GitHub in 10 Simple Steps

1. Get a GitHub account if you don't have one yet. Go to:
   https://github.com/
   Sign up.
   GitHub is all free unless you want some special features, such as hiding your work from the world. (All free GitHub repos are public.)

Note: Think about your username before you choose it. This is a professional space. Your username will be visible to strangers.

You can have only ONE GitHub account per email address.

2. Download GitHub for Mac or GitHub for Windows:
   http://mac.github.com/
   http://windows.github.com/

   Install it. I will refer to this as "the app."

3. Go to the Help section at that website (Mac or Windows, above) and open the "Getting Started" or "First Launch" link there.

   Using the instructions as needed, LINK your GitHub account to the app (GitHub for Mac or GitHub for Windows).

Time for some thinking: GitHub expects things to be stable on your hard drive(s). That means you need to decide very deliberately WHERE on your hard drive a folder associated with GitHub will be. DO NOT associate GitHub with folders on your Desktop! Your GitHub app is going to coordinate things for you between the Web (GitHub) and your hard drive, but GitHub can't think, and GitHub can't figure out that you have moved a folder or changed its name.

One possible way to organize your work is to create ONE folder that will contain all other folders that you will associate with GitHub. You might name that folder Webwork, or Datawork, or Code, or some other general term. Make that folder NOW, inside Documents or My Documents, depending on your OS.

4. Make the folder that will hold all your GitHub work, as explained above.

To get you started, we will **fork** and then **clone** a **repo**. (Those GitHub words will all be defined as we go along.)

5. Go to this repo:
   https://github.com/macloo/html_css_templates

   "Repo" means "repository." You see one repo when you go to the URL above. A repo can contain lots of files and folders, or few, or none (that would be an empty repo). Each GitHub user can have unlimited public repos. You can have as many as you want.

6. Fork that repo.
   Find the button that says **Fork** (near the top of the page, right side). Click it.
   This makes a new, independent copy of that entire repo under YOUR GitHub username.

   Fork: Your repo will not be changed when the original owner changes the original. Likewise, you can change your copy, and it will not affect the original. You will only fork ONCE for a repo.

   The repo does not exist on your computer YET.

7. Clone that repo.
   Find the button that says **Clone in Desktop** (middle of the page, right side). Click it.
   You will be asked to choose a place on your hard drive for this repo. Put it inside the folder you made in step 4.

   Clone uses your app (GitHub for Mac or GitHub for Windows) to make a copy of the entire repo on your hard drive, and this copy is tethered to the version on the GitHub.com site (as you will see). You might clone a repo more than once.

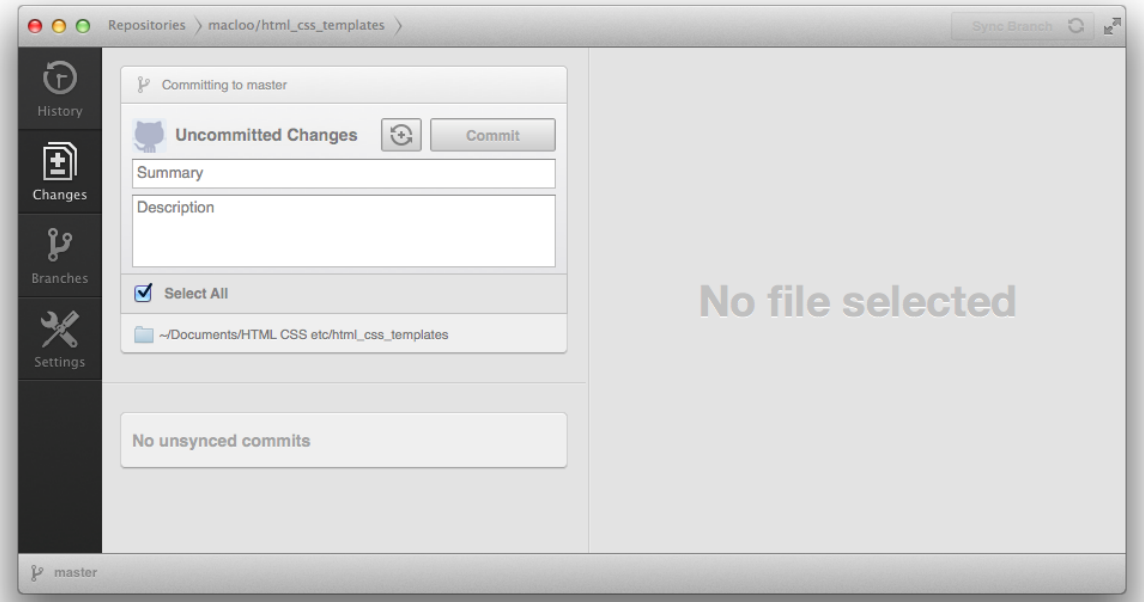
What you should have now: At GitHub.com, when you go to your user page/profile, you'll see a link to your new repo (click it to view the repo). On your hard drive, you'll see a folder with the same name as that repo. (These two are tethered together, thanks to Git.[1])

- *My* user page is: https://github.com/macloo
- Yours will be similar, but with your username instead of *macloo*.
- The name of the repo you forked and cloned is: *html_css_templates*
- That name is both the name of your new repo AND the folder name on your hard drive.

8. Go to your app (GitHub for Mac or GitHub for Windows) now.
   Click the name of the repo (*html_css_templates*) in the app.

   You should see something very similar to this (next page):

[1] Git is a version control system (VCS). Git can be run at the command line, but you have an app, so you don't need to do that. GitHub uses Git to manage and sync your files.



Look at the four buttons on the left. If your app looks different, click the **Changes** button there. Then it should look like the one above.

If you want to return to the list of all the repos on your hard drive, click the word **Repositories** on the top edge of that window.

Notice the word **master** in the lower left corner. This tells you which **branch** you are in. This will be important later.

Click each button on the left side to have a look. Then return to the view you see above (Changes).

9. Change two things in your repo, and then **commit** and **sync**.
   To edit files or create new files in your repo, you use the normal software tools you would use in any other circumstances. The files in this repo are all HTML and CSS files (except for README.md, which is a Markdown[2] file), so use whatever you use to edit or create regular HTML documents.

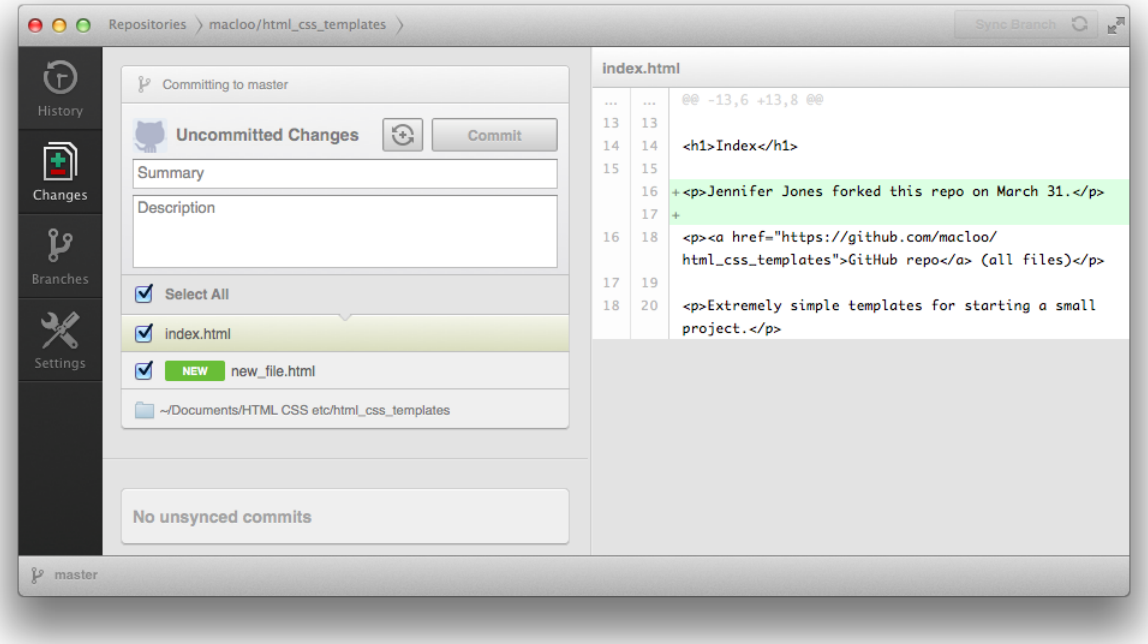
   a) Open the file *index.html* for editing. (Use your usual editing program.)
   Add a new paragraph below the top H1 heading. Write a full sentence that includes *your own name.* Example:
   `<p>Jennifer Jones forked this repo on March 31.</p>`

[2] See http://support.mashery.com/docs/read/customizing_your_portal/Markdown_Cheat_Sheet to find out how to write documents with Markdown. See also http://en.wikipedia.org/wiki/Markdown

Save and close the file.

b) Open the file *template.html* and SAVE AS with the new name *new_file.html.*
Then change the text in the paragraph to a full sentence that includes *your own name.*
Save and close the file *new_file.html*.

c) Go back to the GitHub app.



Note how the app shows exactly what you have changed or added.

d) **Commit** the changes.
GitHub requires you to write something in the field that (now) says Summary. Usually you would write something (brief) to describe what you have changed or added.
Click the **Commit** button.

e) **Sync** your local repo with your repo on GitHub.
There's a button at the top edge of that window, on the right, labeled **Sync Branch**. Click it to upload all the changes to GitHub.com. (Note: You can change the **Commit** button to say "Commit & Sync" by clicking the square button beside it, which also changes it back—it toggles. Some people like to combine the two steps into one.)

f) Go to GitHub and see your changes there.

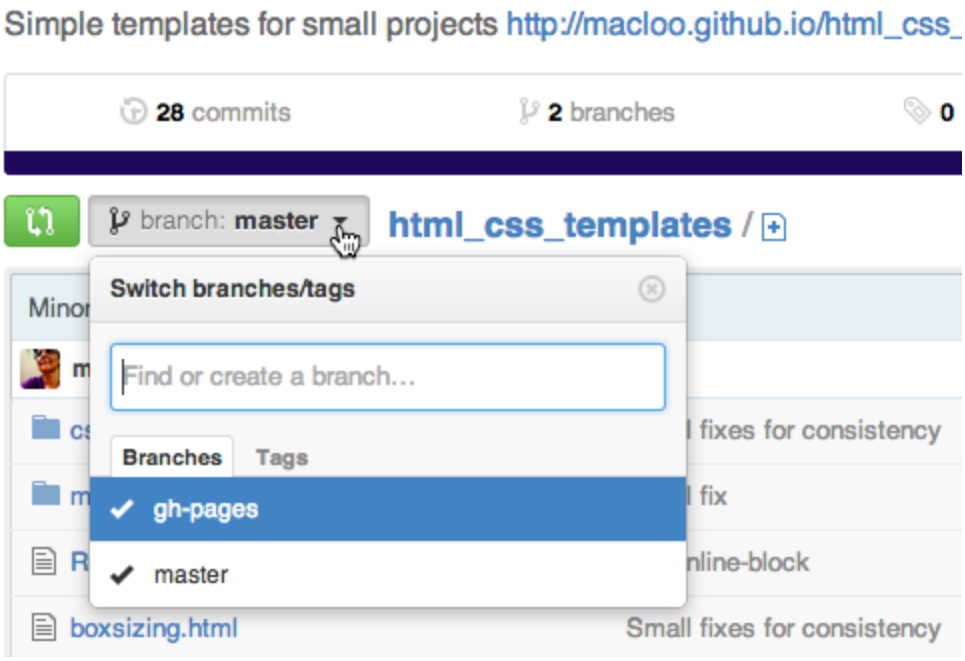
10. Use GitHub Pages to publish your site as Web pages.

If you're happy just to be able to save and share your files on GitHub, you can stop here. But if you want to use GitHub as a publishing platform, this step shows you how to do that. Admittedly, this is a LONG step. It will help you understand **branches**.

Go to GitHub.com and look at your repo for *html_css_templates*. Near the top of the repo page, you'll see this text:

Simple templates for small projects http://macloo.github.io/html_css_templates/ — Edit

That URL points to *my* pages (at the original repo, the one you forked), and you cannot change those. (Click the link at GitHub and see the live Web pages.)
Your version, your repo, also has these files.
The files are in a separate **branch** in your repo.
That branch is named *gh-pages*. To create a website on GitHub, you create a branch with that exact name. All HTML files in that branch will be live Web pages.
To change to any different branch (or to create a new one), open the menu on GitHub, as shown below.



(Why branches? Teams of people who are writing code together often create separate **branches** for different parts of a coding project. They can all work together in one repo, but each person is working in a branch that's isolated from the *master* **branch**. When the side project is finished, the programmer **merges** her branch into *master*.)

a) In your repo, change to the **branch** *gh-pages*. Open the menu and click *gh-pages*.

- Look for your file *new_file.html*. (It's not there.)
- Click to open *index.html* on GitHub. Your new paragraph is not there.

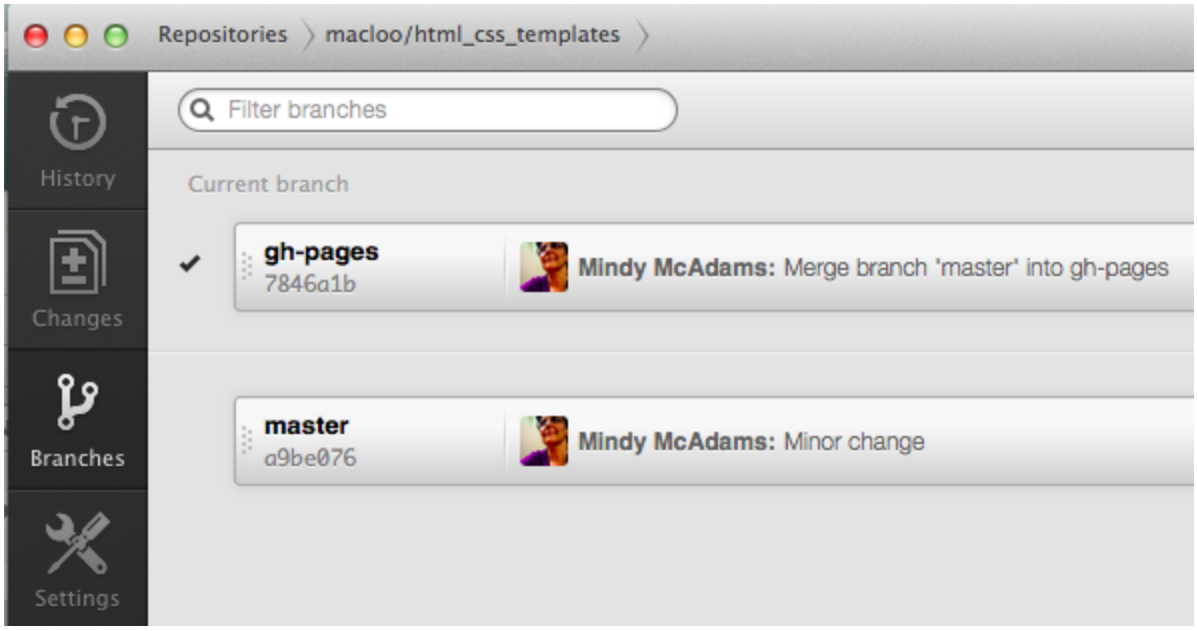
This illustrates how branches stay separated from one another until you deliberately **merge** them.

b) How to **merge** branches.

1. Go to your app.
2. Click the left-side button **Branches**.
3. Click the button **Merge View** (top right).
4. In the gray bar that has opened, there is a left box and a right box. Drag the branch that has your new changes (*master*) into the left box.
5. Drag the unchanged branch (*gh-pages*) into the right box.
6. Click **Merge Branches**.
7. Click **Sync Branch**. (Nothing is uploaded to GitHub unless you do this.)
8. Check GitHub.com. You should see your changes now in the *gh-pages* branch. In other words, all your changes to *master* have now been copied to *gh-pages*.

If you don't see the changes in *gh-pages,* read below. You might need to change the checkmark — which means changing the current branch. Then **Sync Branch** (again).

NOTE: When you are in the **Merge** panel in your app, you change the contents of the folder on your hard drive also. Look for the checkmark below. When that checkmark is on *gh-pages,* anything you change on your hard drive in your repo will be changed ONLY in that branch. Make sure you know WHICH branch you are working on!



To change the current branch, move the checkmark! How? Double-click the name of the (other) branch, and the checkmark will go there.

Note: You can keep your *master* and your *gh-pages* branches totally separate and never merge them. Why would you do that? Maybe your *master* contains a Python library (which you are sharing with the world—good for you!), and your *gh-pages* contains a tutorial for how to use that Python library. The two branches would be completely different.

Publish your GitHub repo as Web pages

All the files in *gh-pages* will automatically become live Web pages. All you need to do is create a branch named *gh-pages* and put files into it. Any time you **sync** the *gh-pages* branch, those changes will automatically be live on the Web. The URLs will be like this:

http://yourname.github.io/repo_name/filename.html

However, files in the *gh-pages* branch of any **forked repo** are NOT IMMEDIATELY generated as live Web pages under your username. The pages *will* all be generated as soon as you **push** a change to *gh-pages*. That means you change something, anything, in *gh-pages,* and **commit,** and **sync.** This causes GitHub to generate pages from a repo that was forked.

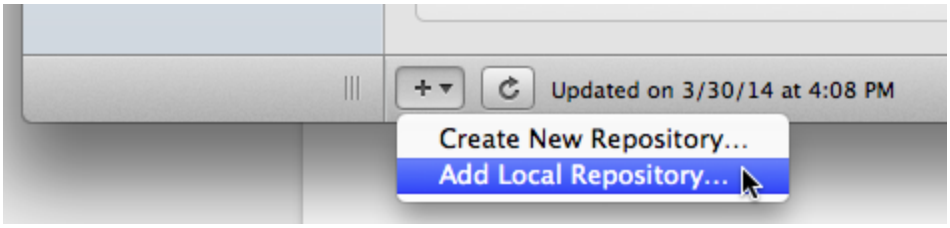
Note: If you just want to do everything directly in *gh-pages,* you can make *gh-pages* the default branch. How?

1. Click **Settings** on the far right side of your repo page.
2. This takes you to a new Settings page.
3. Find the menu button there labeled **Default Branch**.
4. Open that menu, select *gh-pages*.
5. There's no Save button, so as soon as you select this, it's done.

Bonus: How to start a new website from scratch

1. Create a new folder *inside* that folder you created on your hard drive to contain all your GitHub repos.
2. Name the folder with a short, Web-friendly name, such as *web_test*.
3. Copy the following 3 files from *html_css_templates*:
   * template.html
   * reset.css
   * styles.css
4. Paste the 3 files into your new folder.

5. Go to your GitHub app. (The app, not the site.)
6. Go to the list of all the repos on your hard drive (if you don't know how, see page 3 in this tutorial).
7. Bottom edge, Plus (+) menu, click this:



8. Find that new folder you just added 3 files to. Follow the prompts. You'll have a brand-new repo on GitHub.com in almost no time!

   a) "This folder is not a repository": **Create and Add**
   b) Type "First commit" and click **Commit**
   c) Button, top right: **Push to GitHub**
   d) Keep the folder name. **Push Repository**
   e) View your new repo at GitHub.com
   f) Add new branch: *gh-pages*
   g) View on Web

Your new *gh-pages* is not on your hard drive yet. **Clone** it at GitHub.com, and it will appear in your app, on the **Branches** panel (see page 6 in this tutorial). Use your app to switch between branches.

Working on more than one computer

If by chance you keep local repos on more than one computer (hmm, is this a bad idea?), you can easily make the local one match the repo on GitHub.com by syncing in the app. This will transfer newer changes on GitHub down to your local hard drive.

Resources

GitHub Pages main site
http://pages.github.com/

GitHub Pages (instructions, FAQ)
https://help.github.com/categories/20/articles

GitHub Help (general)

https://help.github.com/