

1.1.8 Publish with GitHub

Awesome—all your hard work is now saved to Git! There is one problem, though. These commits only live on your local computer. So if it were to die a tragic death at the hands of spilled coffee or a lightning surge, your work would be lost. The Run Buddy company would be pretty upset if that happened!

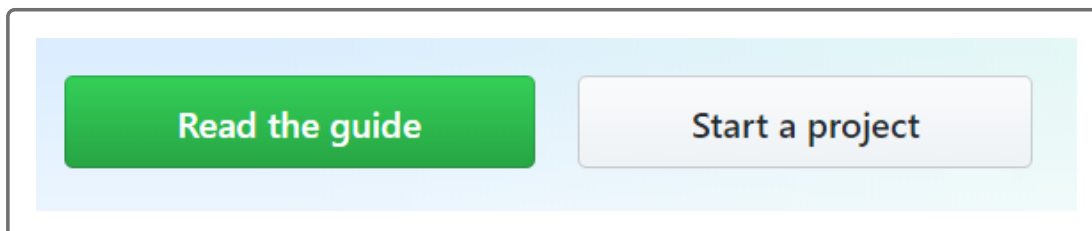
Wouldn't it be helpful if we could save this repository in a remote location (i.e., on someone else's computer!)? That's where GitHub comes in. GitHub is a website that hosts Git repositories. It's free and very popular among developers, so make sure you have an account!

IMPORTANT

A common misconception is that Git and GitHub are the same thing. Git is the version control software that you installed on your computer. GitHub is a service that lets you create Git repositories on the internet. There are other websites that perform the same service as GitHub, but the software on your computer (Git) wouldn't need to change to use them.

So how do we get our fledgling Run Buddy webpage into GitHub? The first step is to create a new, remote repository on GitHub. Then we'll link our local Run Buddy project with it.

After logging in to GitHub, click the "Start a project" button, shown in this image:




This will route you to a page where you'll be able to create a new repository. Give the repository a name (run-buddy) but don't change any other options. Then click the "Create repository" button, shown in the below image:

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 *


 username ▾

 /


run-buddy ✓

Great repository names are short and memorable. Need inspiration? How about [verbose-fiesta?](#)

Description (optional)

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▾

 |

Add a license: **None** ▾ [i](#)


Create repository

The next page will display a blue Quick Setup banner that includes a link to this repository. The link will look something like

`git@github.com:username/run-buddy.git` if the SSH option is clicked, or
`https://github.com/username/run-buddy.git` if HTTPS is clicked.

Here's an image of how it looks with SSH selected:

Quick setup — if you've done this kind of thing before


 Set up in Desktop

 or

HTTPS

SSH

git@github.com:username/run-buddy.git



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

IMPORTANT

If you were able to set up your SSH keys in the prework, you should definitely use the SSH option. If not, we highly recommend getting SSH

to work before moving on. [GitHub has a great set of tutorials on using SSH.](https://help.github.com/en/articles/about-ssh) [\(https://help.github.com/en/articles/about-ssh\)](https://help.github.com/en/articles/about-ssh)

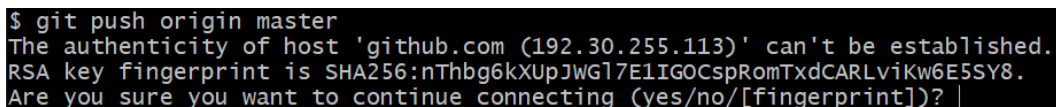
Now that we have a remote repository on GitHub, we need to link it with our local project. Copy the link from the website, then open the terminal again in the `run-buddy` directory and run the following commands, replacing the link below with the link you got from GitHub:

```
git remote add origin git@github.com:username/run-buddy.git
git remote -v
```

The second command, `git remote -v`, shows us which remote links, if any, we've established for our project. We could certainly add others—and later in the boot camp, we will—but the link to GitHub is all we need for now. Take note that we also named this remote link `origin`. That name is important, because we're going to reference it when running this next command:

```
git push origin master
```

If you used the SSH link, Git will ask you to enter your SSH passphrase. The first time you try to push code to GitHub, your computer will also want to know if you trust this website, as this image shows:

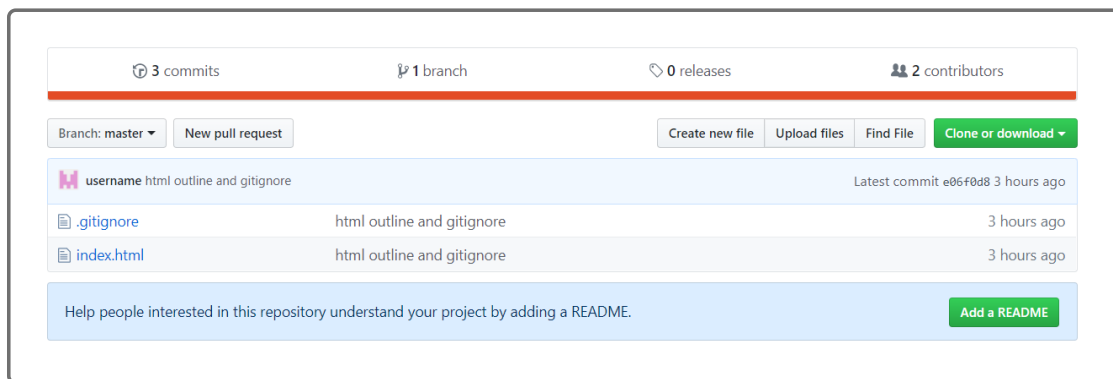


```
$ git push origin master
The authenticity of host 'github.com (192.30.255.113)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWG17E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? |
```

Type `yes`, because we do trust them. If you used the HTTPS link, on the other hand, you'll be asked to enter your GitHub username and password (every time you push to GitHub, which is why SSH is more convenient).

So what exactly did `git push origin master` do? The `git push` command sends any local commits to a remote location. In this case, the location is `origin` (GitHub), and we want to update the origin's `master` branch. For now, you only have one version of your codebase, and it's called `master` by default.

Go back to your GitHub repository in the browser and refresh the page. It should look like this:



GitHub now has our two files and a record of our three previous commits! This means we could download this repository on another computer and continue working without missing a beat, or we could add other developers to the project and let them commit and push to this same location.

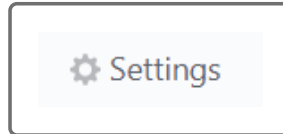
From now on, `git push` should become part of your normal Git routine to ensure that your changes always make it to GitHub. Memorize these three commands:

```
git add -A
git commit -m "<message-that-describes-the-commit>"
git push origin master
```

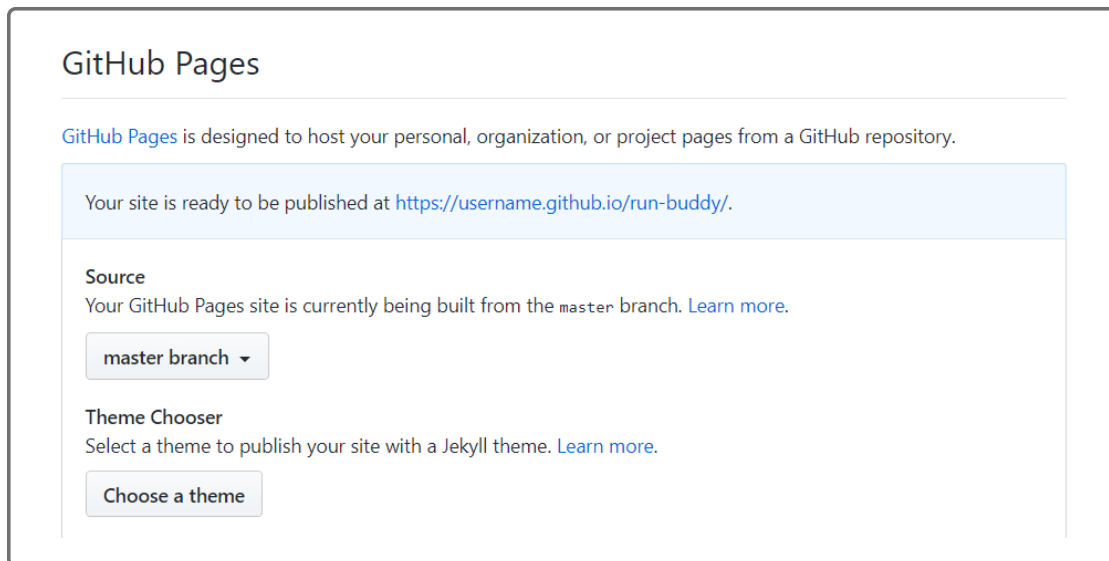
GitHub comes with a lot of other helpful features like contribution stats, code reviews, and bug tracking. GitHub will also host your HTML projects

as live websites, so you don't have to worry about paying for a separate hosting service like GoDaddy. That's pretty nice of them!

Let's flip the switch to make our current HTML project live. That way, we can show the project manager of Run Buddy our progress. On your repository's page, click the Settings tab. It looks like the following image:



Then on the next page, scroll down to the GitHub Pages section. Change the dropdown under Source to say "master branch", as this image shows us:



This tells GitHub to use the contents of your master branch for your live website. Once you do this, GitHub will then display the message: "Your site is ready to be published at [https://.github.io/run-buddy/](\"https://.github.io/run-buddy/\")."

It might take a minute for this to actually kick in, so wait a moment before visiting this link (replacing "username" with your GitHub username, of course). When you do open it, though, you'll see that your Run Buddy webpage is now live on the internet!

And as we continue to improve this project and push new commits to GitHub, this live link will automatically update. Thanks, GitHub!

ON THE JOB

GitHub has become a vital resource to recruiters and hiring managers because it helps them see a candidate's recent activity, samples of their code, and their thought process via READMEs.

Let's back up and look at this link again: <https://.github.io/run-buddy/>. Notice how it loaded the contents of our `index.html` file without us having to write <https://.github.io/run-buddy/index.html>, although that shows the same page, too. The reason for this is that `index.html` is defined as a default by the server.

If you visit something.com/contact.html, for instance, the something.com server knows to send back `contact.html` because it was explicitly asked for. But if a file isn't specified, the server sends back `index.html`. Knowing this, make sure every project has an `index.html` file!

Almost finished with this lesson. Take a quick assessment to help hone your new skills.

Final score: 100%

 [Retake](#)