SIGN IN

SIGN UP

> *In a rush? Skip to tutorial steps or live demo.*

Did you know Pelicans can eat freaking **turtles**?

No wonder they walk around like badass OGs.

Yeah: pelican the bird is indeed awesome.

But so is Pelican the static site generator. **Especially for whipping up a lean dev blog.**

In this Pelican tutorial, I'll show you how to use the Python-powered static generator to quickly create a sleek blog.

I'll also spruce things up by adding static comments and a search function to the site.

Note: this isn't my first rodeo with static sites—I've shown our readers how to add e-commerce to Jekyll, Hugo, Gatsby, and many others. But today, no e-commerce, no Snipcart. Just a plain, simple tutorial for a dev blog! :)

This post will cover:

- How to create a Pelican blog and change its theme
- How to add the Tipue Search plugin
- How to enable static comments with Staticman
- How to host a Pelican site on GitHub Pages

Ready for take-off?

Can I Pelican? (teammates blackmailed me)

Pelican, a Python-powered static generator

Simply put, Pelican is a neat static site generator (SSG) written in Python.

Like all SSGs, it enables super fast website generation. No heavy docs, straightforward installation, and powerful features (plugins & extendability). Plus, you get all the benefits of a static site: cheap, portable, secure, lightweight, easy to host. As a blogging platform, Pelican also allows you to own all of your content—even comments, thanks to Staticman. No need to rely on trusted third parties like Medium.

With Pelican's well-crafted CLI, you can generate your site, switch themes, and push content without leaving your console. Perfect for a dev blog!

There's also a huge open source collection of themes to choose from.

Sidenote: if you're a Python fan, check out our Django e-commerce tuts with Wagtail!

Pelican Tutorial: a dev blog with comments & search

All right amigos, let's get to the crux of the matter.

Pre-requisite

- Python installed with pip's package manager

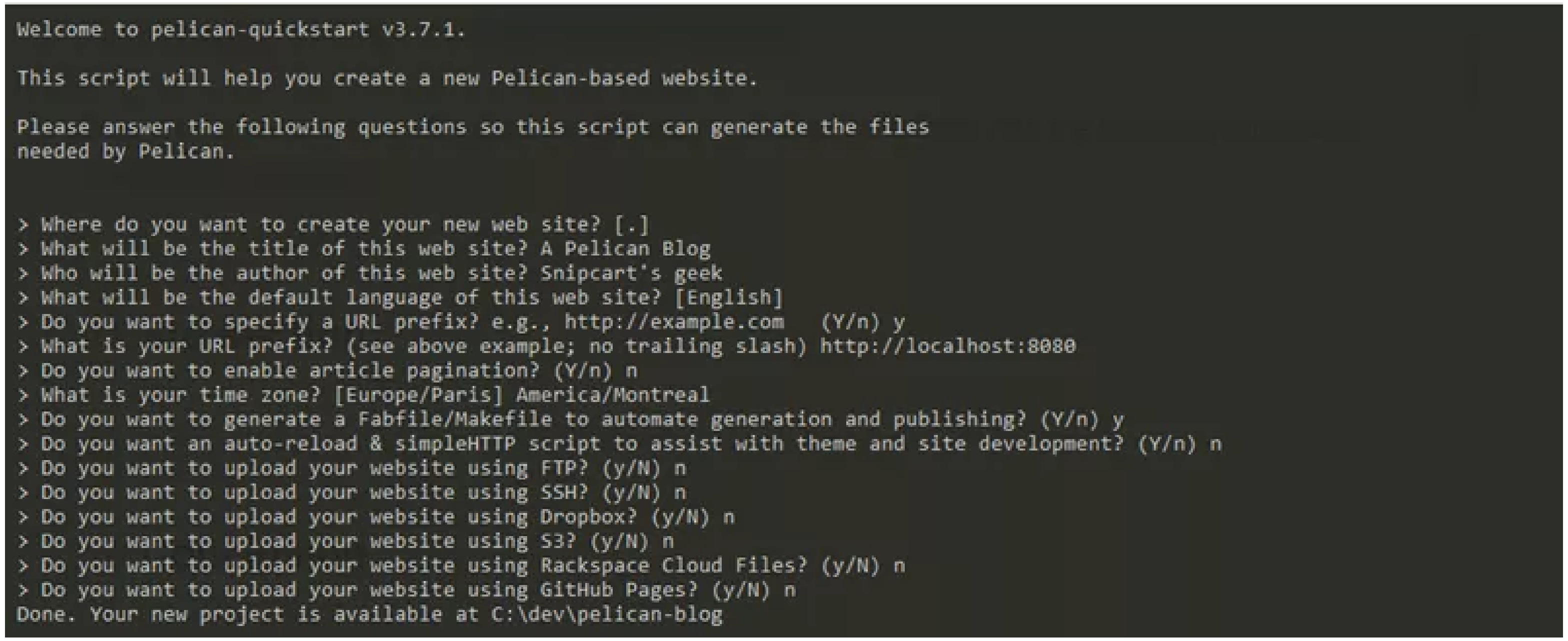
1. Scaffolding the static blog

First thing to do is simply scaffold a website using the CLI. It will give us the file structure needed to customize our setup right away.

Create a new folder for your project and open a console in it. Install Pelican's Python package with: `pip install pelican`

Once it's done, use the CLI straight away to do the scaffolding with: `pelican-quickstart`

Here's how your configuration should look like:



```
Welcome to pelican-quickstart v3.7.1.

This script will help you create a new Pelican-based website.

Please answer the following questions so this script can generate the files
needed by Pelican.


> Where do you want to create your new web site? [.]
> What will be the title of this web site? A Pelican Blog
> Who will be the author of this web site? Snipcart's geek
> What will be the default language of this web site? [English]
> 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://localhost:8080
> Do you want to enable article pagination? (Y/n) n
> What is your time zone? [Europe/Paris] America/Montreal
> Do you want to generate a Fabfile/Makefile to automate generation and publishing? (Y/n) y
> Do you want an auto-reload & simpleHTTP script to assist with theme and site development? (Y/n) n
> 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
> Do you want to upload your website using GitHub Pages? (y/N) n
Done. Your new project is available at C:\dev\pelican-blog
```

For the demo, I wanted to play around a bit with some themes, so I chose to clone the whole `pelican-themes` repo. You can do so with `git clone git@github.com:getpelican/pelican-themes.git` in the folder of your choice.

2. Creating blog content in Markdown

To make this demo less contrived, I used real, "open source" Aeon content.

You could decide to generate a new folder to keep content organized. But for the sake of keeping this demo simple, I created everything directly in the content folder.

Fire up your favorite text editor and open the content folder. First file name: `are-you-just-inside-your-skin-or-is-your-smartphone-part-of-you.md`. As you can see, the file extension is `.md`, so we will use the Markdown format to define our content. A Markdown file is declared with metadata (at the start of the file), followed by the actual content. Pelican's files support many metadata fields—it's worth reading their docs about this here.

Now, back to the file you just created. Open and fill it up with actual content (our Aeon source):

```
Title: Are ‘you’ just inside your skin or is your smartphone part of you?
Date: 2018-02-26
Category: Psychology
Slug: are-you-just-inside-your-skin-or-is-your-smartphone-part-of-you

In November 2017, a gunman entered a church in Sutherland Springs in Texas, where he killed 26 people and woun
```

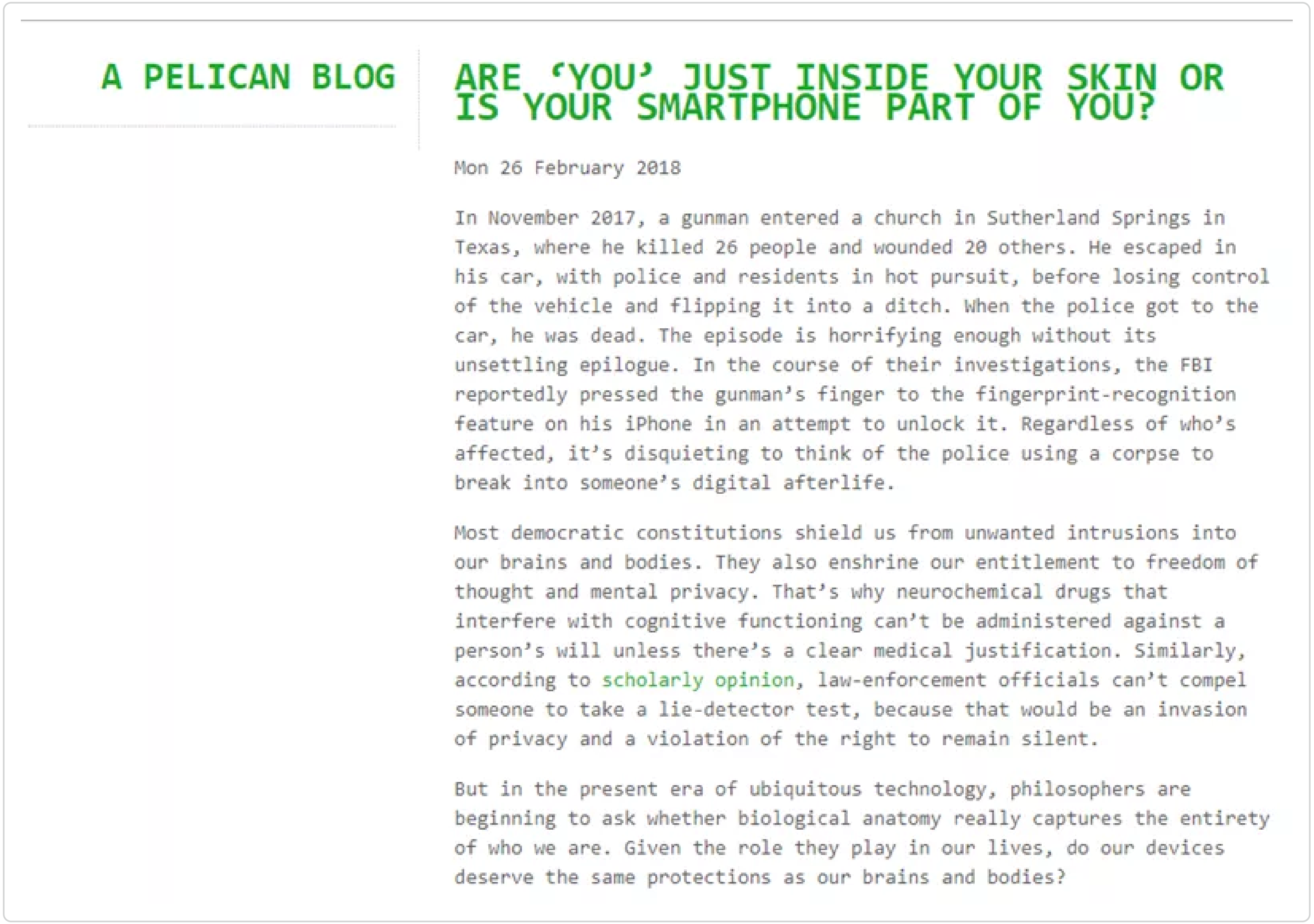
You can repeat this step with all the content you need. Once you've got your content, you'll be ready to generate the blog.

3. Changing the Pelican theme

Go into the folder cloned earlier, and copy your chosen theme in your project's root folder. Then, in the same folder, run `pelican-themes -i {your_theme_name}` .

You're all set to give your site a first shot. To generate it, again, in the same folder, run `pelican -t {your_theme_name}` . This will generate your website with the specified theme and put it inside the output folder.

Simply serve with whatever suits you after. I opted for Node's `http-server` . Here's my result at this point, using the monospace theme:


A PELICAN BLOG

ARE ‘YOU’ JUST INSIDE YOUR SKIN OR IS YOUR SMARTPHONE PART OF YOU?

Mon 26 February 2018

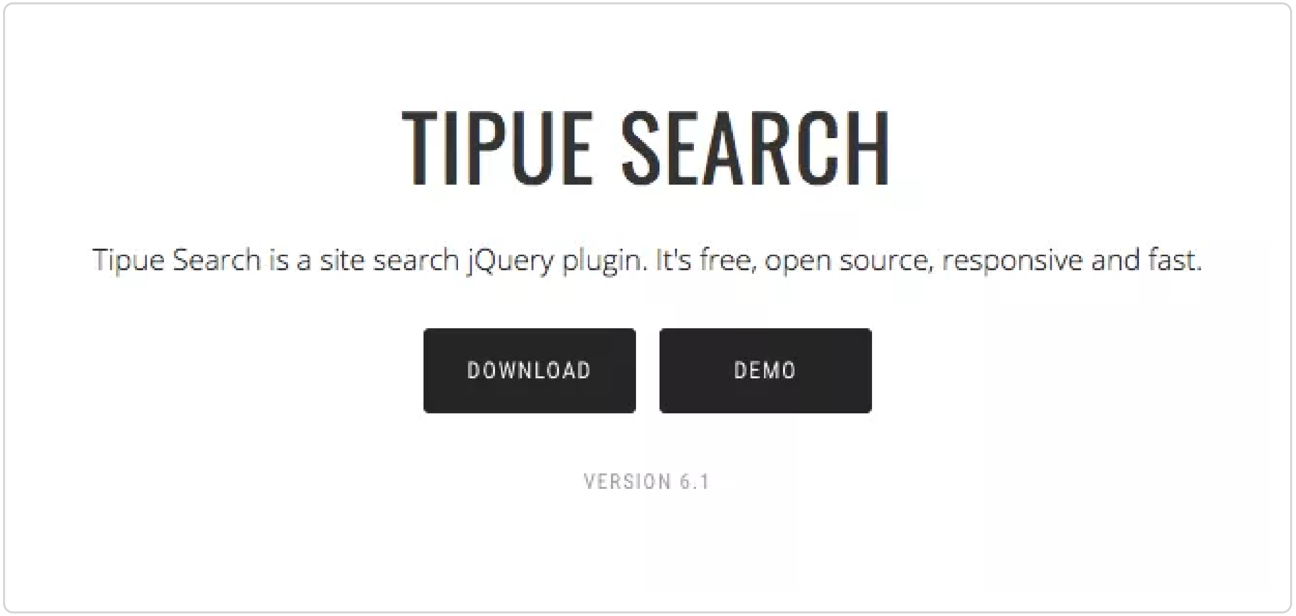
In November 2017, a gunman entered a church in Sutherland Springs in Texas, where he killed 26 people and wounded 20 others. He escaped in his car, with police and residents in hot pursuit, before losing control of the vehicle and flipping it into a ditch. When the police got to the car, he was dead. The episode is horrifying enough without its unsettling epilogue. In the course of their investigations, the FBI reportedly pressed the gunman’s finger to the fingerprint-recognition feature on his iPhone in an attempt to unlock it. Regardless of who’s affected, it’s disquieting to think of the police using a corpse to break into someone’s digital afterlife.

Most democratic constitutions shield us from unwanted intrusions into our brains and bodies. They also enshrine our entitlement to freedom of thought and mental privacy. That’s why neurochemical drugs that interfere with cognitive functioning can’t be administered against a person’s will unless there’s a clear medical justification. Similarly, according to scholarly opinion, law-enforcement officials can’t compel someone to take a lie-detector test, because that would be an invasion of privacy and a violation of the right to remain silent.

But in the present era of ubiquitous technology, philosophers are beginning to ask whether biological anatomy really captures the entirety of who we are. Given the role they play in our lives, do our devices deserve the same protections as our brains and bodies?

Note: if you want to customize & create Pelican themes, check out its Python templating language Jinja2

4. Adding the Tipue search plugin


TIPUE SEARCH

Tipue Search is a site search jQuery plugin. It's free, open source, responsive and fast.

Now now, I know there are tons of way to handle search on a static site. Third parties—Algolia et al. Server-side searches. Client-side searches. Truth is most of these are overkill for my humble demo.

¯_(ツ)_/¯

So to add this feature to the blog, we'll use a Pelican-specific search plugin. There are many plugins in this sub-repo, but the one of interest to us here is `tipue-search`.

Install the required Python package with `pip install beautifulsoup4`. Next, clone the project's folder, and register it inside your `pelicanconf.py` file. You'll be able to do so simply by adding the following line: `PLUGINS = ['tipue_search.tipue_search']`. If you re-generate your website, you'll see a new file; `tipuesearch_content.json`. That's the static content the search will use. Now, you only need to modify your theme's templates to add the search.

Hop into `monospace/base.html`—the template used for every page. There, add both `jQuery` and `tipue-search` with the following HTML:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/Tipue-Search/5.0.0/tipuesearch.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/Tipue-Search/5.0.0/tipuesearch_set.js"></script>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/Tipue-Search/5.0.0/tipuesearch.css"/>
```

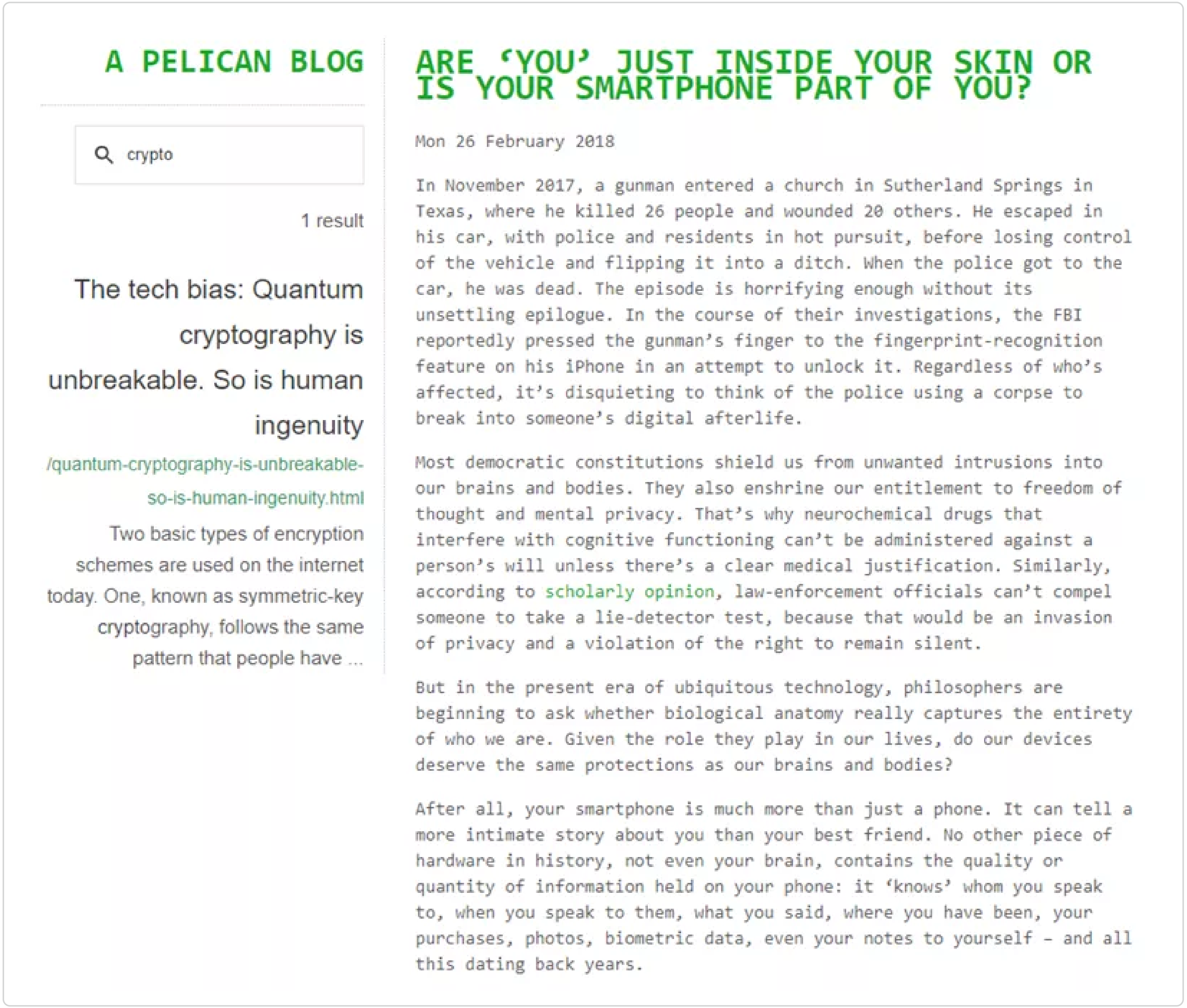
Now add this code to the `<div id ="sidebar">` section:

```
<form action="/">
    <div class="tipue_search_right"><input type="text" name="q" id="tipue_search_input" pattern=".{3,}" title=
    <div style="clear: both;"></div>
</form>

 <div id="tipue_search_content"></div>
```

And the following JavaScript, again in the same `base.html` file:

```
<script>
        $(document).ready(function() {
             $('#tipue_search_input').tipuesearch({
                  'mode': 'json',
                  'contentLocation': 'tipuesearch_content.json'
             });
        });
    </script>
```

That's it! You should have static search directly on our website now. Rebuild your blog and see the result. Here's how my search box & results look:



A PELICAN BLOG

crypto

1 result

The tech bias: Quantum cryptography is unbreakable. So is human ingenuity

/quantum-cryptography-is-unbreakable-so-is-human-ingenuity.html

Two basic types of encryption schemes are used on the internet today. One, known as symmetric-key cryptography, follows the same pattern that people have ...

ARE ‘YOU’ JUST INSIDE YOUR SKIN OR IS YOUR SMARTPHONE PART OF YOU?

Mon 26 February 2018

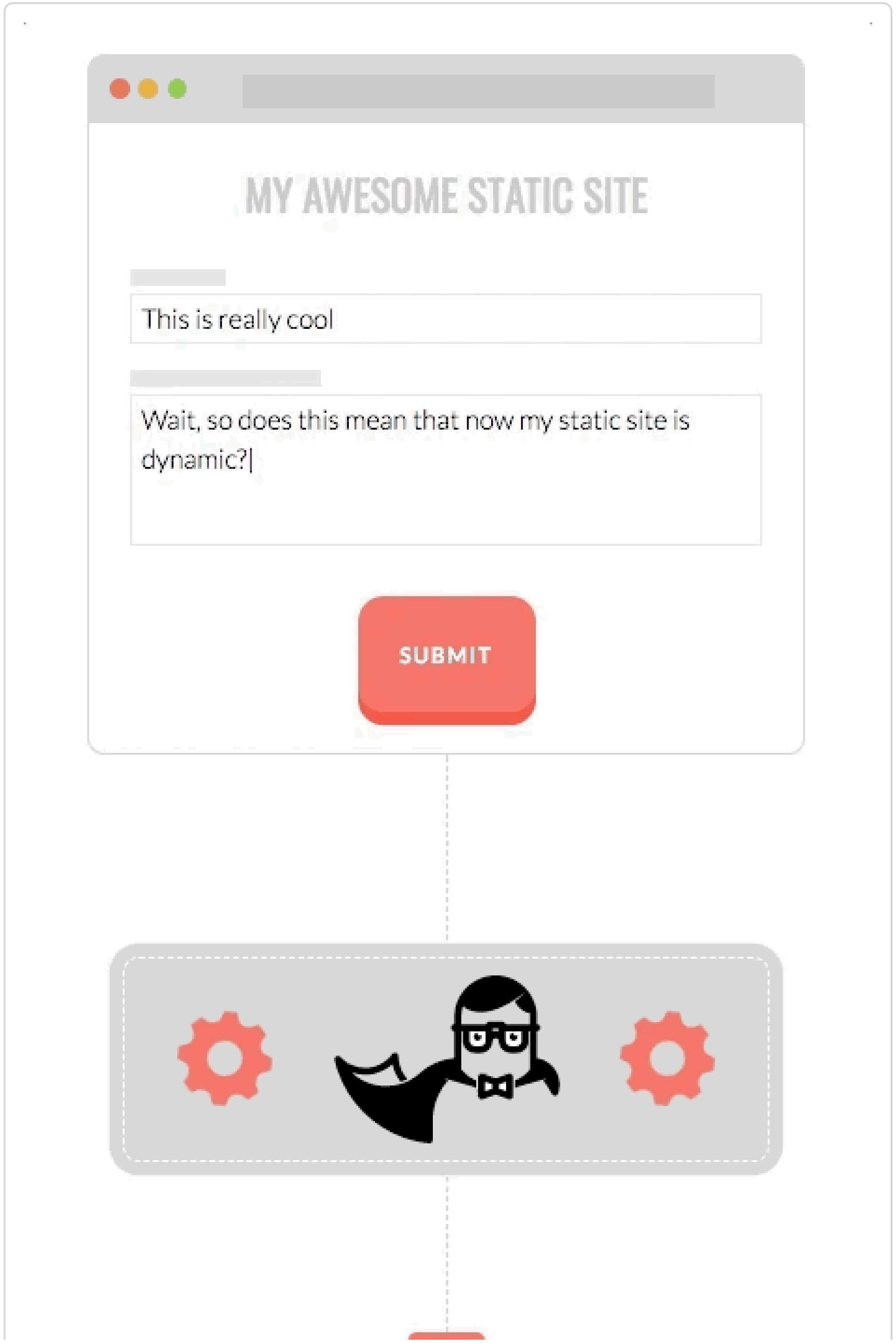
In November 2017, a gunman entered a church in Sutherland Springs in Texas, where he killed 26 people and wounded 20 others. He escaped in his car, with police and residents in hot pursuit, before losing control of the vehicle and flipping it into a ditch. When the police got to the car, he was dead. The episode is horrifying enough without its unsettling epilogue. In the course of their investigations, the FBI reportedly pressed the gunman’s finger to the fingerprint-recognition feature on his iPhone in an attempt to unlock it. Regardless of who’s affected, it’s disquieting to think of the police using a corpse to break into someone’s digital afterlife.

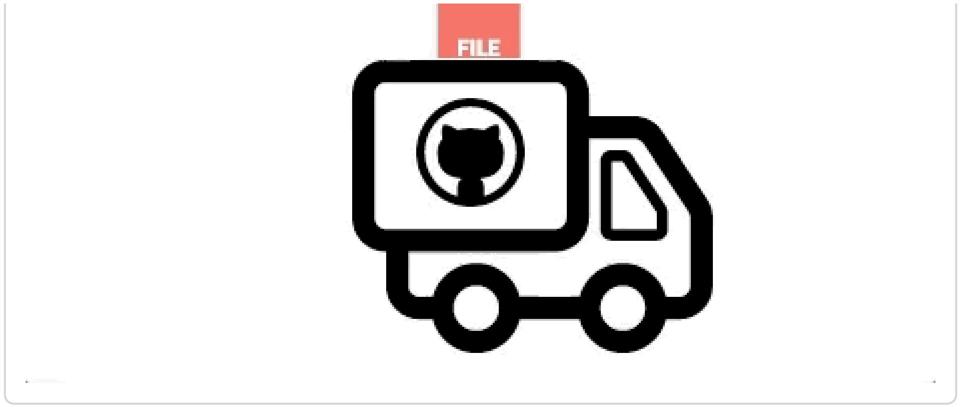
Most democratic constitutions shield us from unwanted intrusions into our brains and bodies. They also enshrine our entitlement to freedom of thought and mental privacy. That’s why neurochemical drugs that interfere with cognitive functioning can’t be administered against a person’s will unless there’s a clear medical justification. Similarly, according to scholarly opinion, law-enforcement officials can’t compel someone to take a lie-detector test, because that would be an invasion of privacy and a violation of the right to remain silent.

But in the present era of ubiquitous technology, philosophers are beginning to ask whether biological anatomy really captures the entirety of who we are. Given the role they play in our lives, do our devices deserve the same protections as our brains and bodies?

After all, your smartphone is much more than just a phone. It can tell a more intimate story about you than your best friend. No other piece of hardware in history, not even your brain, contains the quality or quantity of information held on your phone: it ‘knows’ whom you speak to, when you speak to them, what you said, where you have been, your purchases, photos, biometric data, even your notes to yourself – and all this dating back years.

5. Adding static comments with Staticman


MY AWESOME STATIC SITE
This is really cool
Wait, so does this mean that now my static site is dynamic?|
SUBMIT



Most people default to Disqus to handle comments on their sites. There's also a Disqus static comment plugin for Pelican. Now I have no personal beef with Disqus. But I'd be hard-pressed to ignore the number one issue related to trusting this third party, even in its "static" form:

→ **Comments are hosted on Disqus servers--no ownership of content.**

Staticman solves all of the above. That and, of course, a simple problem most static sites suffer from: **adding forms for user-generated content without a backend**. Staticman is perfect for reader comments, but also for voting systems or user reviews. For more details on its superpowers, check out this full post by my colleague Jean-Seb aka Mumbledore.

Disclaimer: Snipcart sponsors the Staticman open source project.

So for the sake of transparency, here's a list of static comments alternatives you might consider.

Now for the technical part:

I will skip the default configuration of Staticman as it is really straightforward and well explained here.

Once, you have a repo with the app as a contributor running, you'll need to add a configuration file and the necessary templates to send and render data. Create a staticman.yml file with the following content:

```
comments:
  allowedFields: ["name", "email", "url", "message"]
  branch: "master"
  commitMessage: "Add Staticman data"
  filename: "entry{@timestamp}"
  format: "yaml"
  moderation: false
  name: "pelican-blog.netlify.com"
  path: "content/comments/{options.slug}"
  requiredFields: ["name", "email", "url", "message"]
  generatedFields:
    date:
      type: date
      options:
        format: "timestamp-seconds"
```

Now, hop in `monospace/templates/article.html` and add the following just after the `{{ article.content }}` line:

```
<div id="comment-post">
        <form method="POST" action="https://api.staticman.net/v2/entry/snipcart/pelican-blog/master/comments">
            <input name="options[redirect]" type="hidden" value="{{ SITEURL }}/{{ article.slug }}">
            <input name="fields[url]" type="hidden" value="{{ article.slug }}">
            <div><label>Name: <input name="fields[name]" type="text"></label></div>
            <div><label>E-mail: <input name="fields[email]" type="email"></label></div>
            <div><label>Message: <textarea name="fields[message]"></textarea></label></div>

            <button type="submit">Submit</button>
        </form>
    </div>
```

This is the form used to send comments to Staticman. The latter will push then include them in your repo. This way your website will be able to render them without any external calls.

To first render them, let's expose these comments in JSON to your templates, since they are in YML at the moment. First, download the `pyyml` python package with: `pip install pyyaml` .

Once this is done, go in `pelicanconf.py` , this is where you'll expose comments.

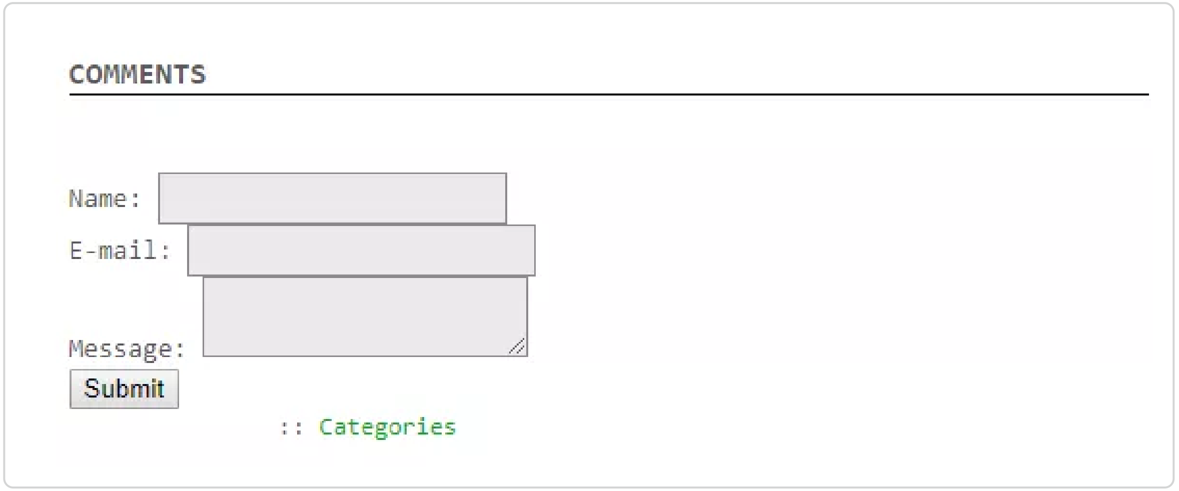
Add these following lines wherever you want:

```
#Staticman Comments
commentsPath = "./content/comments"

def ymlToJson(file):
    with open(commentsPath + "/" + file) as stream:
        return yaml.load(stream)

commentsYML = [f for f in listdir(commentsPath) if isfile(join(commentsPath, f))]
COMMENTS = list(map(ymlToJson, commentsYML))
```

This will load every .yml comment files, parse them into JSON, and expose them as an array. Any all-caps variable in this file will be exposed inside Jinja2 templates, which means you'll have access to the comments in any template. What you want to do now is render the comments on the matching articles. In `article.html` , add this section just between the `{{ article.content }}` line and the form we just added:

```
    <div>
        <h3 id="comments">COMMENTS</h3>
        {% for comment in COMMENTS %}
            {% if comment.url == article.slug %}
                <p class="comment">{{ comment.name }}: {{ comment.message }}</p>
            {% endif %}
        {% endfor %}
    </div>
```

Aaaand a quick look at our static comments now:



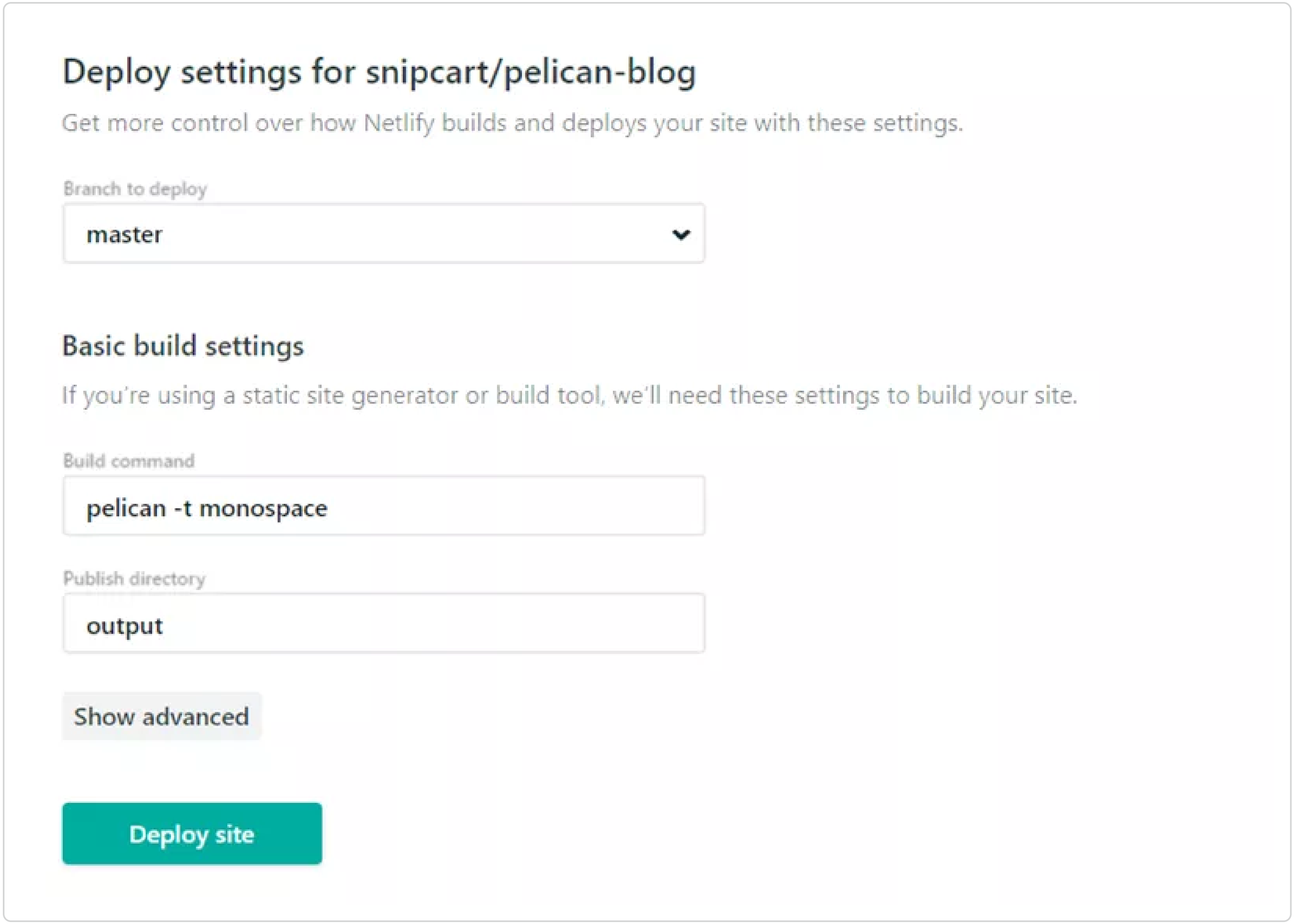
Now let's host that static blog!

6. Hosting your Pelican blog on Netlify

I would have liked to host everything on GitHub Pages, as it's a good fit with Pelican. But since I needed to rebuild the project once a comment is pushed to keep the website updated, I decided to go for Netlify. Once a new comment is pushed, Netlify will be notified, rebuild the website, and host it with the new comment. To do so, add a `requirements.txt` to your project's folder and add these lines to it:

```
pelican
pyyaml
markdown
beautifulsoup4
```

This is the file that will be used by Netlify to download the project's dependencies. Now, push your code to a repo and hit netlify.com. Once you're logged in, click the `new site from GIT` and choose your project's repo with these settings:



Deploy settings for snipcart/pelican-blog

Get more control over how Netlify builds and deploys your site with these settings.

Branch to deploy: master

Basic build settings

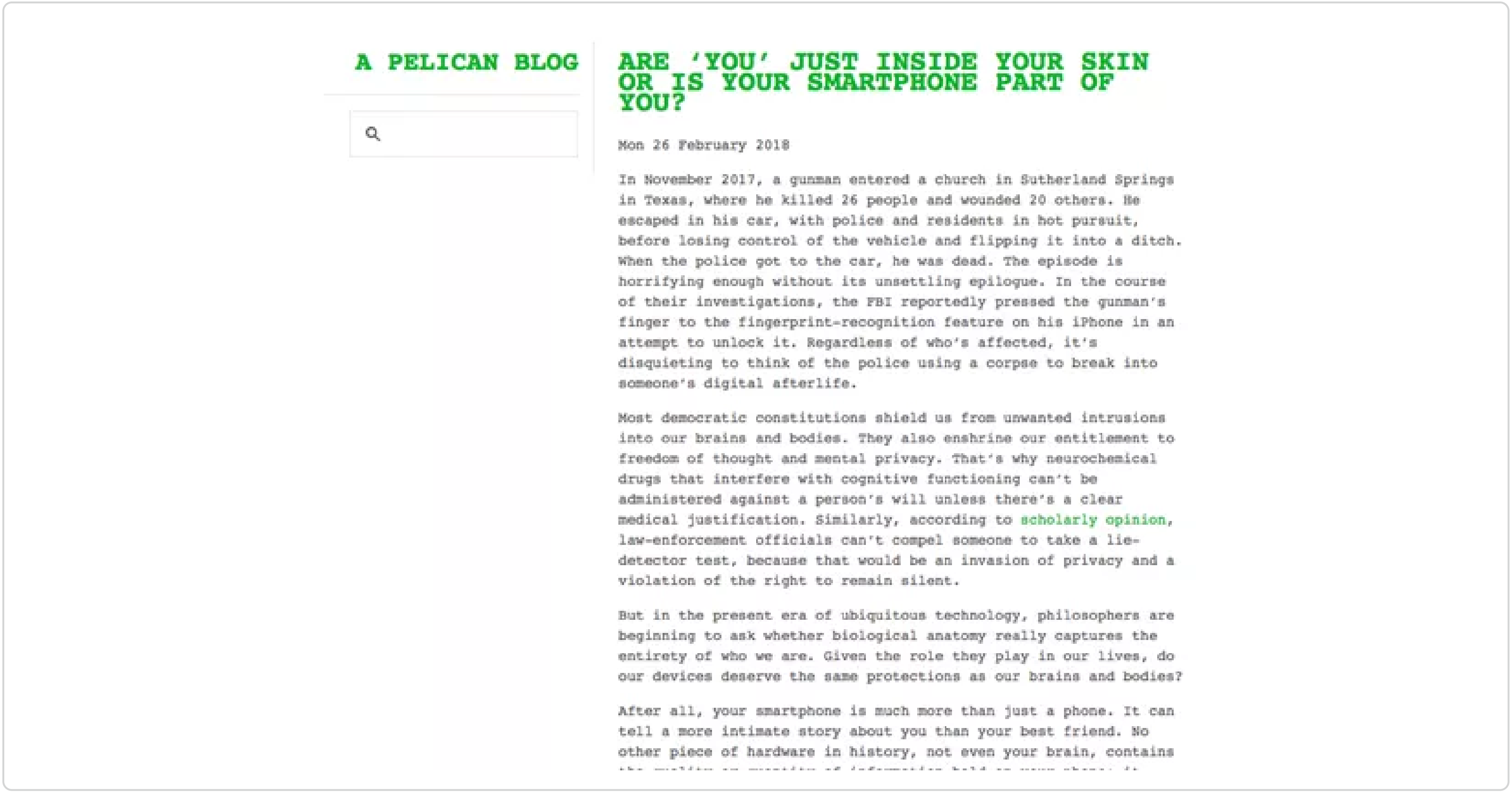
If you're using a static site generator or build tool, we'll need these settings to build your site.

Build command: pelican -t monospace

Publish directory: output

The deploy will start right away, and your website should be live in a minute! Note that the site has to rebuild for new comments to appear.

Live demo & GitHub repo



A PELICAN BLOG

ARE 'YOU' JUST INSIDE YOUR SKIN OR IS YOUR SMARTPHONE PART OF YOU?

Mon 26 February 2018

In November 2017, a gunman entered a church in Sutherland Springs in Texas, where he killed 26 people and wounded 20 others. He escaped in his car, with police and residents in hot pursuit, before losing control of the vehicle and flipping it into a ditch. When the police got to the car, he was dead. The episode is horrifying enough without its unsettling epilogue. In the course of their investigations, the FBI reportedly pressed the gunman's finger to the fingerprint-recognition feature on his iPhone in an attempt to unlock it. Regardless of who's affected, it's disquieting to think of the police using a corpse to break into someone's digital afterlife.

Most democratic constitutions shield us from unwanted intrusions into our brains and bodies. They also enshrine our entitlement to freedom of thought and mental privacy. That's why neurochemical drugs that interfere with cognitive functioning can't be administered against a person's will unless there's a clear medical justification. Similarly, according to scholarly opinion, law-enforcement officials can't compel someone to take a lie-detector test, because that would be an invasion of privacy and a violation of the right to remain silent.

But in the present era of ubiquitous technology, philosophers are beginning to ask whether biological anatomy really captures the entirety of who we are. Given the role they play in our lives, do our devices deserve the same protections as our brains and bodies?

After all, your smartphone is much more than just a phone. It can tell a more intimate story about you than your best friend. No other piece of hardware in history, not even your brain, contains

> *See the live demo here.*

> *See GitHub repo here.*

Closing thoughts

This demo took me about 1-2 hours, thanks to the Pelican documentation and Staticman's simplicity.

I really enjoyed playing with Pelican! I had not worked with a static generator for a while—missed it! Docs were great and concise. The challenge was mostly with Python. I really didn't write much of it, but I had never developed with Python before, so it was fun to try!

I would have liked to push the search further, at the moment it's easy to use and setup but it wouldn't be optimal for a lot of blog entries. The first thing I would do to scale this is outsource the search to something like Algolia. It would be much faster and wayyyy more powerful/scalable than what we have at the moment.

Happy coding!

If you've enjoyed this post, please take a second to share it on Twitter. Got comments, questions? Hit the section below!

Submit to HN   Contribute to the blog

Maxime Laboissonniere
FULL STACK DEVELOPER

Max was the first dev hire for Snipcart back in 2016. Since then, he has stood out by his curiosity towards new technologies. He's the one that introduced the team to Vue.js, for instance. In his 4 years experience as a developer, he’s mastered JavaScript and its ecosystem, as well as C#. These days, he likes to explore Elixir, Clojure, ELM, RxJS, and data science--when he doesn’t have his nose in a book.

Follow him on Twitter.

Suggested posts:

- E-Commerce for Django Developers (with Wagtail Tutorial)
- Carlos in huis: Neat Django E-Commerce with Wagtail CMS
- The State of Node.js & JavaScript for Backend Development
- Launch a Vue.js Blog in Less Than 2 Hours [Live Demo]
- Static Site E-Commerce: Integrating Snipcart with Jekyll

Chad M • a month ago

Regarding the Github pages / comments issue that you mentioned briefly. Can you comment on what goes wrong or what you attempted to get Github pages to rebuild after a comment is posted? I am wonder in a post-commit hook would do the job or if you tried that and it isn't worth the effort.

• Reply • Share ›

Keith Sanders • a year ago • edited

This could have been just me... but to get the tipue search to work when editing the base.html file, I had to place the jQuery script (the one that includes "$(document).ready(function() {" prior to the other script includes. Otherwise, I would get an error about tipuesearch was not a defined function.

• Reply • Share ›

Maxime LaBoissonniere → Keith Sanders • a year ago

Hi Keith!

First, sorry for the delay, this comment slipped us completely.

That's interesting, I tried reproducing the problem but without success.

You could also have used the window.onLoad native function instead of the jQuery one to make this work.

Cheers,
Max

• Reply • Share ›