GitHub Pages is a service offered by GitHub that allows hosting for websites configured straight from the repository. A Gatsby site can be hosted on GitHub Pages with a few configurations to the codebase and the repository's settings.

You can publish your site on GitHub Pages several different ways:

- to a path like `username.github.io/reponame/` or `/docs`
- to a subdomain based on your username or organization name: `username.github.io` or `orgname.github.io`
- to the root subdomain at `username.github.io`, and then configured to use a custom domain

Configuring the GitHub Pages source branch

You must select which branch will be deployed from your repository settings in GitHub for GitHub Pages to function. On GitHub:

1. Navigate to your site's repository.
2. Under the repository name, click Settings.
3. In the GitHub Pages section, use the Source drop-down to select `main` (for publishing to the root subdomain) or `gh-pages` (for publishing to a path like `/docs`) as your GitHub Pages publishing source.
4. Click Save.

Installing the `gh-pages` package

The easiest way to push a Gatsby app to GitHub Pages is by using a package called gh-pages.

```
npm install gh-pages --save-dev
```

Using a deploy script

A custom script in your `package.json` makes it easier to build your site and move the contents of the built files to the proper branch for GitHub Pages, this helps automate that process.

Deploying to a path on GitHub Pages

For sites deployed at a path like `username.github.io/reponame/`, the `--prefix-paths` flag is used because your website will end up inside a folder like `username.github.io/reponame/`. You'll need to add your `/reponame` path prefix as an option to `gatsby-config.js`:

```
module.exports = {
  pathPrefix: "/reponame",
}
```

Then add a `deploy` script to `package.json` in your repository's codebase:

```
{
  "scripts": {
    "deploy": "gatsby build --prefix-paths && gh-pages -d public"
  }
}
```

When you run `npm run deploy` all contents of the `public` folder will be moved to your repository's `gh-pages` branch. Make sure that your repository's settings has the `gh-pages` branch set as the source to deploy from.

Note: To select `main` or `gh-pages` as your publishing source, you must have the branch present in your repository. If you don't have a `main` or `gh-pages` branch, you can create them and then return to source settings to change your publishing source.

Deploying to a GitHub Pages subdomain at github.io

For a repository named like `username.github.io`, you don't need to specify `pathPrefix` and your website needs to be pushed to the `main` branch.

> ⚠️ *Keep in mind that GitHub Pages forces deployment of user/organization pages to the `main` branch. So if you use `main` for development you need to do one of these:*
>
> - *Change the default branch from `main` to something else, and use `main` as a site deployment directory only:*
>   1. *To create a new branch called `source` run this command: `git checkout -b source main`*
>   2. *Change the default branch in your repository settings ("Branches" menu item) from `main` to `source`*
> - ***Note**: GitHub Pages lets you use any branch for deployment, see this docs page on how to do this. This means you do not necessarily have to change your default branch.*
> - *Have a separate repository for your source code (so `username.github.io` is used only for deployment and not really for tracking your source code). If you go down this route, you will need to add an extra option for `--repo <repo>` (works for https and git urls) in the gh-pages command below.*

```
{
  "scripts": {
    "deploy": "gatsby build && gh-pages -d public -b main"
  }
}
```

> *If you are deploying to branch different to `main`, replace it with your deployment branch's name in the deploy script.*

After running `npm run deploy` you should see your website at `username.github.io`

Deploying to the root subdomain and using a custom domain

If you use a custom domain, don't add a `pathPrefix` as it will break navigation on your site. Path prefixing is only necessary when the site is *not* at the root of the domain like with repository sites.

Note: Don't forget to add your CNAME file to the `static` directory.

Deploying to GitHub Pages from a CI server

It's also possible to deploy your website to `gh-pages` through a CI server. This example uses Travis CI, a hosted Continuous Integration service, but other CI systems could work as well.

You can use the gh-pages npm module to deploy. But first, you need to configure it with proper credentials so that `gh-pages` is able to push a new branch.

Obtain a GitHub token for authenticating with CI

To push changes from the CI system to GitHub, you'll need to authenticate. It's recommended to use GitHub developer tokens.

In GitHub go to your Account settings -> Developer settings -> Personal access tokens, and create a new token that provides the `repo` access permissions.

In Travis's settings for the repository, add a new secret environment variable of the name `GH_TOKEN` with the value of the token copied from GitHub. Make sure you **DO NOT toggle the "display in build logs" setting to on** as the token should remain secret. Otherwise, strangers would be able to push to your repository (a big security issue).

Add script to deploy to GitHub Pages via CI

Update the Gatsby project's `package.json` to also include a `deploy` run script which invokes `gh-pages` with two important command-line arguments:

1. `-d public` - specifies the directory in which the built files exist and will be pushed as a source to GitHub Pages
2. `-r URL` - the GitHub repository URL, including the use of the secret GitHub token (as a secret environment variable) to be able to push changes to the `gh-pages` branch, in the form of `https://$GH_TOKEN@github.com/<github username>/<github repository name>.git`

Here's an example (be sure to update the user and repo names to your own):

```
  "scripts": {
    "deploy": "gatsby build --prefix-paths && gh-pages -d public -r
https://$GH_TOKEN@github.com/lirantal/dockly.git"
  }
```

Update .travis.yml configuration

The following `.travis.yml` configuration provides a reference:

```
language: node_js
before_script:
  - npm install -g gatsby-cli
node_js:
  - "10"
deploy:
  provider: script
  # Note: change "docs" to the directory where your gatsby-site lives, if necessary
  script: cd docs/ && yarn install && yarn run deploy
  skip_cleanup: true
  on:
    branch: main
```

To break down the important bits here for deploying the Gatsby website from Travis to GitHub Pages:

1. `before_script` is used to install the Gatsby CLI so it can be used in the project's run script to build the Gatsby website
2. `deploy` will only fire when the build runs on the `main` branch, in which case it will fire off the deploy script. In the above example, the Gatsby site is located in a `docs/` directory. The script changes into that directory, installs all the website dependencies, and runs the deploy script as was set in the previous step.

Committing and pushing both the `.travis.yml` and `package.json` files to your base branch will be the final step in the process.