Amy Hanlon

Migrating to GitHub Pages using Pelican

February 22 2014

Over the past week I've been dog-paddling through the ocean of misery that is migrating a blog from one host (WordPress) to another (GitHub Pages) and attempting to learn enough CSS and Jinja to handle setting up my site using Pelican. I have no experience with CSS! And my HTML experience is limited to injecting angst into my MySpace profile! And I became aware of Jinja and Pelican's existence about a week ago! So obviously I've drowned myself in 1.5 bottles of my neighborhood liquor store's 2-bottles-of-wine-for-$10 special.

The great part about this whole process is that with Pelican, I can write my blog posts and pages in Markdown (about which I also knew little until last week, but it's *wonderfully easy to learn*.) I am so tired of wrangling with WordPress's built-in editor trying to get my code blocks and in-line code to format correctly. Markdown is a blissful alternative.

There's a plethora of material online on Pelican and GitHub pages, but it is fairly disconnected and presumes a certain level of front-end development experience, of which I have none. Hopefully this post can help others make this transition with less misery.

GitHub Pages Setup

1. Create a GitHub repo following the GitHub Pages instructions (the first step only!)

A note on GitHub Pages: I believe your HTML files (particularly your index.html file) must be in the *main directory* of your git repo for this to work. This will be important later. More detail is given in the **Posting to GitHub** section.

Pelican Setup

1. Install necessary packages

2. Run Pelican quickstart

   This will ask you lots of questions that probably seem foreign. These questions will set up some configuration files that you can later edit with your preferred settings. As an example, here's how I

answered:

```
$ pelican-quickstart
Where do you want to create your new web site? [.]
What will be the title of this web site?
> Amy Hanlon
Who will be the author of this web site?
> Amy Hanlon
What will be the default language of this web site? [en]
Do you want to specify a URL prefix? e.g., http://example.com   (Y/n)
> y
What is your URL prefix? (see above example; no trailing slash)
> http://amygdalama.github.io
Do you want to enable article pagination? (Y/n)
> y
How many articles per page do you want? [10]
Do you want to generate a Fabfile/Makefile to automate generation and publishing?
> y
Do you want an auto-reload & simpleHTTP script to assist with theme and site devel
> y
Do you want to upload your website using FTP? (y/N)
> n
Do you want to upload your website using SSH? (y/N)
> n
Do you want to upload your website using Dropbox? (y/N)
> n
Do you want to upload your website using S3? (y/N)
> n
Do you want to upload your website using Rackspace Cloud Files? (y/N)
> n
```

Now if you type the `tree` command within your blog's main directory, you should see:

```
$ tree
.
├── Makefile
├── content
├── develop_server.sh
├── fabfile.py
├── output
├── pelicanconf.py
└── publishconf.py
```

If you don't have `tree`, you should! It's neat. `brew install tree`. If you're on OSX and don't have Homebrew, you should! It's neat.

I'll briefly explain each of these files/directories:

- `Makefile` tells the command `make` what to do. This file defines commands like

`make devserver`. More information on `make` can be found here. I'll cover more on how to use this command for developing your site in the **Generating Your Site** section.

- `content` is the directory that should house all of your Markdown files. Pelican assumes that your articles/blog posts will be inside this directory. Additionally, there are some special directories you should create within `content`:

  ```
  $ mkdir content/pages
  $ mkdir content/images
  ```

  Pelican by default is configured to know that your pages (i.e. static pages like About Me, Contact, etc) are found within this `pages` directory and that images are found within the `images` directory.

- `develop_server.sh` is a bash script that I believe handles serving your site locally during development (i.e. it serves your site to http://localhost:8000).

- `fabfile.py` is a configuration file for Fabric which allows you to generate your site using the `fab` command. You'll need to `pip install fabric` if you want to use it. Alternatively you can just use `make`.

- `output` is, by default, where Pelican will store your HTML files when you run `pelican content`. This can cause issues which I describe in the section **Posting to GitHub**.

- `pelicanconf.py` houses your Pelican configuration settings.

- `publishconf.py` is like `pelicanconf.py` in that it houses Pelican configuration settings, but is not intended to be used for local development. The reasoning behind having two separate files is described in this Stack Overflow answer.

Exporting Existing Content

This section assumes you have existing content on a WordPress blog. Pelican also has an importer for Dotclear and RSS/Atom feeds. You can skip this section if you don't have existing content living elsewhere that you want to port to your site on GitHub Pages.

1. Export WordPress content to XML
2. Imperfectly convert the XML to Markdown using Pelican
3. Manually export your images from your WordPress Media Library (I know. This sucks.) Move these images to `content/images`.
4. Manually edit the Markdown output (your code blocks, links, embedded images will likely need editing).
5. Move your Markdown files to the `content` directory within your website's main directory. Content

intended to be static pages (i.e. About Me, Contact, etc) should go in the `content/pages` directory. Articles/blog posts should go in the `content` directory.

Pelican Themes

1. Clone the available Pelican Themes into your blog's main directory.

```
$ git clone https://github.com/getpelican/pelican-themes
```

2. Choose a theme you'd like to use. Pelican by default comes with the notmyidea and simple themes. Most other themes have a sample image in the pelican-themes repo to help you decide.

3. After you've chosen a theme, set the THEME variable in your `pelicanconf.py` file to the absolute or relative path to the theme. For example, I'm using the subtle theme and added this line to my `pelicanconf.py` file:

```
THEME = "pelican-themes/subtle"
```

   This method is better than using `pelican-themes` as described here, because it ensures that the Pelican HTML output will reflect any changes you make to the theme (without having to re-install the theme by running the `pelican-themes` command).

Customization

All elements of your theme are customizable! You can change attributes of text like font, size, color, and more in the `main.css` file found in your theme's directory. For example, I've made many edits to the file `pelican-themes/subtle/static/css/main.css`.

Similarly, you can change layouts of your pages (like what shows up in your site nav menu) by exploring the HTML files in the `templates` folder within your theme. There will usually be a `base.html` file (or something similar) that provides the foundation for things like your header and site nav menu that will apply to every page.

There should also be HTML files that serve as templates for specific types of pages. For example, `article.html` defines the basic structure for your articles/blog posts. If you want to change the metadata that displays above article content, you should look there.

If you see something on your website that you want to change, and you're not sure where to look in your theme's CSS/HTML files, right click on the element in the browser and go to "Inspect Element". This will show you where in the HTML the element is (on the left) and what parts of the CSS file define its style (on the right). You can adjust things here in the browser to test out different fonts, colors, etc, but changes you make to the code in your browser will not be reflected in your source files.

Generating Your Site

Once you have markdown files in your `content` folder, navigate to your blog's main directory and run:

```
$ cd blog
$ make devserver
```

`make devserver` does a number of things: first it runs the `pelican` command on your `content` folder to generate HTML for your site using the theme you specify in your `pelicanconf.py` file, and serves your site locally at http://localhost:8000. `make devserver` will also automatically regenerate your site (i.e. run `pelican` on `content` every time you save a change to a content, configuration, or theme file! Just refresh the page in your browser, and you should immediately see the changes. If this doesn't work, it's probably due to the settings you have in your configuration files (`pelicanconf.py`, `Makefile`, and/or `develop_server.sh`).

Posting to GitHub

Recall that you need a repository on GitHub named *username.github.io* (this will be the remote repository for your blog), and that your HTML files need to be in this repository's main directory (not within a subdirectory).

It's intuitive to initialize a local repository for your blog within your blog's main directory, because in addition to posting the HTML, you'd also like to backup your content Markdown files, configuration files, and customized theme. This is a reasonable desire!

However, if you do this, GitHub won't generate your site! It isn't smart enough to know that the HTML files it needs to serve are actually contained within the `output` folder (recall that Pelican by default saves the HTML it generates in this folder).

The best solution I've come up with so far (and please email me if you know of a better solution!) is to create two separate repositories - one inside the `output` directory where Pelican generates your HTML (this repo should have *username.github.io* on GitHub as a remote), and another in your blog's main directory with your source Markdown files (in `content`), theme, and configuration files (this repo should have a different remote on GitHub).

In the terminal, move to the `output` directory, and initialize a git repo. Add a remote pointing to the repo you created on GitHub (called *username.github.io*), add all the files you want to commit, commit, and push changes to the remote repository.

```
$ cd output
$ git init
$ git remote add origin https://github.com/username/username.github.io.git
$ git add --all
$ git commit -m "commit message"
```

```
$ git push origin master
```

If you use this method, you'll want to change the following setting to `False` in your `publishconf.py` file:

```
DELETE_OUTPUT_DIRECTORY = False
```

Otherwise if you use the `publishconf.py` file as your settings file when running the `pelican` command, you'll delete your git repo!

Similarly, don't use the `make clean` command! If you poke around the `Makefile`, you'll see that `make clean` runs `rm -rf output` which will delete all files (including your git repo) in your output folder.

If you accidentally delete the repo in your output folder, it's not a *huge* deal (I've done it like 5 times playing with different commands and settings). Just clone your remote *username.github.io* repo into a new, empty `output` folder, re-generate your site with any changes you've made since your last push to the remote, and then commit and push the changes to the remote:

```
$ cd blog
$ git clone https://github.com/username/username.github.io.git output
$ pelican content
$ cd output
$ git add --all
$ git commit -m "commit message"
$ git push origin master
```

You'll also need to set up another repository for your source content, configuration files, and theme, which is annoying. I added a .gitignore to this repo to ignore the files in the output folder, but that isn't necessary.

Within about 10 minutes of pushing your changes, your site should be up and running! (Later changes should be reflected on your site almost instantaneously.)

Custom Domain Setup

If you have your own domain name that you'd like to use instead of *username.github.io*, you'll need to follow these instructions.

Fin

Feel free to poke around my blog's GitHub repos (beware: there are unpublished draft posts in there). My configuration files in particular might be useful to you.

If any of you Hacker Schoolers have trouble migrating your blog, I'd be happy to help!

tags: pelican blogging wordpress github pages markdown hacker school

Comments

Sandeep Nangia • 3 months ago

Nice post. Thanks. Was looking for using this on my pelican based blog.

I also came across this http://docs.getpelican.com/en/... (see section "Publishing to GitHub") using ghp-import. Looks like, that would make the process to push to github easier.

• Reply • Share ›