

Ishill Example Site presents

Ishill Monograph

A Monograph By

Brendan Francis O'Connor

And Published

Sep 23, 2026

This monograph demonstrates how to use Ishill to create zines easily, and attempts to explain why you'd want to. It's documentation for an active project as well as an example of using that project successfully, so it may contain mistakes, but you can rest assured that all of the mistakes were generated by humans. If you have questions, comments, or concerns, please open an issue at <https://codeberg.org/ussjoin/ishill-examplesite>, or send an email to ishill@ussjoin.com.

Release history:

- September 23, 2026: Initial release of zine, tied to release 1.0 of Ishill and the Ishill example site.

What's Ishill? Why does it exist? How do I use it?

Brendan Francis O'Connor

Hello ideally-human person! You've probably found this at Codeberg [1] or GitLab [2] or GitHub [3], or possibly on a bench somewhere in meatspace. Read on to learn what you've found!

Note: this monograph is always going to be incomplete, because the set of all things one can do with software is vast. We hope it's useful to you even as-is. You're welcome to reach out for further assistance using the Issues function on Codeberg [4], and I'll do my best to aid you (and add the things that confused you to this document so that they can help others!).

Quick Definitions

- Hugo: it's a tool that makes websites, and it's what Ishill is built on. It's available at <https://gohugo.io/>. You can run it on Windows, Mac, or Linux.
- Ishill: it's a theme for Hugo that uses Hugo to handle rendering formatted text, and some other tools to turn that text into PDFs and "impress" them into printable zines. It's available at `/ussjoin/ishill` on any of <https://codeberg.org>, <https://gitlab.com>, or <https://github.com>. It only supports being built on Mac or Linux; it's technically possible to write a version of this that would run on Windows, but I don't currently intend to do that; that said, you can *still use it from Windows* if you use a code hosting platform to "build" the zines for you. Instructions on that are in this zine!
- Ishill Example Site: it's the code to build a website using the Ishill theme on the Hugo platform. It, along with this zine, forms the documentation for how precisely to use Ishill to make different kinds of zines. The code is available at `/ussjoin/ishill-examplesite` on any of <https://codeberg.org>, <https://gitlab.com>, or <https://github.com>. A view of the website generated by this code is available at any of four places, currently, primarily to show how it can be deployed there (with instructions in this zine!):
 - <https://ishillexamplesite.nfshost.com/> (on NearlyFreeSpeech.net, a low-cost static site hosting platform);
 - <https://ussjoin.gitlab.io/ishill-examplesite/> (on GitLab Pages, a free static site host attached to the GitLab commercial code hosting platform);
 - <https://ussjoin.github.io/ishill-examplesite/> (on GitHub Pages, a free static site host attached to the GitLab commercial code hosting platform owned by Microsoft); and

□ <https://ussjoin.codeberg.page/ishill-examplesite/> (on Codeberg Pages, a free static site host attached to the Codeberg nonprofit code hosting platform).

The Pitch (Short)

Ishill (pronounced [5] EE-sheell) is a publishing stack, built on the excellent Hugo static site generator, to enable people to go quickly from “I have an idea I want to hand out as a zine” to “I have a stack of zines.” It is named in honor of Joseph Ishill [6], a printer.

The Pitch (Human)

Want to write/make zines but don’t have Illustrator or InDesign, and aren’t a good enough artist to do twenty pages of lettering by hand? Want to use a computer to make longer content than the (lovely, if short) minizine [7] format? Optionally, want to simultaneously publish to print and the web, so that people who come across one of your zines can find more?

Ishill is for you. It’s a zine publishing toolkit built on top of a blog toolkit, Hugo, that lets you quickly create for web, PDF, and zine. If you want to, you can use it to publish your content *for free* on the Internet, on a site only you control (no Facebook/Instagram/etc. nonsense), and with no lock-in; you can move it anywhere at a moment’s notice. You can use a domain name you own, or use a free subdomain of either of two gigantic code providers (GitLab and GitHub) (or a reasonably-sized nonprofit code provider, Codeberg, which offers the same functionality), and use their computer time and web servers to distribute your content to the world. You can also host it on any web host that can host static sites, which is... pretty much all of them, for either free or nearly no money. You can also publish it on Tor [8], or even keep it offline and just use it as a fast Markdown-to-PDF pipeline.

Hugo is a “static site generator,” so while it takes a teensy bit of tech knowledge to get it set up (we’ll help you! That’s what this zine is), you don’t need to worry about securing it or getting hacked; most hosts for static sites don’t let you configure anything that could get you in trouble, which is just how you should like it.

Sound good? Read on.

The Pitch (if you spend a lot of time working with tech)

Ishill is a zine+web publishing stack built on top of Hugo [9], the static site generator, with some add-on tools to do Markdown-to-zine efficiently. It has “happy path” deployments to

CodeBerg Pages [10], GitLab Pages [11] (using GitLab CI/CD), and GitHub Pages [12] (using GitHub Actions), but since it's basically Hugo plus a couple of shell scripts, you can run it anywhere *nix-y with a modicum of tech knowledge. If you need a full tech stack:

- Hugo [13] for core Markdown-to-HTML rendering
- WeasyPrint [14] for HTML/CSS-to-PDF rendering
- PDFImpose [15] for imposition (arranging the order of pages so you can print them double-sided and fold them into a zine)
- Some Shell Scripts [16] to tie it together.

Why?

Why do this? A few reasons.

- For one thing, it's *obnoxiously* difficult to go from Markdown (or any simple text format) to zine. If it could fit on a minizine [17] (so less than half a normally-printed page), great. If it can fit on one printed page, front and back, one can make a four-half-page brochure with a bit of mucking about in Word, Pages, or similar. After that, it's not simple unless you have Adobe InDesign [18] or other dedicated (and very expensive) prepress software; as far as I've found, there's no "happy path" for making the transition.

□ OK, technically The Anarchist Library Bookbuilder [19] has a stack to do this as well, but it doesn't use Markdown or anything else popular, instead defaulting to something only known to the kind of people who use Emacs.

Ain't nobody got time for that, we want to publish zines!

□ Note that "not simple" doesn't mean "impossible;" folks comfortable with stacking command-line tools can get it done—but there's nothing turnkey to make it happen.

- For another, Travis Goodspeed [20]'s groundbreaking and iconic International Journal of Proof of Concept or Get the Fuck Out (POC||GTFO) [21] introduced hackers to the idea that our research wasn't immune from memoryholing, at the end of the day; a lot of people, myself included, learned about samizdat [22] from Travis. Zines have a lot of influences, but samizdat is certainly one of them.

□ I don't think it's a real surprise in 2026 that memory holing research and communication is at top of mind for many communities.

- Why make a website for printed zines? Personally, when I find a zine that I like, I try to find the author and/or publishing collective to find more things that might be enjoyable, just as I do with authors. I wanted to make it as easy to publish a zine as a single blog post, and as easy to start a zine publishing house as it is to start a blog site. If you don't want the website parts, this is a bit over-engineered, but you can still use it to generate your zines and print them (which is, as noted above, a major pain).

Neat Features of Ishill

There are a few neat things that come from using blog software to host and render zines. If you only care about using Ishill as a tool to go from Markdown to printable PDF, you may not care about all of these, but I thought they were useful.

Endnotes from Links

One of the goals of Ishill is to generate quality printed matter from normal Markdown, because Ishill generates both a website and a set of printable zines at the same time. One issue arises: while normal PDFed HTML (e.g., from “Print to PDF”) will preserve links (that is, the same words will be clickable on the website and in the PDF), once you print the PDF, that ability goes away; you can’t click a page. To make sure that the context provided by a link isn’t lost, Ishill takes links and replaces them with numbered endnotes, *in the PDFs only*; the web pages preserve the normal links.

Deployable Anywhere

Websites come in two very broad categories; “static” websites, where content cannot change from moment to moment (or in response to who’s viewing it), and “dynamic,” where it can. This is closely related to (but technically separate from) whether the site uses JavaScript, where part of the site runs on your computer/phone. The vast, vast majority of sites people use in 2026 are “dynamic” *and* use JavaScript, but neither is required for a website, particularly one that isn’t monetized. Similarly, most “easy ways to make websites” (WordPress, Squarespace, etc.) are dynamic.

Ishill is not dynamic. It’s just a theme for the Hugo [23] static site generator, and once Hugo generates a site, it’s just a set of files. This means it can be hosted not just on expensive “dynamic” host providers, but on a huge array of “static web hosts.” Those are often free (supported by hypervisor corporations, like GitHub Pages [24], or other code hosts, like Codeberg Pages [25] or GitLab Pages [26]) or very low cost. We use NearlyFreeSpeech.net, whose prices we like (and whose politics are excellent [27]), as our paid example site host. Many domain registrars have some sort of static site hosting for free when you register a domain. You get the idea.

Ishill also does not use JavaScript. That’s not a Hugo requirement, we just don’t wish to do so. You can certainly add JavaScript to it, but there’s no need to do so. (Most JavaScript is used to track and monetize visitors, so, frankly, we don’t want to do it and we don’t have to.)

Nothing External

Since there’s nothing that requires that users run JavaScript or access any other server (this is atypical for websites in 2026, with Google Fonts, CDNs, tons of web trackers, ... but not actually difficult to do if I’m not trying to market to you and sell your data to brokers!), this means

anyone can preserve your site forever, as long as the HTML, CSS, and PDF formats are still readable (they've been around since 1990, 1996, and 1993 respectively, and two are written in ASCII, a format written in 1963; it's not a clay tablet, but it's not bad for something intangible). There's nothing by default that can go offline and make your site stop working. That's pretty neat for the next feature:

Self-Archiving

Ishill's build process (the thing that turns it from a pile of Markdown files into a website and bunch of zine PDFs) takes a copy of the whole site and puts it in a zip file, linked in the website footer. This means that anyone visiting the site can download the whole site, read it offline, or even put up a copy should yours go offline. Since there's nothing that requires that users run JavaScript or access any other server (this is atypical for websites in 2026, with Google Fonts, trackers, ... but not actually difficult to do if I'm not trying to market to you and sell your data to brokers!), this means anyone can preserve your site forever, as long as the HTML, CSS, and PDF formats are still readable (they've been around since 1990, 1996, and 1993 respectively, and two are written in ASCII, a format written in 1963; it's not a clay tablet, but it's not bad for something intangible). (And see the note above about samizdat.)

If you want to turn this off, set `enableArchive = false` in the `[params]` section of your `config.toml` file.

NB: the process creates a copy of your finished site, not the Markdown and build scripts used to create it. If you want to keep a backup of that (you should), make sure to back up your site's folder on your computer—either by creating your own fork on a Git host (Codeberg, GitLab, GitHub, whatever), or just by copying it elsewhere from time to time.

RSS

Remember Google Reader [28]? The same technology (RSS/Atom, invented by someone driven to suicide to punish him for not believing in copyright enough [29]—in the same way now being done by every LLM corporation, which seems a bit on the nose even for 2026) still exists, and there are many tools (paid and free, cloud-hosted or self-hosted or desktop apps or mobile apps) to allow users to follow the feeds. Every new zine in Ishill will appear in your RSS feed, and any feed reader can find your RSS feed (technically an Atom feed, but it doesn't matter) given your site's URL. (It's `yoursite.place/index.xml`, FWIW.)

You Don't Have to Have a Website

Ishill uses a blog platform and generates web pages that you can send to a web host; however, it also turns Markdown files into viewable and (saddle-stitched, letter-size) printable PDFs. You can just use it for that feature and not use the website, if you wish. It's a bit tricky to pull the “make this into a zine” thing out of the rest of the website tooling for various boring reasons, so it's not a tool I can release on its own, but if you just want to make printed zines

and not deal with the website, you can follow the guide and use the tool without the “upload it to a web host” part and you’ll be just fine.

Easy to Change Styles

Hugo makes it very easy to change anything about the theme; you add a file with the same name to your site’s folder, copy the theme file in there, and then make any changes you want. If you only want to add CSS, put the file in `assets/css/` and add the filename to the `[params].customcss` array in your `config.toml`. If you only want to change the typeface or font, look at `assets/css/newtypeface.css` for how to do that (font files will go in `static/fonts`).

While you can make a local copy of the theme folder itself and edit the theme that way, I recommend you don’t; if you do, it will be difficult to upgrade the base theme (and get any bug fixes) without losing your changes in the future.

Types of Content in Ishill

There are three types of publications that Ishill supports: monographs, leaflets, and imprints. The example site contains instances of each; you might find it helpful to browse through the `content` folder as you look at this section.

Hugo itself doesn’t really define different types of content; it’s all just content. Ishill uses the different types primarily to make it easy to make different front matter in the PDFs.

Each subfolder of `content` (`imprints`, `leaflets`, `monographs`) can be organized however you would like. In the example site, the `imprints` folder uses a folder scheme based around years and months; the `leaflets` folder uses one based around years only; and the `monographs` folder does not have additional organization other than the zines themselves. You can change these however you’d like; Hugo will use the folder scheme to create paths on your website, but doesn’t otherwise care how things are organized. (You can even disorganize them; you could have some zines inside an annual folder, and others simply inside the publication folder. It really doesn’t matter.)

Renaming the publication types (so you could call something a “folio” instead of a “monograph,” for instance) is certainly doable, but as of this release, not implemented. If you wish it existed, let me know (file an issue at <https://codeberg.org/ussjoin/ishill>) and I can work on it.

Each zine folder has an `index.md` which contains the front matter (title, author, publication date, and so on) for the zine. Zines other than `imprint` then have one or more other `.md` files, which contain individual articles. The articles are ordered by the `weight` value in the

article front matter; there isn't any particular significance to the `weight` values *other* than which are lower or higher.

There's one other key feature in the `index.md`: the parameter called `extra_spacer_pages` in the `[params]` section. It's the number of spacer pages you want to insert between the end of your content and the rear cover page. This matters because a saddle-stitched zine needs to have its number of pages divisible by 4 so that it can print (front and back sides of one sheet, then folded in half). The tooling will pad out your page count automatically, but it can only do so by putting blank pages at the *end*, which will mean that the back cover (your publisher contact information, etc.) won't be in the right place. Use `extra_spacer_pages` to insert pages *before* the back cover so that the count comes out correctly. The way to do this is to build the zines and look at them, then count; I'm really sorry to make you do this manually, but I haven't yet figured out how to do it automatically in the context of Hugo's build system. (This all sounds complicated but takes about three seconds; the important bit is to make sure you do it just as you're ready to publish, to make sure that added content doesn't change your page count—but if you forget, it'll be obvious when you look at the impressed zine.)

Technical explanation for this flaw (feel free to skip): Hugo doesn't know what the page count is, because WeasyPrint acts as the "browser" that views the HTML generated by Hugo (which determines the page size). It uses the parameter to turn three `div`s on or off at build time (see `layouts/partials/spacer.html`). It seems like there ought to be a way for some sort of CSS `@media` query to know what the page count is going to be, *and therefore turn display to block or none on those three elements*, in roughly the same way that the page numbers are generated—but I couldn't figure out how to make it work. This embarrasses me, and one day I'll figure it out, but I've spent an inordinate amount of time on it and decided to do the dirty hack instead. If a CSS expert knows how to do this, *please* contact me: ishill@ussjoin.com. (Or open a pull request on Codeberg, of course!)

Leaflet

Leaflets support single- or many-author zines, and on the title page will say "curated by" rather than "written by." Their title pages will have a table of contents that lists the name and author of each article in the zine (again, in order of `weight`). Endnotes are generated at the end of each section.

Monograph

Monographs are designed for single-author publications, and generally have one article inside them. Their front pages say "written by." A Table of Contents is *not* generated on the front page. Endnotes are generated at the very end.

Imprint

Imprints are a special kind of publication. Unlike leaflets and monographs, imprints aren't rendered to PDFs by Ishill. Why do they exist? It's so that you can publish art zines, minizines, or anything else where creating the "final form" is done outside Ishill and Hugo, on the same website as your other zines (and include it in the RSS/Atom feeds, etc.). The front matter content controls what's displayed on the site for the zine entry, but otherwise is unused, and the `index.md` should contain a link to the PDF(s). Again, take a look at the example site for how this works.

How to Get Going as Quickly as Possible

This section of the zine is written for people who don't know how to use Hugo and don't feel like they want to learn much of it right now. You shouldn't really go straight into production (as in, create a live website and put this up there) this way as your first experiment, but it shows you how things work and lets you get a feel for the way a Hugo deployment works. You will need to enter commands on the command line, but we'll try to provide clear instructions for them (or give terms you can put in a search engine).

If you already know how to use Hugo and Git, great! Some of this will be pretty trivial. You'll still want to work using the example site as a base, just because the theme makes some assumptions about certain paths (mostly around the publication types documented above).

This is mildly poor software engineering practice, but the complexities of fixing it exceed the wins from doing so, because it's unlikely that someone is going to change their "normal" Hugo website into an Ishill instance to zineify, say, their entire blog portfolio all at once; if they want to do that, they probably need to change other things anyway (like the front matter on each post). If you disagree, however, I am happy to listen to a counterargument!

0. Pick a code host to work from

Because I'm a gigantic nerd, and for some other reasons ranging from sentimental to paranoid, the Ishill theme and Ishill example site are hosted on three different coding platforms: GitHub, GitLab, and Codeberg. The URLs to find the Ishill example site on all three are at the beginning of this document. It doesn't matter which you pick, just pick one and open up the example site page.

1. Set up Hugo

Go to <https://gohugo.io/installation/> and follow the installation instructions for your operating system. If you're on macOS, `brew install hugo` is what we use, with the Homebrew package manager (<https://brew.sh/>). You'll also want to run `brew install`

`git`. If you're on Linux, I doubt I need to tell you how to install packages, but check out <https://gohugo.io/installation/linux/> for relevant options.

If you're on Windows, you can install Hugo, but you needn't, because Ishill can't build on Windows; that is, the tools that turn text into zines can't run on Windows. All hope is not lost for you, however; you just need to use one of the code hosting platforms to do the building for you (and then either host the site on their platform or not; the two decisions are separable).

2. Grab the example site repository

You can clone the repository to your local machine; use `git clone [repository_clone_url]`, with the clone URL coming from whichever code host you're getting (e.g., for Codeberg, it should be <https://codeberg.org/ussjoin/ishill-examplesite.git>). Once you have it, change into the directory, and run `git submodule init` and then `git submodule update`; that will pull in the Ishill theme itself.

3. Change the configuration

Open the folder in a text editor. (A coding text editor is helpful here; if you're new to this sort of thing, use something like Sublime Text [30] to get started. It's paid software (and if you like it, it's worth paying for it), but you can use it for free to evaluate it as long as you need, so just figure out what editor you want to use later and use it as we go through this rapid start.) You don't need to edit anything in the `themes/ishill` folder, but otherwise you can look around and play with things. You should start in the `config.toml` file, which contains a lot of the settings for your site, including changes to the rear zine page and the like. Play with anything; if you forget what you've changed and want to go back, `git diff` will show you what you did.

4. Build

From the example site folder, run `./setup.sh` to get some initial software ready to use (it's mostly getting some Python libraries like WeasyPrint [31] ready), and then run `./build.sh` to run hugo and zineify the three zines in the example site.

5. View the website (locally)

Now run `hugo serve`. This will run hugo in local webserver mode; in your browser, go to `http://localhost:1313` to see the site. Hit Control-C in the terminal to stop it.

6. Add and change content

You know how it works to make it build zines, and you've got some steps you can follow. Fantastic! Now explore the files and folders in the example site as you read the rest of this, and you can experiment from there. If you don't know Markdown, you can mostly learn it just by

reading the `.md` files and looking at the resulting web pages (and PDFs), but web searches for “learn Markdown” or similar will get you going in no time.

While `hugo serve` is running, the web pages will automatically reload as you save files, but the PDFs won't change. To regenerate the PDFs, hit Control-C in the terminal, and then run `./build.sh` again, followed by `hugo serve`. (I often run it as one command: `./build.sh && hugo serve`.)

There's no rush to deploy to a working website, but when you want to, the last part of this zine is here.

Putting it on the Internet

Don't be in any hurry. The example site is your base, but you're building a new zine publisher; make it look like your own. You should change the site name and contact information of course, but you probably also want to change the quotes for the spacer pages and other stylistic choices. You may want to choose a new typeface; check out `assets/css/newtypeface.css` and `static/fonts` for how to do that. (You can use Google Fonts to find and download typefaces, but please don't use a Google Fonts-*hosted* font by inserting their code into the head of your website; that means every visitor to your site will be known by Google, which is just silly and lazy.)

You might want to set a custom favicon for your site as well; that's the icon you see in a browser tab, or in a variety of ways on mobile. There are eleventy billion favicon generator sites that can turn text or images you upload into the favicons your site needs, with many different levels of fidelity and complexity. <https://favicon.io/> is a non-LLM-infested site to do the transformations for you, if you'd like. Take the resulting files and put them in `static`, and the HTML code can go in `layouts/partial/custom_head.html`.

Deployment Options

See Deployable Anywhere, above, for an explanation of what Ishill needs and doesn't need in a host. We'll use those four as examples, and if you're doing something else, you can likely use the information here to get what you need. (If you think you need additional information or something could be better-organized, open an issue at <https://codeberg.org/ussjoin/ishill-examplesite> and I'll update the documentation to make it clearer!)

NearlyFreeSpeech (or any Other Static Site Host)

NearlyFreeSpeech has excellent documentation, and there's no need for me to rehash it; check out their Getting Started [32] after you've created an account. When you create a site, pick

“Apache 2.4 Static Content”; that’s what you have (a static website with no need for PHP, CGI, or Daemons). After you run `./build.sh`, you’ll have a `public` folder filled with your whole site; use any of their documented methods for sending its contents to their server.

If you wish to use a CI pipeline to do this, see `gitlab-ci.yml` (and the GitLab Pages section below) for an example that uses `rc1one` to ship the example site to NearlyFreeSpeech. (On GitLab’s CI, the environment variable `$RCLONE_CONFIG` is set from GitLab’s CI/CD variables.) Note that the script also deploys the code to GitLab Pages; if you don’t want it to do that, remove that part.

GitHub Pages

GitHub’s documentation [33] will help you get up and running with an account. You’ll need to create a new repository (make sure that “Add README” is off), and then follow the instructions for “...or push an existing repository from the command line,” prepending `git remote rename origin old-origin` so that you don’t get an error. Then:

1. Click Settings, then Pages (on the left).
2. Under Branch, where it says “GitHub Pages is currently disabled,” click None, then select `main`.
3. Click Save.
4. Under Source, click “Deploy from a Branch,” then select “GitHub Actions.”
5. Then you’ll need to trigger a new build. Click Actions, then click “Build and Deploy an Ishill Site to GHP” on the left.
6. Click “Run workflow,” then the green “Run workflow” button.
7. Wait a bit (you can click into the resulting item to view progress) and your site will be online at <https://YOURUSERNAME.github.io/NAMEOFREPOSITORY/>

Subsequent site builds and deploys will run automatically when you push new code to the repository; the above only needs to be clicked through one time.

Codeberg Pages

Codeberg’s documentation [34] is easy to read, and will help you create a new account and create a new repository; make sure “Initialize repository” is unchecked, and then follow the instructions for “Pushing an existing repository from the command line,” prepending `git remote rename origin old-origin` so that you don’t get an error. Then:

1. Click Settings, Units, Overview. Check the box next to “Actions” and save.
2. Click Actions, then `buildexamplesite.yml`.
3. Click “Run workflow,” then “Run workflow.”
4. Wait a bit (you can click into the resulting item to view progress) and your site will be online at <https://YOURUSERNAME.github.io/NAMEOFREPOSITORY/>

Subsequent site builds and deploys will run automatically when you push new code to the repository; the above only needs to be clicked through one time.

GitLab Pages

GitLab has excellent documentation on most things, so use <https://docs.gitlab.com/> to figure out how to create an account, create a new repository, and push your code to GitLab. (You'll want to create a blank repository, fill out the form, and uncheck "Initialize repository with a README," then follow the instructions for "Push an existing Git repository.")

I found it... complex to get GitLab Pages to work. Here's my summary of what you need to do (please use this in conjunction with their GitLab Pages documentation [35]):

Click Settings (on the left), then General. Scroll to "Visibility, project features, permissions" and click on it. Scroll down to "Pages," click "Only Project Members," and select "Everyone With Access." Scroll to the bottom of the section and click the blue "Save changes" button.

Now, click Deploy (on the left), then Pages. Then do the following:

1. Enter `node:lts` under "Select your build image."
2. Hit Next.
3. Next, Next, Commit.
4. Now GitLab has successfully overwritten the perfectly good build file the repository started with. This is suboptimal. Luckily, you have a fix!
5. Click the name of your repository (it's likely just to the right of your name) near the top of your screen, so you see your files again.
6. Click `.gitlab-ci.yml`, which will almost certainly have a message next to it like "Update .gitlab-ci.yml."
7. Open <https://codeberg.org/ussjoin/ishill-examplesite/-/blob/main/.gitlab-ci.yml> in another browser tab. Select all the text that's in the file (in the monospaced typeface). Copy it (Ctrl-C, Command-C, etc.)
8. Back in your main browser, click the blue Edit button on the upper-right, then "Edit Single File."
9. Select all in the editor.
10. Paste what you copied from the other tab (Ctrl-V, Command-V, etc.).
11. Click the blue "Commit Changes" button, then in the popup, click the new blue "Commit Changes" button.
12. Click the name of your repository, just like you did in step 5.
13. On the left, click Deploy, then Pages.
14. Click "Domains & settings."
15. Uncheck the box that says "Use unique domain," then click "Save changes."
16. Now you'll have a URL in the box marked "Access pages." Your site will be there!
17. If the page looks weird (e.g., unformatted), run a new build job. (This is a one-time issue when the URL to your site changes, but you wanted to do that to get rid of the "unique" domain.)

18. Click Build, then Pipelines.
19. Click the blue “New pipeline” button.
20. Click the (new) blue “New pipeline” button. You don’t need to enter anything on that page.
21. When the build completes, check the site again.

Subsequent site builds and deploys will run automatically when you push new code to the repository; the above only needs to be clicked through one time.

Parting Notes

Send me an email at ishill@ussjoin.com if you use this; I’m always looking for good things to read, and maybe I’ll add you to a directory in the theme repository, so people can see what it looks like. Also, keep an eye occasionally on the theme repository, because I may release bugfixes or new features from time to time. (No guarantees, but hey.) If you have suggested features, please feel free to submit them as issues at <https://codeberg.org/ussjoin/ishill>.

Footnotes:

1. <https://codeberg.org/ussjoin/ishill>
2. <https://gitlab.com/ussjoin/ishill>
3. <https://github.com/ussjoin/ishill>
4. <https://codeberg.org/ussjoin/ishill/issues>
5. https://en.m.wikibooks.org/wiki/Romanian/Pronunciation_and_alphabet
6. https://en.wikipedia.org/wiki/Joseph_Ishill
7. <https://www.icaboston.org/articles/make-your-own-mini-zine/>
8. <https://www.torproject.org/>
9. <https://gohugo.io/>
10. <https://docs.codeberg.org/codeberg-pages/>
11. <https://docs.gitlab.com/user/project/pages/>
12. <https://pages.github.com/>
13. <https://gohugo.io/>
14. <https://weasyprint.org/>
15. <https://framagit.org/spalax/pdfimpose>
16. <https://xkcd.com/1319/>
17. <https://www.icaboston.org/articles/make-your-own-mini-zine/>
18. <https://www.adobe.com/products/indesign.html>
19. <https://theanarchistlibrary.org/bookbuilder>
20. <https://github.com/travisgoodspeed>
21. <https://github.com/angea/pocorgtfo>
22. <https://en.wikipedia.org/wiki/Samizdat>
23. <https://gohugo.io/>
24. <https://docs.github.com/en/pages>
25. <https://docs.codeberg.org/codeberg-pages/>
26. <https://docs.gitlab.com/user/project/pages/>
27. <https://blog.nearlyfreespeech.net/2025/07/27/a-quick-note-to-our-queer-members/>
28. https://en.wikipedia.org/wiki/Google_Reader
29. https://en.wikipedia.org/wiki/Aaron_Swartz
30. <https://www.sublimetext.com/>
31. <https://weasyprint.org/>
32. <https://faq.nearlyfreespeech.net/section/gettingstarted>
33. <https://docs.github.com/en>
34. <https://docs.codeberg.org/>
35. <https://docs.gitlab.com/user/project/pages/>

"A demonstration of the Ishill zine publishing stack. Click About to learn more."

Ishill Example Site makes it easy to create good-looking zine content on the web and in print. <https://example.com/>. Inquiries and submissions may be sent to donotemail@fake-email-address.com. Ishill Example Site publishes with the Ishill Stack, built on Hugo. Want to be able simultaneously to publish to the web, in PDF, in zine format, ...? Check out the free Ishill Stack at <https://codeberg.org/ussjoin/ishill>.

Document revision: 3632lc6.