top right corner of the navbar:



After clicking the button, GitHub will ask you to name your repo and provide a brief description:



When you're done filling out the information, press the 'Create repository' button to make your new repo.

GitHub will ask if you want to create a new repo from scratch or if you want to add a repo you have created locally. In this case, since we've already created a new repo locally, we want to push that onto GitHub so follow the '....or push an existing repository from the command line' section:

mnelson:myproject mnelson$ git remote add origin https://github.com/cubeton/mynewrepository.git

mnelson:myproject mnelson$ git push -u origin master

Counting objects: 3, done.

Writing objects: 100% (3/3), 263 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/cubeton/mynewrepository.git

\* [new branch] master -> master

Branch master set up to track remote branch master from origin.

**Step 7: Push a branch to GitHub**

Now we'll push the commit in your branch to your new GitHub repo. This allows other people to see the changes you've made. If they're approved by the repository's owner, the changes can then be merged into the primary branch.

To push changes onto a new branch on GitHub, you'll want to run [git push](http://git-scm.com/docs/git-push) origin yourbranchname. GitHub will automatically create the branch for you on the remote repository:

mnelson:myproject mnelson$ git push origin my-new-branch

Counting objects: 3, done.

Delta compression using up to 8 threads.

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 313 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

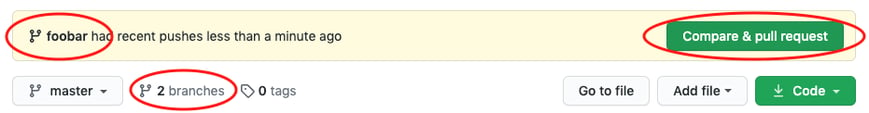
To https://github.com/cubeton/mynewrepository.git

\* [new branch] my-new-branch -> my-new-branch

You might be wondering what that "origin" word means in the command above. What happens is that when you clone a remote repository to your local machine, git creates an alias for you. In nearly all cases this alias is called "[origin](https://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes)." It's essentially shorthand for the remote repository's URL. So, to push your changes to the remote repository, you could've used either the command: git push git@github.com:git/git.git yourbranchname or git push origin yourbranchname

(If this is your first time using GitHub locally, it might prompt you to log in with your GitHub username and password.)

If you refresh the GitHub page, you'll see note saying a branch with your name has just been pushed into the repository. You can also click the 'branches' link to see your branch listed there.

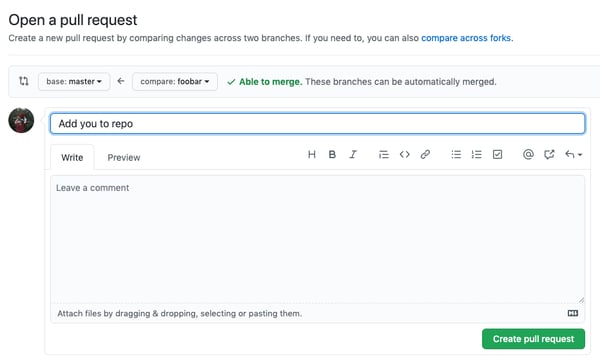
[](https://cloud.githubusercontent.com/assets/5241432/9189475/da30eb86-3fb6-11e5-934f-ca596a2cac69.png)

Now click the green button in the screenshot above. We're going to make a **pull request**!

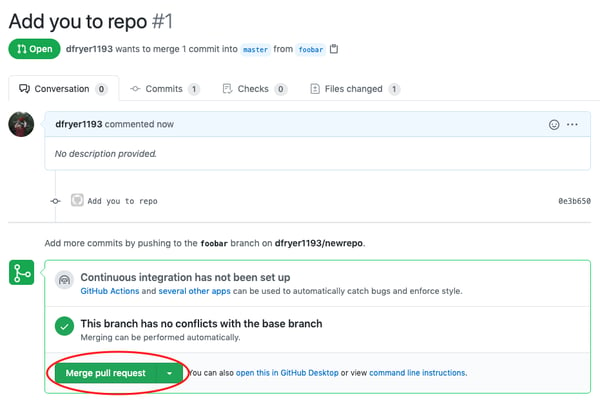
**Step 8: Create a pull request (PR)**

A pull request (or PR) is a way to alert a repo's owners that you want to make some changes to their code. It allows them to review the code and make sure it looks good before putting your changes on the primary branch.

This is what the PR page looks like before you've submitted it:

[](https://cloud.githubusercontent.com/assets/5241432/9189500/4688c07e-3fb7-11e5-99ed-d75b50ed9e48.png)

And this is what it looks like once you've submitted the PR request:

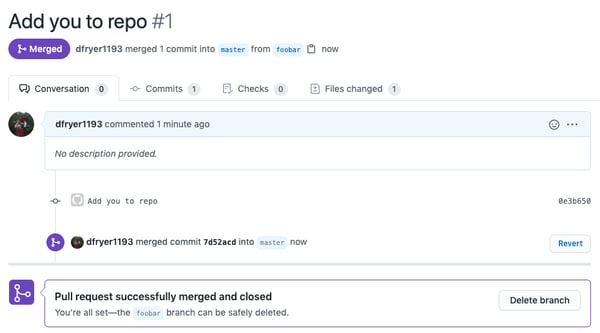
[](https://cloud.githubusercontent.com/assets/5241432/9189528/b39a7176-3fb7-11e5-87b1-7fed3e63b6bb.png)

You might see a big green button at the bottom that says 'Merge pull request'. Clicking this means you'll merge your changes into the primary branch..

Sometimes you'll be a co-owner or the sole owner of a repo, in which case you may not need to create a PR to merge your changes. However, it's still a good idea to make one so you can keep a more complete history of your updates and to make sure you always create a new branch when making changes.

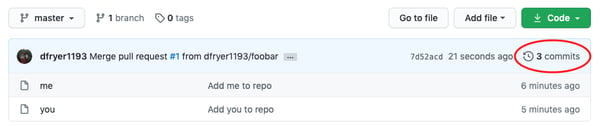
**Step 9: Merge a PR**

Go ahead and click the green 'Merge pull request' button. This will merge your changes into the primary branch.

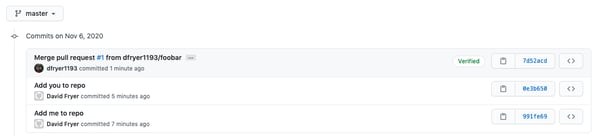
[](https://cloud.githubusercontent.com/assets/5241432/9189587/76631d98-3fb8-11e5-9fdb-17e7dec1c2a4.png)

When you're done, I recommend deleting your branch (too many branches can become messy), so hit that grey 'Delete branch' button as well.

You can double check that your commits were merged by clicking on the 'Commits' link on the first page of your new repo.



This will show you a list of all the commits in that branch. You can see the one I just merged right up top (Merge pull request #1).



You can also see the [hash code](https://git-scm.com/docs/git-hash-object) of the commit on the right hand side. A hash code is a unique identifier for that specific commit. It's useful for referring to specific commits and when undoing changes (use the [git revert](http://git-scm.com/docs/git-revert)<hash code number> command to backtrack).

**Step 10: Get changes on GitHub back to your computer**

Right now, the repo on GitHub looks a little different than what you have on your local machine. For example, the commit you made in your branch and merged into the primary branch doesn't exist in the primary branch on your local machine.

In order to get the most recent changes that you or others have merged on GitHub, use the git pull origin master command (when working on the primary branch). In most cases, this can be shortened to “git pull”.

mnelson:myproject mnelson$ git pull origin master

remote: Counting objects: 1, done.

remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (1/1), done.

From https://github.com/cubeton/mynewrepository

\* branch master -> FETCH\_HEAD

b345d9a..5381b7c master -> origin/master

Merge made by the 'recursive' strategy.

mnelson.txt | 1 +

1 file changed, 1 insertion(+)

[**view raw**](https://gist.github.com/cubeton/48b5c726b496d50c3975/raw/fe2c68e0988c467fd218587e2397552076355b52/pulloriginmaster.md)[**pulloriginmaster.md**](https://gist.github.com/cubeton/48b5c726b496d50c3975#file-pulloriginmaster-md)hosted with ❤ by [**GitHub**](https://github.com/)

This shows you all the files that have changed and how they've changed.

Now we can use the [git log](http://git-scm.com/docs/git-log) command again to see all new commits.

(You may need to switch branches back to the primary branch. You can do that using the git checkout master command.)

mnelson:myproject mnelson$ git log

commit 3e270876db0e5ffd3e9bfc5edede89b64b83812c

Merge: 4f1cb17 5381b7c

Author: Meghan Nelson <mnelson@hubspot.com>

Date: Fri Sep 11 17:48:11 2015 -0400

Merge branch 'master' of https://github.com/cubeton/mynewrepository

commit 4f1cb1798b6e6890da797f98383e6337df577c2a

Author: Meghan Nelson <mnelson@hubspot.com>

Date: Fri Sep 11 17:48:00 2015 -0400

added a new file

commit 5381b7c53212ca92151c743b4ed7dde07d9be3ce

Merge: b345d9a 1e8dc08

Author: Meghan Nelson <meghan@meghan.net>

Date: Fri Sep 11 17:43:22 2015 -0400

Merge pull request #2 from cubeton/my-newbranch

Added some more text to my file

commit 1e8dc0830b4db8c93efd80479ea886264768520c

Author: Meghan Nelson <mnelson@hubspot.com>

Date: Fri Sep 11 17:06:05 2015 -0400

Added some more text to my file

commit b345d9a25353037afdeaa9fcaf9f330effd157f1

Author: Meghan Nelson <mnelson@hubspot.com>

Date: Thu Sep 10 17:42:15 2015 -0400

This is my first commit!

**Step 11: Bask in your git glory**

You've successfully made a PR and merged your code to the primary branch. Congratulations! If you'd like to dive deeper, check out these more advanced tutorials and resources: