

Pantsbuild in-repo docs via Readme.com

Eric Arellano & Christopher Neugebauer, June 1 2022

Rewrite June 6, 2022: split out Sphinx as followup change

Background

V1 docs

We used a [custom site generator](#), with help from BeautifulSoup and Pystache (Mustache templating).

End-user limitations:

- No Pants versioning: 1.0 and 1.30 had the same docs.
- Not mobile responsive.
- Styling looked out-of-date. (Bootstrap + custom styling)
- No rich blocks, like warning blocks and tabbed code blocks.

Contributor limitations:

- Clunky to add pages
 - Had to create a `page target w/ explicit dependencies field`.
- Little file organization. Most pages were in a top-level folder.
- Unfamiliar tooling for first-time contributors (creating a burden for drive-through docs contributions)
- Maintenance burden for adding any new features

Readme.com

For v2, we needed a new site in 2019.

Improving our custom sitegen was not seen as viable, given how many problems we needed to fix.

We found Readme.com as an easy way to address every one of the above issues. It was ready-to-go: all we had to do was write docs, no engineering work.

I do not recall us seriously considering in-repo docs (e.g. Sphinx), which in hindsight, was short-shorted. At the time, we did not have a good understanding of the limitations of Readme.com (see "Motivation" below), and we did not spend time investigating that because it was such an improvement over custom sitegen.

Motivation for in-repo docs

Harder to keep docs up-to-date

Thus far, docs updates have generally been delayed until release candidates, when one of us (usually me, Eric) exhaustively audits the docs and tries to update. This has been extremely hard for me to do because I can't keep track of every change in a release. I am burnt out and dread every release.

The scope of the docs audit has increased significantly since we started this practice. We started performing audits when there was only Python support. There is now support for 5+ languages, with a featureset that can no longer fit in the head of one docs manager.

A key issue is that code changes are decoupled from docs changes. Ideally, when you make a code change to a user API, you change the docs in the same PR. We do not have a process that can enforce this, or even highlight when this step is missed.

This contrasts with our `help` messages, which we generally do a better job of fixing when making code changes.

A major reason that it's easier to bundle code & docs changes at the same time is *grep*. Currently, when making a code change like renaming a field, it is too hard to quickly audit the docs to see if you broke anything. Using *grep* would be invaluable.

In-repo docs are no panacea for stale docs. We must still have robust processes, like high quality code reviews.

No formal docs review process possible

Admins can make changes without anyone else being notified. While we do trust our team, in code-land, we have decided it is always valuable to have a second eye as basic quality control.

Admins can actively solicit feedback via Slack, but they must tell the reviewers precisely what changed—there is no diff. And it's easy to forget to ask for feedback.

I, Eric, usually don't ask for checks on minor changes made during docs audits because it is too hard to keep track of what I changed. I would like more double checking of my work.

Inadequate VCS

README has limited version control, i.e. seeing a prior version of the page and what changed. However, it is not as powerful and ergonomic as Git.

For example, not having "commit descriptions" makes it hard to quickly glance at what the change was.

Edited By Eric Arellano	about a year ago	View Diff	Revert to This Version
Edited By Benjy Weinberger	about a year ago	View Diff	Revert to This Version
Edited By Benjy Weinberger	about a year ago	View Diff	Revert to This Version
Edited By Benjy Weinberger	about a year ago	View Diff	Revert to This Version
Edited By Eric Arellano	about a year ago	View Diff	Revert to This Version


We also can't compare between "branches", e.g. v2.11 vs v2.12 of our docs.

Proposal

For now, still use Readme.com. But store the docs in the pantsbuild/pants repo as our "source of truth" and generate the docs from there.

We expect most editing to still happen in Readme's (recently improved) WYSIWYG editor. You iterate directly there because it's more convenient, and then must copy and paste the page into the markdown file for our repository so that the change can be persisted.

Readme has apparently improved their "Raw Mode" to be much more understandable. For example, a "tip" block:

```
>  Use `//:tgt` for the root of your repository
>
> Addressed defined in the `BUILD` file at the root of your repository are
prefixed with `//`, e.g. `//:my_tgt`.
```

Non-maintainers can either directly contribute to our docs in GitHub, or still use "Suggest Edits". With the latter, an admin will accept their change and then copy and paste it into a PR.

Downsides of in-repo docs via Readme.com

Easy to forget to update GitHub

If you edit Readme.com directly, and don't copy over the change, it will be lost the next time the docs are generated.

Slower iteration cycle

We're adding an extra step anytime you want to change docs, so it will be slower.

Fortunately, you can still edit with the WSIWGY editor online when you want rapid iteration cycles.

More repository activity

It can be seen as noisy that we will have many more docs related PRs now than before.

On the other hand, this is more transparent. Anyone following the project can see when we changed the docs, vs. currently us having to notify people.

Sphinx followup proposal (kept for history)

Motivation

No DRY

With documentation, it can be good to duplicate things. One reason is that you cannot assume a reader has read docs in a certain order.

However, the duplication is hard for us to maintain in sync. Ideally, we can write the snippet in code only once, and then insert it into all relevant pages.

Case study: our initial onboarding instructions.

- We currently have one overly generic guide for all languages.
 - E.g. Golang users don't need to know about source roots, but we still mention it.
- We want to move to several language-specific guides, e.g. Python vs Golang
- Several of the instructions will be the same like "downloading the install script".
- Current workaround: link to common pages
 - But we've gotten feedback that our docs jump around way too much. It's better to avoid relying too heavily on links.

Limited nav bar nesting

Readme.com only allows two levels of nesting.

PYTHON
▼ Python overview
Enabling Python support
Third-party dependencies
Interpreter compatibility
Linters and formatters
Pex files
Building distributions
> Goals
> Integrations

Technical writing wisdom strongly encourages each page to have one purpose, [e.g. guide vs. how to vs. reference](#). This nesting limitation prevents us from breaking out subpages and results in overly crammed pages.

Case study: Python third party dependencies. Multiple lockfiles is a niche feature that most won't use. It crowds the current page, and we haven't added a guide on how to effectively use them because we don't have somewhere to put it.

Implementation

We use the site generator Sphinx, given the maturity of its ecosystem and its use in many open source projects like Python itself, pip, and Black.

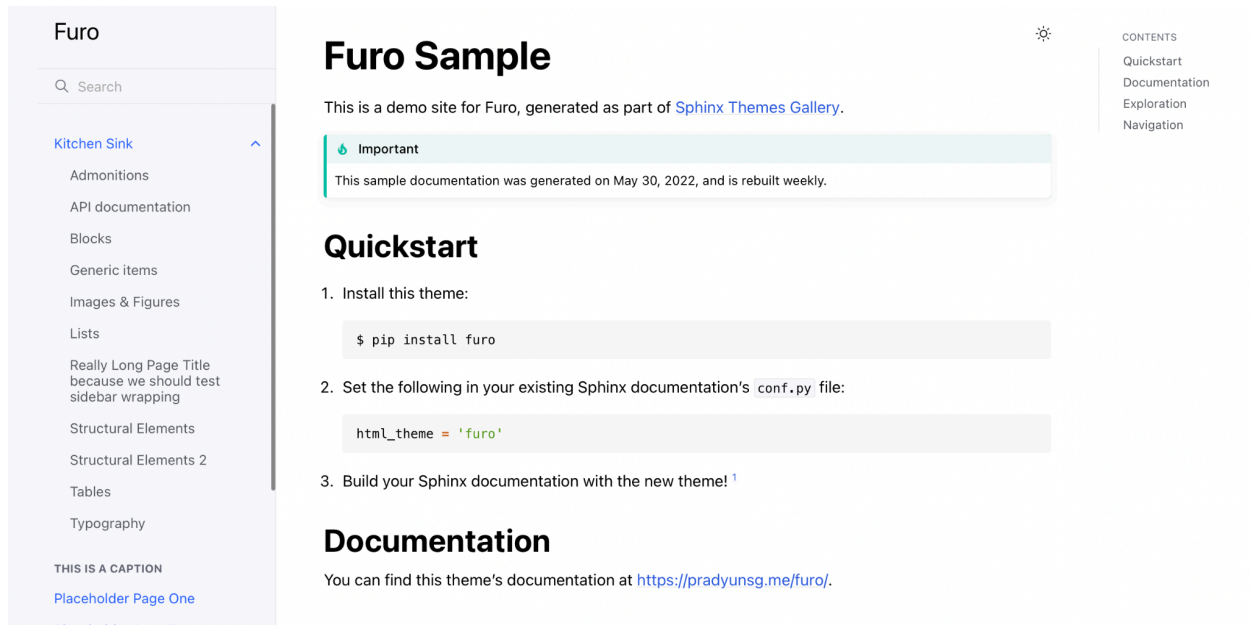
Sphinx has the additional benefit of familiarity to external contributors, and would encourage us to have first-class support for a tool that many Python projects would need before considering adopting Pants.

For accessibility to contributors, we would use Markdown via [Myst](#), which is Common Markdown plus some extensions for adding inter-page navigation. Myst still has all the power we need from Sphinx. Carol Willing from Jupyter (a primarily Python project that supports multiple languages) has vouched for Myst and said it has helped non-Python contributors.

Theme

See <https://sphinx-themes.org> for pre-written options that won't require us to maintain a custom theme.

I suggest using [Furo](#), which is used by [pip](#). Sphinx allows custom CSS, so we can adjust theme colours and fonts to be more Pansy.



Some benefits:

- Modern
- Dark mode available
- Looks good on mobile
- Nav bar can open/close sections
- Back to top button



This just shows up as a vanilla `<pre>`, which is... both nice a

Feature Parity

Sphinx supports plugins that can add support for features that we rely on at readme.com. For example, we can use [Sphinx Tabs](https://sphinx-tabs.readthedocs.io/) so that we don't lose code tabs.

Users can be guided to each page's source with an edit button that links to GitHub (which can be added [automatically on Read The Docs](https://readthedocs.org/docs/en/stable/guides/adding-edit-buttons.html), a sphinx hosting service), e.g. this picture from [pip's](https://pip.pypa.io/en/stable/) Furo-themed docs:



Sphinx on its own does not offer versioning support: that is generally implemented via [ReadTheDocs](https://readthedocs.org/).

Migration plan

We hope to fully migrate our docs once we have proven that the tooling is a viable option for us. There are four steps, each of which offers us an opportunity for evaluation once complete.

1. Create proof of concept skeleton site.
 - a. Not using Pants to run Sphinx for now, for simplicity.
2. Privately port Pants 2.13 or 2.14 docs to Sphinx
 - a. Don't rewrite anything, only a port.
3. Get new docs to coexist with old Readme.com docs
 - a. See <https://docs.readme.com/docs/exporting-docs> for how to export the old data
 - b. Do we have to get old docs to load via Sphinx? Or can we host the static site?
4. Switch DNS to point to Sphinx docs

We will then be free to reorganize the docs how we'd like, e.g. using more nesting in the left nav bar.

Hosting: Readthedocs

They host Sphinx-based websites, and are heavily used, including by Pex.

We can get it for free since we're open source. Or choose to pay for benefits like no ads: <https://readthedocs.com/pricing/>.

As Sphinx generates static HTML, other static site hosting services are available, but these would involve engineering work.

Other discussion points

CI/CD

With ReadTheDocs, we can trigger builds from Pull Requests: <https://docs.readthedocs.io/en/stable/tutorial/#trigger-a-build-from-a-pull-request>.

As with V1, we can set up our CI to skip running tests for docs only changes.

Generated docs

Sphinx can also generate reference documentation from Python docstrings as necessary. This should make keeping the reference section up to date a lot easier.

Unversioned pages

Stu asks: Currently the docsite has a series of pages which are "unversioned" (everything along the top header), and thus don't need to be re-published for each version. Would that be possible with ReadTheDocs (without republishing from all branches for each edit), or would we need to host a separate site for those?

Currently it is not clear if we can manage unversioned pages within the same logical "site" as the versioned documentation. If this is not possible, we will need to add a second "marketingware" site as a separate project.

Note that there are limitations to the "marketingware" features of readme.com (for example, the "get started" link must point to the first page of documentation. This puts onboarding at odds with a coherent navigation structure).

It may well be the case that we retain Readme.com as our "marketingware" site.

Pants plugin for Sphinx punted on

A commonly requested feature is for Pants to directly run Sphinx. Yet that is an entirely separate project from this: this proposal is focused on easing the maintenance burden of our own docs, rather than a new Pants feature.

It is easier to implement Sphinx without Pants, e.g. less engineering work.

This also gives us more time to better understand how Sphinx is used in the wild.

Downsides

As with any proposal to change tooling, there are benefits and drawbacks to this proposal. The level of perceived impact of any downside depends on the target audience – who we expect to be the people eventually interacting with this tooling and any process built around it.

The authors propose that our "prototype" target audience consists of technical users who are familiar with software engineering tools such as Git and GitHub workflows. If there are significant classes of documentation contributors who do not match this use case, then we should consider the proposal through a lens that includes them.

Also note that there is significant community experience with this specific tooling, as well as the concept in general from other open source projects: In-repo docs are used by Python projects such as Django, Pip, and Python itself; and by commercial services like the Azure SDK. If these downsides prove contentious when we discuss full implementation of this proposal, it would be useful to seek experience from these projects to understand how they have perceived these trade-offs.

New contributors must be familiar with GitHub pull requests and markdown. Readme.com offers a [WYSIWYG](#) editor. No need to know Git or Markdown.

In comparison, Sphinx requires that you fork `pantsbuild/pants` in GitHub and then open up a pull request. You must also know Markdown/Myst.

It is possible to do this all via the browser, although that might not be obvious to contributors.

Other open source projects include direct links to the GitHub file editor, which guides the use through editing, forking, and creating a pull request; for first-time contributors, the initial forking can be a somewhat slow process.

Slower iteration cycle for non-controversial edits

Readme.com's WYSIWYG editor means that you can immediately preview how the change will render. Admins can also save those changes immediately. Thus, it's very easy for us to fix small and non-controversial edits like typos.

With in-repo docs, you must trigger a rebuild of the docs to preview the change. To save anything, you must open a pull request.