ally

from start to finish

ally

from start to finish

Madison Scott-Clary

Also by Madison Scott-Clary

Arcana — A Tarot Anthology, ed.

Rum and Coke — Three Short Stories from a Furry Convention

Restless Town

Eigengrau — Poems 2015–2020

ally

This book uses the fonts Gentium Book Basic and Merriweather Sans and was typeset with XeLaTeX.

ISBN: 978-1-948743-15-0

Digital edition, not for print purposes. For the paperback version, please visit makyo.ink/publications/ally

ally from start to finish

First Edition, 2020.

10 9 8 7 6 5 4 3 2 1

ally began in the form of an interactive website.
The project continues at https://ally.id

How does one start a project?

With a bang, or with a whimper?

Very funny.

In all seriousness, though. How? Does one come up with an idea and just...what, go? Just start going and when you get to the end, stop? Can everything be, as NaNoWriMo would have it, pantsed? Run by the seat of your pants such that everything is done without planning, and thus nothing is unsurprising to the author?

Or does one plan meticulously? Does one craft an outline of such startling beauty that to finish the project itself feels almost a betrayal?

Which are you guilty of?

Both, of course. I have my fair share of projects I planned so thoroughly that they fell through, just as I have my fair share of projects that I tried worked and worked and worked on and kept adding and adding and adding, and by the end they were so wandery as to be incomprehensible. They didn't hold together, and the story had gone so far off the rails that it was unfixable without a total rewrite.

And which type was I?

Don't preempt me. All of the projects that I've actually succeeded at have been somewhere in the middle. It's important to plan, as I've learned from all those countless unfinished projects, but there is also a fine balance of planning required, lest you plan your work out of existence.

Qoheleth, the book that follows *ally*, has, as a major theme, the difference between honing and forging. To hone is to take an idea and work it to an ever sharper point, whereas to forge is to take an idea, work until its good enough, and forge onwards.

- ☒ *untitled furry thing*
- ☒ The Manifesto Project
- ☒ Tarot Experiment
- ☒ On Music
- ☒ Consequences of Dissonance
- ☒ Inner Demons
- ☑ Rum and Coke
- ▣ On Furry
- ▣ Sawtooth Universe
- ▫ No Thoughts Our Own
- ▣ Jaroudi
- ▣ ally
- ▣ Post-Self
- ▫ *untitled surgery novel*
- ▫ It's Not About The Dishes
- ▣ Poetry

Neither is bad, of course. There is no value judgement in this distinction. Neither, also, is there any sense of permanence to the label. *ally* was a project borne of forging: I was always trying to do something new with the typography, the wordchoice, the colors and textures of each of the sidequests, and so on. *Restless Town* was a project borne of honing, though. My goal with those stories was to try and somehow come to the finest possible point of the lives involved and the tropes and identities that drive them. I wanted to take aspects of myself — my gender, my mental health, my sexuality, my polyamory — and hone each into a story worth reading.

But, as with outlining versus pantsing, one can go too far in either direction.

> *And still, you will never not giggle when you write 'pantsing'.*

Correct.

All that to say that, as Herbert would hvae it, beginnings are such delicate times. To start a project is to kill a portion of yourself, because, whether or not you succeed in finishing the project, whether or not you are trying to hone something to a cruel point or to forge into new territory, you will never start that project again. You will never again be the you who started that project.

And so why am I here?

May I throw your words back at you?

By all means.

"Can an ally disinhabit a mind so easily?"

A question I remember you being decidedly uncomfortable with.

Yes.

*But why am I here **now**? Why when we are talking about how this project was made?*

Do you not deserve to be here for such a conversation? I trust that you will have little to say of much of the mechanics, but much to say about the process of research.

I do not doubt you. And yet you began this as a list of neat LaTeX things you learned along the way. How often does one write a LaTeX cookbook with one's imaginary friend?

I don't know. Probably not often. There is precedent, though, for overused literary devices in technical writing. `Coy`[1], anyone?

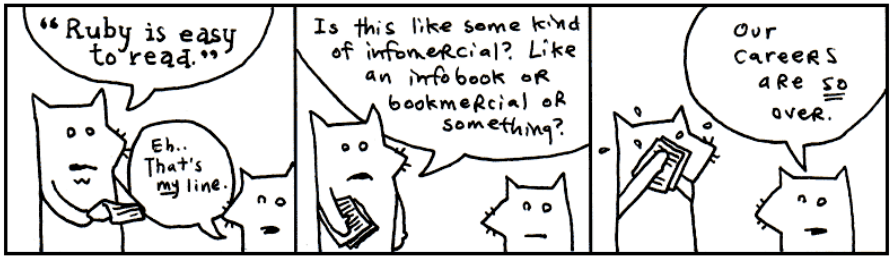
Error messages
strewn across my terminal.
A vein starts to throb.

Their reproof adds the
injury of insult to
the shame of failure.

[1] Coy module on CPAN

When a program dies
what you need is a moment
of serenity.

Or perhaps you enjoy foxes[2] as I do:



There are all sorts of instances of folks writing technical things in a decidedly non-technical fashion.

> *If you say so. What, then, are you going to talk about in this technical guide?*

Thanks for writing my segue for me.

ally.id How the interactive side of ally is built, including some fun examples.

Page 5

The ally book How the book itself was built.

Page 15

Gotchas Some problems I ran into along the way.

Page 25

[2]From *Why's Poignant Guide to Ruby* by Why The Lucky Stiff, licensed under a CreativeCommons Attribution-ShareAlike license.

ally.id

We promise ourselves that we live our memories in a linear fashion. And who knows, perhaps we do.

What we emphatically do not do is remember our lives in linear fashion. *ally* began as an interactive project specifically to explore this aspect. The goal was to use the concept of interlinked pages to represent the way that one memory can be interlinked to another, and another, and so on.

> *And dreadfully distinct within the dark, a tall white fountain played?*

Something like that.

And here, we lean on a very specific deinition of hypertext. Hypertext is used to imply that some portion of a document can link to another portion of a document. This linking goes beyond simply the links that one clicks, as it can mean inclusions, such as when an image is included on a page full of text. It can, indeed, mean a link that leads from one page to the next, but what means 'next' here? Does it mean moving on to the next page in a series of pages, or does it mean moving from one section of the site to another? Perhaps it means moving from this site to the next.

With *ally.id*, this became a core component of exploring memory. It means something different to wind one's way down the singular path a

memory treads than it does to jump the track onto something wholly different. The central axis of the story is the death of Matthew as told through conversations with–

Me!

–with an imaginary alter-ego who, by virtue of that 'ego', knows all the same things I do.

Or more.

Perhaps, yes.

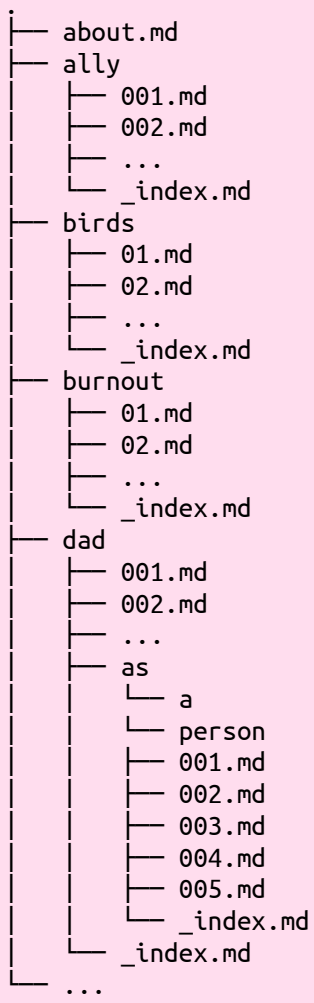
As a memory would be touched upon, it would spark a new branch of exploration that would proceed in much the same way. This is why the project is described as "arborescent": there is a central trunk with a defined beginning at the root, and from there, it blossoms up and out.

Or, it turns out, down and out.

Hypertext types

Axial	A set of linked documents that travels down a single axis, from start to finish.
Arborescent	A set of linked documents with a central axis, of of which may sprout other documents (which may in turn be axial or arborescent hypertexts).
Networked	A set of linked documents with no discernable axis. No start or finish, no direction to travel in.

While working on *ally.id*, Each page was kept in a single Markdown file, and each exploring branch was kept in a folder. Thus, we wind up with a file structure akin to what we see on the right.



> *Never content to condense your thoughts into something simple and easy to read, were you?*

To your scattered files go, I suppose.

Do keep in mind that this is a website, however. This directory, these filenames, this structure all play a role in how the whole project works. These whole tree of files are in the `content` directory of a Hugo project. Hugo is a program which knows how to take these Markdown files and turn them into HTML files while following a simple set of rules.

In each of those `_index.md` files, one will usually find nothing. That is, in terms of content, for at the top of each file comes a header which describes some of those rules. For instance, in some directories, we want the background to be a certain color, or perhaps we want there to be a link down at the bottom saying "back to where we left off".

> *There is no going, and there is no back.*

We can always pretend.

Another rule that is stated in these files is that they are intended to list the pages in that directory. Usually, this is done on a blog page where you might see the titles and first paragraphs of ten blog entries in a list, each with a "read more" link, followed at the bottom by a list of 'pages of results'. In my case, though, I set Hugo up so that, when it listed all of the 'posts', it would only do one per page, and instead of showing only the first paragraph, it would show the whole page. This essentially made it work as a book would: you simply turn the page when you're done reading. My

`list.html` file for this 'serial' layout loops over the pages in the directory as follows:

```
{{ $paginator := .Paginate .Pages.ByWeight 1 }}
{{ $content := .Content }}
{{ range $paginator.Pages.ByWeight }}
...
```

With this theme in place and with the files all in the right orders (each with a `weight` key in their own headers), Hugo will build the site as it stands.

> *Oh, but there's more to it than that.*

Of course.

Early on in *ally.id*'s life, I decided that there needed to be a map of the site. A literal map, too. No carefully broken down list of links for you to click on, but something more clearly representing the paths one takes through memory.

Your "Catastrophically Maddy" counter is ticking up.

Did you expect anything less?

I suppose not. Carry on.

Score one for Maddy.

So, even thought it takes a bit of work by hand everytime I add a page, I decided that it would be worth it to construct a graph that showed the arborescent nature of the project. I was tempted at one point to do the whole thing in Javascript using SVG so that it would track your movement through the pages, but even that was too much for me.

Wonder of wonders.

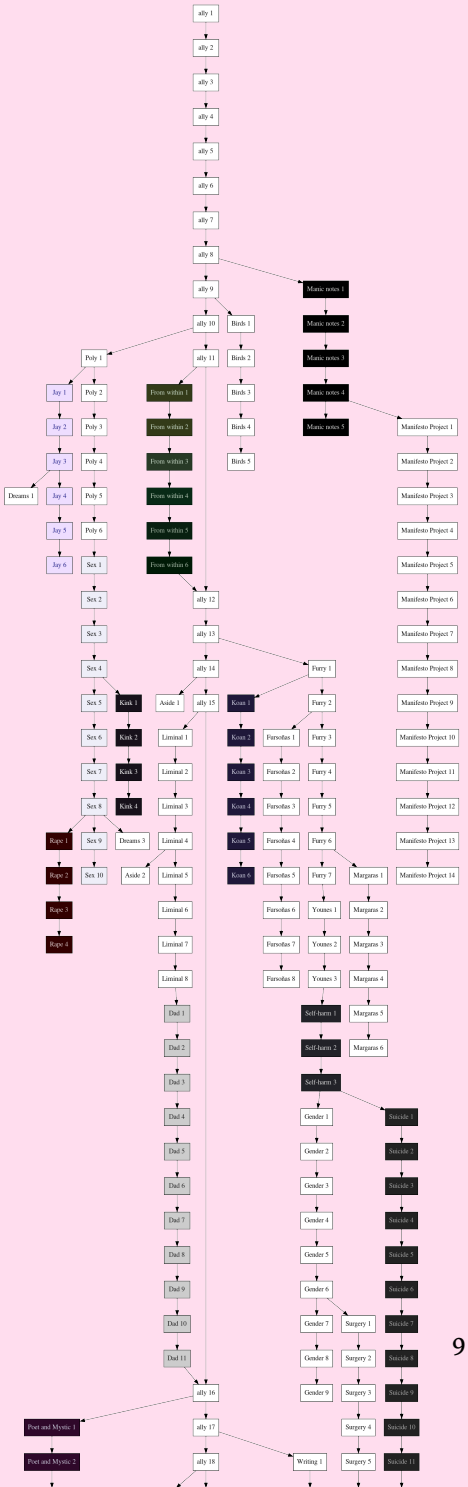
So instead, I leveraged existing tools–

Gag.

Right, sorry. So instead, I decided to use what I already had installed, which meant dusting off my knowledge of Graphviz.

In my `assets` folder lives a `map.dot` file which contains a node for every page and the edges that connect them. It's easy to add to; every time I add a new branch, I list every page inside of it along with its URL, and then draw all of the links that connect those pages together. At the bottom, there is the links from the trunk (or parent branch) to the branches.

```
digraph Map {
```



```
    node[group="dad",
         style="filled",
         fillcolor="#cccccc",
         fontcolor="#222222"]
    "Dad 1" [href="/dad"]
    "Dad 2" [href="/dad/2"]
    "Dad 3" [href="/dad/3"]
    "Dad 4" [href="/dad/4"]
    "Dad 5" [href="/dad/5"]
    "Dad 6" [href="/dad/6"]
    "Dad 7" [href="/dad/7"]
    "Dad 8" [href="/dad/8"]
    "Dad 9" [href="/dad/9"]
    "Dad 10" [href="/dad/10"]
    "Dad 11" [href="/dad/11"]
    "Dad 1" -> "Dad 2" -> "Dad 3" -> "Dad 4" -> "Dad 5" ->
    "Dad 6" -> "Dad 7" -> "Dad 8" -> "Dad 9" -> "Dad 10" ->
    "Dad 11"
```

And so on, until:

```
    "ally 29" -> "Burnout 1"
    "As a person 5" -> "ally 16"
    "From within 6" -> "ally 12"
    "Younes 3" -> "Self-harm 1"
    "Furry 1" -> "Koan 1"
    "Jay 3" -> "Dreams 1"
    "Liminal 8" -> "Dad 1"
}
```

While the whole file is much, much larger than just this, it is really no more complicated[3]. I rarely go back and change orders, and almost never tack a branch onto the trunk in a different place, so I just trace the lines once and let Graphviz sort the rest out.

Once I've updated the Dot file with the new pages, I type `make`, which, on seeing that the file has changed, runs `dot -Tsvg map.dot -omap.svg`, which generages the image itself.

The cool part about SVG is that it renders in the browser just as well as HTML, including with clickable links, so I get a *navigable* sitemap for free. All I need to do is copy and paste the contents of that `map.svg` file into the proper place in the site[4].

[3] Except... — page 25

[4] ...sorta — page 25

We have our content, we have our map, now we just need a way to get it up on the web so that others can see it, right?

> *A question like that never has an easy answer.*

Correct.

Hugo generates a set of folders filled with HTML files and static assets. It's no good on my machine, since that'd mean that I'm the only one that can see it. I could FTP the project up to a server and drop it where everyone could see that but...you know, now that I think about it, I can't remember the last time I used FTP.

Better, instead to just have another tool do it for me. Two other tools, actually.

> *You know, if you are trying to sell this as an easy way to approach a project, I don't know that you are succeeding.*

This is fair. One of the reasons it feels easy to me is that I was already running several other projects using the same technology. The reasons those work so well for me are closely tied with how I run my writing setup, and how I use my writing setup is closely tied with how I program.

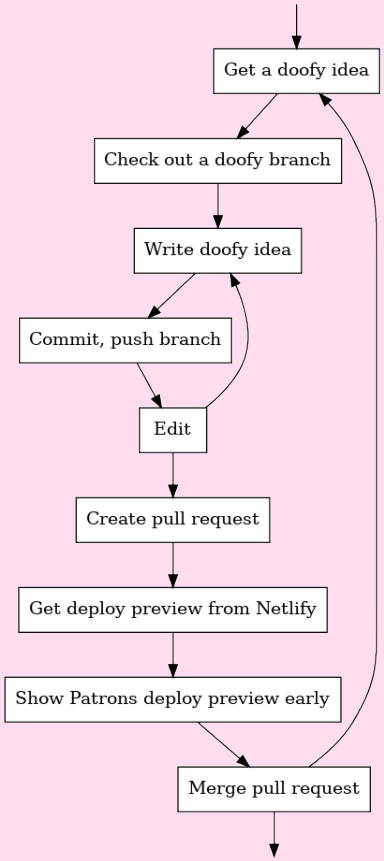
It makes sense for me to have a system that relies on convention over configuration, given how many software projects have worked that way. It makes sense for me to have a system that relies on publishing stories in files rather than database entries as I might with Wordpress or Ghost, given how much of my life is spent tooling around with other files. I'm hardly recommending this as a path forward. There are doubtless easier ways, regardless of how well this worked for me.

> *Right.*

Right.

So, given this Hugo site, the best way for me to work with deploying it uses a pair of tools: Github, which allows me to keep the entire site in a version-controlled repository synced remotely, and Netlify which will automatically build and serve static sites such as this one.

When I get an idea for a "sidequest" to work on with *ally*, I create a branch away from the `master` branch — 'branch' both in terms of git as well as in terms of hypertext, here — and work there. When I push that branch up to Github and create a pull request, Netlify sees this and automatically creates a deploy preview which I can share with those Patrons who get early access. When I click the big grean "merge" button on github, Netlify then builds the main site which everyone can see



The *ally* book

Okay, tell a lie.

> *Oh no.*

It's not that bad.

Another benefit that I get out of working with a set of files on the file system is that it is startlingly easy to get the very same data that is published on the web into the eventual book. I don't have to pull data out of a database to do anything, I just have to run a few commands.

As mentioned, I write in Markdown. It's a plain-text format — meaning I can edit it anywhere without additional software — with the ability to generate HTML. For instance, the first few lines of this chapter in Markdown look like this:

```
Okay, tell a lie.

> Oh no.

It's not that bad.
```

Using the `pandoc` command, it's super easy to translate Markdown to LaTeX, which is what is used to typeset the book[5]. For instance, `pandoc -o out.tex --wrap=none in.md` will do all the translation for me. Had I written this file in Markdown originally, that would have netted me:

```
Okay, tell a lie.

\begin{quote}
  Oh no.
\end{quote

It's not that bad.
```

I don't particularly like the look of the quote environment for you.

> *Neither do I.*

Right. It just looks like this:

> Yar har I'm the *ally* I'm not your friend.

Which is...not great.

> *Do I really sound like that?*

No. But I, of all people, am allowed to poke fun at you.

> *True enough.*

So instead I created an `ally` environment in LaTeX that closely matches the styling of the blockquotes on *ally.id*. It's easy enough to search for all instances of `\\(begin|end)\{quote\}` and replace it with `\$1{ally}`

```
\newenvironment{ally}{
\noindent\ignorespaces
\begin{quotation}
    \allyFont\itshape
    \noindent\ignorespaces}{
\end{quotation}\ignorespacesafterend }
```

[5] Well, a subset of LaTeX called XeLaTeX which allows custom fonts and colors and whatnot.

To help simplify this process, I made a `make` target which pandocs all Markdown files in the Hugo site over to LaTeX for the book.

> *Easy, as they say, peasy.*

Lemon squeezy.

Deciding what to put into the book was almost as hard as writing the content itself. Deciding what to put in, where to put it, and how to display it when suddenly left without the benefits of clickable links was overwhelming.

And yet.

And yet.

I toyed with a few ideas. Originally, I considered printing the book wider than it was tall and giving each page five columns that would be filled in as the story branched.

This proved to be largely illegible. The mind can deal with perhaps two columns on a page, and even then, tying them together across pages rather than within a page requires additional visual clues such as different widths or different background colors.

So I took that and ran with it. I gave each page two columns of unequal widths and ensured that I could mess with the background colors separately — much like these pages here.

Actually *finding* a package that could accomplish this was surprisingly difficult. I was used to the `multicol` package, but that proved less than flexible. I poked a bit at `parcolumns`, but that also left much to be desired.

My solution wound up being the `paracol` package[6], which had everything that I needed, and much more besides. *Actually* learning it, however, proved to be a bear. Much of the package documentation seemed to be written for those who already had some familiarity with its usage, and that lead to a steep initial learning curve.

In the end, though, I wound up with a setup that worked well for me.

And me.

[6] When it didn't bitch about being too full — page **??**

Well, yes.

In order of appearance:

Uneven column widths Before beginning the `paracol` environment (with the second argument being `2` for two columns), one needs:

```
\columnratio{0.65}
\twosided
```

Now, the main body of the text can take place in the `leftcolumn` environment, and the bullshit notes can appear in the `rightcolumn*` environment[7].

Column colors `paracol` allows you to set the background color of the column as well as the surrounding area, selecting the former with the `c` option and the latter with the `C`[8] option like so:

```
\backgroundcolor{c[1]}[HTML]{eeddff}
\backgroundcolor{C[1]}[HTML]{eeddff}
```

Font colors The `fontspec` package proved to be most useful here. One can specify the color of text to use for the font family as arguments in the font family creation[9]. This often meant I had to renew font families:

```
\fontspec{Gentium BookBasic}[Color=222288FF]
\renewfontfamily\allyFont{%
  Merriweather Sans}[Color=4444AAFF]
```

[7] the starred version begins the switch at the current position in the page, rather than the beginning of the `paracol` environment

[8] There are some margin issues, granted — page 25

[9] Along with some other garbage — page 25

Typesetting funkiness (surgery poem, speaktome)

full-page pictures

pdftk for attaching covers

Generating index[10]

[10] Aw heck, really? — page **??**

Gotchas

includepdf for image pages

- pdftk for covers
- munging the index for page breaks
- Branches to left or right in dot file
- Graphviz sets fonts to stupid things
- Setting color margins in paracol
- Needing `Ligatures=TeX` in fontspec when renewing