Deploying Your React Website to GitHub Pages

Abdulrahman Y. idlbi
Mar 23 · 5 min read ★

Making *React* + Dynamic *Routing* + *GitHub Pages* Play Nicely.

I had been finalizing my portfolio website over the weekend and wanted to deploy it to GitHub Pages. However, what I thought would be a straightforward task wasn't really that. I built a React app that uses dynamic routing to show different pages with unique links and wanted to host it on GitHub Pages. I faced a few issues and wanted to share the process I followed in one place.

I assume you already have your working React app (as in the one you created using `create-react-app`) and it's hosted in a repo on GitHub. So let's start.

First, install the **gh-pages** module in your app directory:

```
$ npm i gh-pages
```

Now, you have to add the deployment configuration. Head to your `package.json` file to make some edits.

What edits you do next depend on whether you're creating a project website or a user/organization website. You can create as many project sites on GitHub Pages as you wish. Their addresses will have have the format: https://*username*.github.io/*repository*.

On the other hand, you can have only one user website per account, which will have a nicer-looking address: https://*username*.github.io. You can learn more on GitHub Pages.

If You're Creating a Project Site

Here's what you have to do inside `package.json`:

- Add your homepage address (just before `dependencies`): `"homepage": "http://username.github.io/repository"`
- Inside `scripts`, add the commands to create a production build and deploy: `"predeploy": "npm run build"` and `"deploy": "gh-pages -d build"`

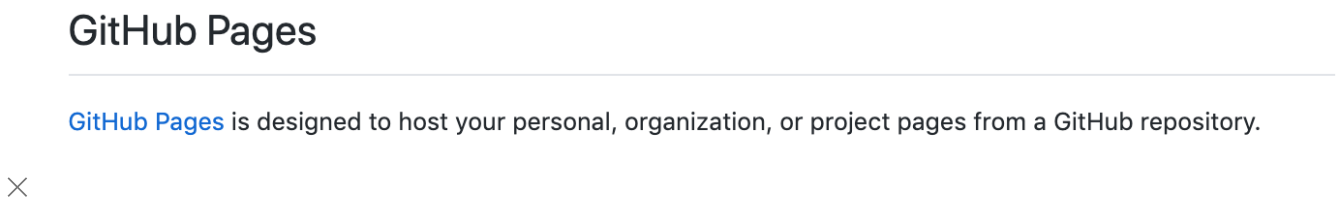
The `package.json` file should look somehow like this. (don't forget to use proper values in place of ***username*** and ***repository***!):

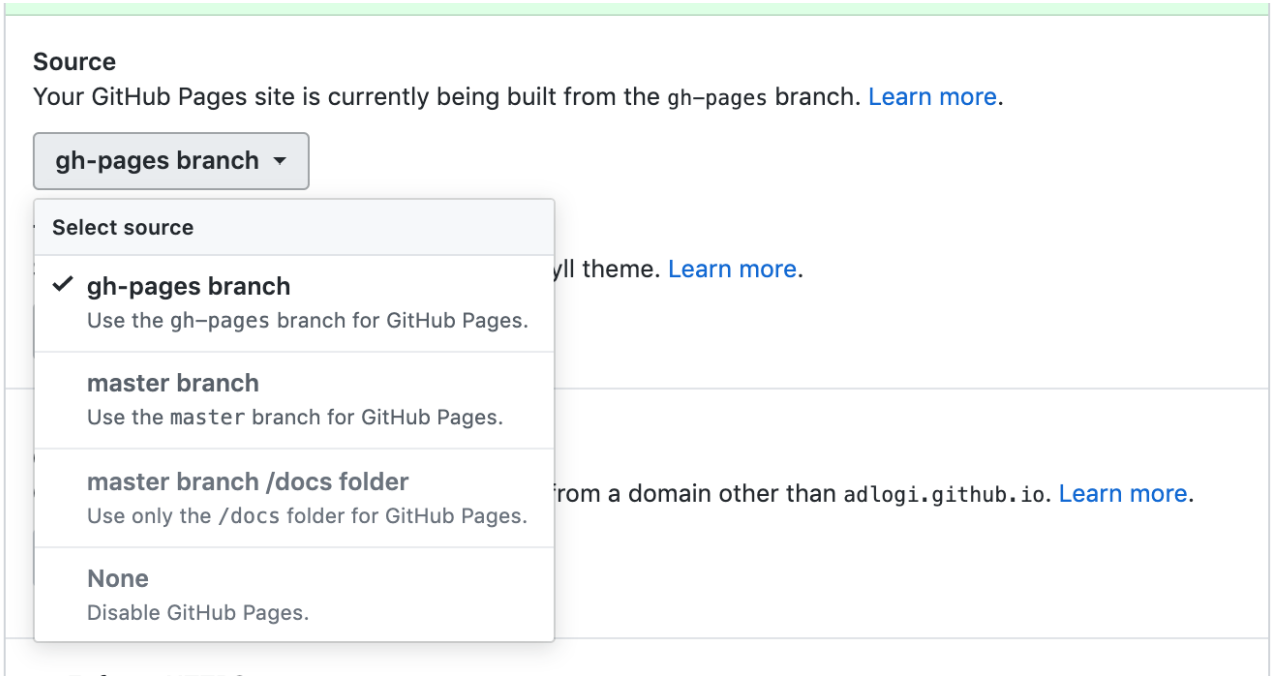
```
{
  "name": "YourReactApp",
   ...
  "homepage": "https://username.github.io/repository",
   ...
  "scripts": {
     ...
    "predeploy": "npm run build",
    "deploy": "gh-pages -d build",
     ...
  }
}
```

To deploy, all you have to do is running the following command in the terminal:

```
$ npm run deploy
```

We're almost there. If you go to your repo on GitHub, you'll notice a new branch called `gh-pages` that has been created. This is the branch that will serve your website. From your repo's main page on GitHub, go to **Settings**. You'll find the **GitHub Pages** section near the end of the page. Select the **`gh-pages`** branch as the source for your GitHub Pages website.





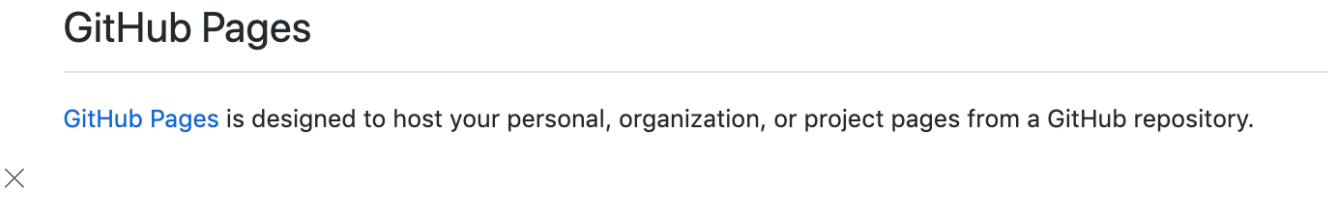
Yay! You can now see your website live! If you're using routing in your app, there's still one more thing to do, so continue reading (you can pass the next section).

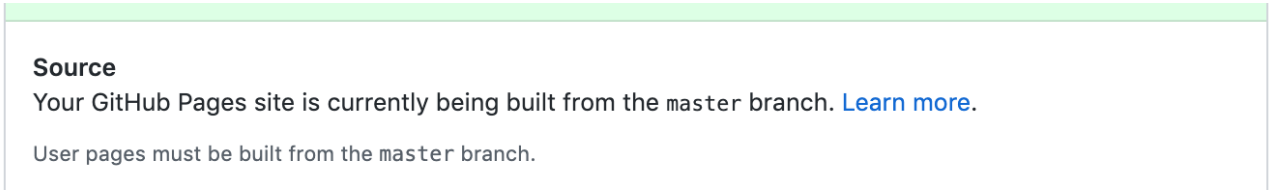
If You're Creating a User Site

I started all this because I was creating my portfolio website. As I was about to deploy to GitHub Pages, I thought it would be nicer to use *adlogi.github.io* as an address instead of *adlogi.github.io/portfolio* for example. Creating a user website is not very different from creating a project one. So, here's what you have to do:

You can have only one user site per account, and the repo for that site should have a specific name. Your app repo's name should be `username.github.io`. If you're already using a different name, don't worry! Navigate to the repo's main page on GitHub and click **Settings**. Then, under the **Repository Name** heading, type the new name: `username.github.io` (using your own username, of course).

While you're still on the **Settings** page, scroll down towards the bottom of the page to the **GitHub Pages** heading. You'll notice that, unlike project pages, user pages can only be served from the `master` branch. This requires some changes to our branches and the way we configure the deployment in `package.json`.





First of all, let move what we have on the `master` branch to a new one. Let's call it `source`:

```
$ git checkout -b source
$ git push origin source
```

Back on your repo's **Settings** page, click **Branches** from the menu on the left and change the **Default branch** to `source` and click **Update**.

Next, configure the deployment for a user website in `package.json`:

- Add your homepage address (just before `dependencies`): `"homepage": "http://username.github.io/"`
- Inside `scripts`, add the commands to create a production build on the `master` branch and deploy: `"predeploy": "npm run build"` and `"deploy": "gh-pages -b master -d build"`

The `package.json` file should look somehow like this. (don't forget to use *your* username values in place of `username`!):

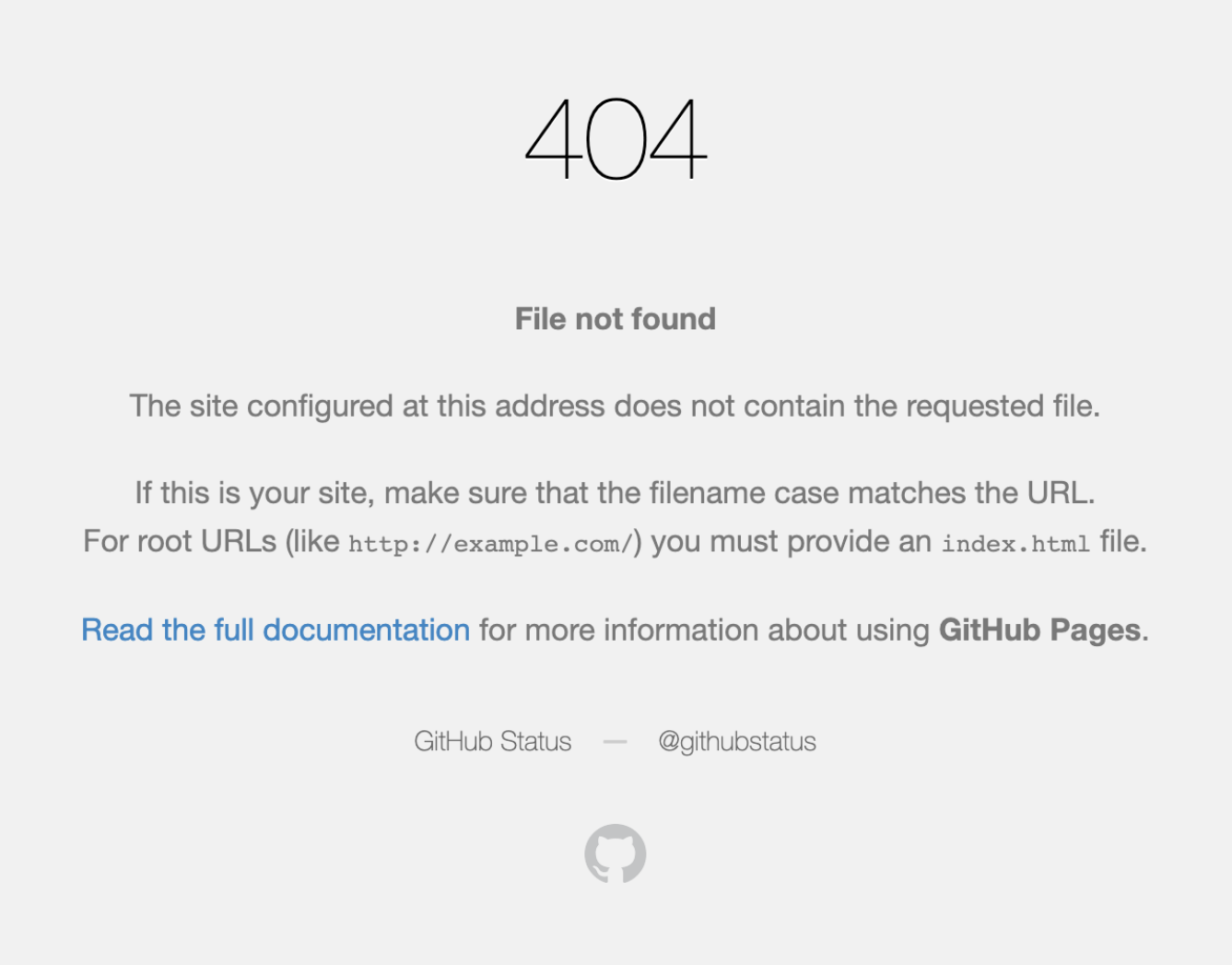
```
{
  "name": "YourReactApp",
   ...
  "homepage": "https://username.github.io/",
   ...
  "scripts": {
     ...
    "predeploy": "npm run build",
    "deploy": "gh-pages -b master -d build",
     ...
  }
}
```

To deploy, all you have to do is running the following command in the terminal:

```
$ npm run deploy
```

Let's Talk about Routing

I used React Router to make my website look as if it had separate pages with unique addresses. I deployed it as a GitHub user site following the steps explained above and the links seemed to be working fine. I was on one of those pages when I hit refresh, and the page disappeared and a 404 showed instead. Trying to access a page by entering its address directly (instead of navigating starting from the homepage) gave the same result. It seemed that a fresh page load was failing in retrieving the page in production (while it was handled correctly in development).



It seems there are two ways to overcome this problem (see this note on client-side routing and GitHub Pages). One is using React's `<HashRouter>` instead of `<BrowserRouter>`. The other is handling 404s and redirecting them to the proper pages. I followed the second solution, using code (and explanation) available on this repo

In two simple steps:

1. Copy the `404.html` file from that repo to your project's `/public` directory (just next to `index.html`). There's a variable, `segmentCount`, that may need to be set in the script inside `404.html`. For a GitHub Pages user site, `segmentCount = 0` works fine.
2. Copy the script found here (lines: 58–88) to the `<head>` section in `/public/index.html`.

That should be it. Deploy (`$ npm run deploy`) and your links should be working as expected.

JavaScript React Github Pages React Router