

GitHub Part II: Building a Static Website

Nicole Cote, Spring 2023

Introductions

What We'll Cover

- PreReqs/Installation Guide Reminder
- Establishing Background (incl. a conceptual intro to static sites, examples, & etc.)
- Building A Site—The Basics (incl. a workflow for building and editing, file basics, and Markdown)
- Bringing Your Site to Life (incl. workflows for publishing, editing pages, and getting started on your own)
- Expanding Your Knowledge: (incl. additional topical resources and where to go next)
- GCDI Resources: (incl. additional GCDI resources and opportunities to help you grow your skills)

Installation Guide Reminder:

If you wish to follow along in an applied way, please be sure you have already completed the steps in the Installation Guide.

The installations are a pre-req for completing the tasks on your own computer.

NOTE: It is perfectly fine to just hang out, follow-along with what's on my screen, see if you're interested, and take a few notes. These applied concepts can always be attempted after, if preferred!

And, all of these resources will be available to you, and have been designed to facilitate asynchronous engagement as well.

Establishing Background

GitHub and Jekyll

The “Whats” and “Whys” of Static Sites

Real World Examples

Workshop Sample Site

Quick Summary

Building A Site — The Basics

**What Are All These Files and
Folders?**

A Quick Overview of Some Key Files and Folders

- **_config.yml** this YAML file controls your whole site level information and some structure (including the nav bar & footer!) You'll need to update some key information here. Tip: remember for this YAML file, you cannot just refresh the page after editing, but have to re-run the local server. Don't rename this file.
- **_posts** this is a folder that houses any blog posts you want to include. Our blog posts are Markdown files. Ensure that you maintain the naming convention and structure of the files. Don't rename the folder.
- **_site** this is where your site is built by Jekyll and turned into HTML files. We won't be touching this folder. Don't rename the folder.
- **about.md** is a sample about page for your site. You'll want to edit this page! Tip: duplicate this file to make new pages for your site! Make new names for your new pages. But, ensure you retain the format.
- **index.md** is a Markdown file that controls the homepage information. You'll want to edit this! But, don't change the filename. Like any .md files, we will use Markdown syntax.
- **Gemfile/Gemfile.lock** manages extensions. We won't be touching these files today. Don't rename them!

Markdown Basics

Markdown Basics

You'll need to write your posts and pages (.md files) with Markdown. For more advanced styling, refer to the [Markdown Guide](#) website.

Headings	# H1 ## H2 ### H3
Emphasis	bold text or <i>italicize text</i> or <i>bold and italicize text</i>
Unordered Lists (i.e. bulleted lists)	<ul style="list-style-type: none">- Some Point- Some Other Point
Ordered Lists (i.e. numbered lists)	<ol style="list-style-type: none">1. First Point2. Second Point
Links	[words that will display](https://www.some-link.com)
Images	![alt text](my-image.jpg) or ![alt text](images/my-image.jpg)
Horizontal Rules	*****

Let's See An Example Workflow

Basic Workflow for Making/Editing Static Sites

IMPORTANT: this workflow is to be used *after* you have done the installations. If you haven't yet done that, please refer to the [Installation Guide](#)

Part 1: Building the Basic Site

1. open the Terminal
2. cd to your chosen directory
3. \$ jekyll new BlogName
4. \$ cd BlogName

Part 2: Viewing and Editing the Site

5. \$ bundle exec jekyll serve
6. in your browser go to: <http://localhost:4000> (this is where you will preview your site! Use control-C in the Terminal to stop). A tip: do check where your Terminal tells you the server is located, as this will change throughout the process (i.e. Server address: <http://127.0.0.1:4000/>)
7. make some changes to the Markdown/YAML files
8. EITHER refresh the webpage (for Markdown files) OR redo step 5 & 6 (for YAML files)
9. view your new edits!
10. rinse and repeat until you have a site you <3

**Friendly Reminder: Don't type
the \$ signs!**

Any Qs?

Bringing Your Site to Life

Next Steps: Part 1

1. If you haven't already, build a site locally by following the [Basic Workflow for Making/Editing Static Sites](#)
2. Next, edit/continue editing your site per the above workflow
3. For tips and a working example, I would recommend you refer to the Sample Site made for this workshop, where provided instructions explain how to see exactly what's been done to that site
4. Optional: if you want to provide an overview to the repository on GitHub create a README.md file

Next Steps: Part 2

If you have a site you like, try to publish it on GitHub

For instructions, see one of the Basic Workflows for Publishing Static Sites:

[Option 1: with the Command Line](#)

[Option 2: with GitHub Desktop](#)

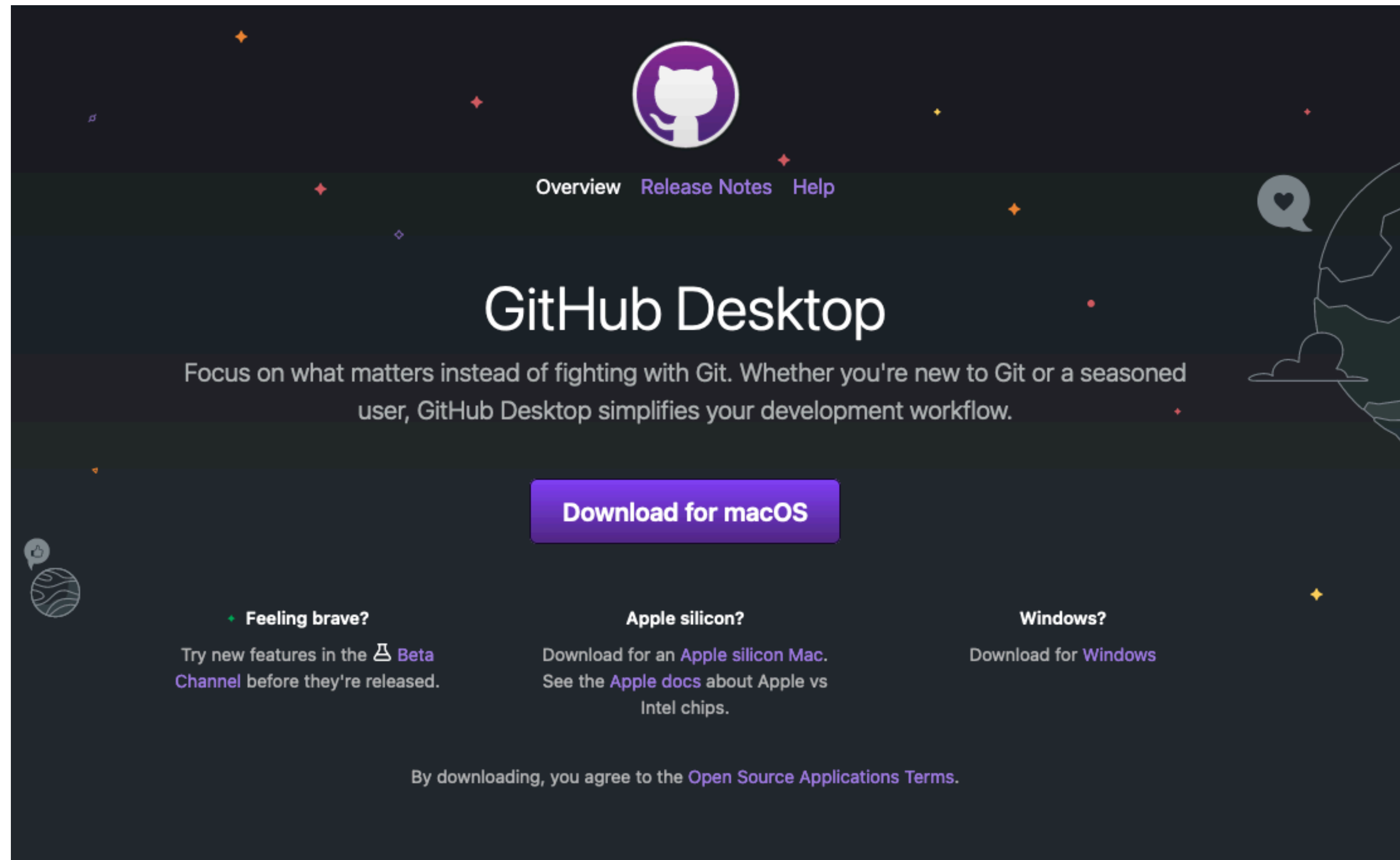
Publish Your Site

Basic Workflow for Publishing Static Sites: Option 1 (Command Line)

Note: this workflow is to be used *after* you have the bones of your initial site/want it publicly viewable.

1. Create a repository on [GitHub.com](https://github.com) for your site (name it the same as your local directory BlogName and don't initialize with a README)
2. Open Terminal and cd to your project directory
3. `$ git init` (transforms your current directory to a git repository)
4. `$ git add .` (stages all of the files—don't forget the dot)
5. `$ git commit -m 'write message'`
6. On [GitHub.com](https://github.com) copy the HTTPS (aka the remote repository URL which looks something like: `https://github.com/your-username/site-repo.git`)
7. `$ git remote add origin <paste the URL you just copied>` (adds the remote ([GitHub.com](https://github.com)) repository where the local will be pushed)
8. `$ git branch -M main`
9. `$ git push -u origin main` (pushes the local changes to [GitHub.com](https://github.com))
10. Now, add your website information to your files. In your local folder: go to the `_config.yml` file.
 1. update “baseurl:” and add slash followed by your repo. (i.e.: `baseurl: /BlogName`). And update “url” (i.e.: `https://username.github.io`)
 2. test it out by running `bundle exec jekyll serve` (as we've done before). But, you will now need to navigate to the address the Terminal says
11. repeat steps 4 and 5
12. `$ git push origin main`
13. Navigate to your repo on [GitHub.com](https://github.com) (i.e. something like `https://github.com/username/BlogName`)
14. Click Settings (at the top of the page) and then click Pages (on the right of the page)
15. Set the dropdown for “Branch” to “Main” and click “Save”
16. Navigate to your site (i.e. something like: `https://username.github.io/BlogName/`)




Don't want to use the Command Line?



GitHub Desktop: <https://desktop.github.com/>

Basic Workflow for Publishing Static Sites: Option 2 (GH Desktop)

Note: this workflow is to be used *after* you have the bones of your initial site/want it publicly viewable.

1. Open the GitHub Desktop App
2. Click “Create a Repository on your Hard Drive”
3. Choose the file path for your project folder (i.e. Desktop/) and name the repo the same exact name as your project folder (i.e. BlogName)
4. Click “Create Repository”
5. Add Commit Message (i.e. “adds the initial blog site”)
6. Click “Create A New Branch” with this icon 
7. Type gh-pages in the “Name” field
8. Click “Commit to gh-pages” (on bottom of page)
9. Click Publish (on top of page)
10. Click “Fetch Origin” with this icon 
11. Navigate to your site (it will be: <https://username.github.io/BlogName/>)
12. You’ll probably see the formatting a bit skewed. To fix this you need to update the baseurl and url in the *_config.yml* file
13. Now, add your website information to your files. In your local folder: go to the *_config.yml* file.
 1. update “baseurl:” and add slash followed by your repo. (i.e.: baseurl: /BlogName). And update “url” (i.e.: *https://username.github.io*)
 2. test it out by running bundle exec jekyll serve (as we’ve done before). But, you will now need to navigate to the address the Terminal says
14. In GitHub Desktop add a new commit message explains what you did (it might just auto populate one for you) and click “Commit to gh-pages” (on bottom of page)
15. Click “Push Origin” with this icon 
16. Navigate to your site (i.e. something like: *https://username.github.io/BlogName/*) (NOTE: sometimes there is a short delay in seeing the updates, ~1-3 mins)

Any Qs?

Expanding Your Knowledge

Where to Go Next with These Tools?

- Follow along with **both** this deck and the sample site created for this workshop.
 - Specifically: start with the Installation Guide to download the prerequisites, then follow the Workflow for Making and Editing a Site, and lastly follow the Next Steps slides to edit and publish your site.
 - Note: the instructions in this slide deck (for the installations, building, and publishing the site) are numerical and intended to be completed in order.
- Check out the additional resources!
- Need more support about what you just learned? Request a follow-up individual consultation. (See more options in the GCDI Resources section)

Additional Resources

GitHub, Jekyll, and Markdown Resources

- A curated list of additional resources is available on the Resources page of our Sample Site

Git/GitHub Refresher

Common Terminal and Git Commands

- `cd` (change directory)
- `clear` (clear what's visible in Terminal)
- `ls` (list contents of folder)
- `git init` (creates git directory)
- `git add` ("stages" your changes)
- `git commit -m 'Type Message'` (commits your changes with a message < 50 characters)
- `git status` (shows current status of your repo)
- `git help` (shows the help feature)
- `git diff` (shows changes)
- `git push origin main` (pushes local changes to the main branch of your GitHub repo)
- `pwd` (print working directory, i.e. the folder you are in currently)

**Making Edits to Your Published
Site?**

Basic Git/GitHub Process

1. code/write something
2. open the Terminal
3. cd to your directory
4. \$ git status
5. \$ git add <filename> (for a website you'll want to add all files like this: git add .)
6. \$ git commit - m 'write message'
7. optional: do another git status (just to double check it)
8. \$ git push origin main

Any Qs?

GCDI Resources

Please take a moment to share your feedback:

cuny.is/qcdifedback

Other GCDI Offerings

Graduate Center Digital Initiatives offers support for digital scholarship, including:

- Workshops
- One-on-One Consultations
- Working Groups
- Events
- Online Resources
- Grants and Fellowships

Keep in Touch with GCDI

- Follow GCDI on Twitter: @cunygcdi
- Follow the GC Digital Fellows on Twitter: @digital_fellows
- Follow the #digitalGC hashtag on Twitter
- Follow the GC Digital Fellows blog, Tagging the Tower
- Contact the fellows at digitalfellows.commonsgc.cuny.edu/contact-us/