Making a Research Website

-

Quick Start Guide

By Yair Franco

Workflow Overview

There’s several easy ways to get a website up and running. I’ll be showing how to launch a GitHub Pages site because it’s the way I’m most familiar with.

Workflow Overview

For this guide, it'll be helpful to be a little familiar with:

- Using GitHub
- Web development coding languages (mainly HTML and CSS, though JavaScript is quite useful too)

It is possible to develop a website without needing to write your own HTML or CSS (via Google Sites, WordPress, etc.)

If you choose to do so, the next relevant section to you is "Domain Names".

However, I personally highly encourage learning how to handwrite your website as it is a valuable skill and greatly increases the flexibility of what your website can do.

Workflow Overview

- The steps shown in these slides will be:
  1. Setting up your GitHub repository and working environment
  2. Creating a sample HTML page to publish
  3. Publishing the site on GitHub Pages
  4. Obtaining a domain name and hosting the site with it

Requirements

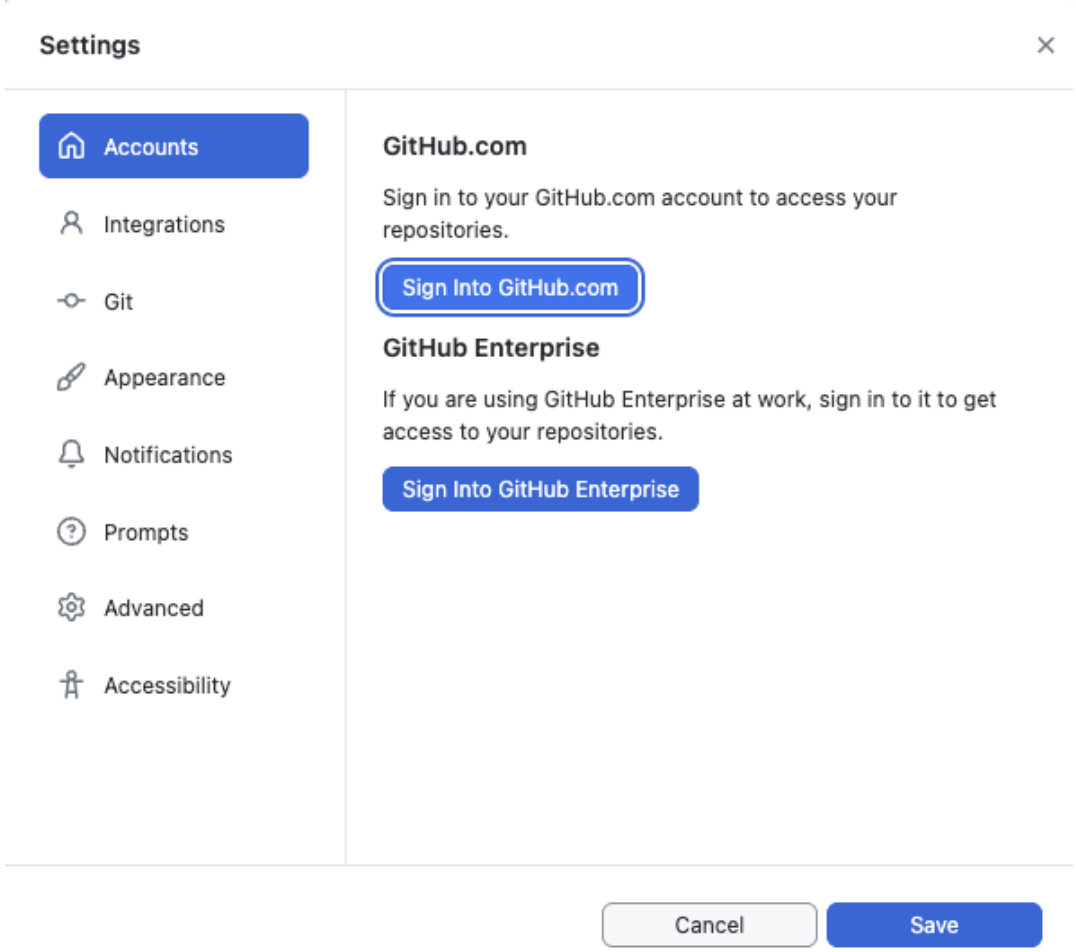
- Necessary equipment:
  - A computer with internet connection
- Necessary software:
  - GitHub desktop (technically not necessary but extremely recommended), and a GitHub account
  - Visual Studio Code (or your favorite IDE)
    - VS Code also strongly recommended because of an extension used in this Guide
  - Python
  - Your favorite internet browser (I will Google Chrome in this guide)

Setting Up Your GitHub Environment

Setting Up Your Workflow

Setting Up GitHub

- Install GitHub Desktop (GHD) on your local machine: https://desktop.github.com/download/
  - You'll also need Git, which will be installed along with GitHub if it wasn't already
- Log in to your GitHub account on GHD
  - (On Mac) GitHub Desktop > Settings > Accounts > Sign into GitHub.com

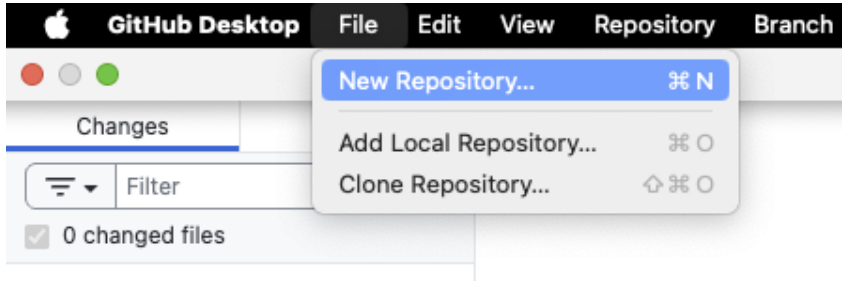


Setting Up Your Workflow

Creating a repository

This repository will be the directory for your website.

- In GitHub Desktop, go to File > New Repository...



Setting Up Your Workflow

Creating a repository

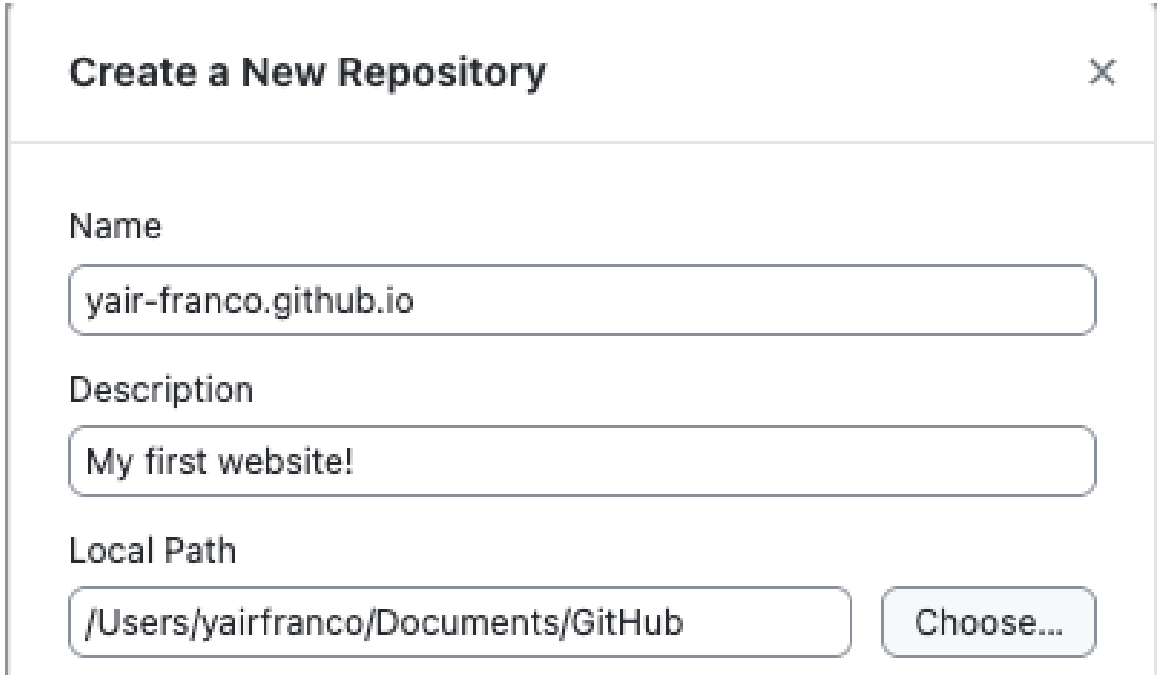
- Name your new repository

  `{your-github-username}.github.io`

  - For example, my username is ‘yair-franco’. My website repository is named

    `yair-franco.github.io`

While technically not necessary, your website repository should follow this naming convention. Otherwise, your GitHub Pages link will be very long and confusing (more on that later).



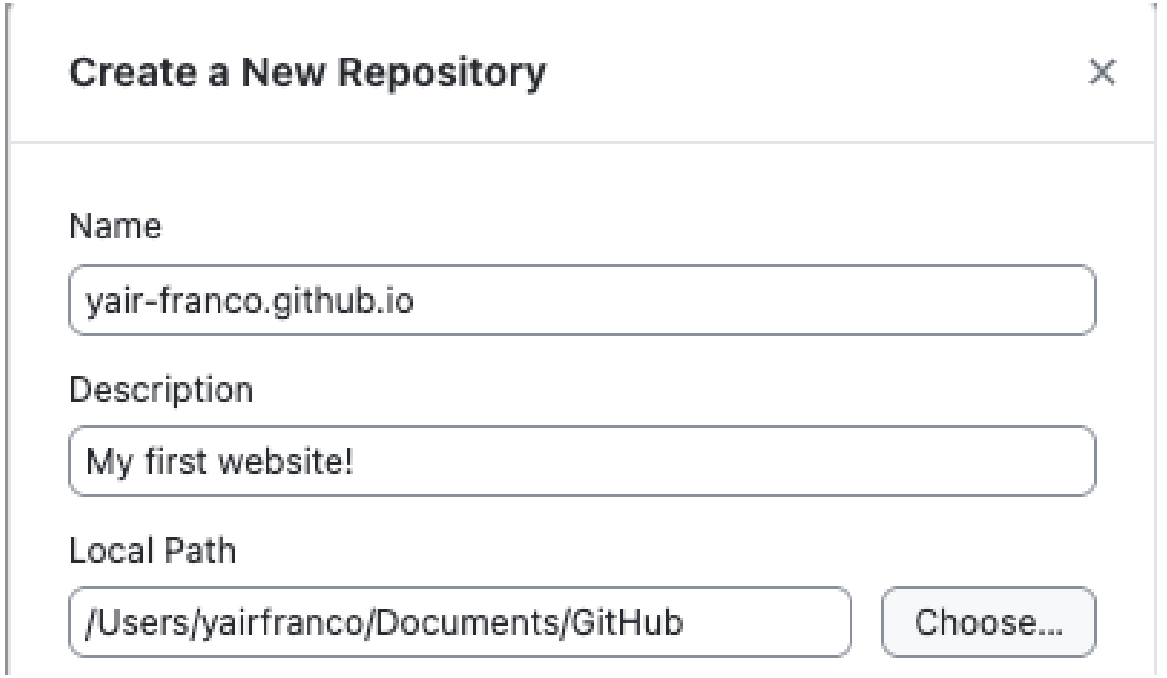
Setting Up Your Workflow

Creating a repository

- Give your repository a brief description if you like

By default, GitHub repositories are saved in a directory in your Documents folder. You can change this if you prefer.

- Choose your preferred Local Path (or keep the default)
- Click ‘Create Repository’

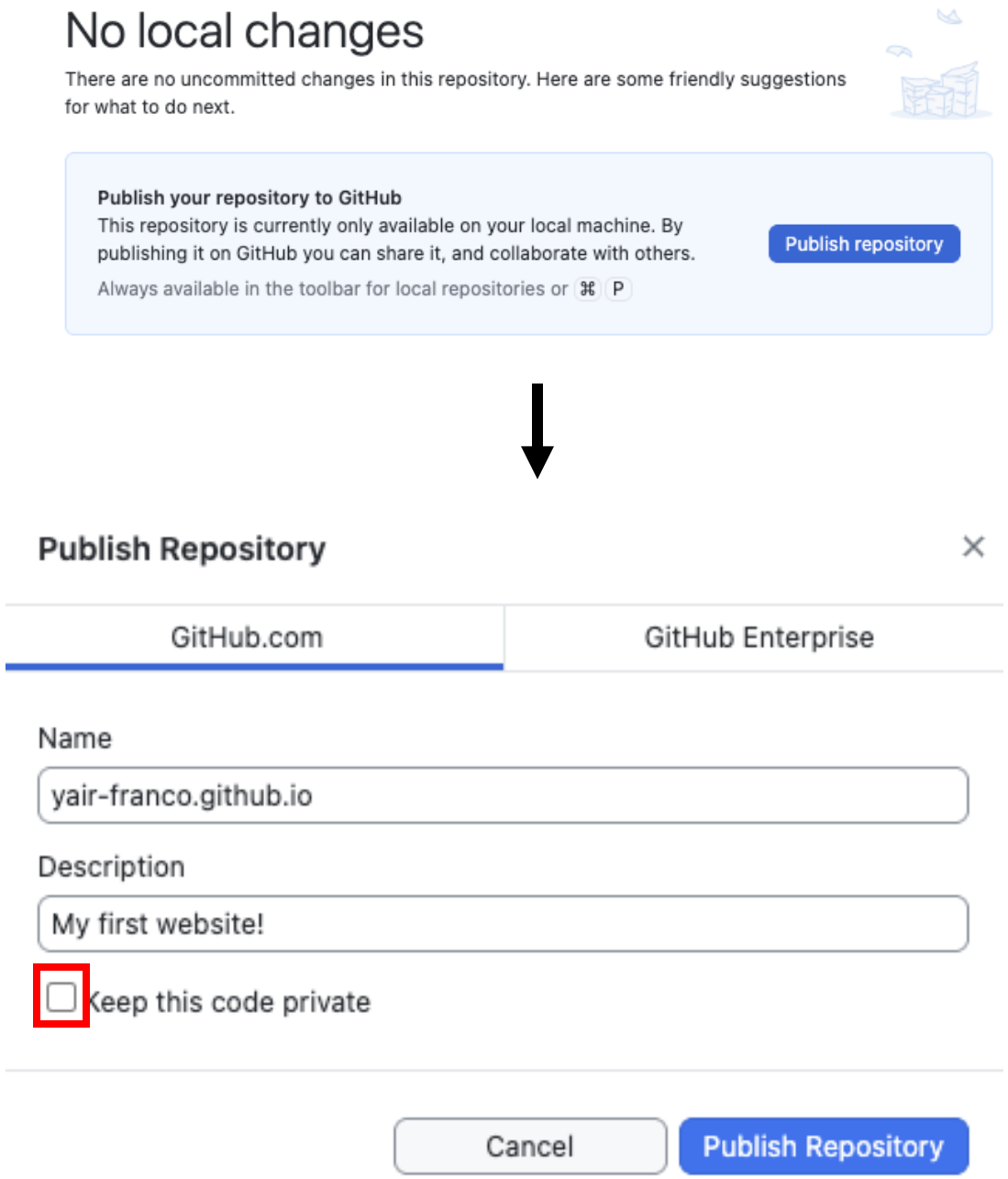


Setting Up Your Workflow

Creating a repository

Congrats! Now you have a repository. Every file related to your website will be stored here. Now we need to publish so it's on the GitHub servers. GitHub Pages are required to be published on public repositories. This just means anyone can see the code for your website on GitHub (which is also possible to do using any browser's dev tools anyways).

- Click 'Publish repository'
- Uncheck 'Keep this code private'
- Publish your repository

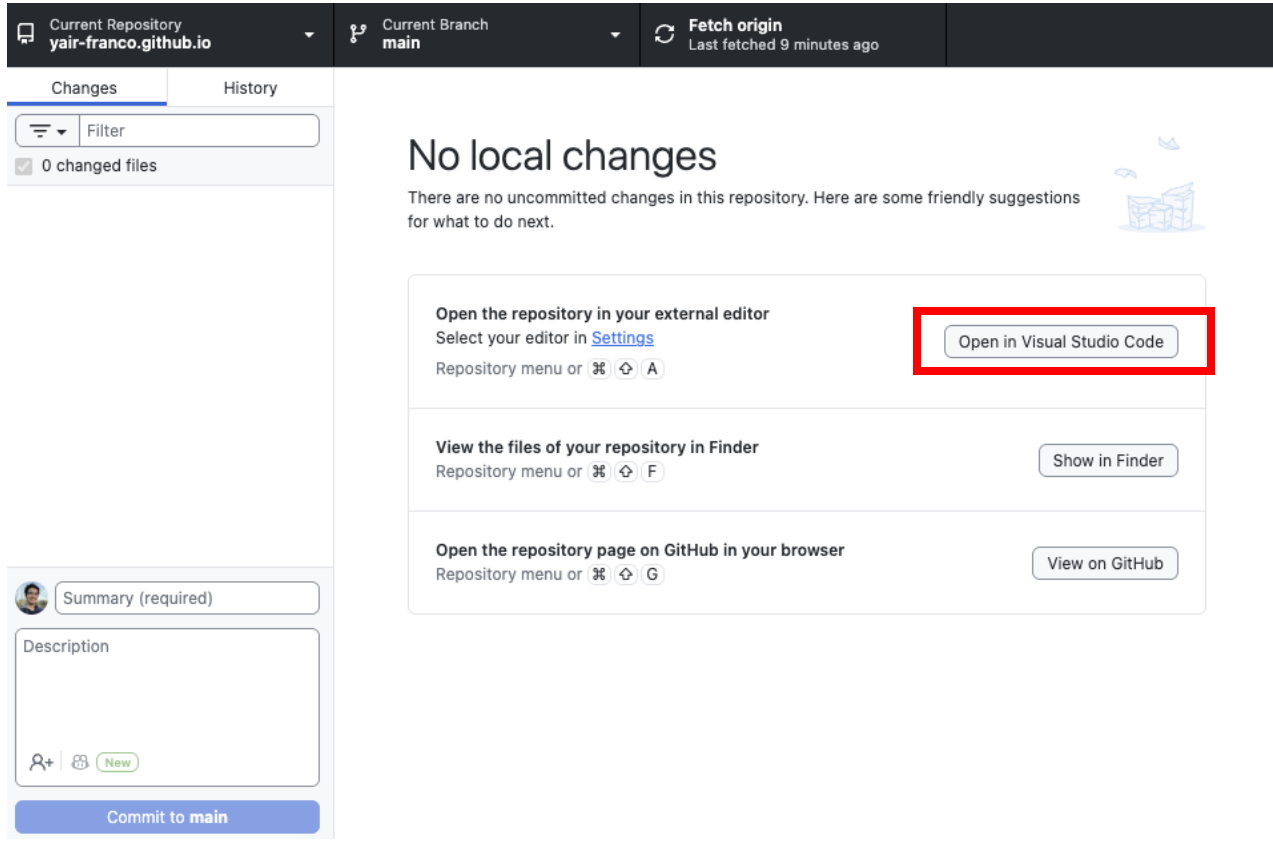


Setting Up Your Workflow

Creating your web dev environment

You should now see this screen. This will be your command center for opening your code and publishing it to GitHub. Let's make your homepage!

- Click 'Open in Visual Studio Code'
  - If GitHub detects another IDE, you might be prompted to use that one. You can change the default in your settings.

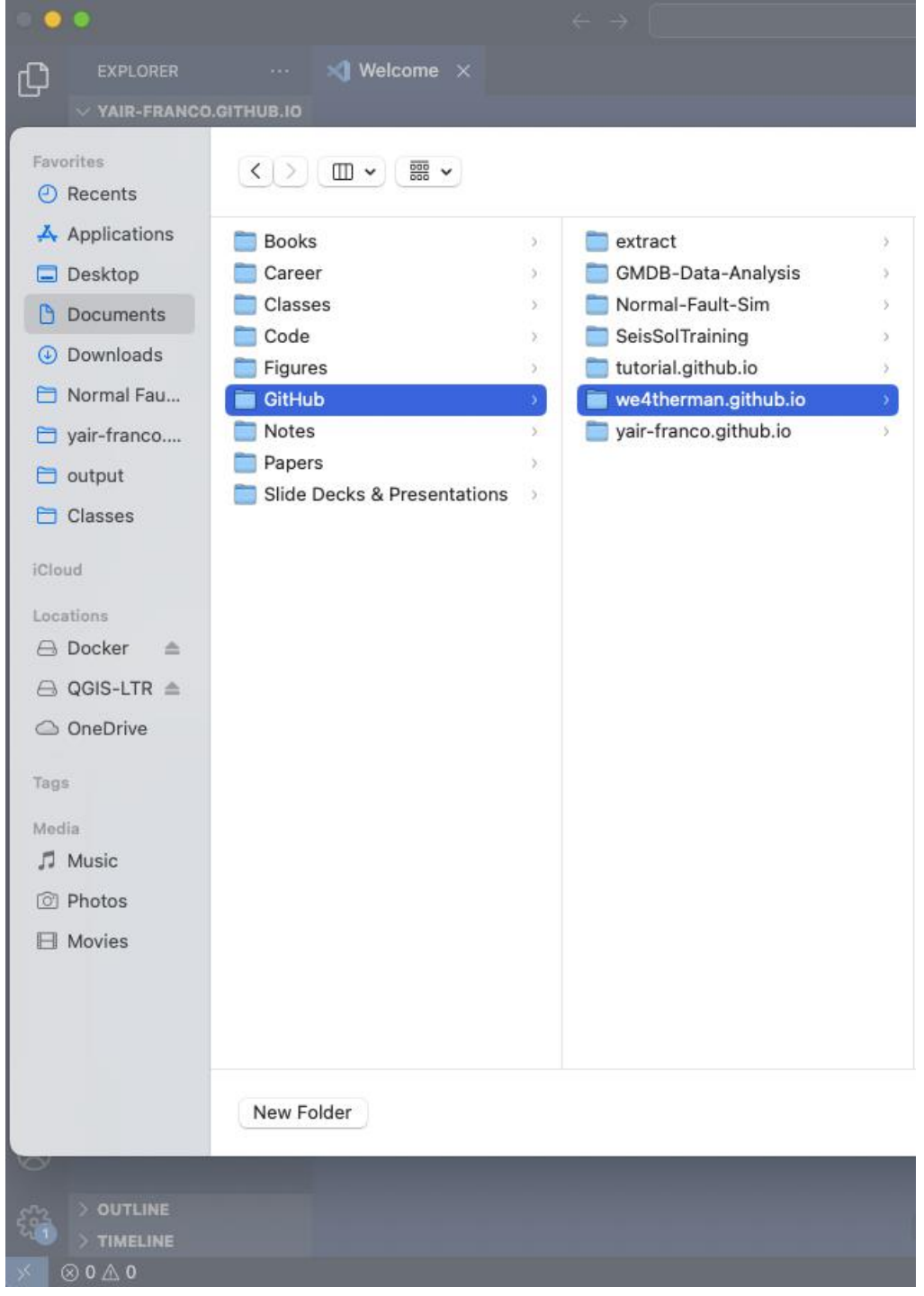


Setting Up Your Workflow

Creating your web dev environment

- In VS Code, open your new repository folder
  - File > Open Folder…
    - Navigate to your repository's path; e.g.,

Documents > GitHub > yair-franco.github.io



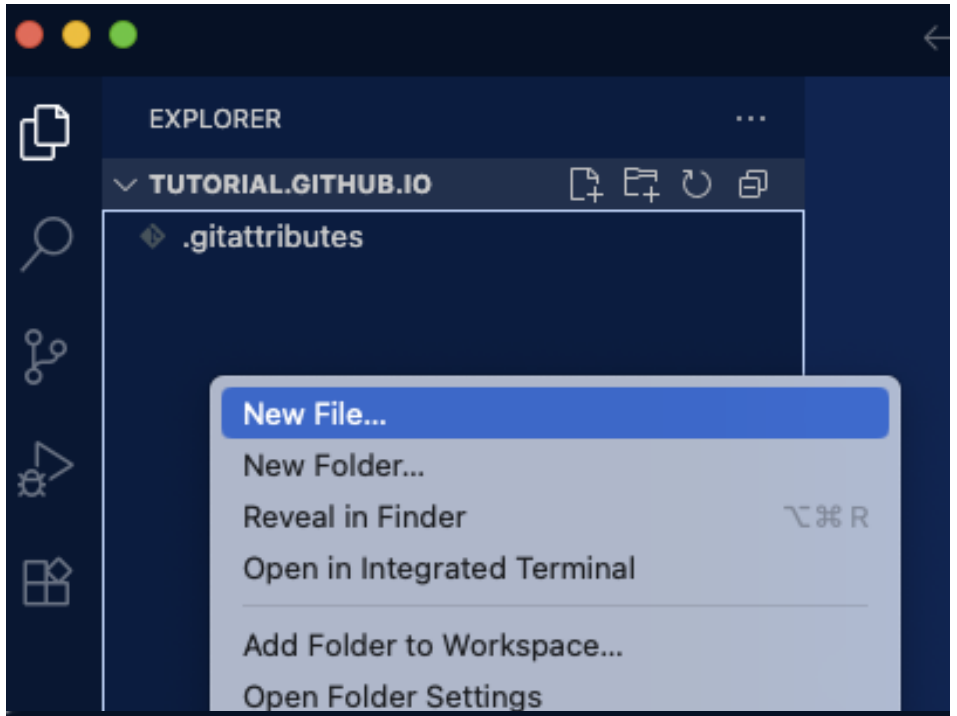
Setting Up Your Workflow

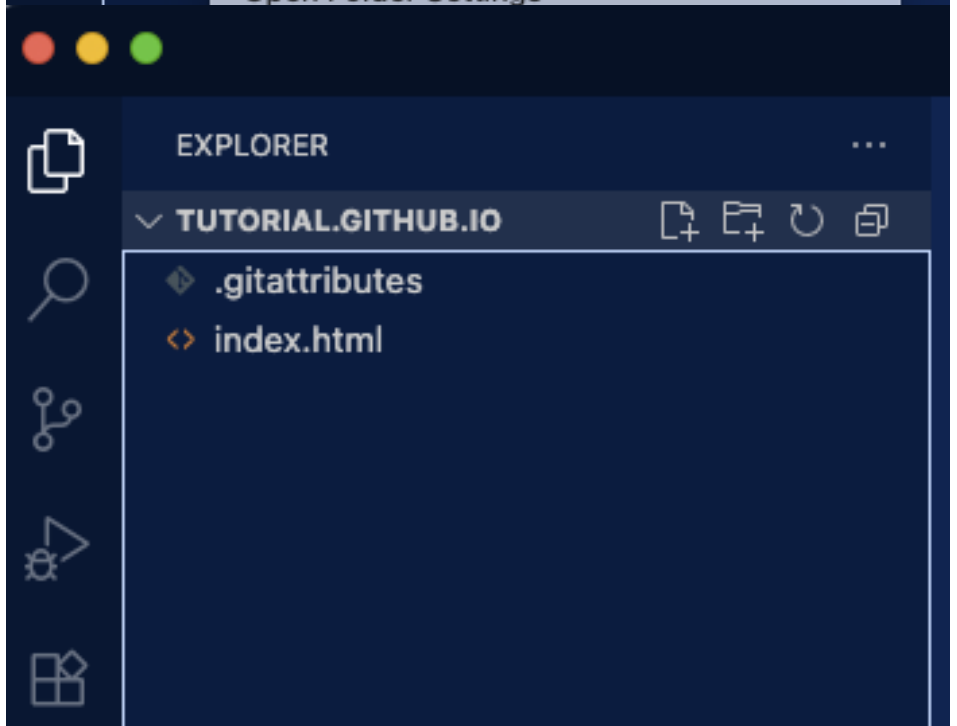
Creating your web dev environment

You should now see a project in VS Code containing a single '.gitattributes' file. We'll leave that little guy alone, it's for GitHub.

- Right click on the file explorer and create a new file
- Name the file 'index.html'

When a browser opens a file directory, it defaults to loading the 'index.html' file as a page. Otherwise, it will not know what the default page is.

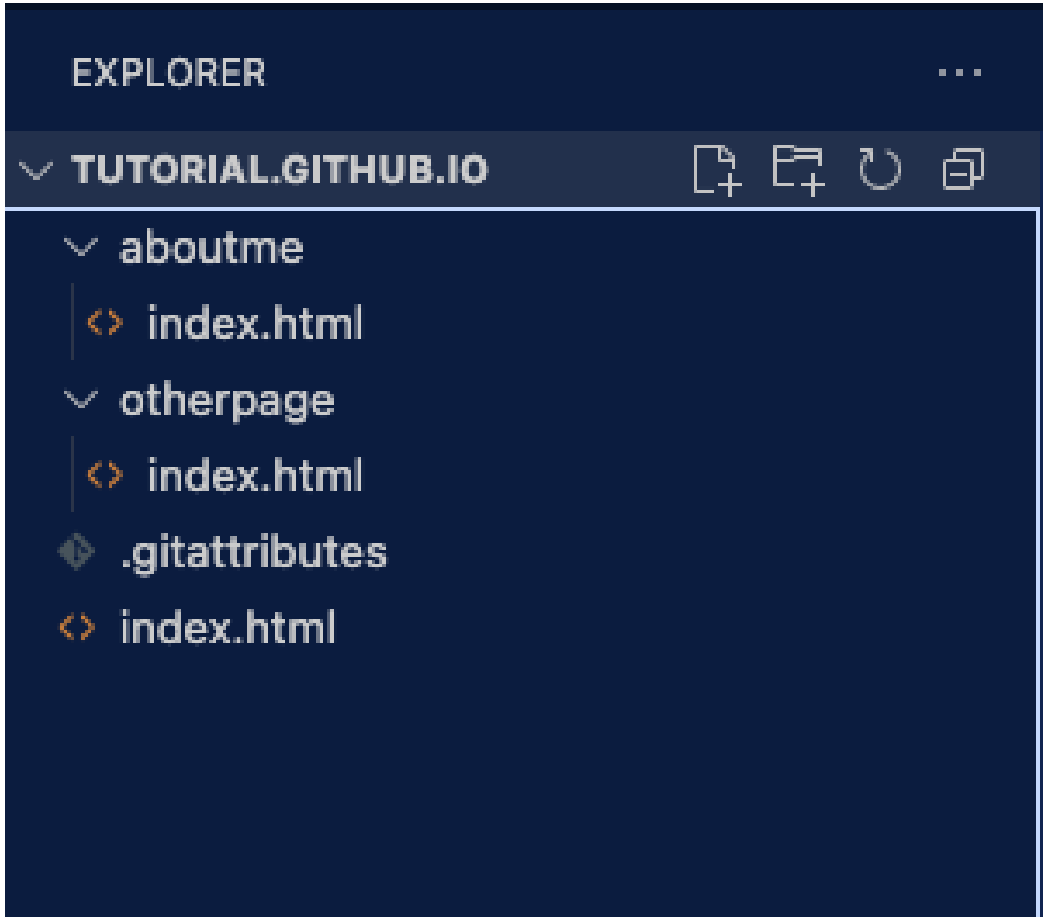




Setting Up Your Workflow

Creating your web dev environment

This ‘index.html’ file in the root of your depository ill be your homepage. Different pages on your site will be on different folders, with their own index pages.



Setting Up Your Workflow

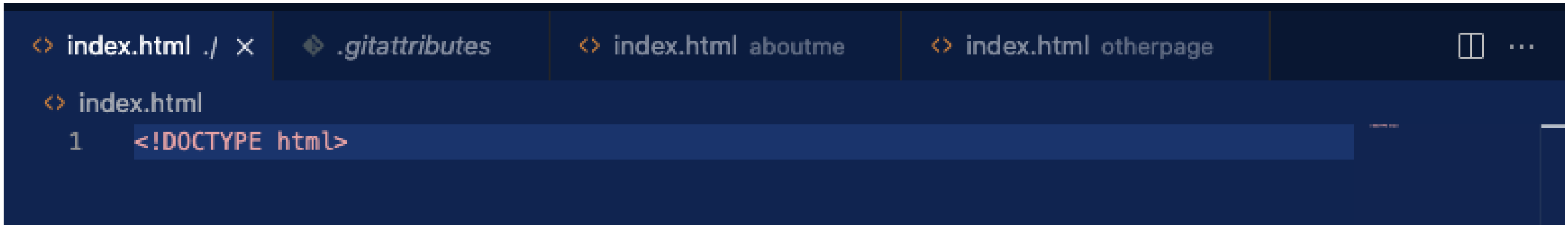
Creating your web dev environment

Let's write a really basic web page to get you familiar with your workflow.

For brevity, I will skip detailed explanations for html code. Just follow along and search up how html works if you are not familiar.

- Every good HTML file starts with

```
<!DOCTYPE html>
```



Setting Up Your Workflow

Creating your web dev environment

Let's write the head of your page. This contains the information that the browser reads about the site. Note the bracket and slash notation for opening and closing html elements.

```
<head>
  <title>{Your website's name}</title>
</head>
```

'title' is used to name the browser tab the page is on.

Setting Up Your Workflow

Creating your web dev environment

Now for the body of the site. This is what is actually displayed on the browser page.

```
<body>
  <h1>{some text}</h1>
  <p>{some text}</p>
</body>
```

‘h1’ and ‘p’ are header and paragraph tags

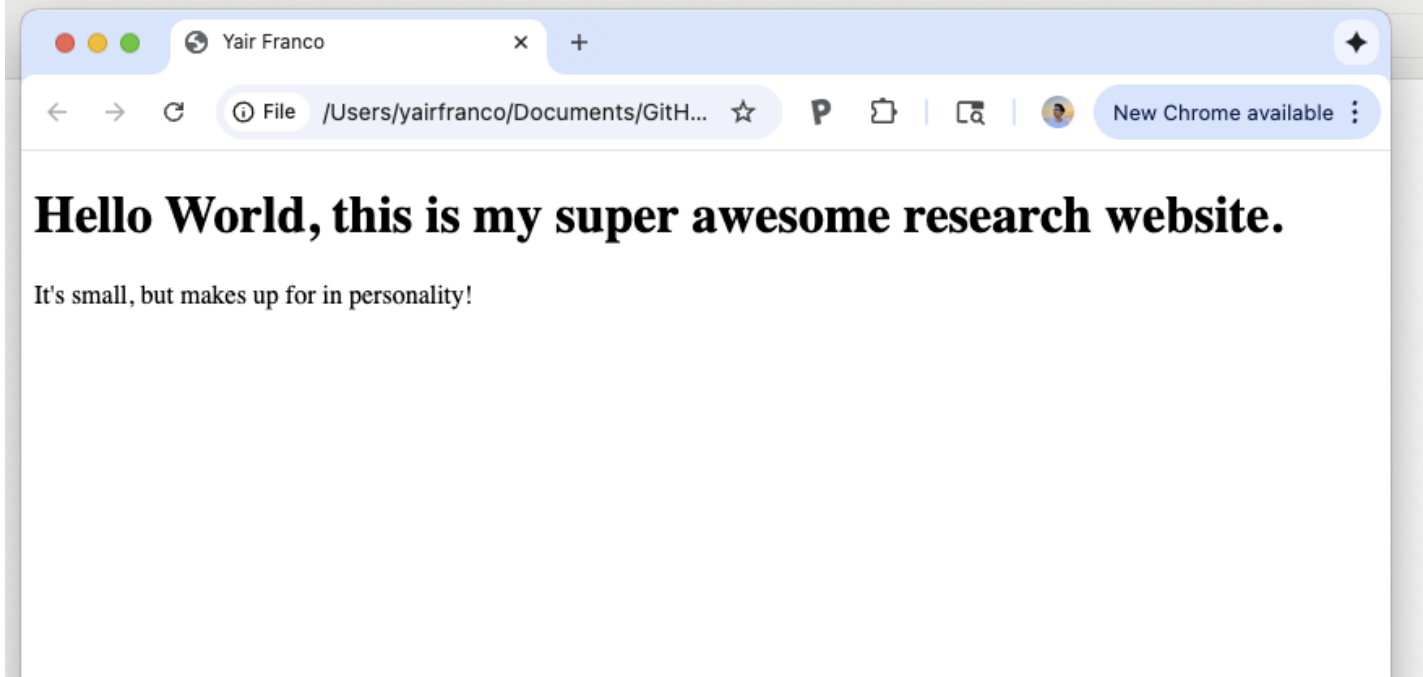
```
1   <!DOCTYPE html>
2
3   <head>
4       <title>Yair Franco</title>
5   </head>
6
7   <body>
8       <h1>Hello World, this is my super awesome research website.</h1>
9       <p>It's small, but makes up for in personality!</p>
10  </body>
```

Setting Up Your Workflow

Creating your web dev environment

This page is ready to be opened!

- Open the index.html file from your computer's file explorer
  - It should open on your default browser



Setting Up Your Workflow

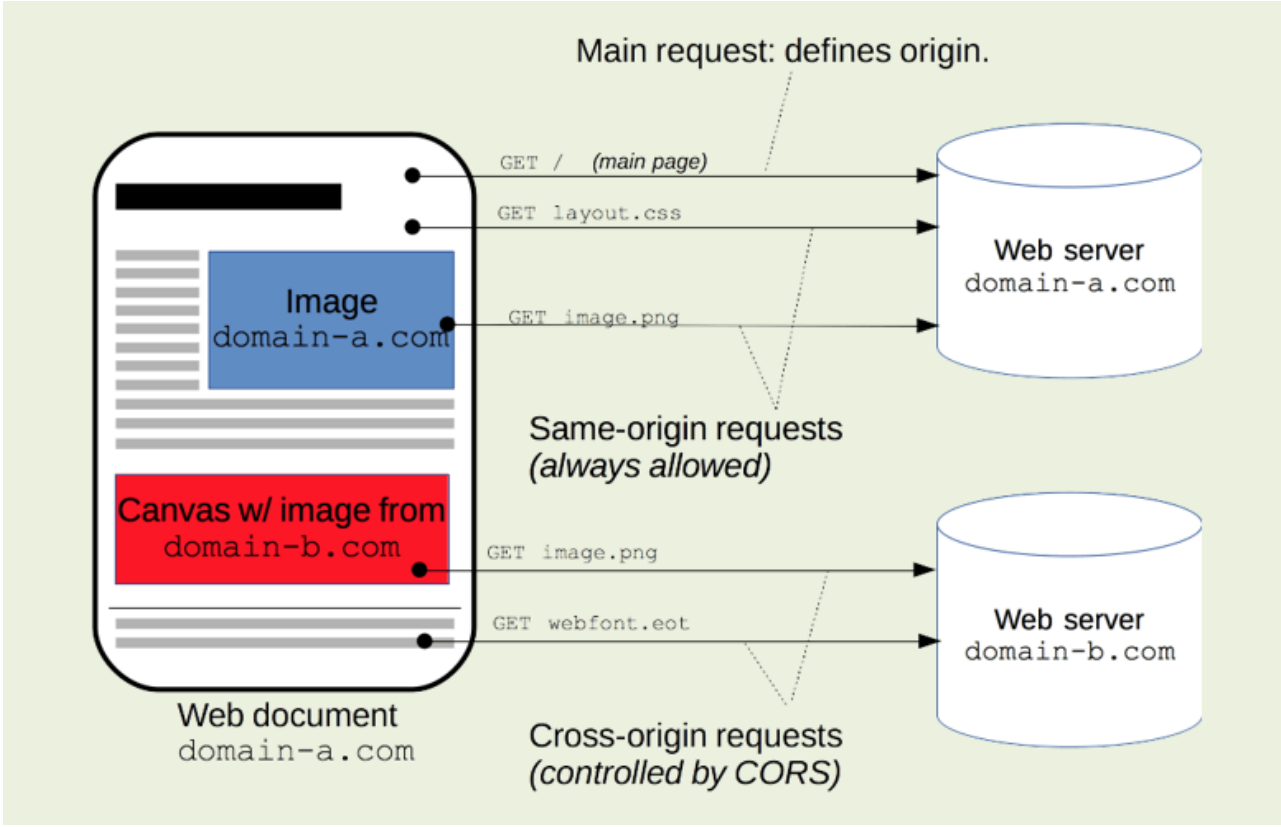
Creating your web dev environment

Opening the page from your local files is the easiest way to see your site.

However, if you implement scripts with multiple files, or that fetch data from APIs, you'll run into issues testing them.

This is because all major internet browsers regulate how files with different origins can communicate with each other (known as a CORS restriction).

To bypass this, you can start a local server on your computer. This ensures your entire repository is a single origin for your browser to use.



Setting Up Your Workflow

Creating your web dev environment

There's many ways to start a local server. The two shown here are among the simplest:

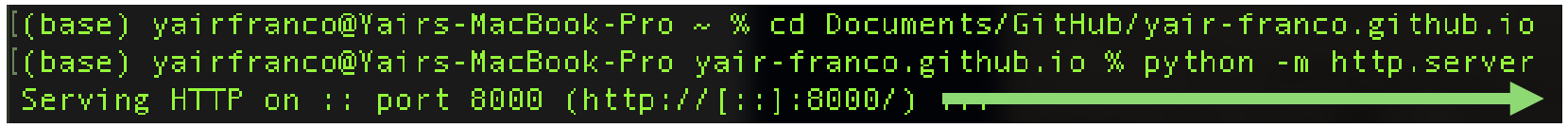
Using Python:

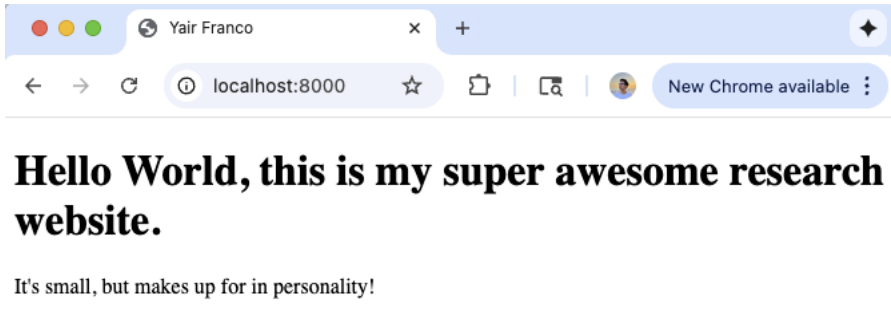
- Go to your repository's directory in your terminal
- From the directory, start a local server with this command:

```
$python –m http.server
```

- In your browser, navigate to http://localhost:8000

You should see your site just as you opened it before.



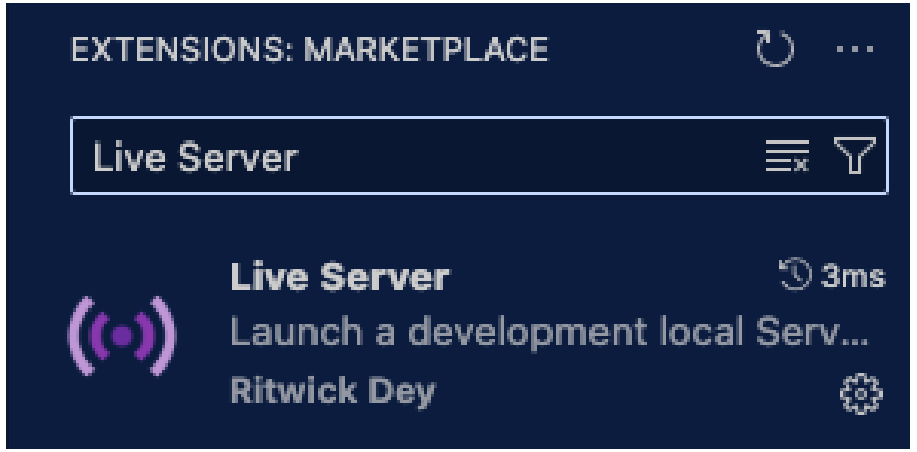


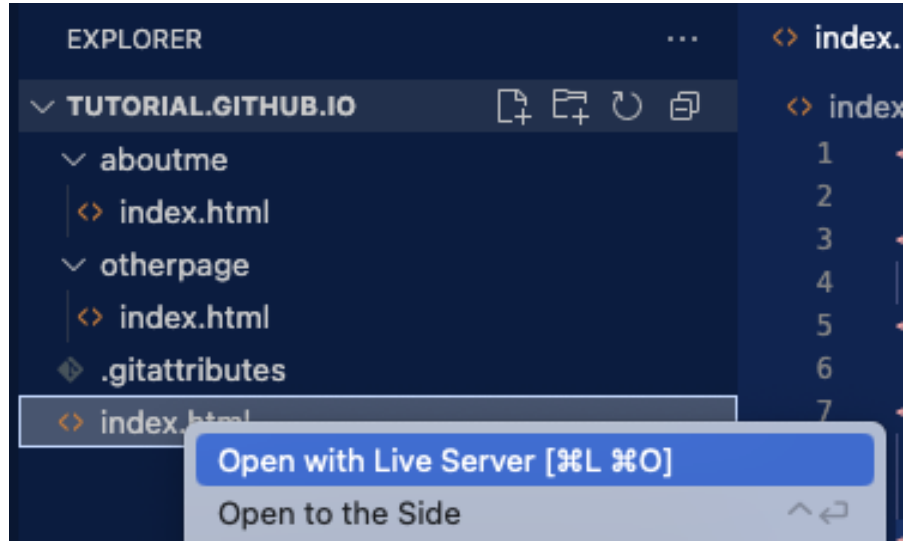
Setting Up Your Workflow

Creating your web dev environment

Using Live Server (in VS Code):

- In VS Code, install the Live Server extension from the extensions marketplace
- In your explorer, right-click the index.html file and click ‘Open with Live Server’
- A local server will be started and the page will open on your browser

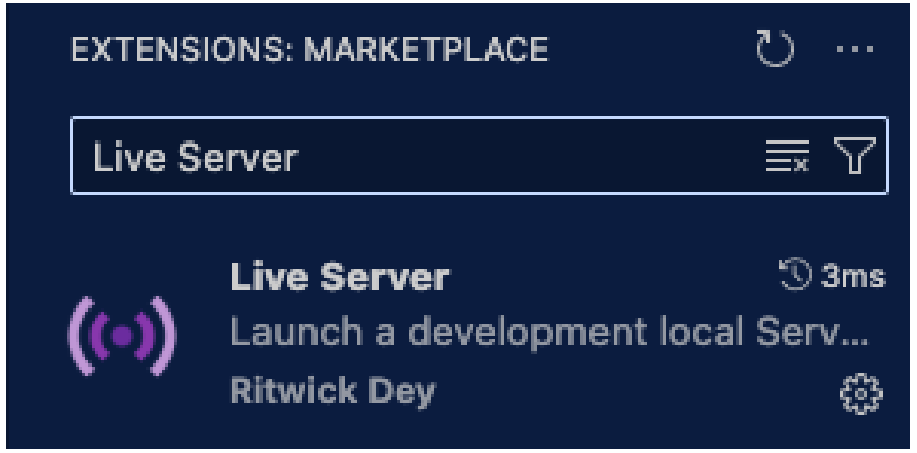


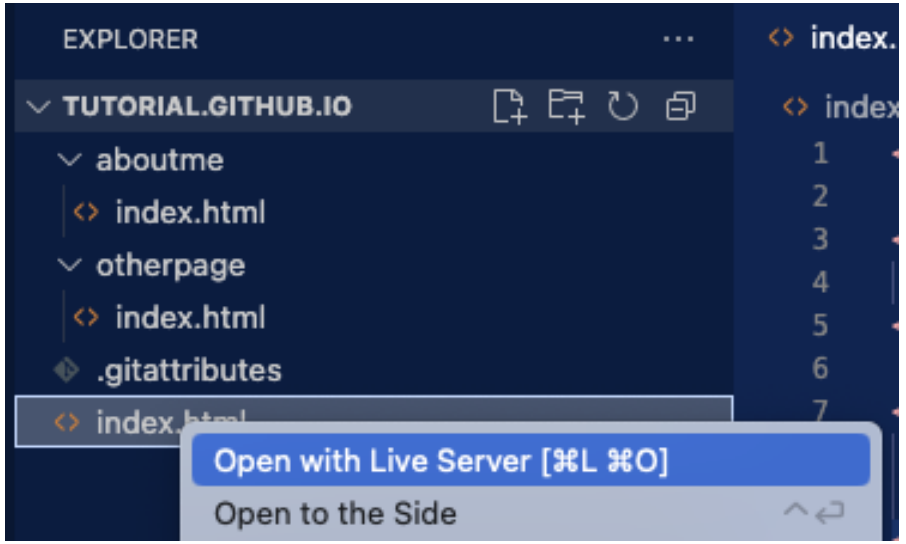


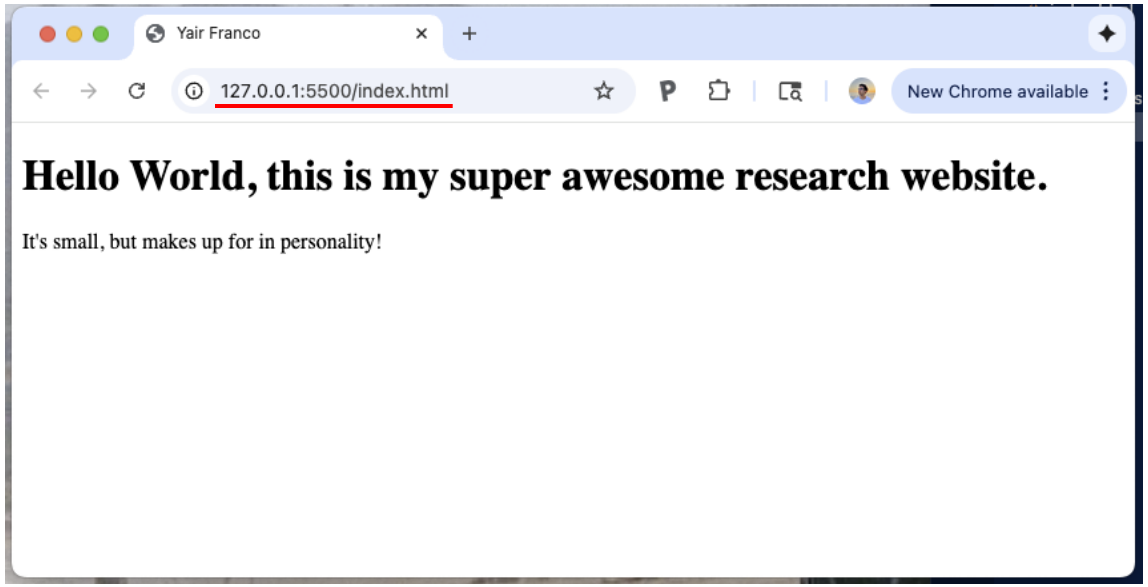
Setting Up Your Workflow

Creating your web dev environment

I highly recommend using Live Server, as it updates the page automatically each time you save a file. This saves a ton of time spent clicking. (It may sound silly, but time spent saving and refreshing when editing a page adds up!)







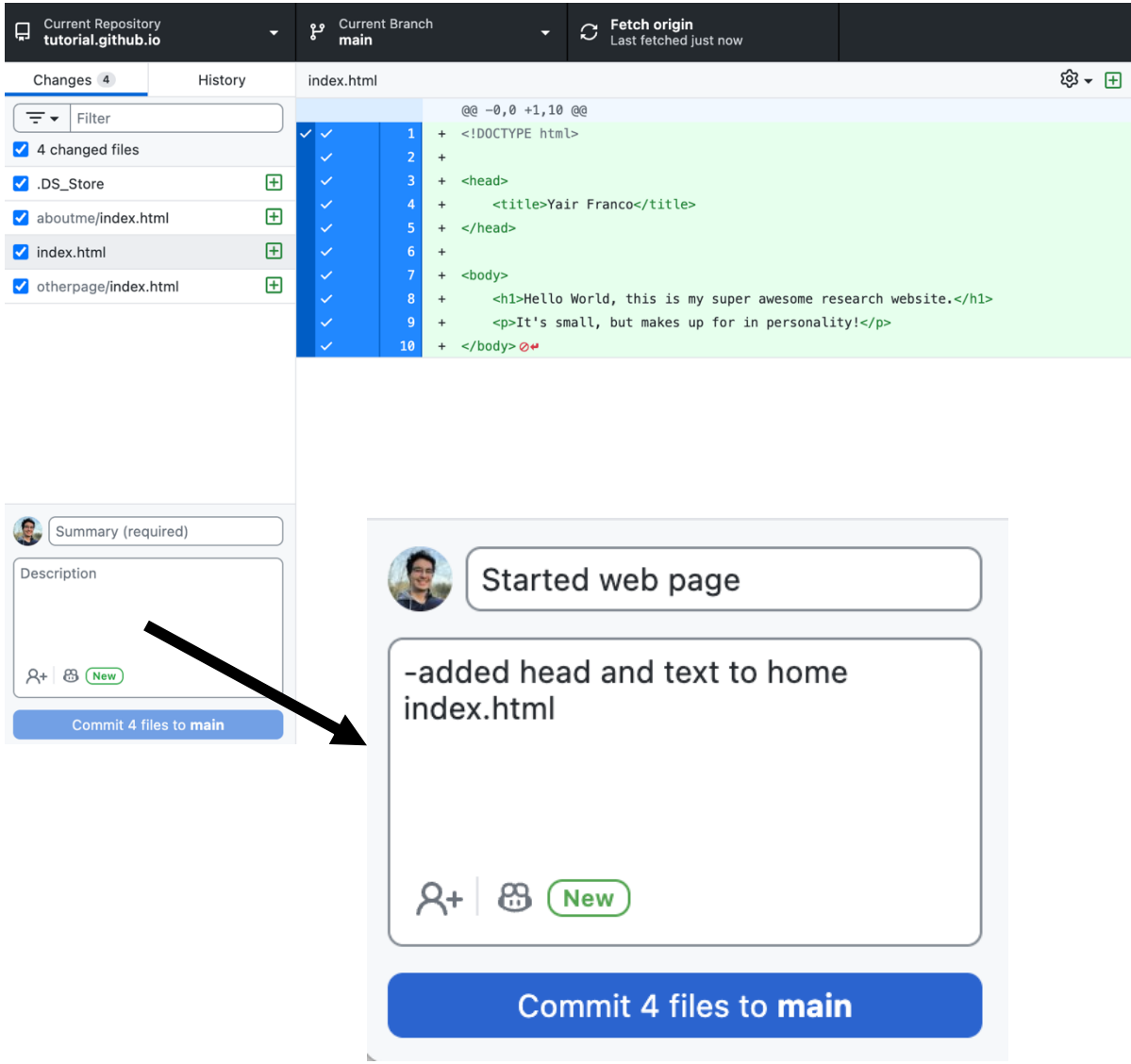
Setting Up Your Workflow

Creating your web dev environment

The last step in the environment is pushing your repository to GitHub so it's public.

You may have noticed a couple of things change on GHD as you worked. These are your file changes.

- On the commit section, add a summary and description of this work session
- Commit the files



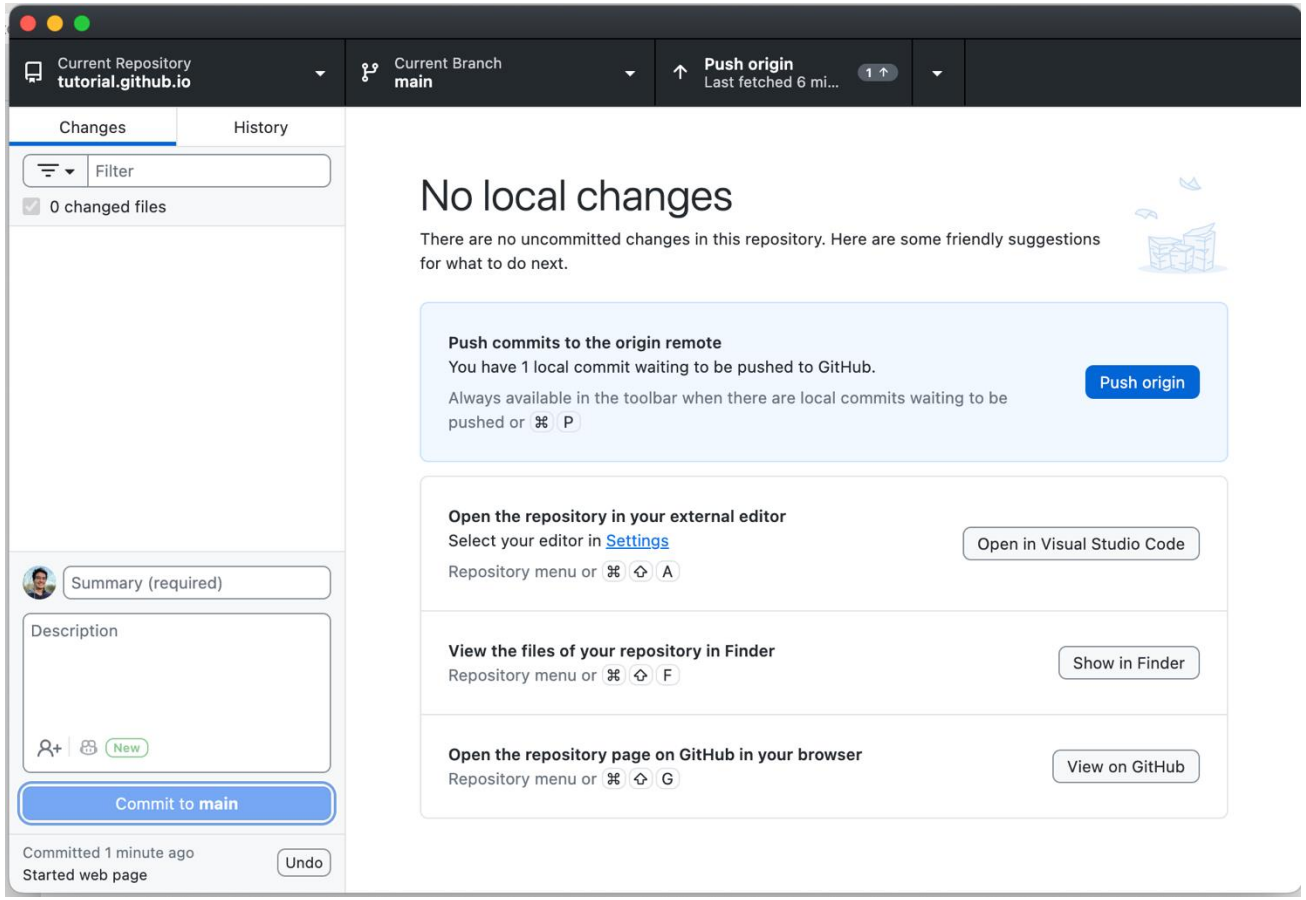
Setting Up Your Workflow

Creating your web dev environment

Your commit should now be saved locally.

- Click ‘Push origin’

This will push your code publicly to GitHub!

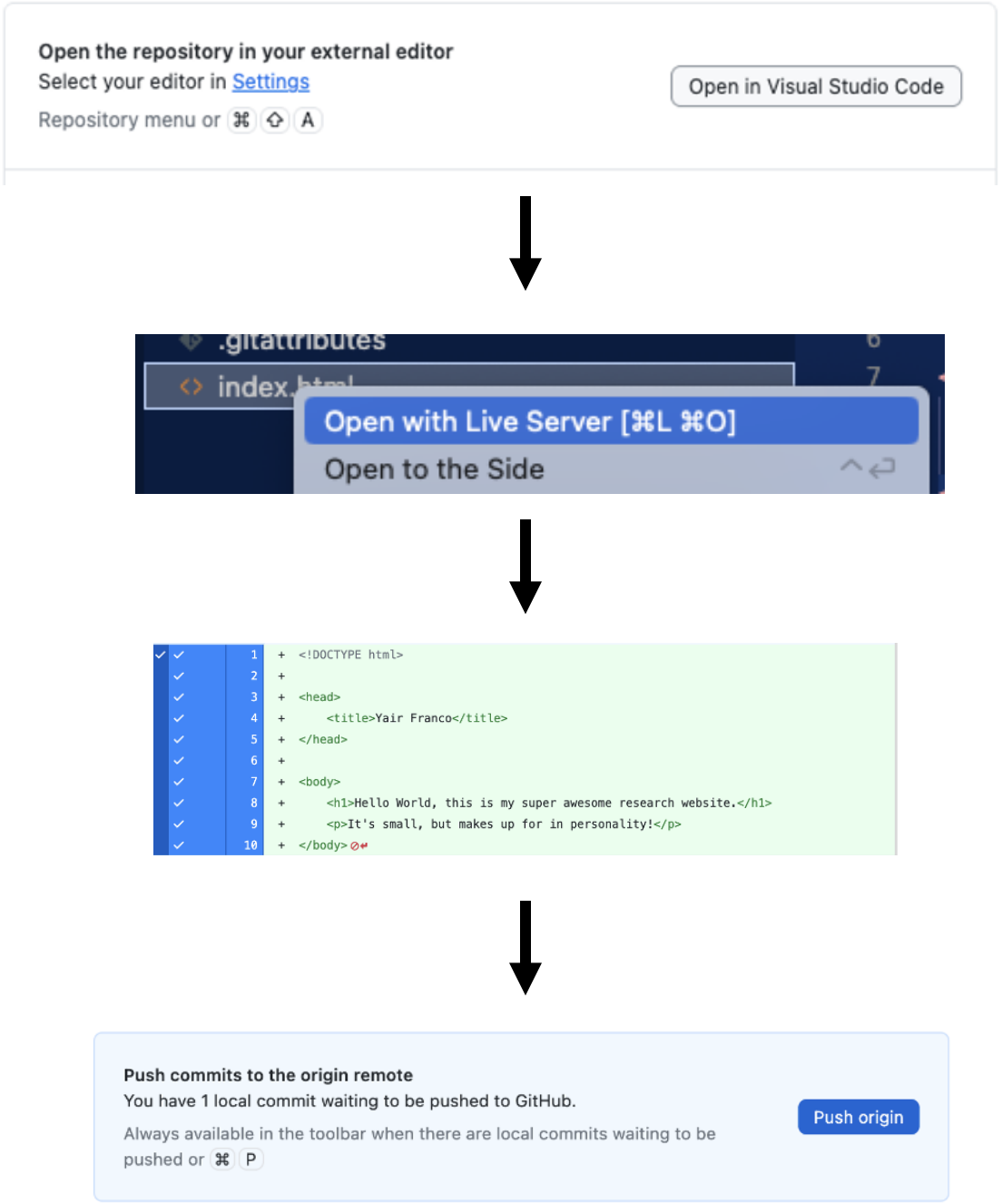


The Workflow

Rinse and repeat

This is a workflow you can use to update your website. To summarize:

- Open repository folder in VS Code (most direct via GitHub)
- Start local server with Live Server
- Edit and save your code
- Push repository to GitHub



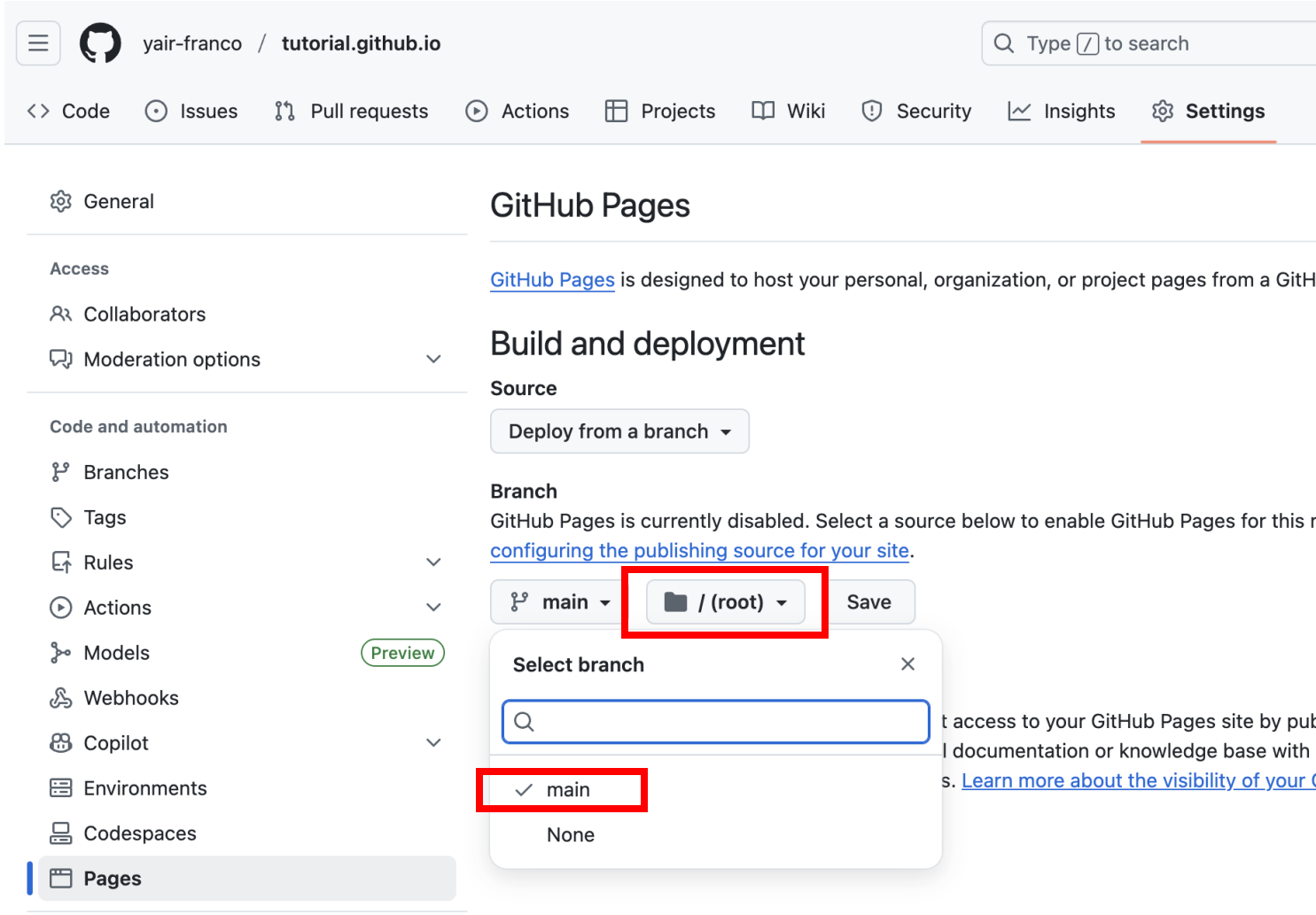
Publishing on GitHub Pages

Publishing on GitHub Pages

Setup

Now let's get your website published on the internet.

- Navigate to your repository on github.com
- Go to Settings > Code and Automation > Pages
- Ensure 'Source' is set to 'Deploy from a branch'
- Under 'Branch', select the 'main' branch and the root directory
- Click 'Save'.

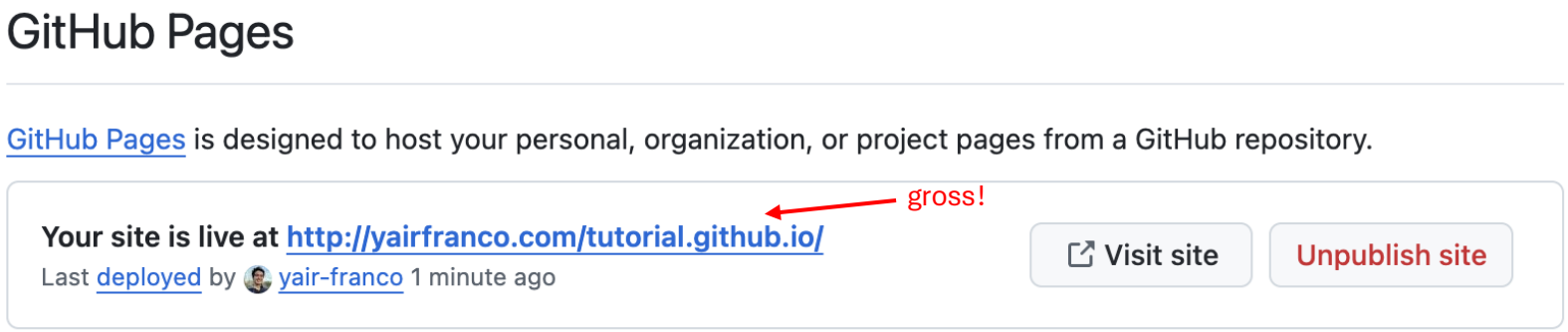


Publishing on GitHub Pages

Setup

After a couple of minutes, the Pages settings page will update and confirm that your site is published.

Note this example, where my domain name does not match the repository name. When this happens, GitHub will include both names in the URL.



In most cases, if done incorrectly, it'll look more like:

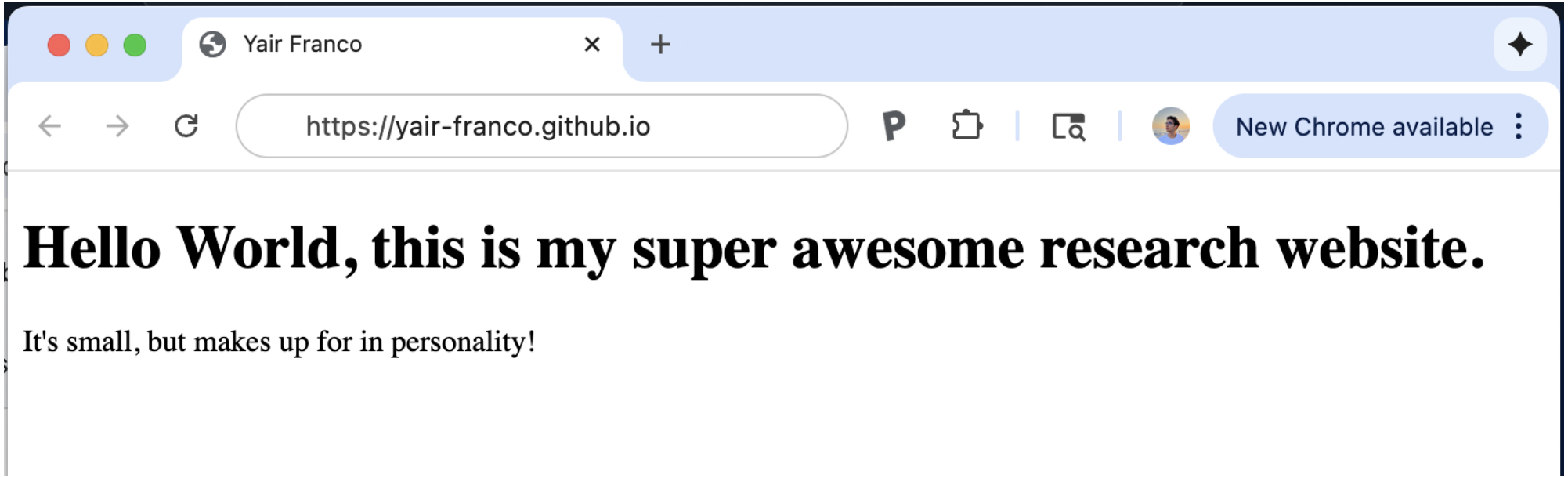
```
http://{yourusername}.github.io/{yourrepositoryname}.github.io
```

This is easily fixed by renaming the repository to the matching name.

Publishing on GitHub Pages

Setup

Now your website is live! Tell your friends. Try accessing it from different browsers and devices.



Domain Names

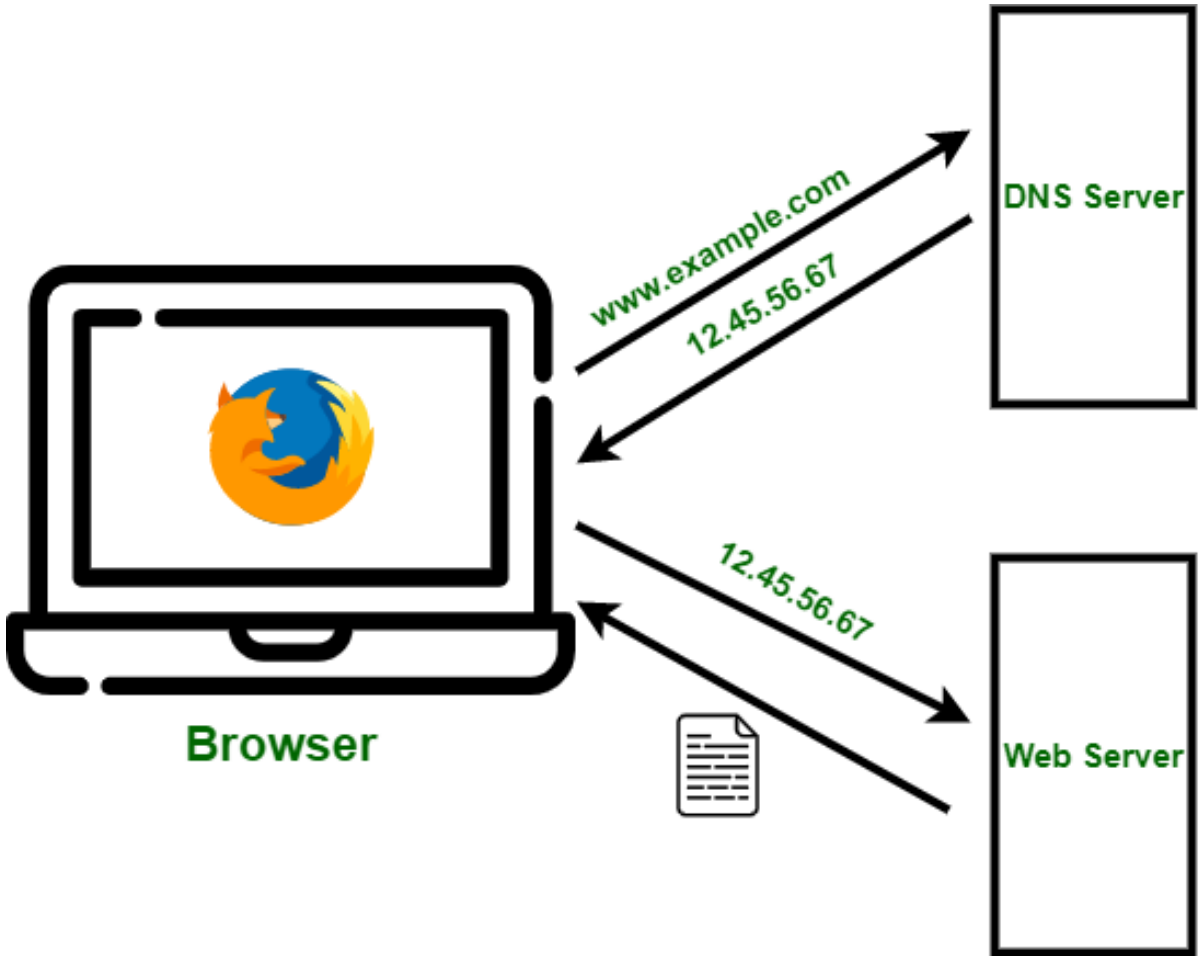
Getting a Domain Name

A What?

A domain name is the readable address that directs the browser to the IP address of a server (via the Domain Name System, or DNS). Rather than memorizing a lot of numbers and dots to get to the servers of my favorite website, I can just type the domain name ‘youtube.com’ in the browser bar instead.

Domain name registrars sell domain names and allow you to direct them to your server.

The next task is to obtain a domain name and direct it to the GitHub servers hosting your website.



Getting a Domain Name

Purchasing a domain name

There are several domain name registrars out there, some more reputable than others.

I have had a good experience with Porkbun. I will show the process of getting a domain name there. However, if you choose a different registrar, the process will be fairly similar.

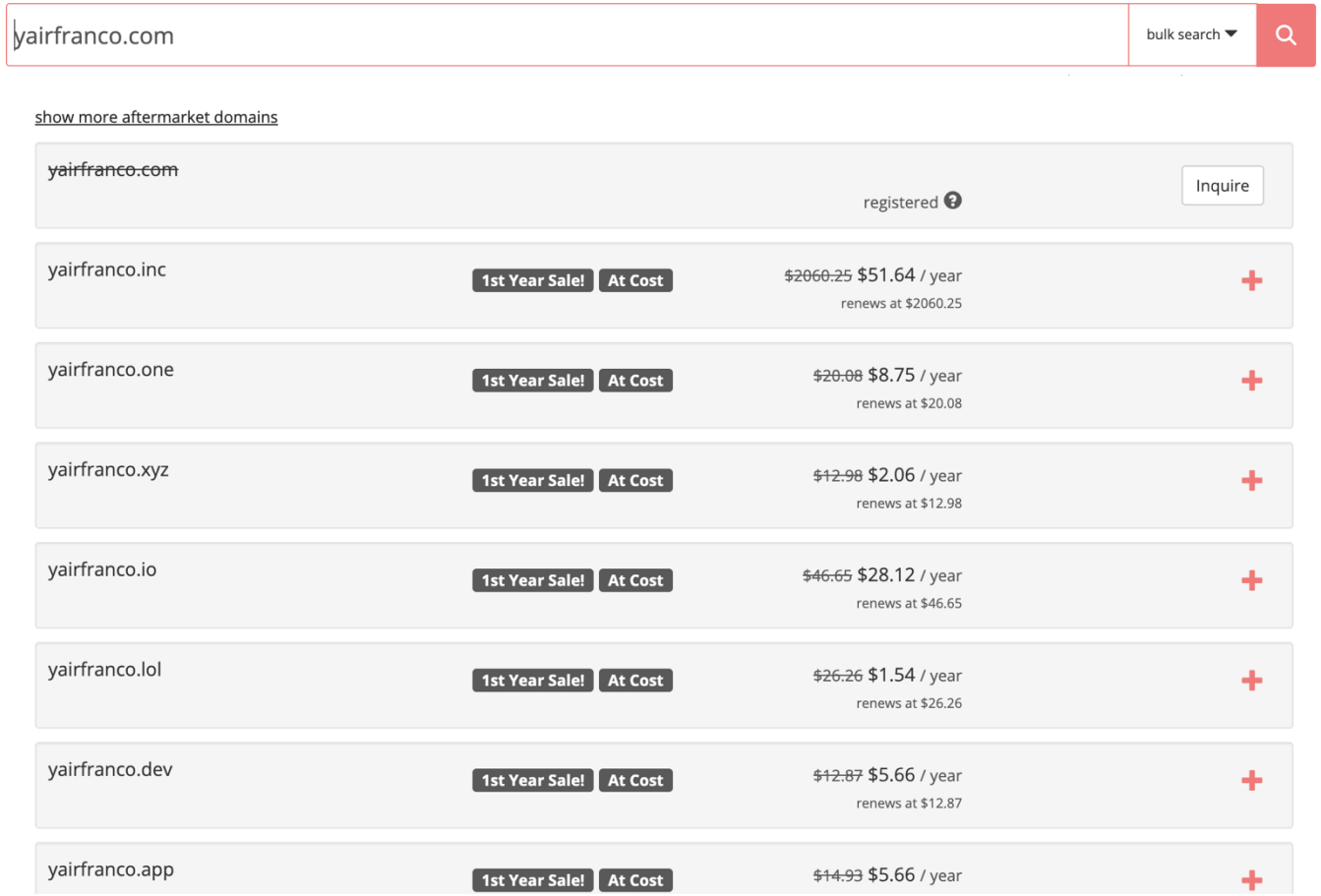
Getting a Domain Name

Purchasing a domain name

- Navigate to porkbun.com and type the domain name you want in the search bar

You'll see the domain name and similar ones to the one you typed, and its availability.

Here, for example, yairfranco.com is taken (because I already claimed it).

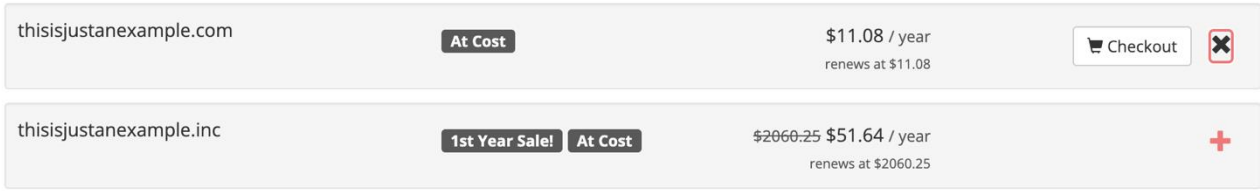


Getting a Domain Name

Purchasing a domain name

Once you've found the right domain name for you:

- Log in to your Porkbun account
- Click the '+' icon on the domain name you want
- Proceed with the checkout steps



Some registrars will push hard to upsell you for things like additional subdomains, web hosting, or email servers. You won't need that for now. I like Porkbun in part because it does not upsell you hard (unless you want them to).

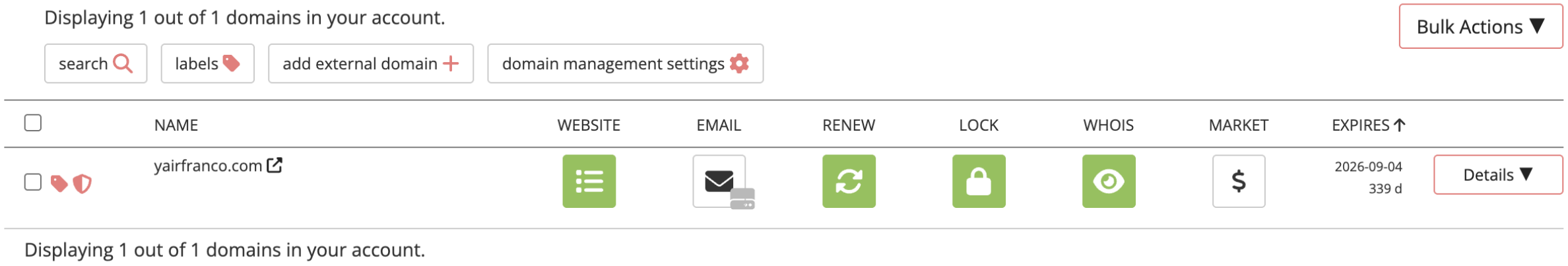
Getting a Domain Name

Purchasing a domain name

Once you've purchased your domain:

- Go to Account > Domain Management

You'll see a list of domains you own.



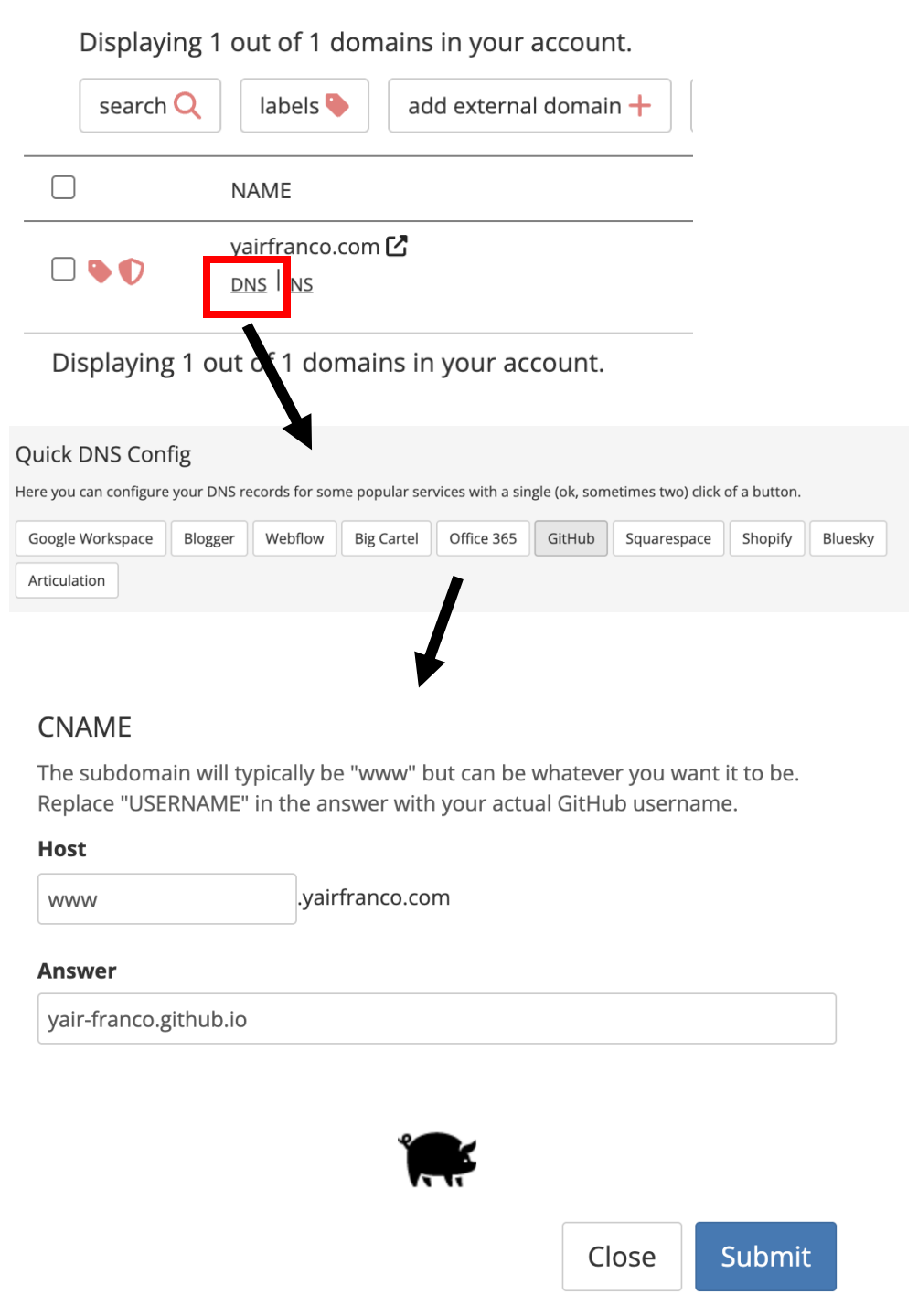
Getting a Domain Name

Setting up your domain

Now that you own this domain, you can do whatever you want with it! Let's add DNS records that will redirect the domain to your github.io page. Luckily, Porkbun makes this really easy!

- In your Porkbun domain management page, click "DNS" under the domain you're setting up
- Under 'Quick DNS Config', select 'GitHub'
- Under 'CNAME', set the 'Answer' field to be your github.io URL.

A CNAME record is a DNS record that points a domain name to another domain name, rather than an IP address.

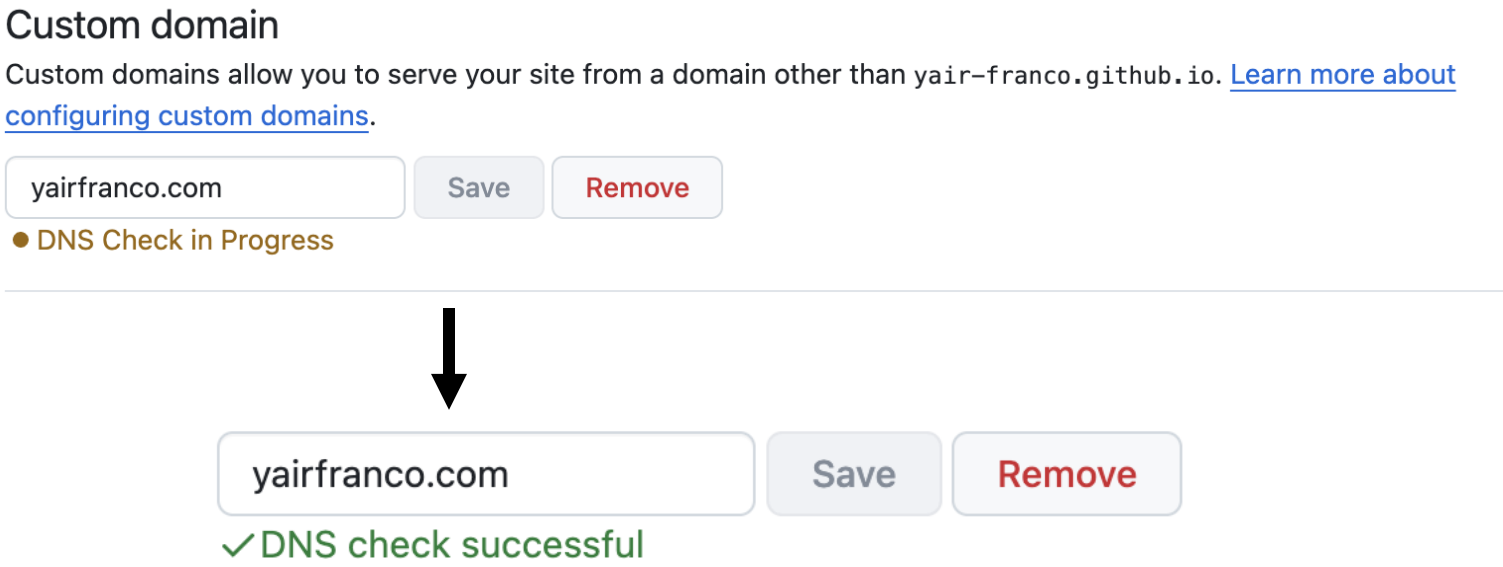


Getting a Domain Name

Setting up your domain

- In your GitHub Pages settings, enter your custom domain

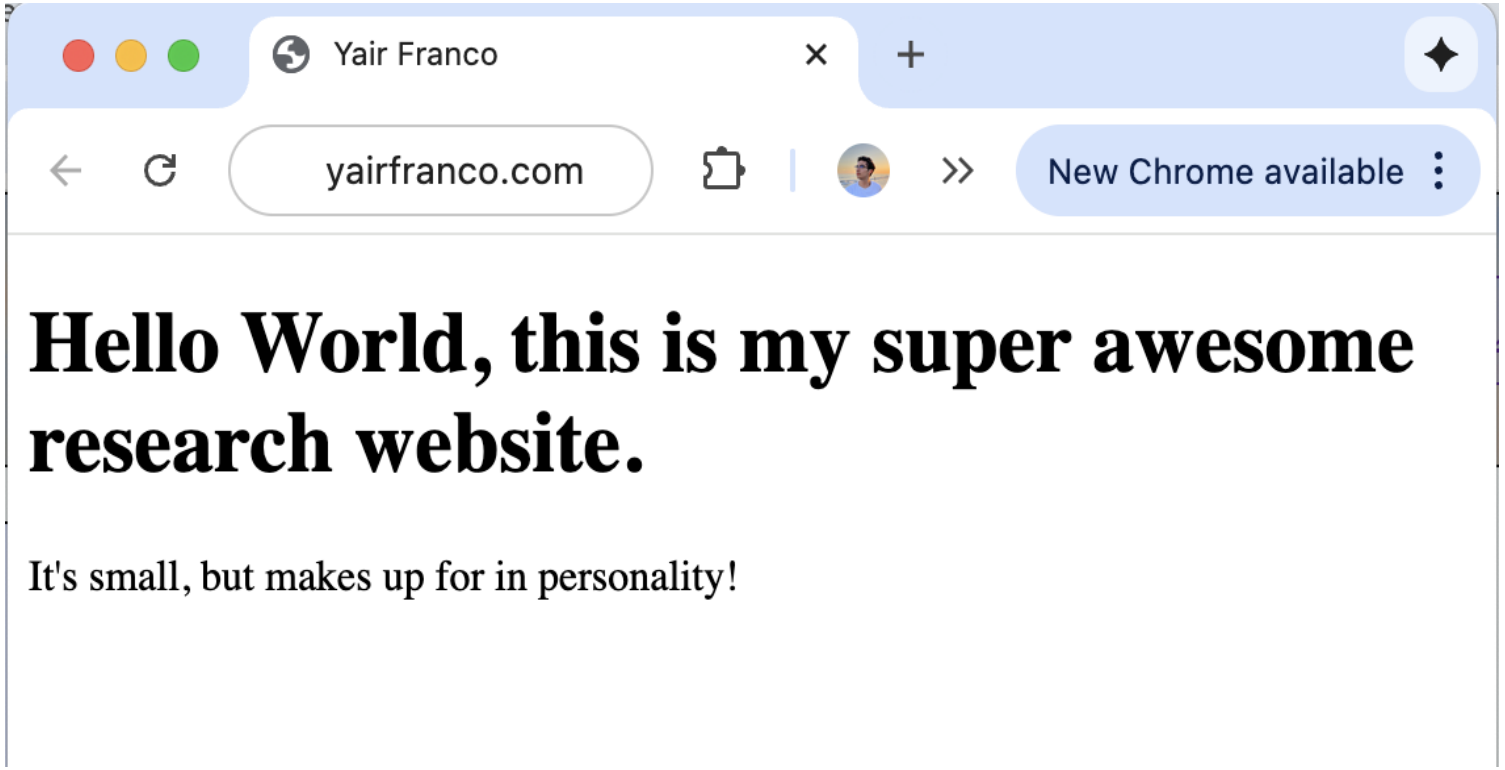
GitHub will then start a DNS check to verify that you own the domain and have set it up correctly. This can take a few minutes to a few days (I've never seen it take that long, but that's what GitHub says).



Getting a Domain Name

Setting up your domain

Once GitHub has finished the DNS check, you'll be able to visit your website using your domain name!



Getting a Domain Name

Setting up your domain

If you choose to use a domain registrar other than Porkbun, there's a chance you'll have to set your DNS records manually (as shown in the image). GitHub provides a guide to do this here: https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site.

TYPE	HOST	ANSWER	TTL	PRIORITY
A	yairfranco.com	185.199.108.153	600	
A	yairfranco.com	185.199.109.153	600	
A	yairfranco.com	185.199.110.153	600	
A	yairfranco.com	185.199.111.153	600	
AAAA	yairfranco.com	2606:50c0:8000::153	600	
AAAA	yairfranco.com	2606:50c0:8001::153	600	
AAAA	yairfranco.com	2606:50c0:8002::153	600	
AAAA	yairfranco.com	2606:50c0:8003::153	600	
CNAME	www.yairfranco.com	yair-franco.github.io	600	

To summarize, you'll have to create A and AAAA records pointing your domain to GitHub's servers yourself. A records point domain names to IPv4 addresses, and AAAA records point to IPv6 addresses.

What Now?

Congratulations! You now have your own website.

Your task now is to develop it to your heart's desire. Learning HTML and CSS can be really fun if you like what you're creating, and I highly recommend it.

Another way to host a website is to make it on a website builder like Google Sites or Jekyll and point your domain name to that site instead. I've personally never done this so I'm not really qualified to show you how to do that, but some of this Guide might be helpful for that.

Happy web designing!