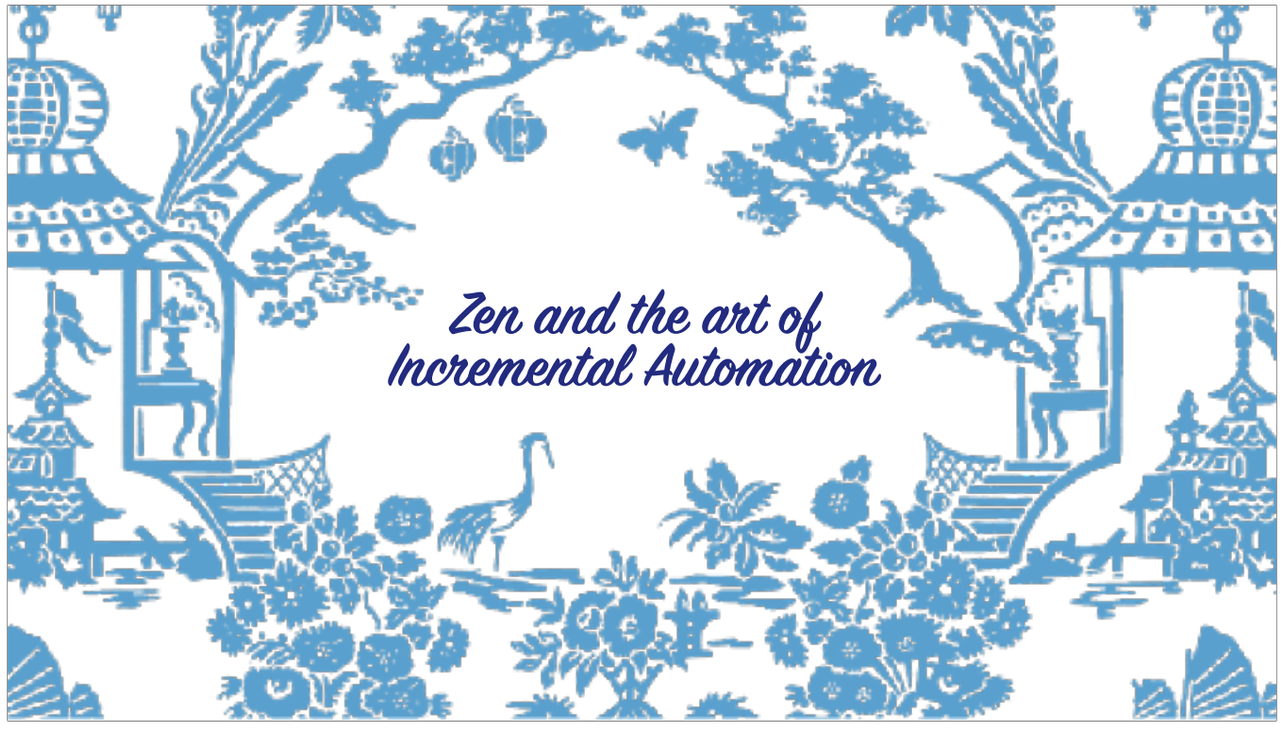

Zen and the art of
Incremental Automation

name

pronouns

workplace

allow me a moment to do the thing

name Aji

pronouns

workplace

My name is Aji

Aji

they/them pronouns

workplace

I use they/them pronouns

Aji

they/them

I work at thoughtbot

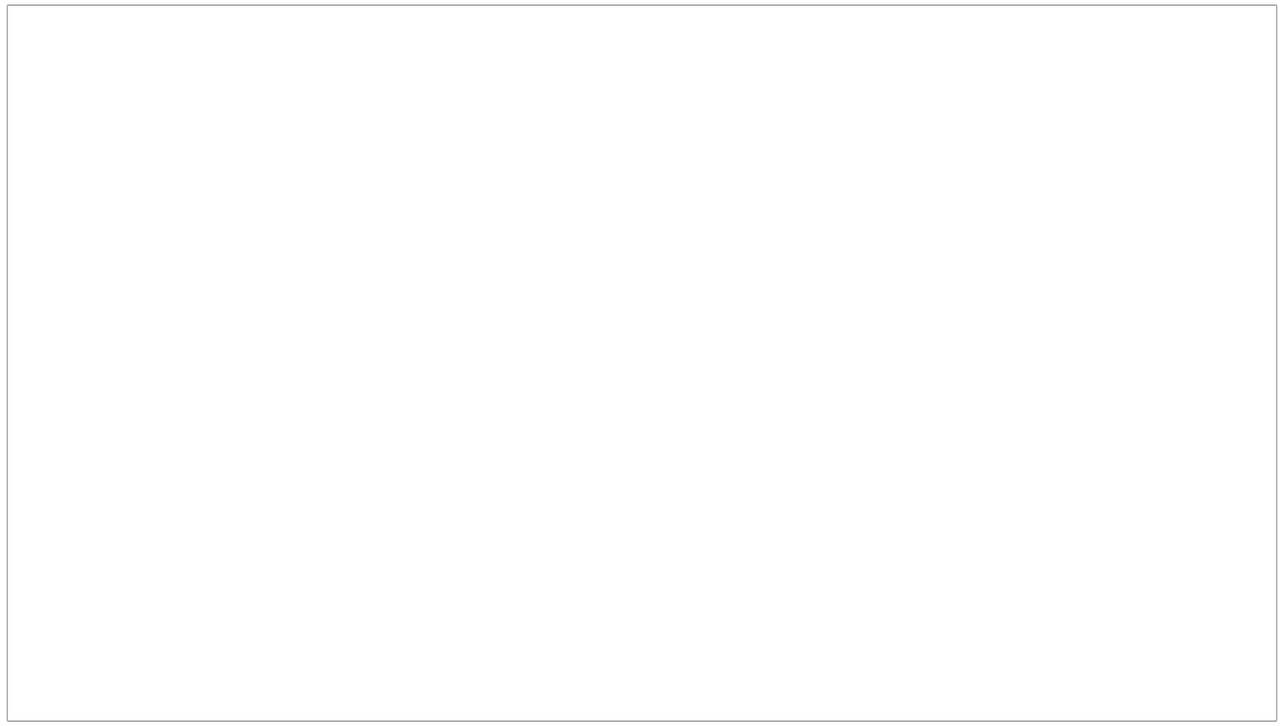
and I work at thoughtbot

Aji

they/them

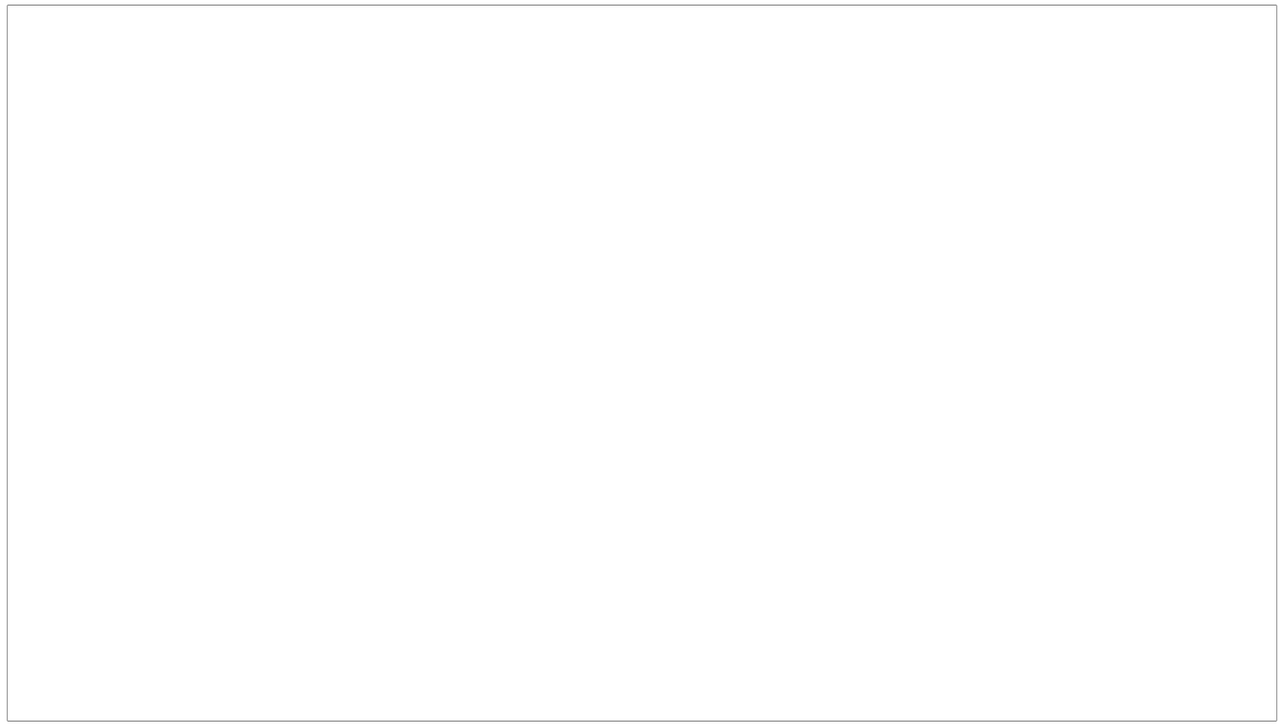
thoughtbot

i never know what else to say for these intros so.. the highways where I grew up had pedestrian walkways that spanned over top of them. Once as a young child I asked, "what's a pedestrian?" to which my father replied "someone who chooses to walk rather than drive a car." For years I thought capital P Pedestrians were a religious group identifiable by their beliefs that forbade them the use of the automobile



You're here to think about process, about automation. The promise of the computer age. Thinking machines that relieve us of the burden of manual work. Steve Jobs spoke of a bicycle of the mind, computers working in tandem with their operators to close the gap between thought and action.

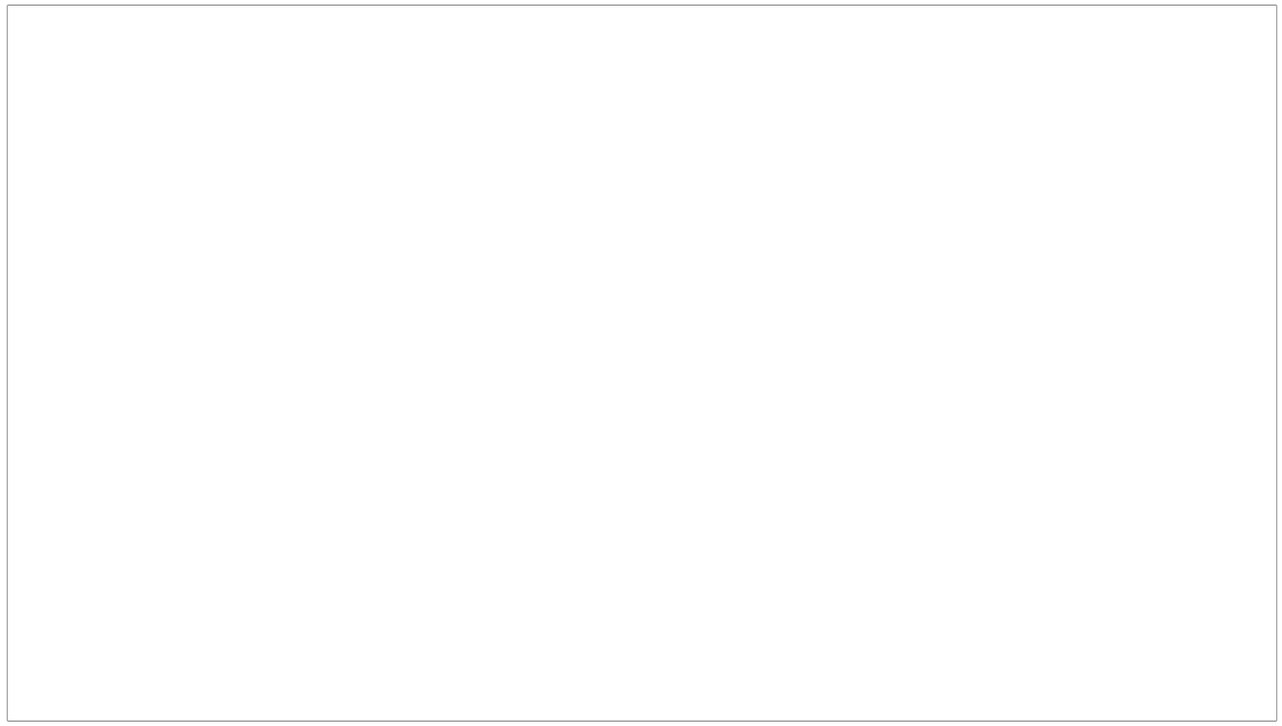
So why in this auspicious time where ruby, the most dynamic of languages has static typing..



Why is there still so much manual toil?

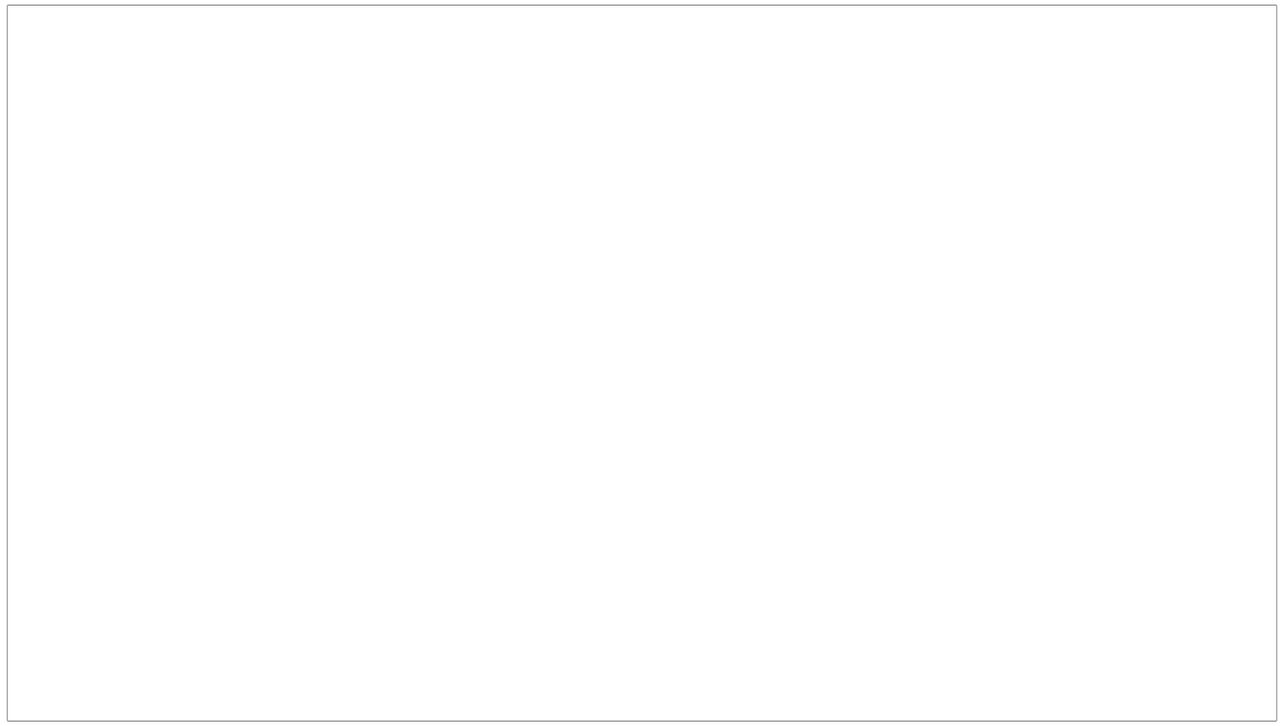
The toolchain keeps getting longer, the tech stacks taller, we have to keep so much in our heads, juggle at least three project management apps, and I guess I'm supposed to start a podcast now

while *still* making progress on delivering value to our users, our businesses, our clients..



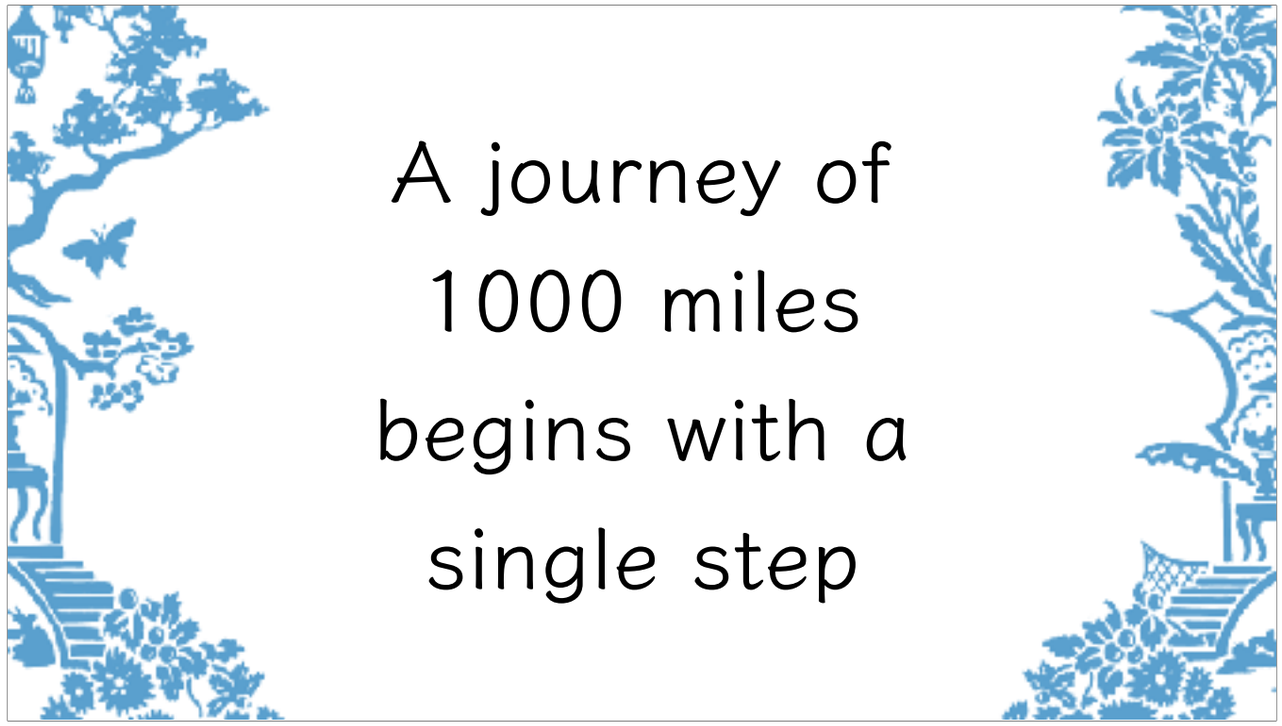
To make any progress at all, we'd have to press pause on feature work and go off into some magical land without
meetings, jira, slack, hybrid offices, basecamp, code review, tps reports, direct reports, import maps, webpacker, the metaverse, terraform, mastodon, Elon Musk, turbo, hotwire, stimulus, strada (coming soon), pattern matching, 2fa, mfa, tcp, ip, udp, php, dns, css, gui, api, fyi, npm, sql, gem install, rails new, make, rake, reek, flog, blog, reflog, and now there's a Rails Foundation? I don't even know what that means!

I don't have time to walk my dogs in the morning without listening to a tech podcast for fear that I might fall behind. Forget one hill, there are a hundred hills, and Jemma can't hold my hand up all of them!

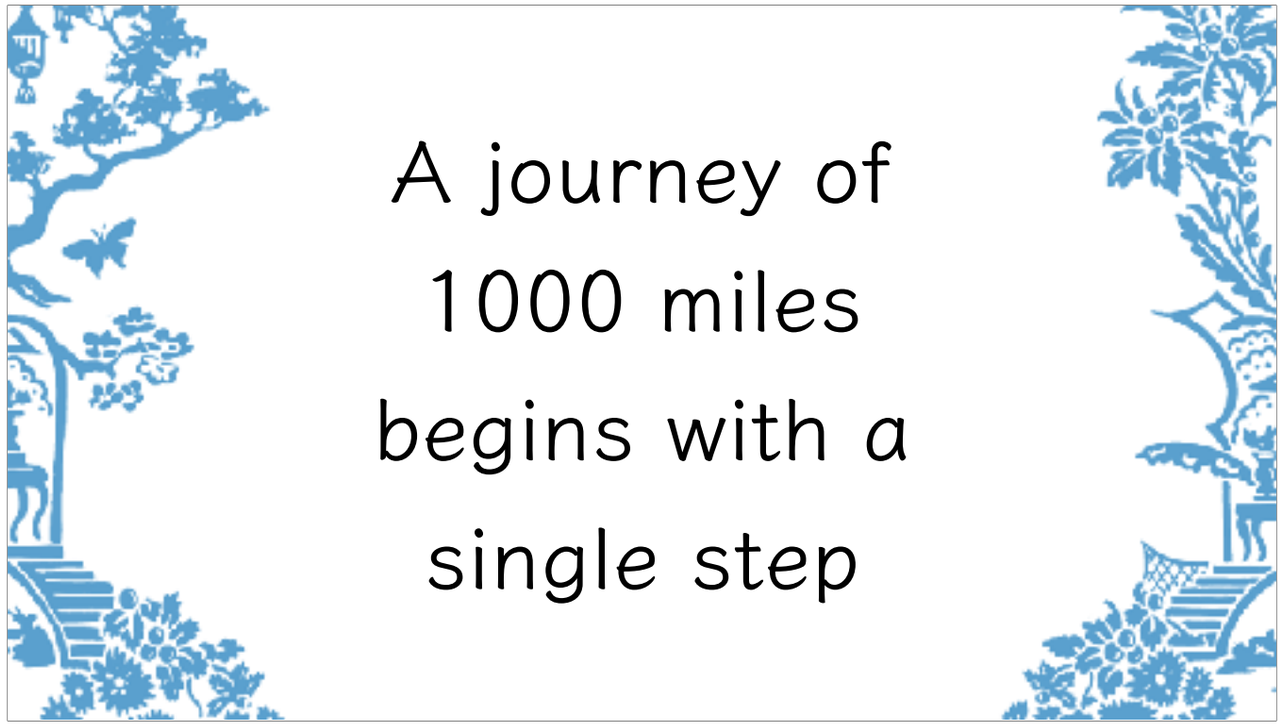


WHEN am I supposed to automate, I still haven't set the clock on my microwave back an hour

I need a moment of zen...

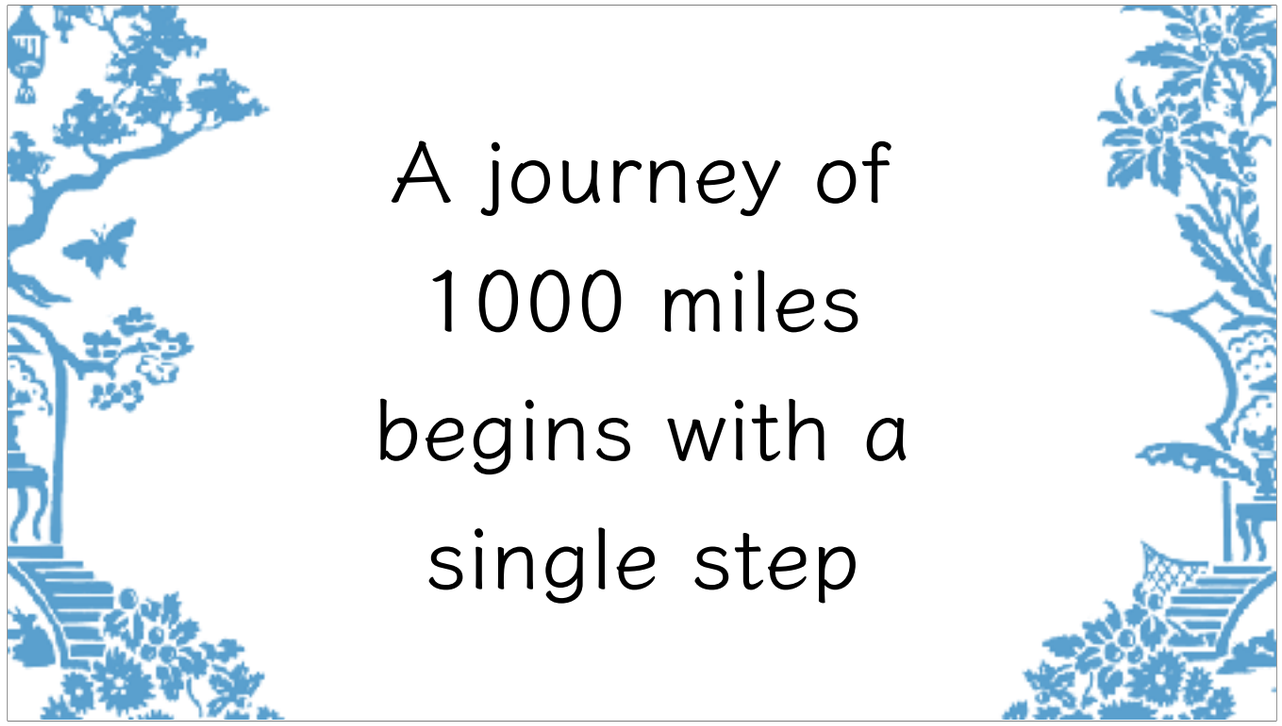


Let's consider this popular quote from the Taoist thinker, Laozi

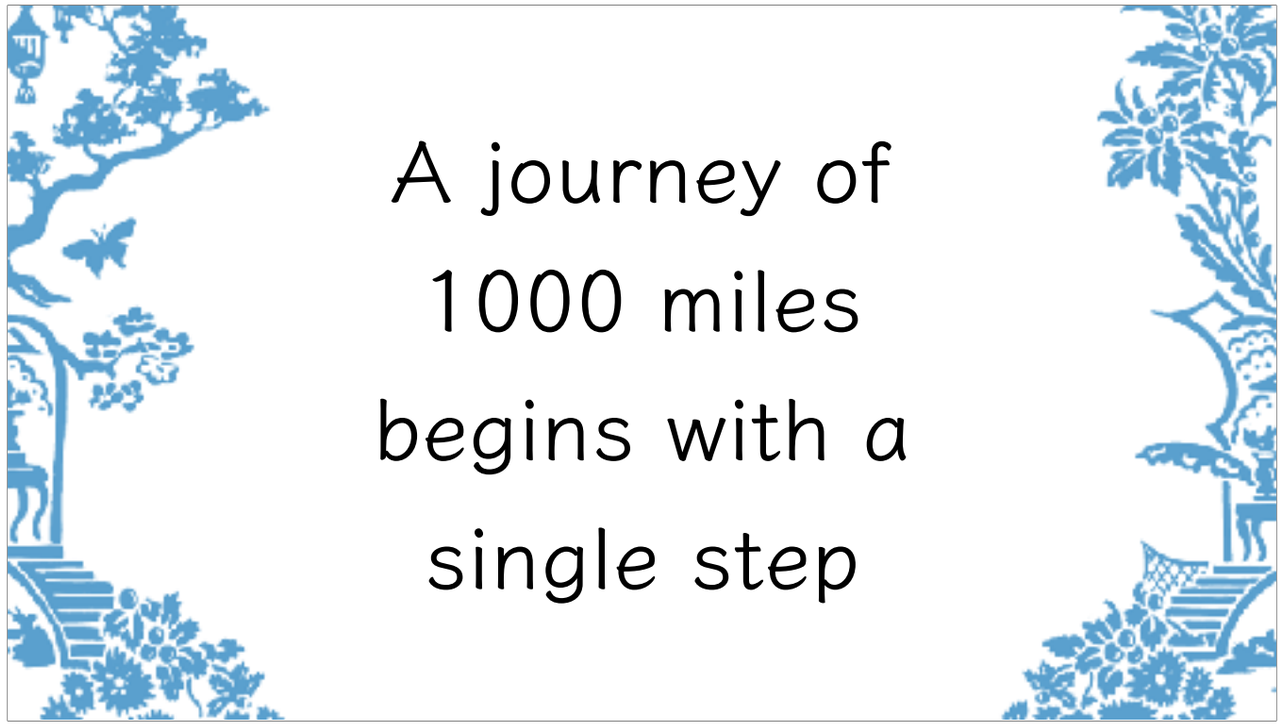


Laozi wouldn't have known the term Zen Buddhism, he (or they because we don't know that it was truly only one person) he was a Chinese philosopher influenced by and building on Central Asian and Indian traditions and mythology.

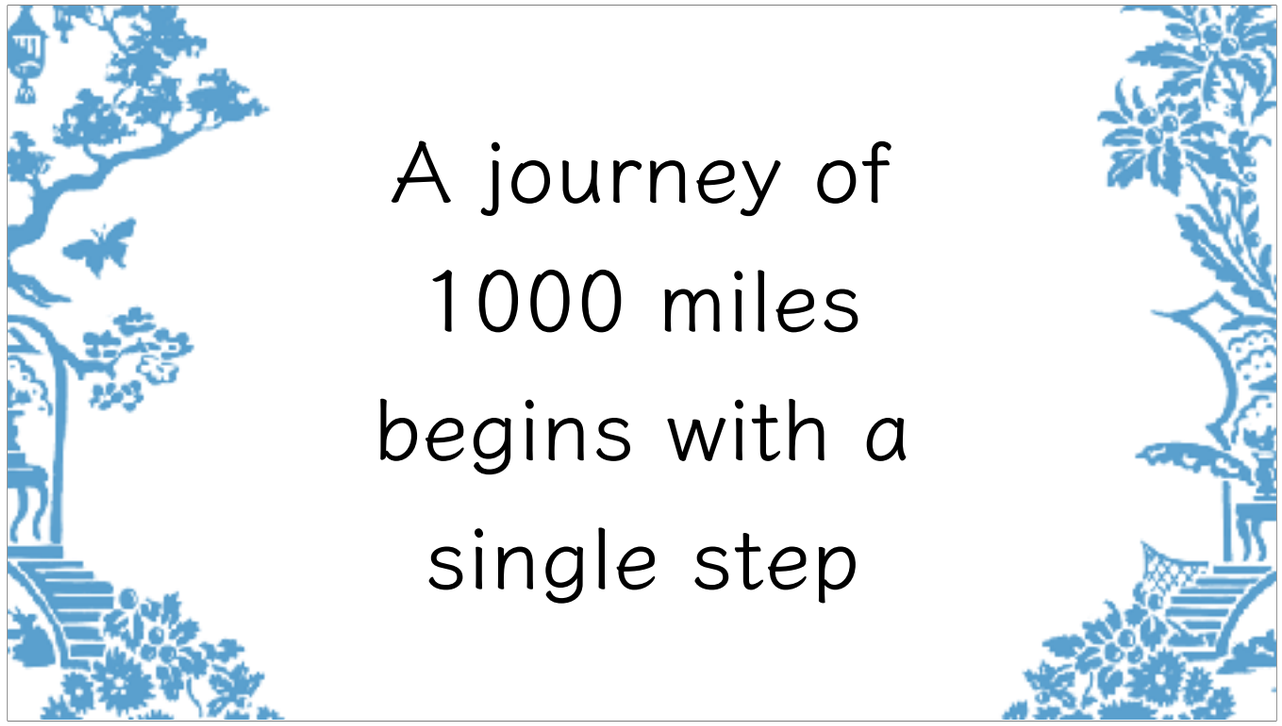
Zen as a school of thought born in the Tang dynasty of China that came into its own in Japan, was influenced by Laozi, and stood on the shoulders of hundreds of years of dedicated practitioners and scholars.



All that is to say that what I'm about to do is a hand-wavy and reductive device in the service of a pithy phrase for a conference talk



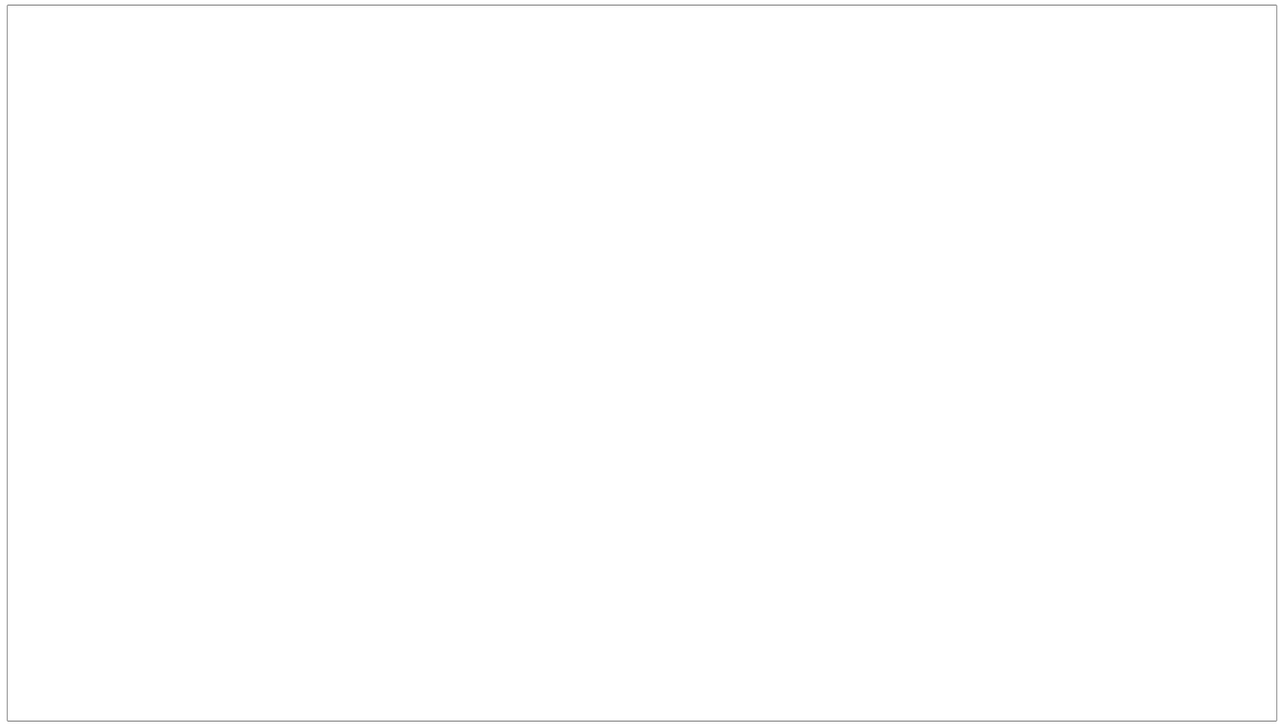
but the best way to describe what needs to be taken on in order to achieve progress in our goal of harnessing automation in service of.. developer happiness... is a mindset. A practice. A relationship to our work.



We're going to explore several principles of this mindset... that in shorthand

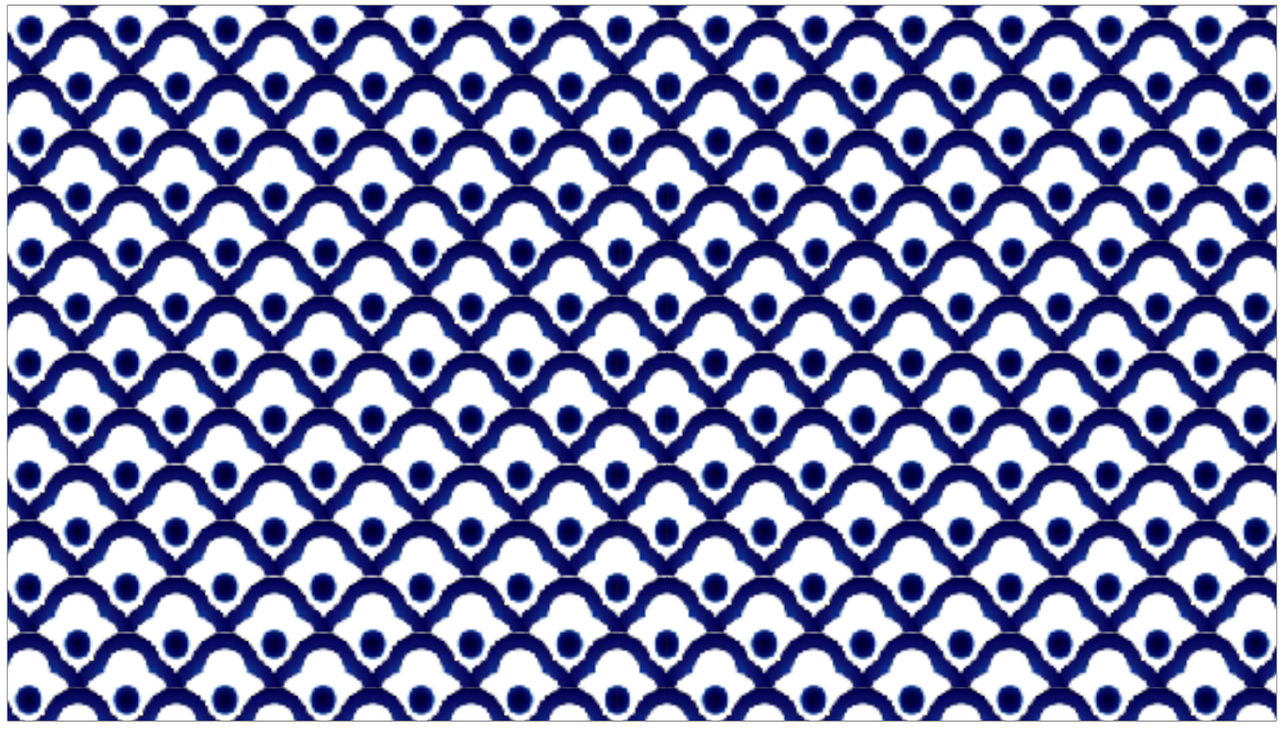
we'll refer to as Zen Automation. Because like Zen, or Test Driven Development, this is a practice, rather than a gem to install.

So let's reflect on Laozi's poetry, and reframe our conception of automation as an all-or-nothing prospect. You wouldn't set out to write a JIRA card that says "Implement tests for order serialization," the tests are part of the work, not a separate deliverable. And we'll find that so too is it often incongruous to write a ticket to "automate new employee account creation," as our change in mindset will carry practitioners of Zen Automation towards that goal... a single step at a time.

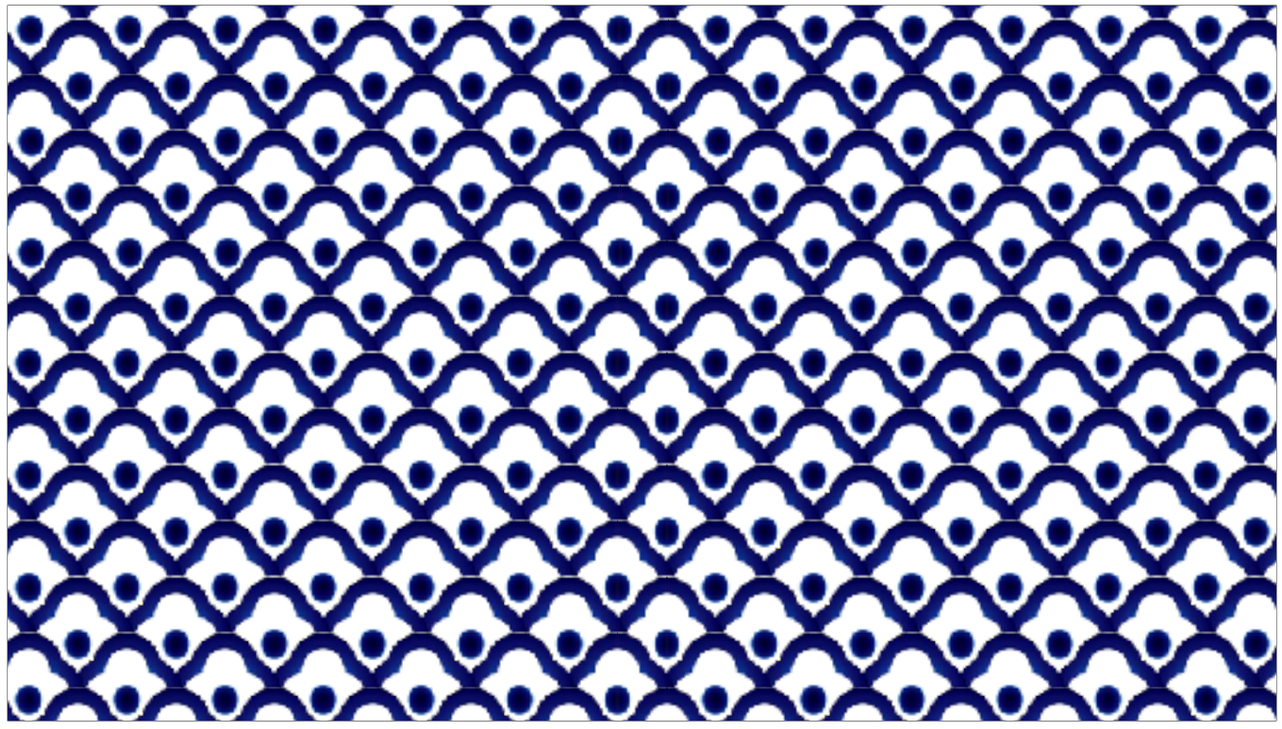


(take a beat)
i promise i love my tech podcasts and my dogs do too. Yes, Andy even the ones where three white guys talk about Apple products.


CONNECTED
Technology without borders

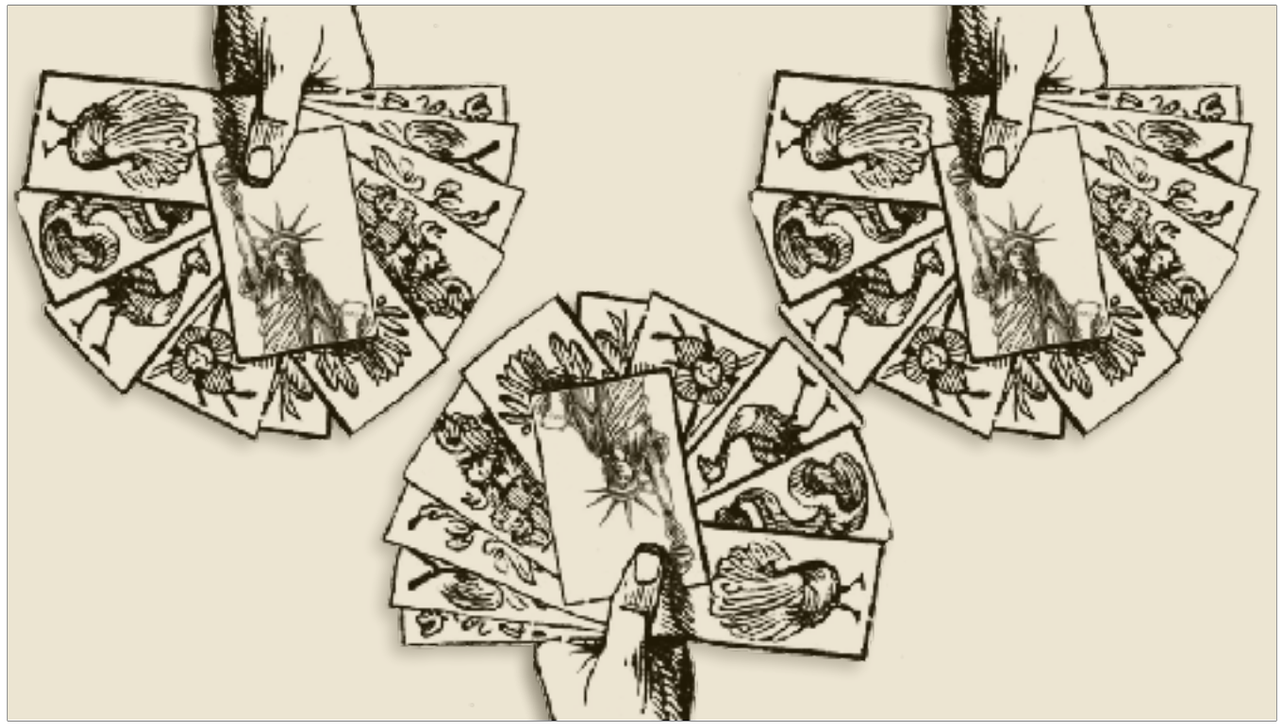


I want to share a practical example of the practice of zen automation through a task that I encountered some years ago, and show how these principles guided my approach.

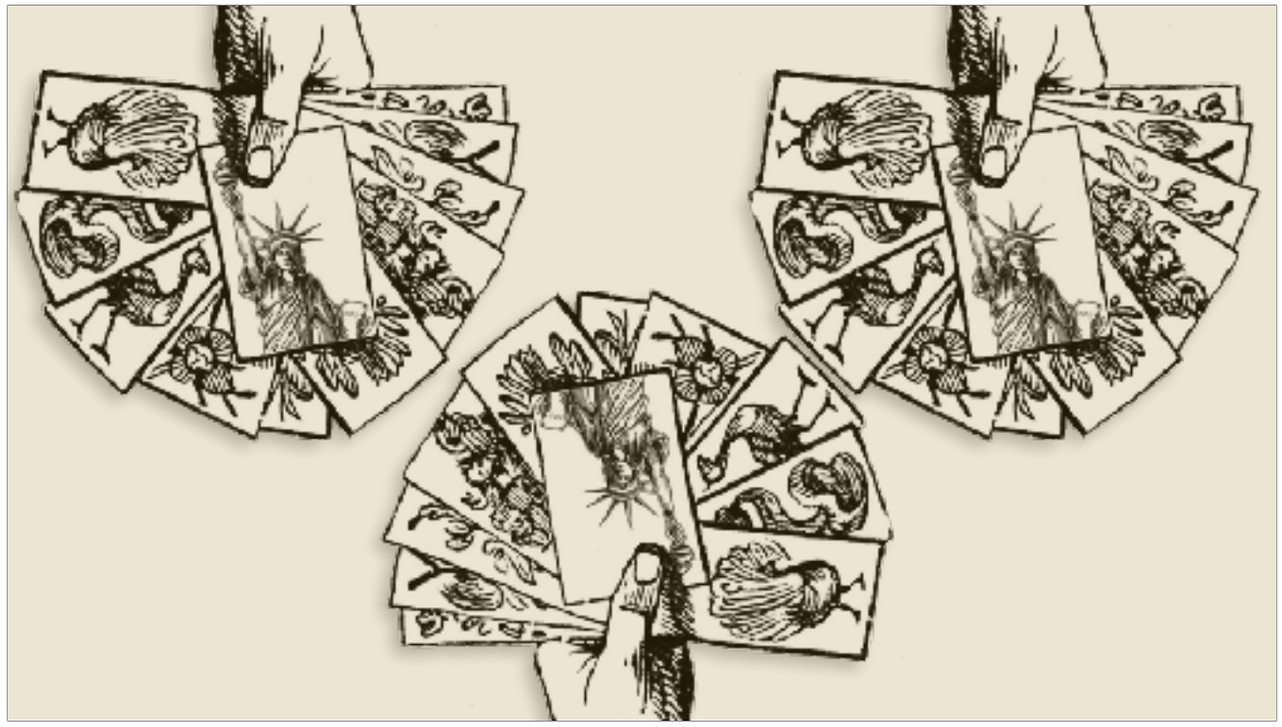


I was working on a product that had a concept of locations. Users would schedule <> their clients to go to a specific location <> for different activities. <> There were hundreds, <> if not thousands, of these locations across the country.

As launch day grew closer, the partial list of locations hadn't changed since seed data from day one...
there wasn't any tooling in place, and no UI existed for new or updated locations



As anticipated, I was given a list of a few dozen locations that needed to be added to the system. A straightforward enough task..



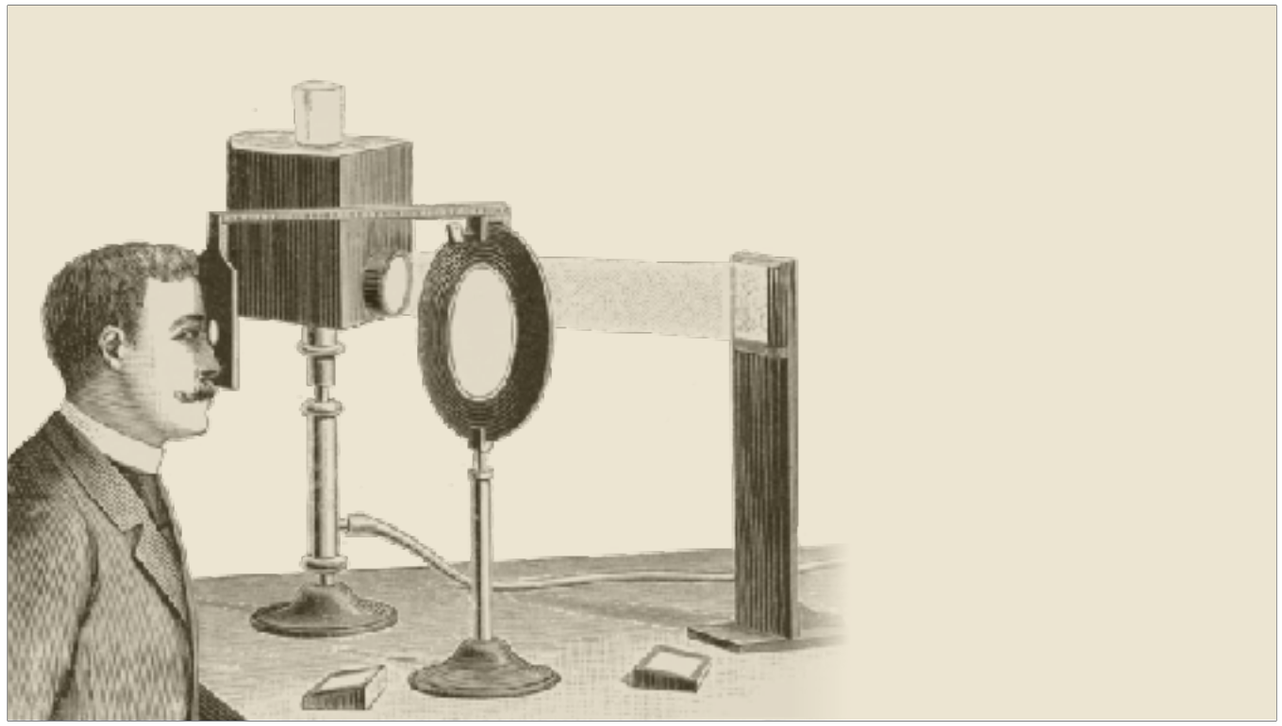
parse and massage the list to the shape of our data, populate it into the database. No business logic or UI changes.

So I opened my text editor
- to a fresh new markdown file
- and began with the (first principle of zen automation...

first principle of Zen Automation... Always be capturing

The importance of this step cannot be over exaggerated. None of the rest of this works unless you start by writing things down.

writing EVERYTHING down



I had only a single list of new locations to add, and yet I started with a markdown file to capture <> documentation.

That file was going to be my scratchpad. The edge cases I considered, links, thoughts, learnings, and of course, the steps I took to complete the work.

I had this one off task, taking a list, dropping a bunch of extraneous data, <> passing that file to some servers, making new records. I could have thought, <> "I'll never have to do this again, it's so quick, why bother with the notes?"

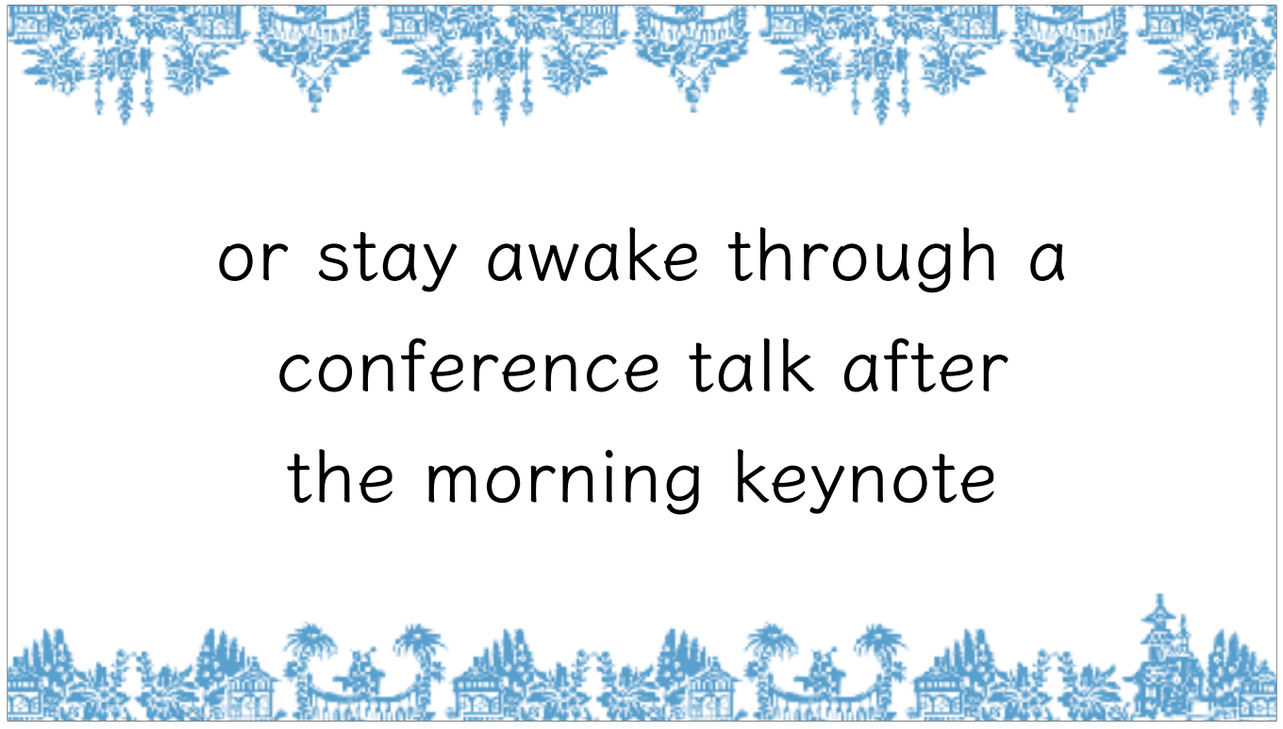
lowering activation energy

But the key to this becoming a habit, embodying our mindset of Zen Automation, is lowering the activation energy to creating & maintaining documentation

activation energy

activation energy is a minimum threshold of energy that has to exist in a system before a reaction will take place

It's that motivation you need to build up to finally go clean the leaves out of the gutters. It's the second cup of coffee high you need to hit before you can make it through the weekly planning meeting.



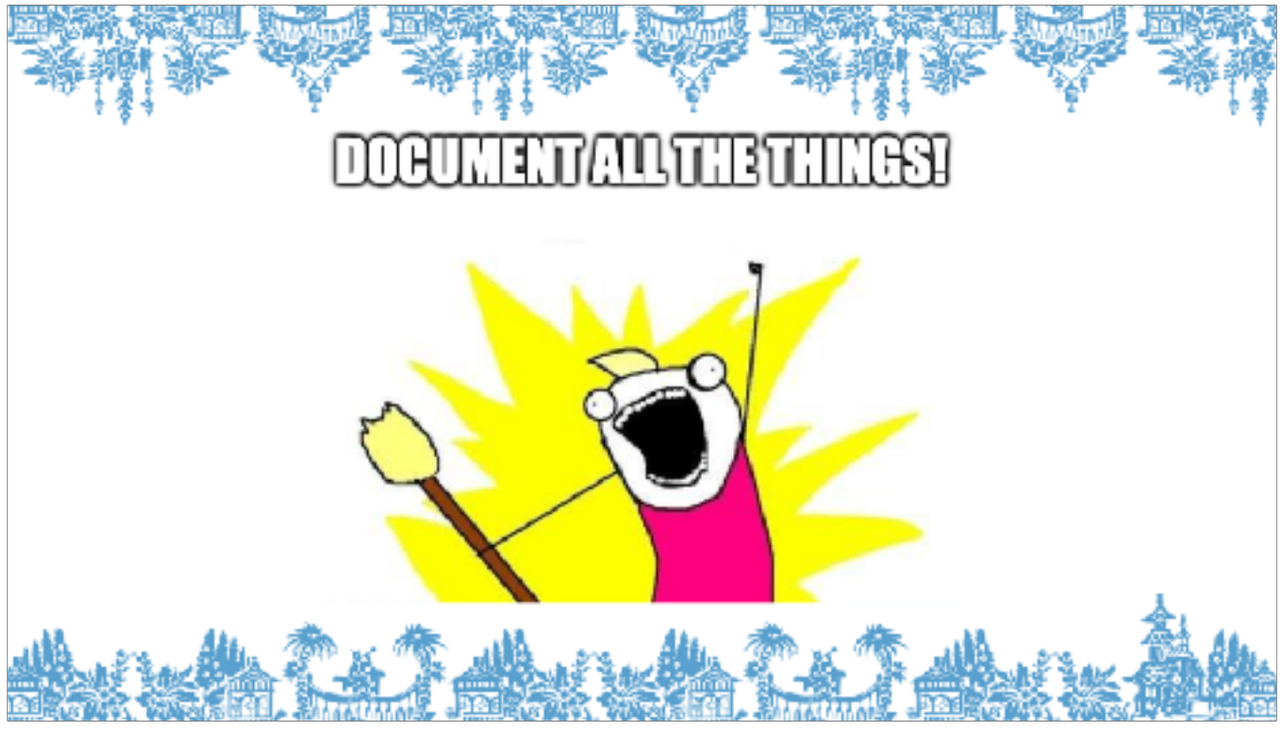
(hold) it means making choices and using the tools at our disposal to reduce the friction that stands between us and the goal. As we reduce the friction, the energy required to reach our goals is less. And what is that goal again?

Always Be Capturing

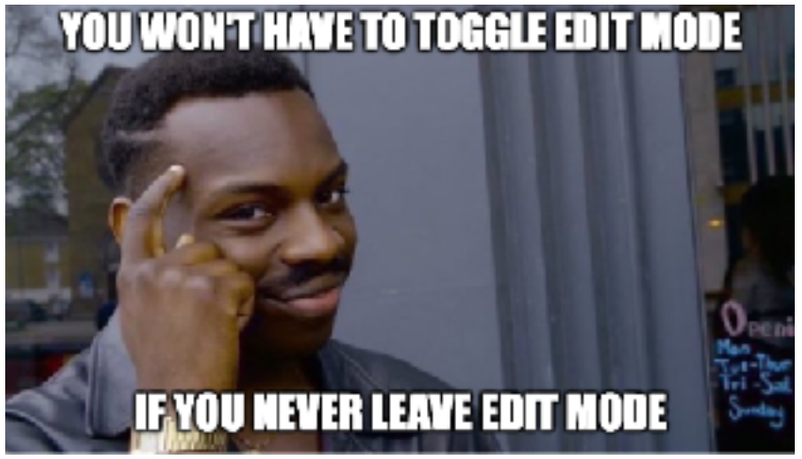
Remove the mental cost of considering the question

"is this worth documenting?" By documenting everything by default.

A practitioner of zen automation has favorite ways to quick start a new doc file, and they do it all the time.



We'll remove the friction of deciding what folder to put something in
with an Inbox, a Pig Pen, a catch all folder to be sorted later when it's place becomes obvious.. or my favorite: lean on search and never worry about arbitrary organization.



Let's live in edit mode. There isn't a cost associated with switching between the document viewer and the document editor, because we never left the editor in the first place.

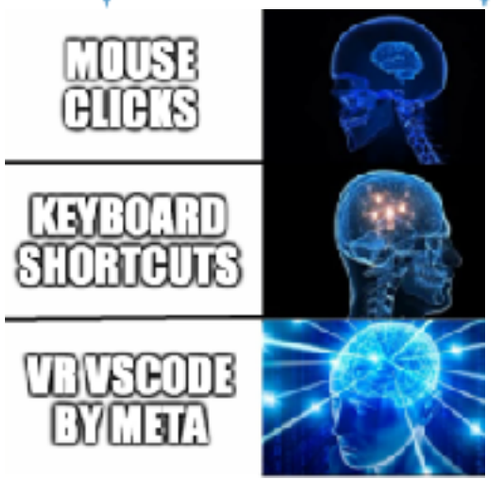
* if the wiki is in something like confluence, we leave that tab open in edit mode instead of view
* we'll prefer raw markdown in vscode over a separate app

Small choices that alone don't cost anything, and relieve minor friction, but in aggregate support a shift in mindset about how we complete a task, and a rebalancing of what we value as part of our work.

And not to get recursive on you, but those workflows can be further smoothed over by personal automation, orchestration of your local machine that bends it to your will.

maybe an alfred workflow to open a specific tab of the browser for the document you're working on

a fuzzy file search so your notes are always right nearby
Not everything has to be a formal document for the historical record, maybe it starts as 3 lines in a file on the Desktop



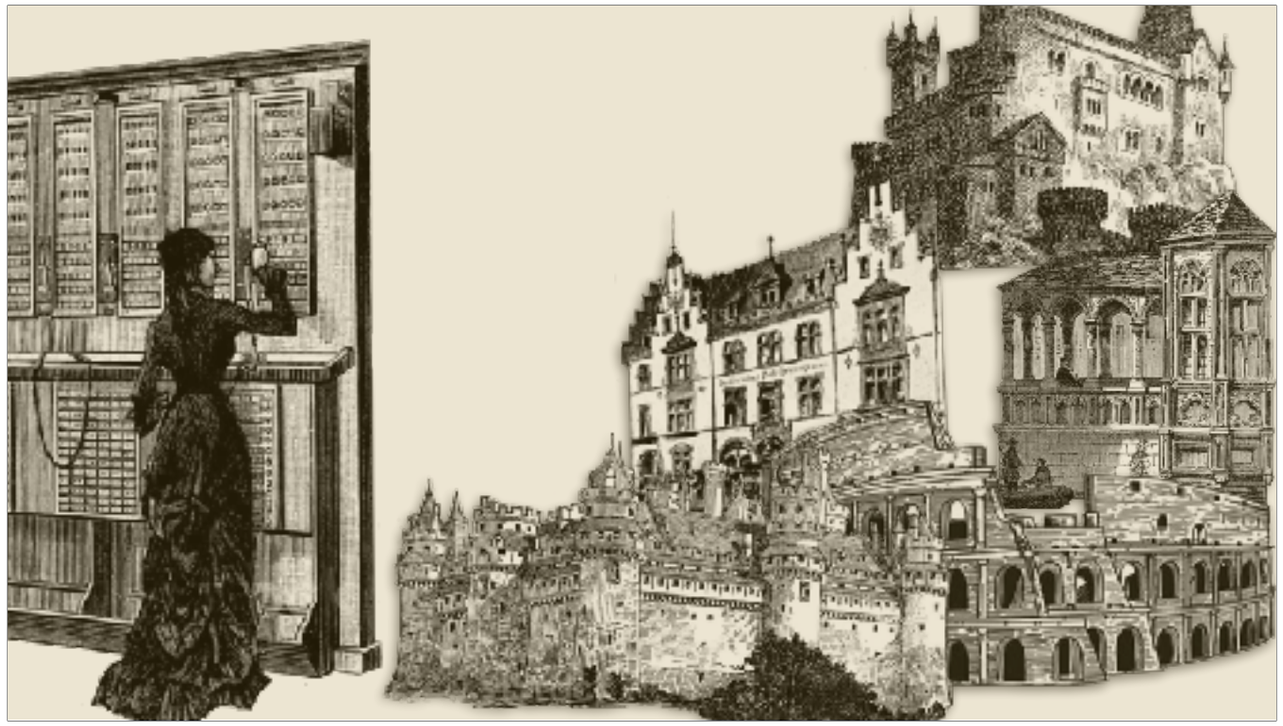
We'll shorten the round trip from thought to action by paying attention to our workflows, capturing them into a text file, and turning the inner eye of Zen Automation on ourselves.

how again?

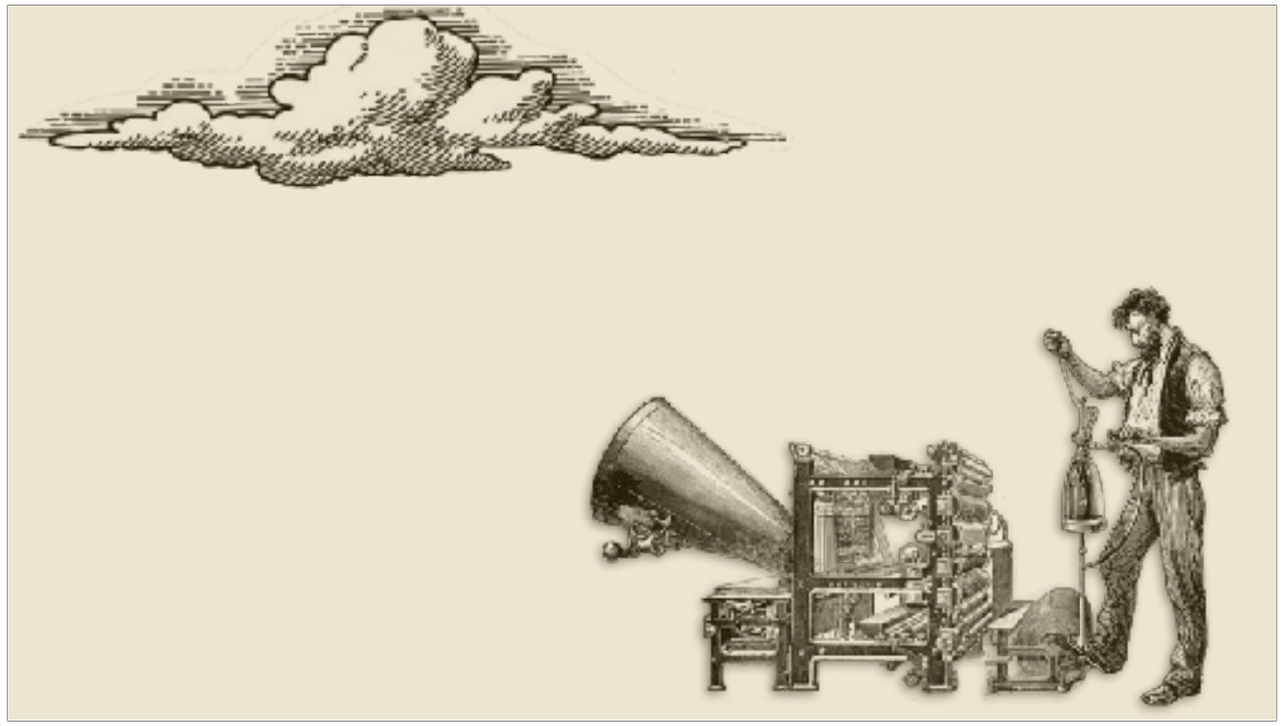
Always Be Capturing

I looked over the file the client had sent over, <> a lot of extra data we didn't need. I wanted name. address.

Used a vim macro to chop out the cruft, <> and since recorded vim commands are just text, I copy pasted them into my notes.

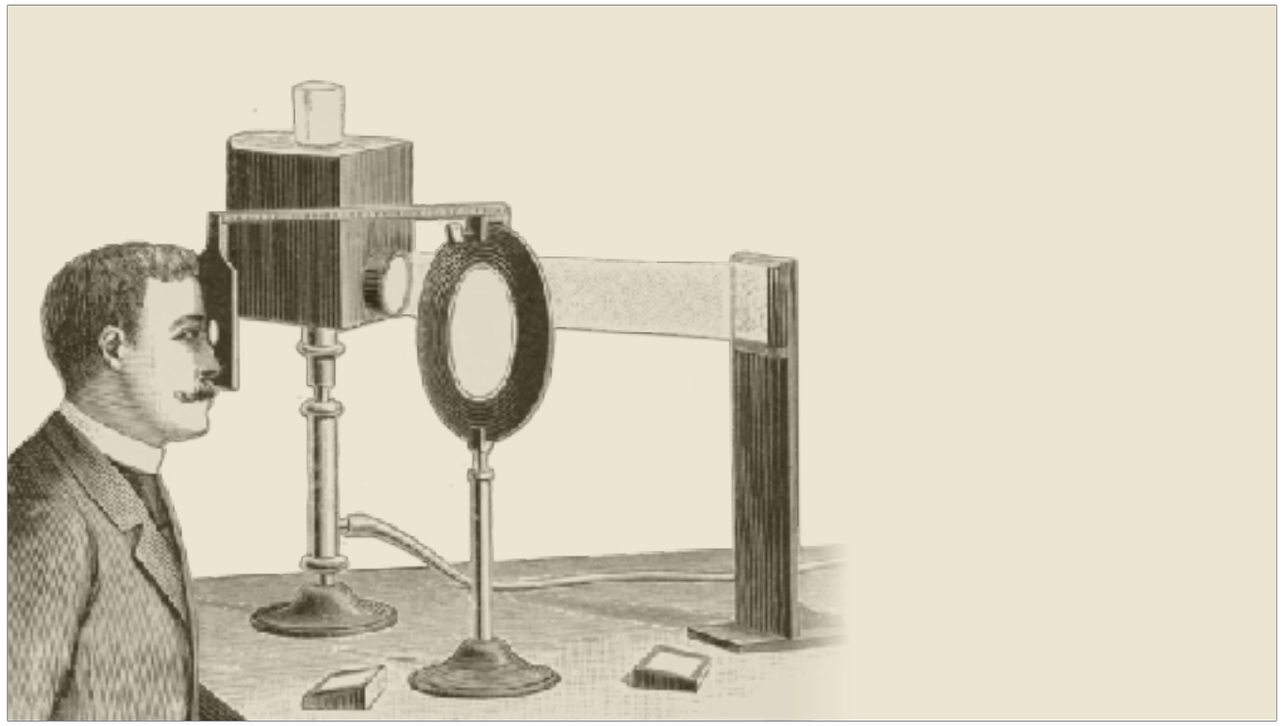


Could have just as easily used excel or google sheets to do the same work and written out the steps.
I suppose you could do with awk if that's your jam.

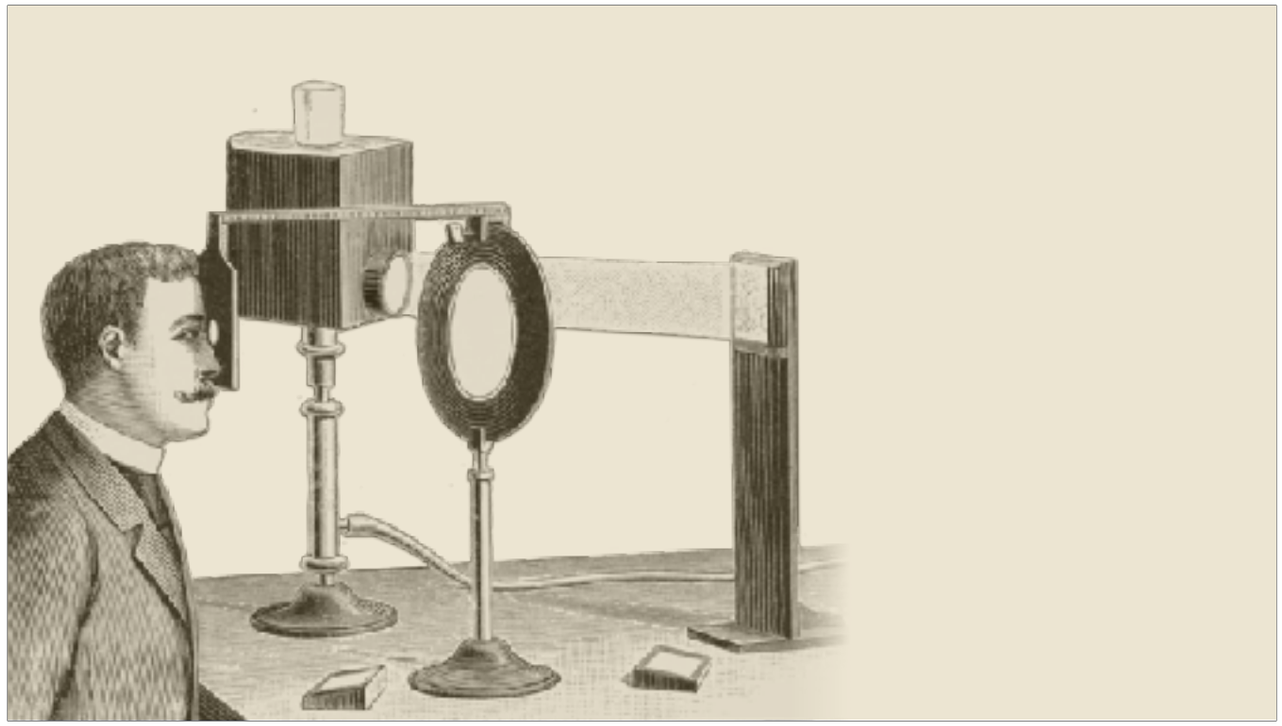


Getting it from a tidy list to the servers it needed to live on was non trivial and I'll spare you the specifics. <> But needless to say my markdown file filled with commands and AWS gov-cloud hoops to jump through. (beat)

Predictably, the "one off task" that I would never have to do again... <> came back the next morning. The client found a few more locations <> that needed to get thrown in, and could I do it again?

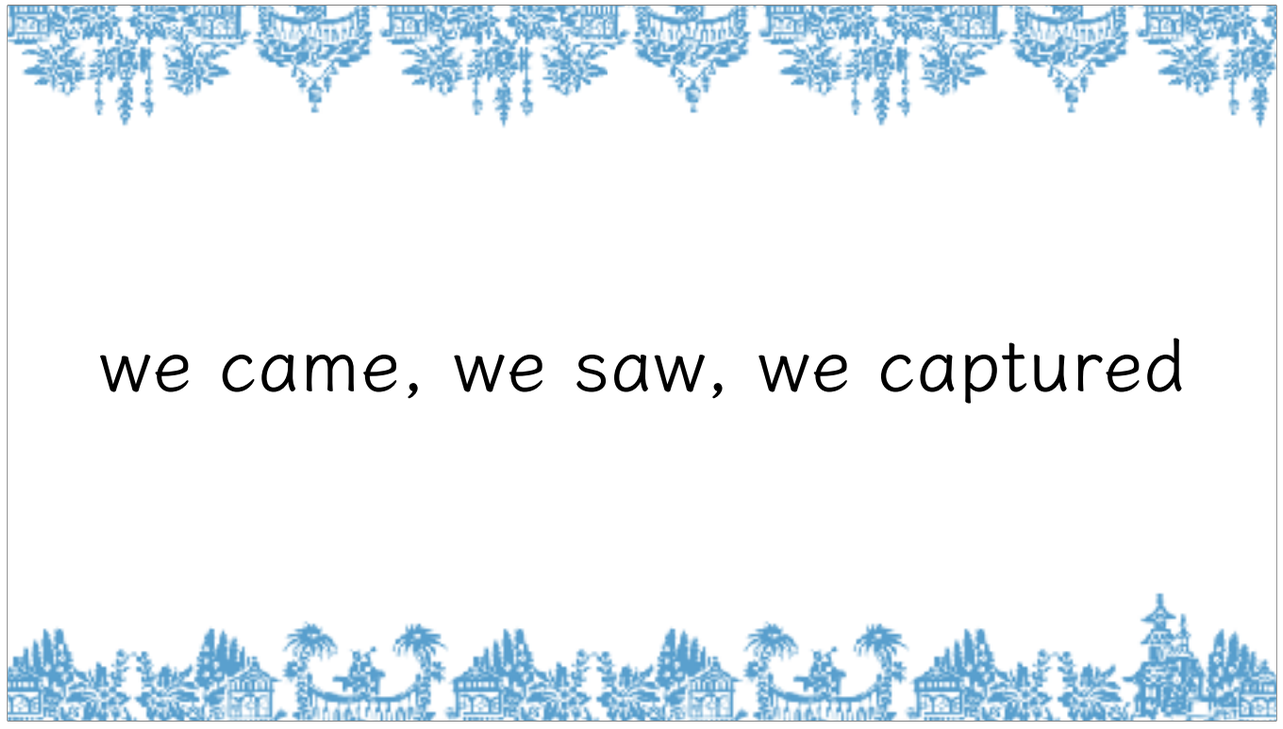


It was less than 24 hours since I had last added locations to the app, but instead of charging ahead with my memory of the steps I took, <> I opened the document from the day before and followed along with myself of yesterday as I went.



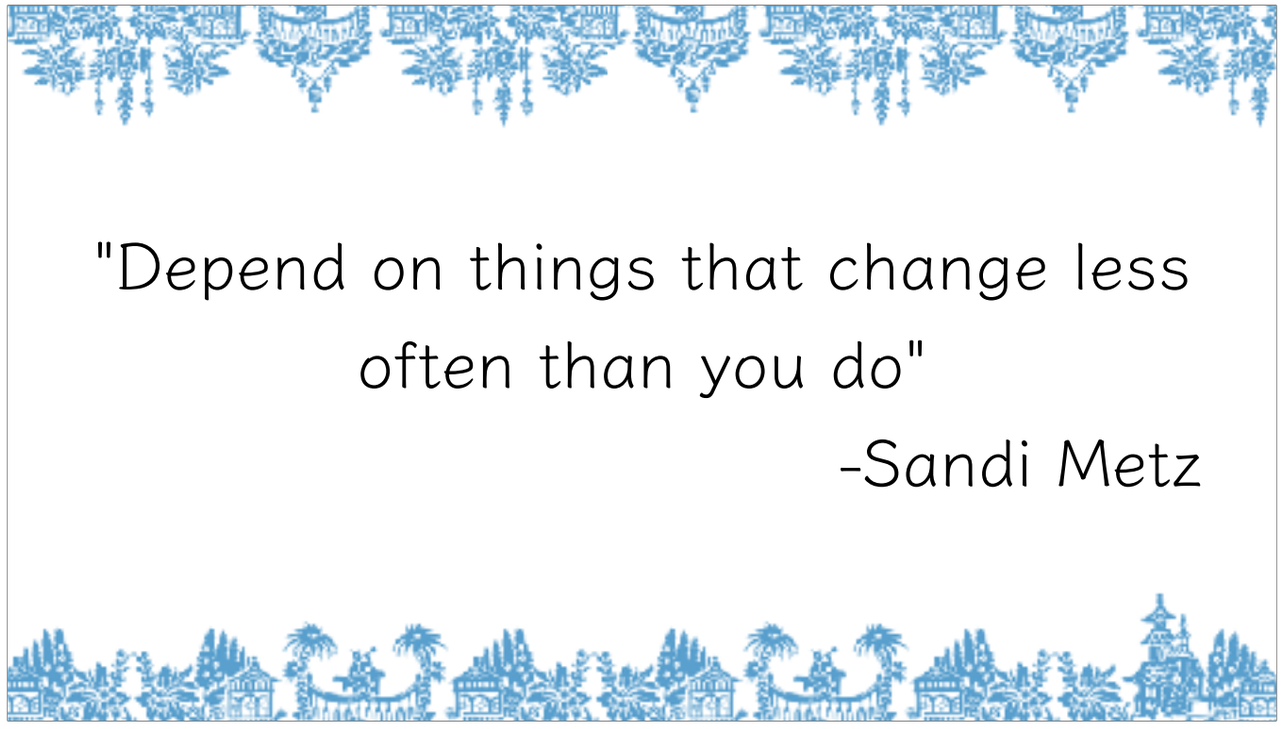
Here is where we encounter the second principle of Zen Automation, in which we will..

"Refine to Be Fine"



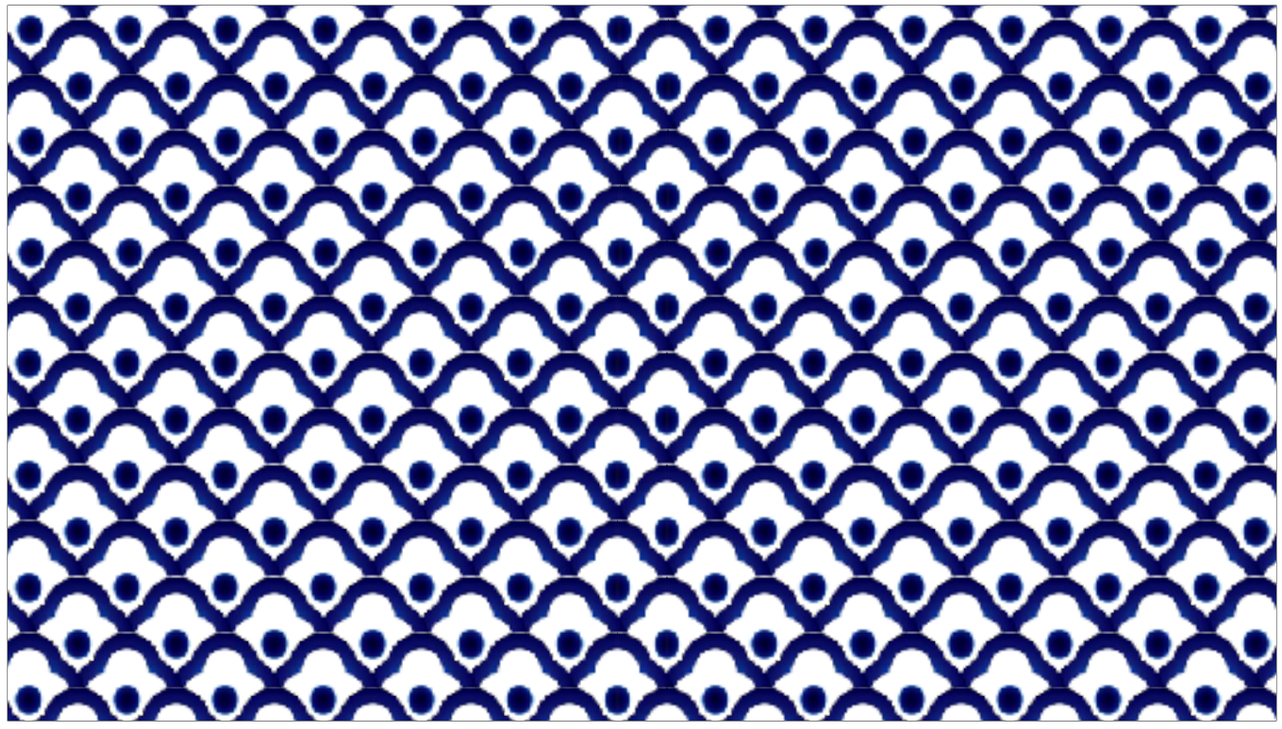
because we've written it down, we can iterate on it

if you don't write it down, you're iterating on your memory of it



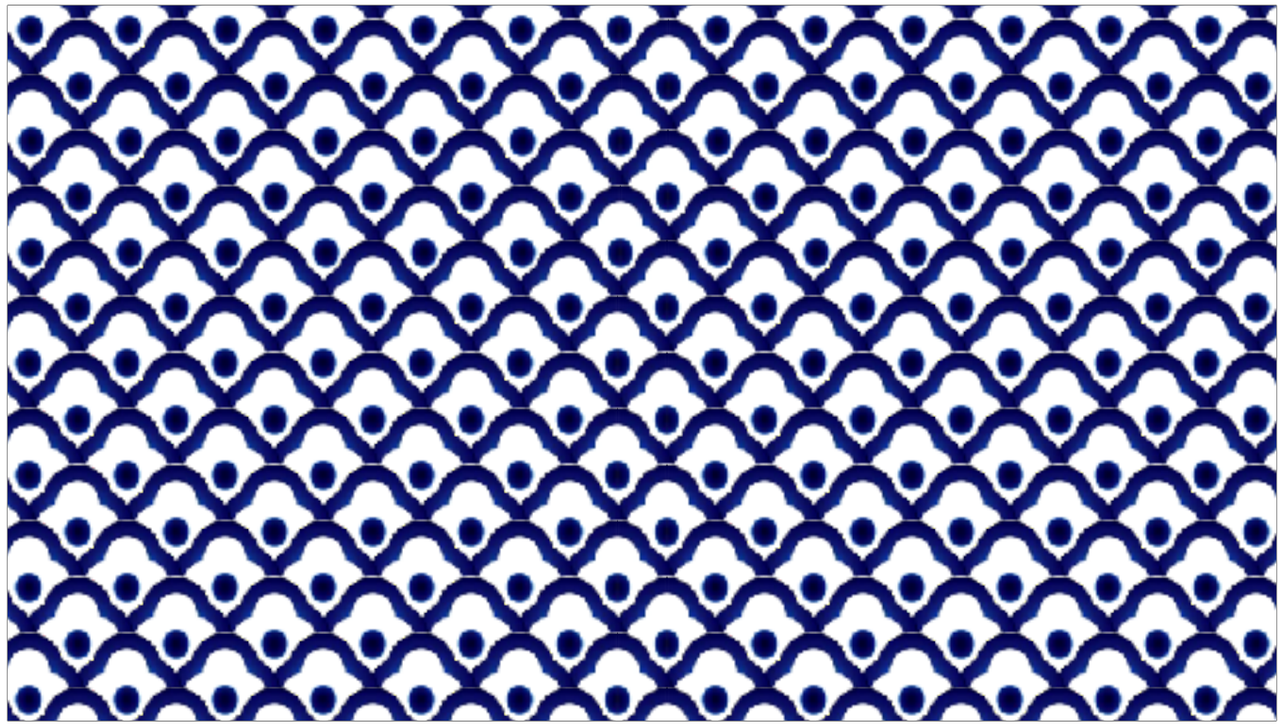
And Sandi Metz said <quote>
So that was written in the context of code dependencies, but it's good advice. Rely on stability, build on solid foundations

So stay away from human memory, it is buggy AF <>



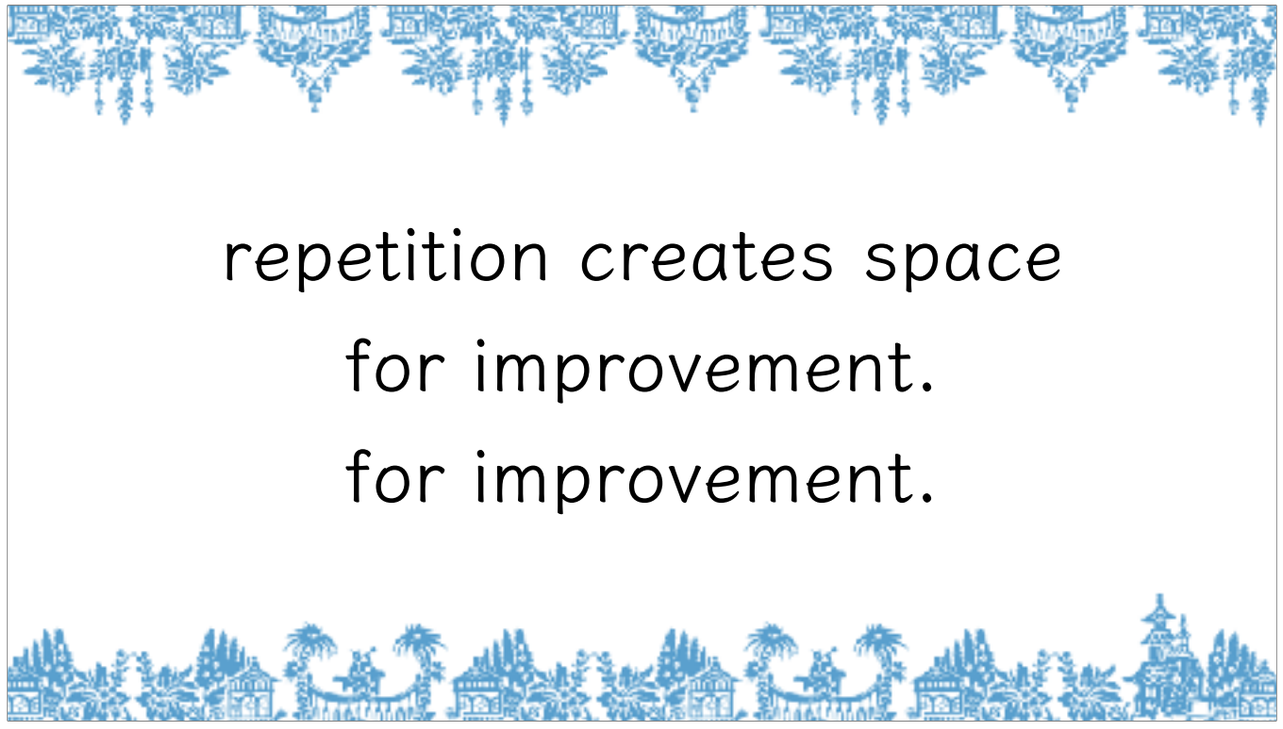
The situation was essentially the same as the day before, but instead of rediscovering the process, I copy pasted over the macro to reformat the text.
<> I had been a diligent practitioner of Zen Automation, each command was in my document. Copy/pasted a line at a time into my terminal... <>
not having to spend grey matter compute cycles remembering what had been effective yesterday, because I still had it.

The documentation essentially code, a checklist akin to pseudocode, automation to be executed by the homo sapiens runtime



Guided by the first and second principles we are already reaping the rewards only the second time through.
Time spent looking up how to use SED for the thousandth time can instead cut straight to the chase, with a fraction of the time, effort, and toil.

And what was the cost? typing into a document? We're professional typers.
Already it's feeling like automation, just following instructions, expending a sliver of the thought & energy.

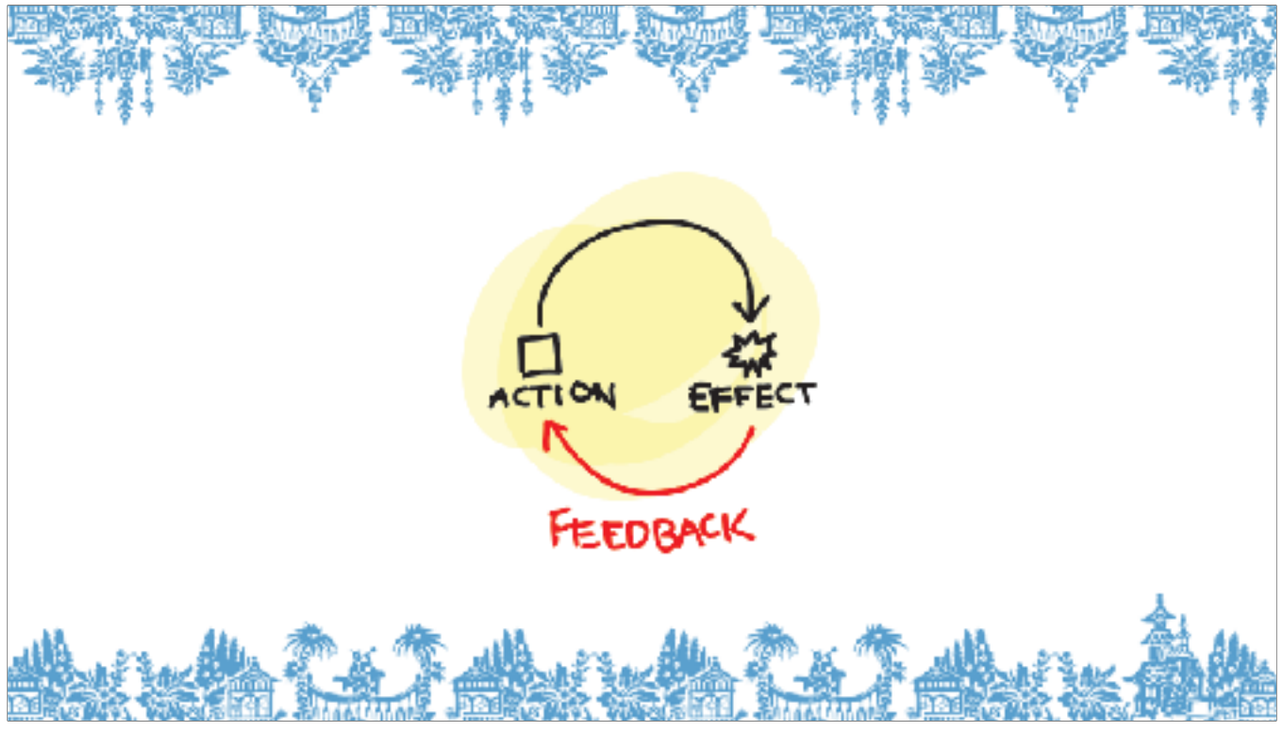


the first time.. we're assembling the path, clearing the way along an uncharted route.
In further iterations, we have the capacity to notice ways to enhance the steps because we're not hacking through the unknown.

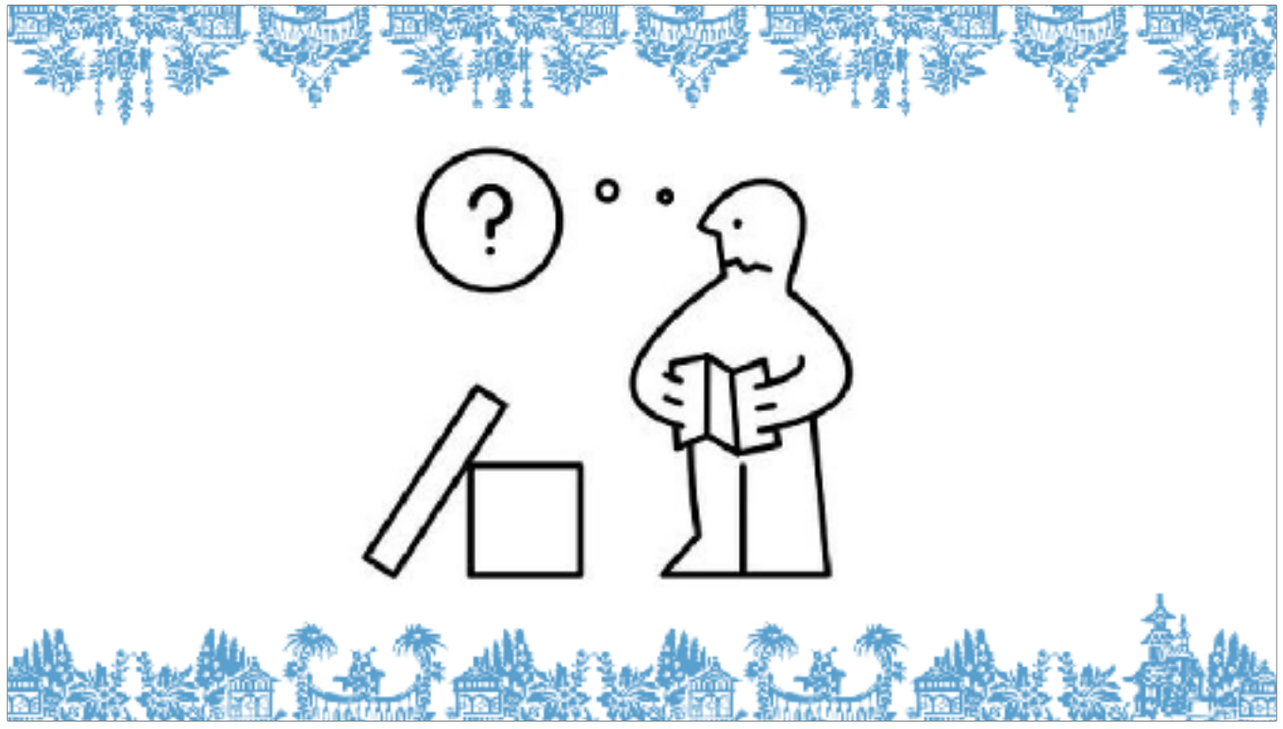
My brain has too
many tabs open

We've freed up mental bandwidth and given ourselves a chance to make things cleaner, faster, more robust

The mentality of working in edit mode comes in handy again.
This time we're not lowering the activation energy to create the thing in the first place.. but the activation energy required for improvements
because our tooling is ready to modify and update, and our mindset is there to match, small changes that make the script better jump off the screen

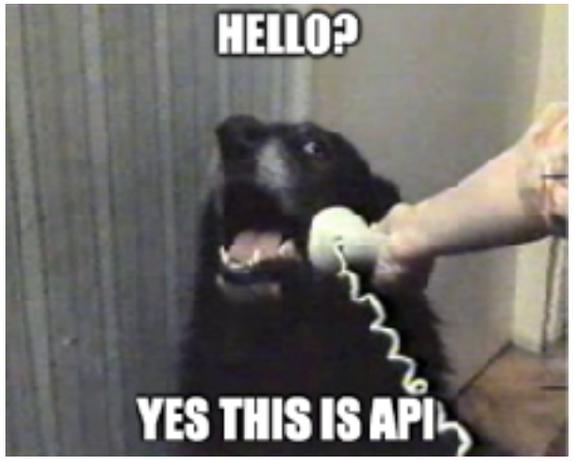


we construct a positive feedback loop that reduces slog and cognitive load one manual iteration at a time
..which frees up more opportunity for improvement
..which in turn frees up more opportunity for improvement!



and what can improvement look like at this step?

descriptions of actions can be converted to command line snippets,
actions that used to take a mouse click get converted to a command



something once done thru a GUI can turn into an API call

combine 2 steps into one with a pipe and xargs

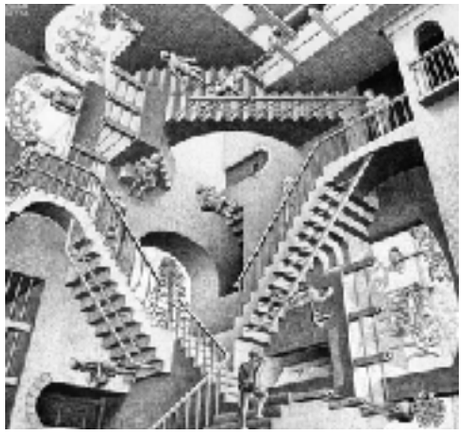
parameterize steps for re-use, anticipating a quicker on-ramp to eventual programmatic control

un-blur steps-- things like "Make sure its running" becomes a checklist of **how** to make sure it's running

Adding QA checks

maybe it's not something that a command line snippet can do, maybe it relies on a gem or a class from the codebase, so we copy-paste the lines we ran in irb or the console last time (which we dutifully captured in markdown) and make a rake task or script that can run in the right context... and then write down the command line snippet for executing it

and these are just some of the more computer-heavy improvements, it can be more human
maybe we read a line of what we wrote and it no longer makes sense so we clean up the description

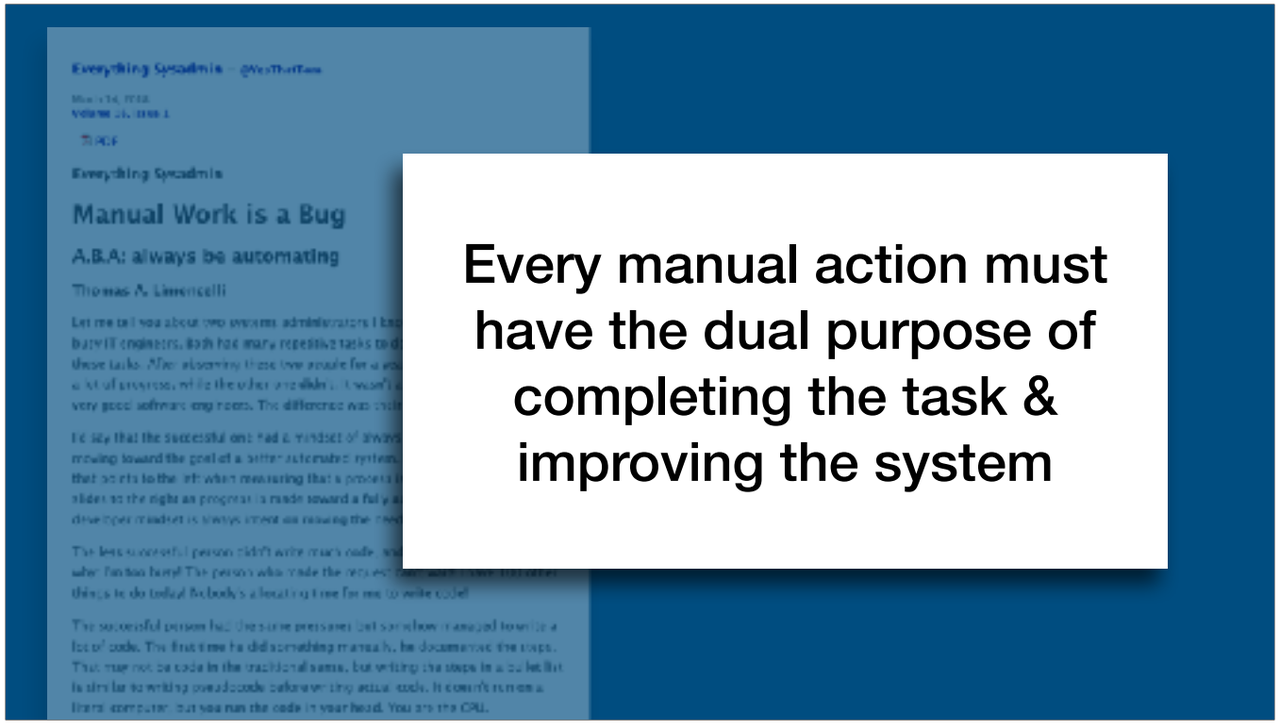


there's a nigh infinite number of ways to tighten up our script.

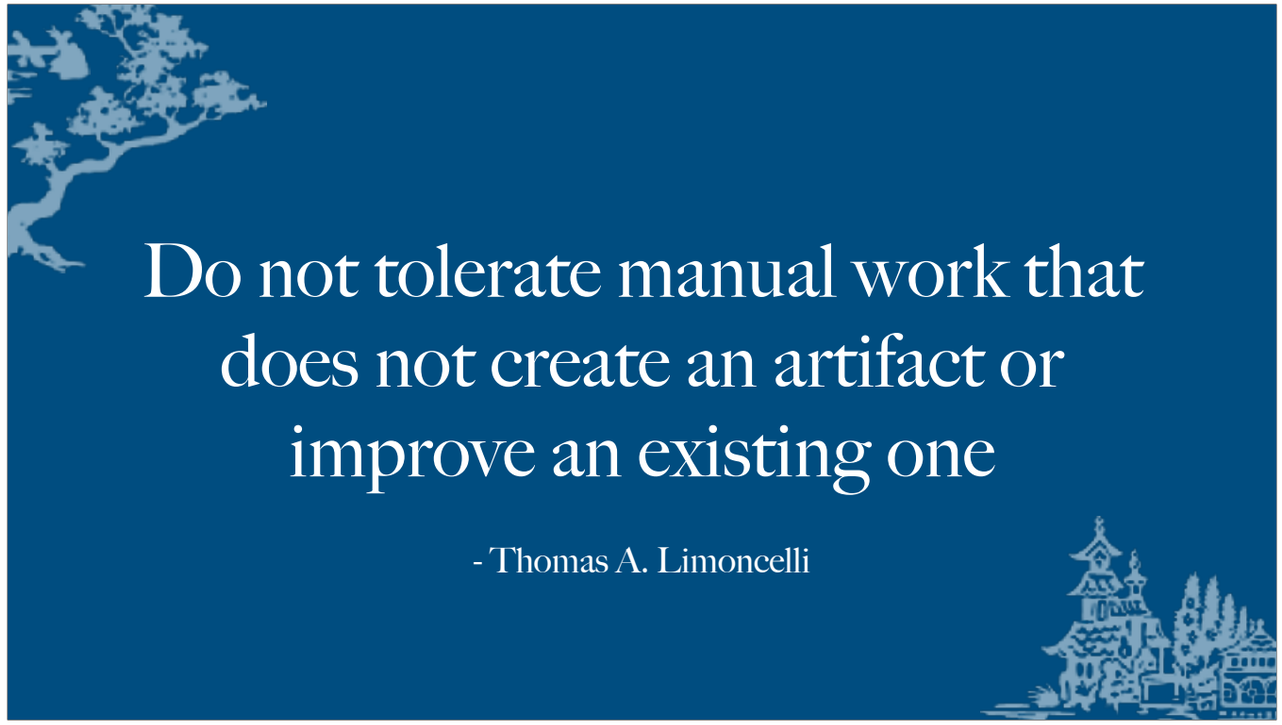
what's at the core of this principle is that each *use* of the script also makes it better. we...


Refine to Be fine

As Thomas A. Limoncelli put it..

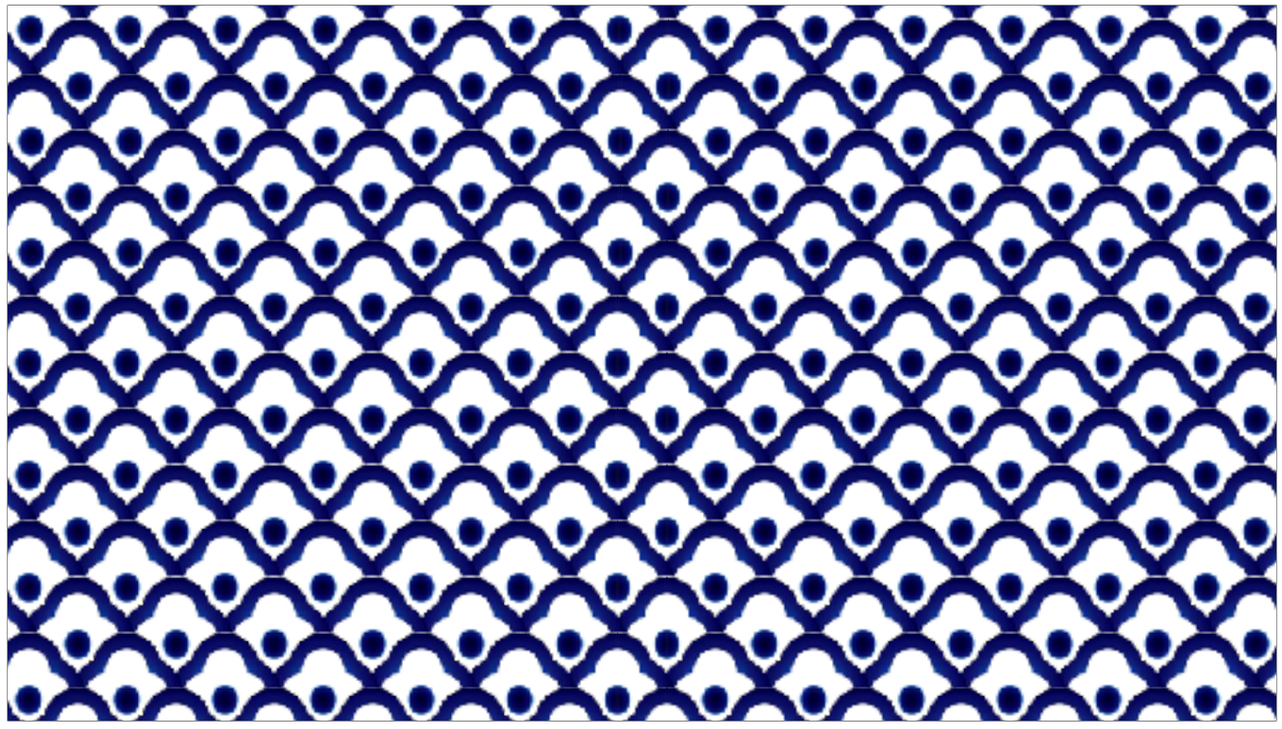


Every manual action must have the dual purpose of completing the task & improving the system
or put another way

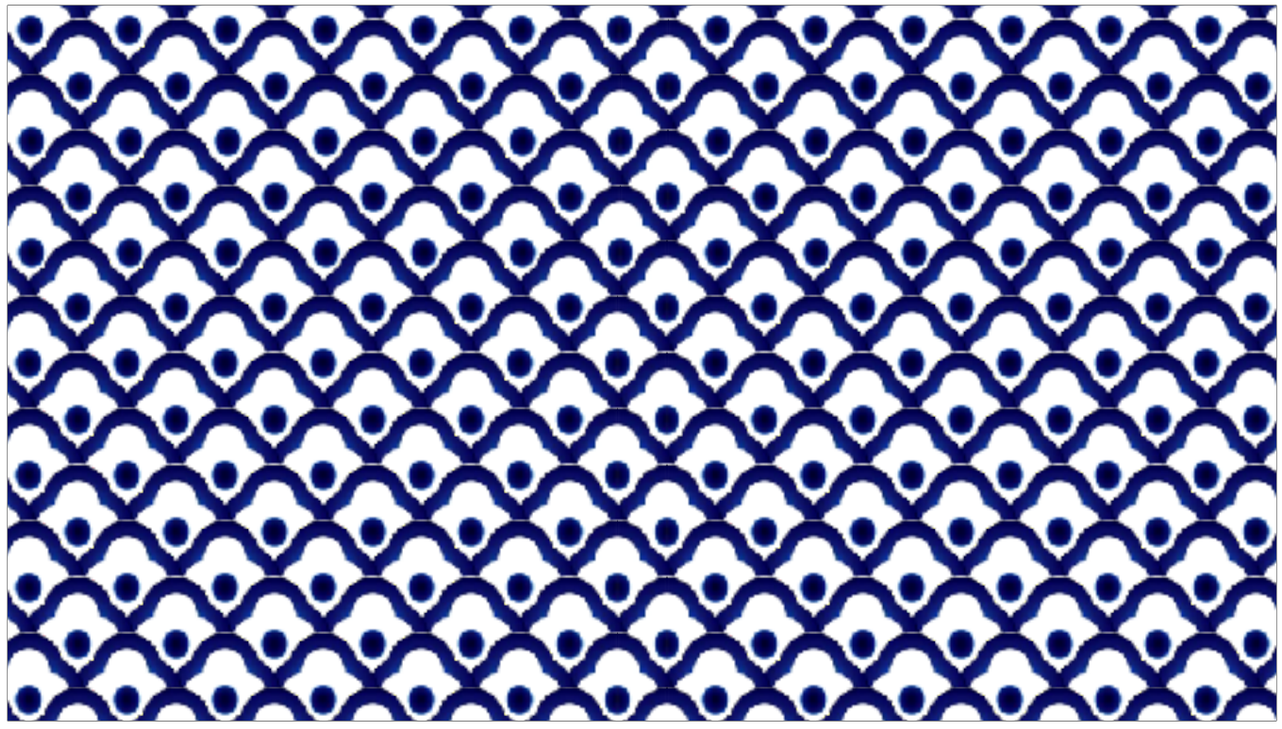


The practitioner of Zen Automation does not tolerate manual work that does not create an artifact or improve an existing one

this doesn't mean that everything has to go smoothly the >1 times through or was a wasted iteration.

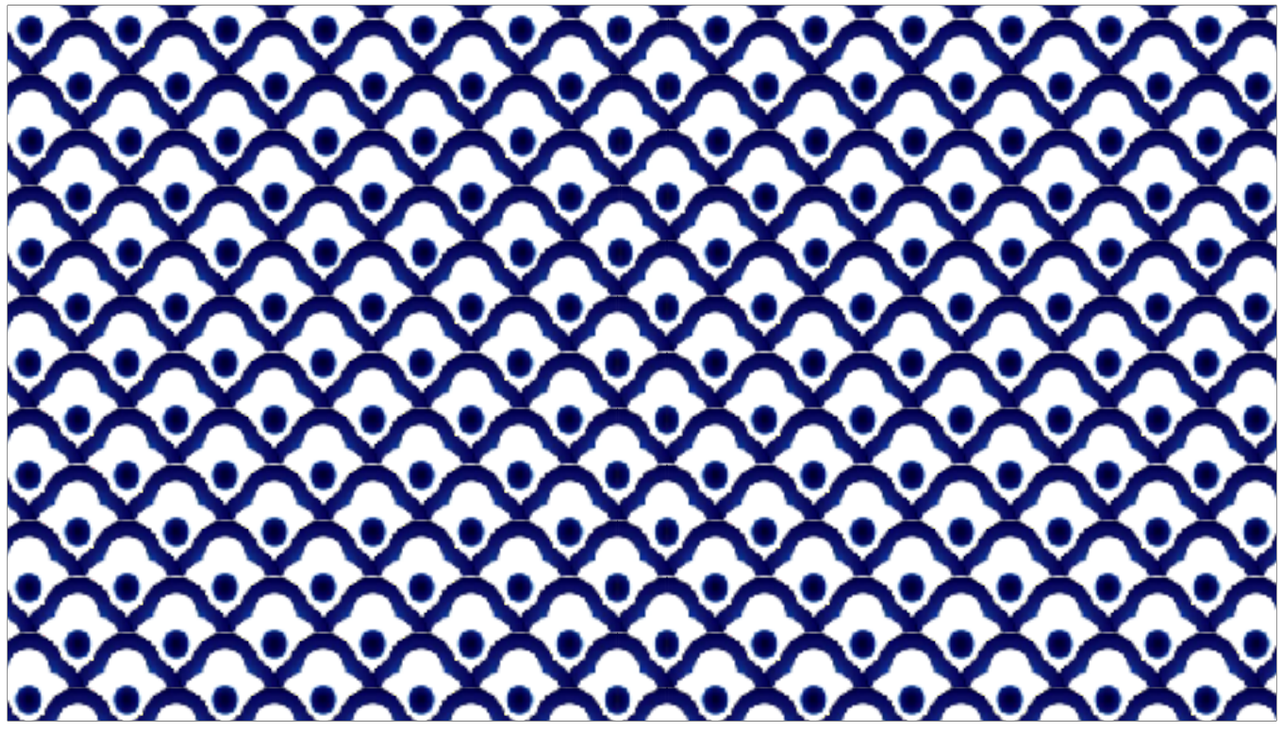


Finding an edge case, new errors, exotic failures, debugging them and carrying on is extremely valuable experience, because we will have written it down for the next dev to use the script, which yes, can include us tomorrow morning



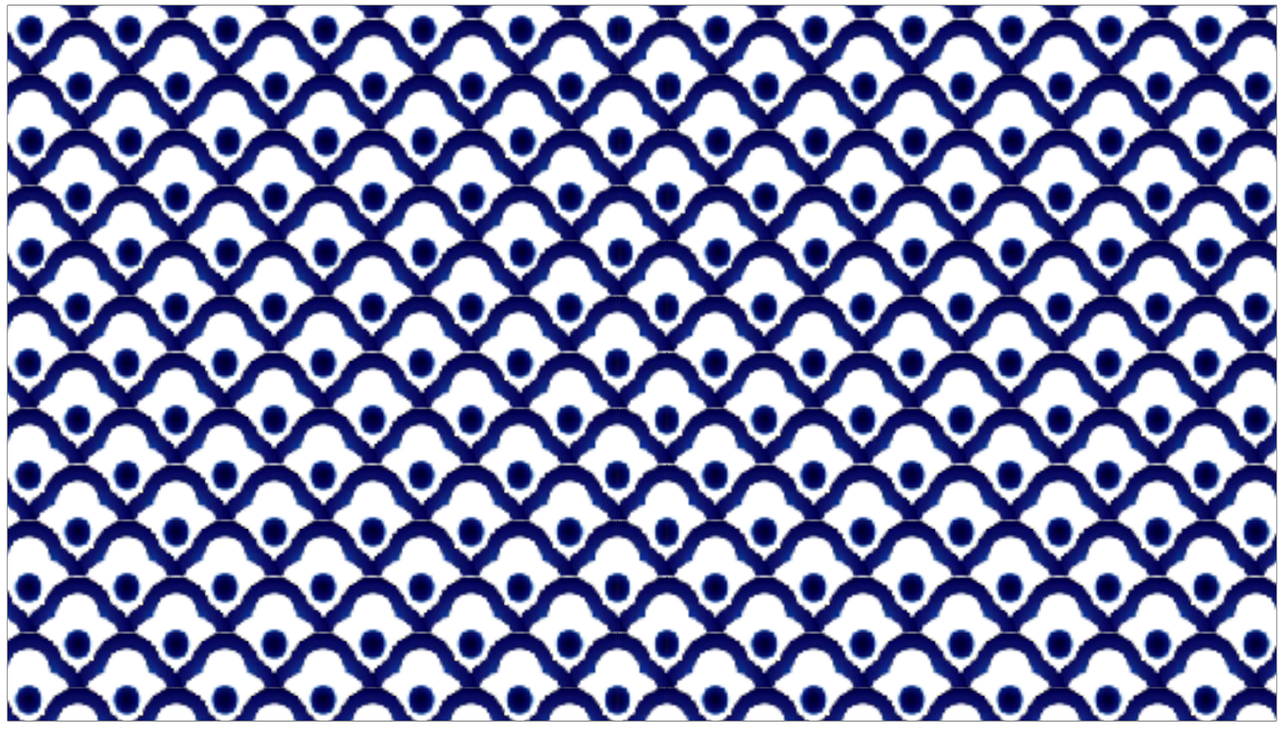
So when the third request to update locations rolled in, I was starting to catch on that this "one off task" wasn't going to be a twice and done deal

I upgraded my markdown file to an executable ruby script



Do I mean that everything was automatic, systematic, hy-dro-matic greased lightning?

No, I made myself a 'do nothing script'



means the script doesn't do anything directly except print out each step, one at a time, waiting for our confirmation to continue each tick.

puts driven automation

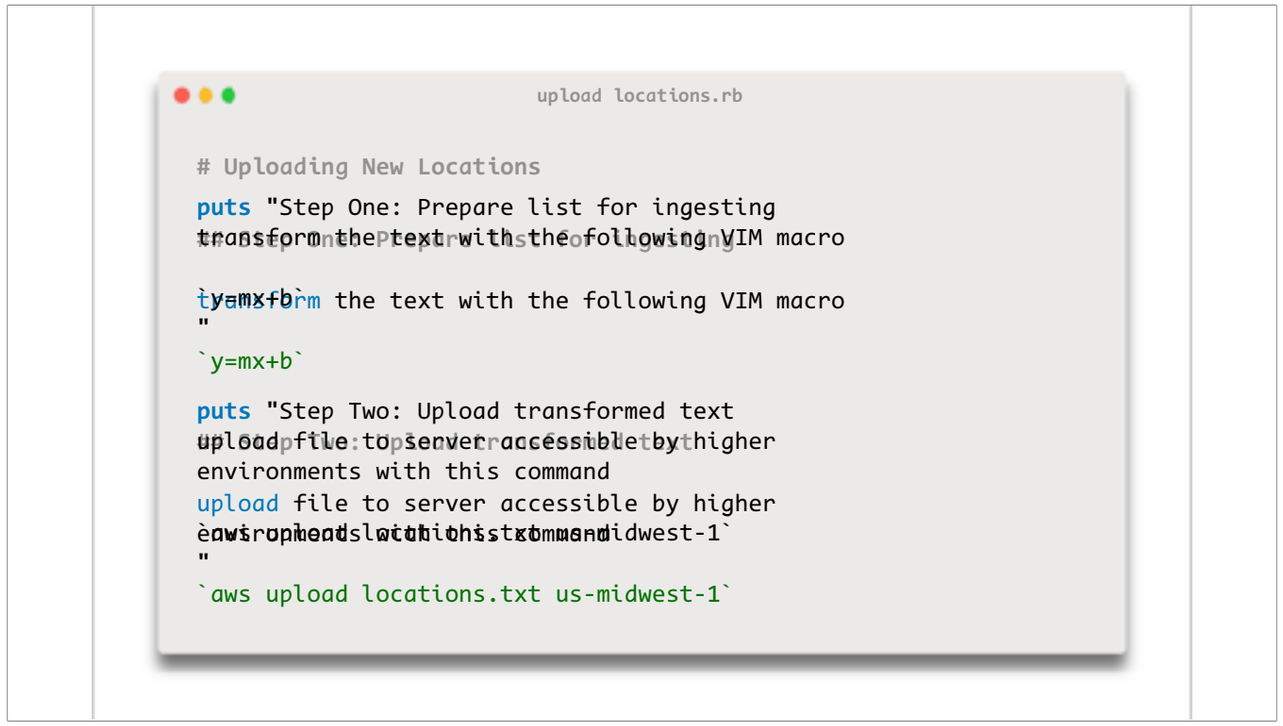
Think of ruby as our co-pilot, running the in-flight checklist. Pair programming terms: Ruby's the navigator, we're the driver.

Might look like this..

upload locations.md

```
# Uploading New Locations

## Step One: Prepare list for ingesting

transform the text with the following VIM macro

`y=mx+b`

## Step Two: Upload transformed text

upload file to server accessible by higher
environments with this command

`aws upload locations.txt us-midwest-1`
```

Let's take for example this markdown file, typical for what you might see in project documentation or a README. But we live in edit mode, it's all plaintext, does this file have to end in ".md"? let's make it ".rb"
<>



The steps written out are not valid ruby syntax, so we'll make them into strings, <> and they can stay essentially as they were. We can read strings in a ruby file just as well as text in markdown to refer to the docs.

upload locations.rb

```
# Uploading New Locations
puts "Step One: Prepare list for ingesting
transform the text with the following VIM macro

`y=mx+b`
"
gets
puts "Step Two: Upload transformed text
upload file to server accessible by higher
environments with this command

`aws upload locations.txt us-midwest-1`
"
gets
```

And we'll use PUTS, everyone's favorite debugging tool, to print the steps out when we run it.
In between, though, we'll put a method call that you might not use day to day, and someone who's coming to ruby from Rails might never have encountered... gets

upload locations.rb

```
# Uploading New Locations
puts "Step One: Prepare list for ingesting
transform the text with the following VIM macro

`y=mx+b`
"
gets
puts "Step Two: Upload transformed text
upload file to server accessible by higher
environments with this command
"
`aws upload locations.txt us-midwest-1`
"
gets
```

gets receives input from the terminal and returns it. But we're not saving it here, there's no variable or method call.. it's a pause, so we can manually do the step, and hit enter in the terminal to move forward as confirmation.

if we were to run this file right now, each step of the instructions would be printed to the terminal, and execution would pause until we came back and pressed enter.

We're less likely to lose our place, but more importantly, we've lowered the activation energy to incrementally automate this process. watch.

upload locations.rb

```ruby
# Uploading New Locations
puts "Step One: Prepare list for ingesting
transform the text with the following VIM macro

`y=mx+b`
"
gets

puts "Step Two: Upload transformed text
upload file to server accessible by higher
environments with this command
"
gets
```

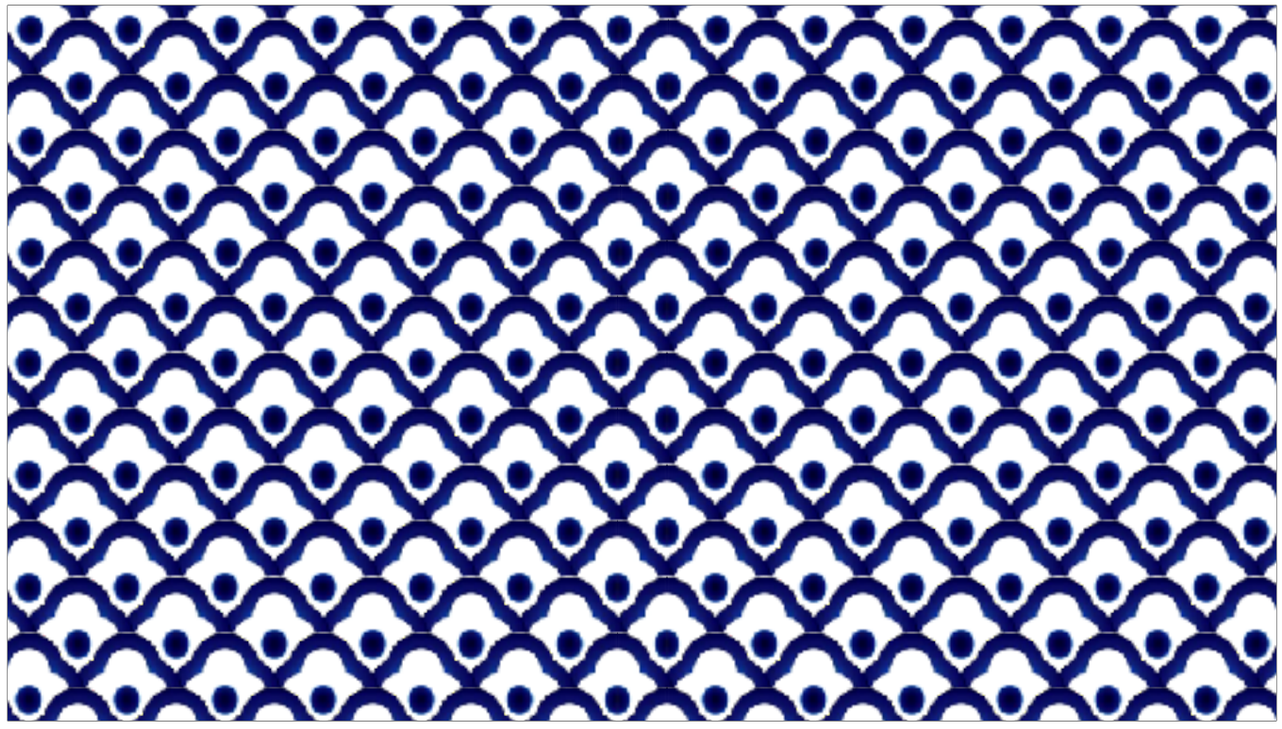
(moves text down, auto advances)

upload locations.rb

```
# Uploading New Locations
puts "Step One: Prepare list for ingesting
transform the text with the following VIM macro

`y=mx+b`
"
gets

system "aws upload locations.txt us-midwest-1"

puts "Step Two: Upload transformed text
upload file to server accessible by higher
environments with this command
"
gets
```

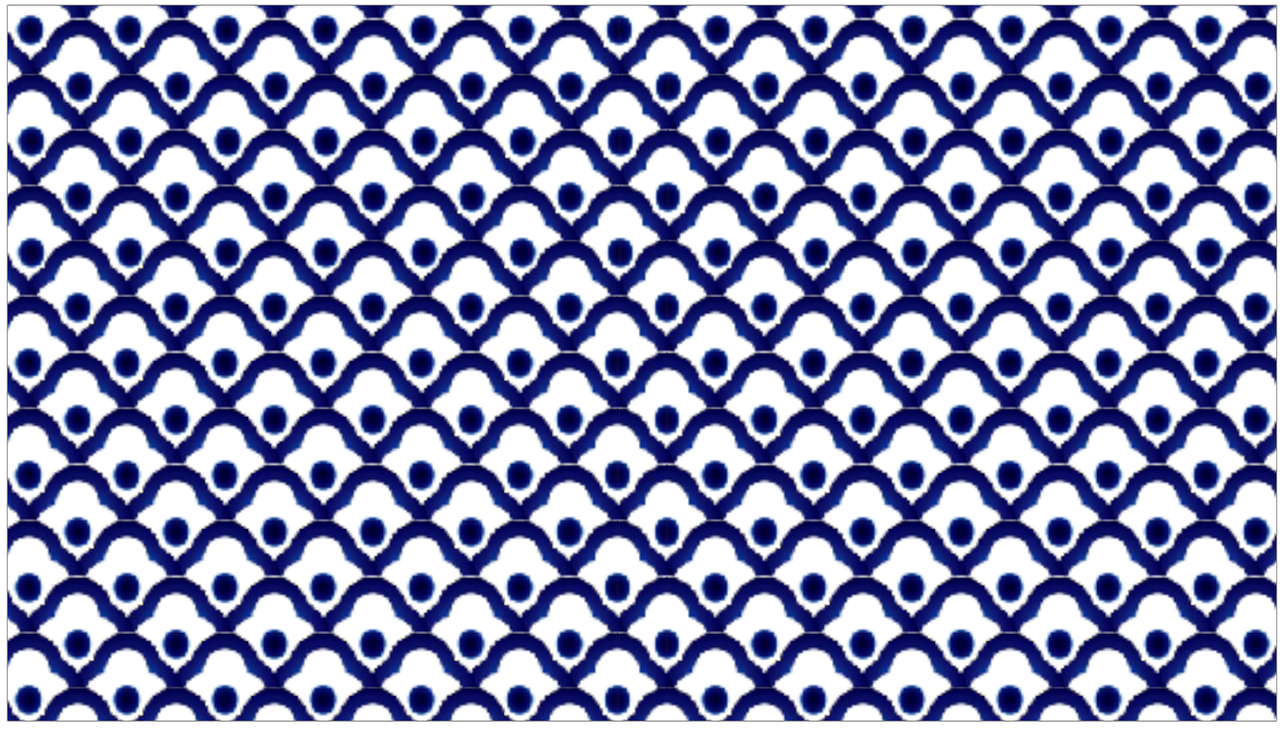
we'll use one of the ways built into ruby for executing shell commands to run that snippet for us. we don't have to copy paste it over, and we've automated step two.

so let's reword that puts



By keeping the description of what's happening, someone new to the project can get vital information about how the system is put together by walking through the steps as it goes.

if this is a script that's going to run all the time, maybe you get fancy and add a command line option that skips the puts-es where possible.



several iterations down,
* I made my plain text file into a do nothing script
<>
* I was the only person on my team using vim, so I switched the macro to some ruby

* I took descriptions of steps that were <> half-explained or written in my own personal jargon, and made them into something other people <> would understand.
Because I was preparing for the third principle of Zen Automation

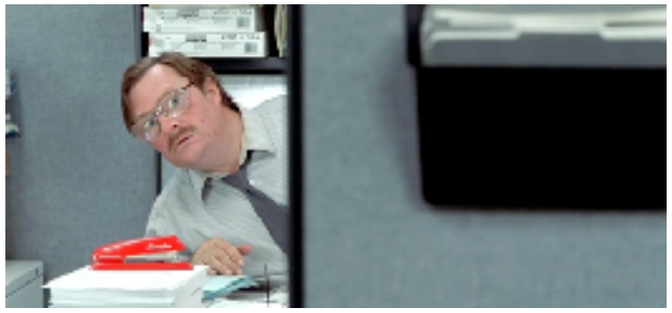
Share Early & Often

simply put, this mindset becomes most powerful when adopted by the group and turned into team norms & culture

if one of the major benefits <salute> of automation is reducing frustration and toil, then making documentation automation

documation?

available to the wider team shares the load, reducing burnout & overwork.

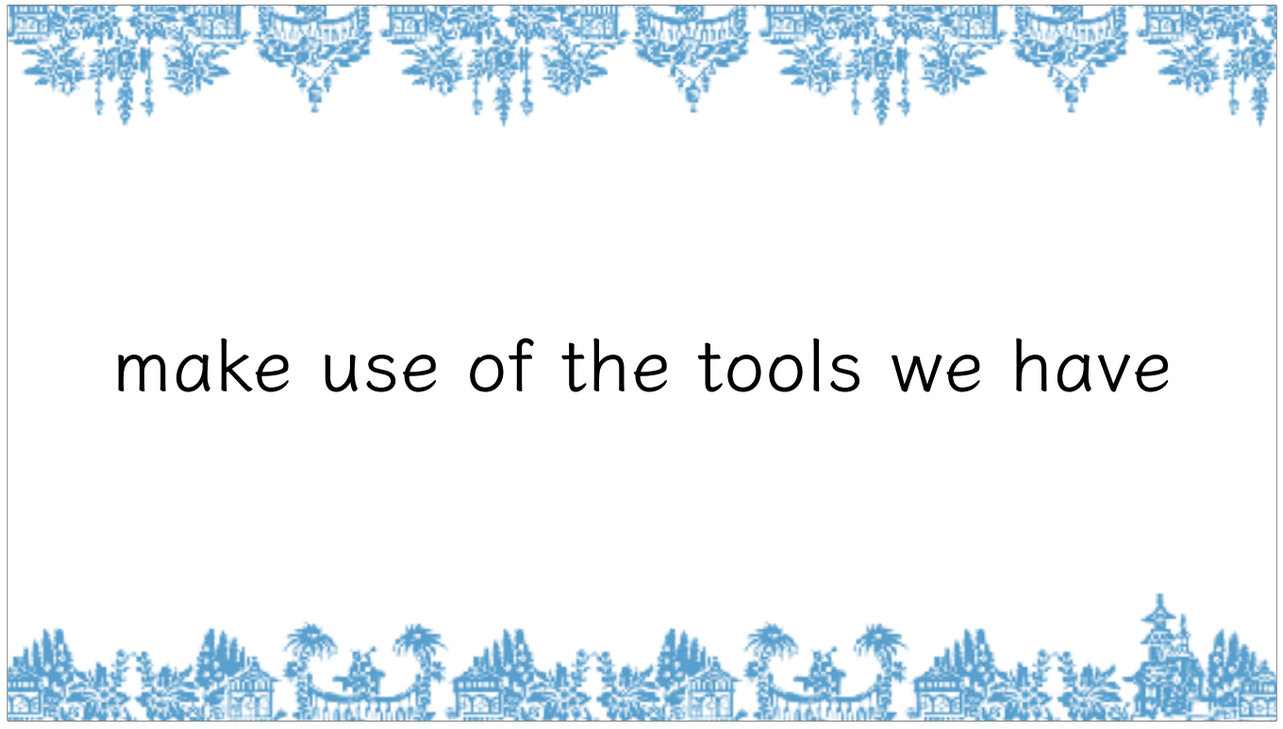


it's a common refrain... one person on the team does a task. Next time the task comes up, well it only makes sense that the same person do it again. they already know how it goes.

and they do it again, and again. soon that developer becomes a single point of failure. A lone sick day grinds the team to a halt, and even worse, it's no fun and not fair to that individual, and increases the chance that your single point of failure took that sick day to talk to a hiring manager

automation can seal
knowledge into a script

Rather than sealing the learning and knowledge of a process behind an individual, do nothing scripts that explain to a new user what's going on...can be a way to share knowledge of the system, while still getting things done



If the documentation and scripts live in a folder of the repo, you can leverage everything that comes along with that. Version control. Code Review, might just end up with some improvements while in PR, right out of the gate.

That folder, merely by existing, will encourage teammates to add to it, to pay attention to their workflows.

we'll celebrate the addition of new scripts, create a positive feedback loop.

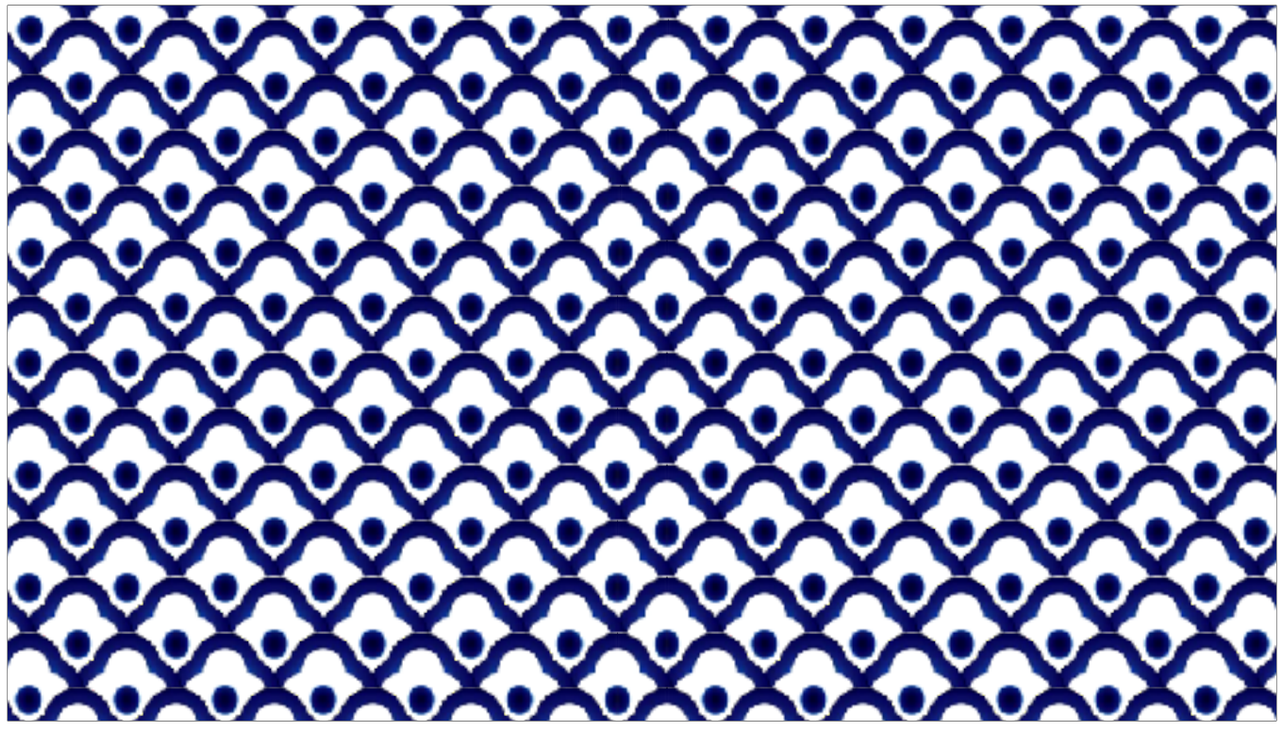
and if it lives near the code, it's more likely to get updated, especially if it IS code and code that the rest of the team relies on as well

the more ambitious teams might even find ways to write tests that fail if a change invalidates a portion of the documentation, as a reminder to keep it up to date.

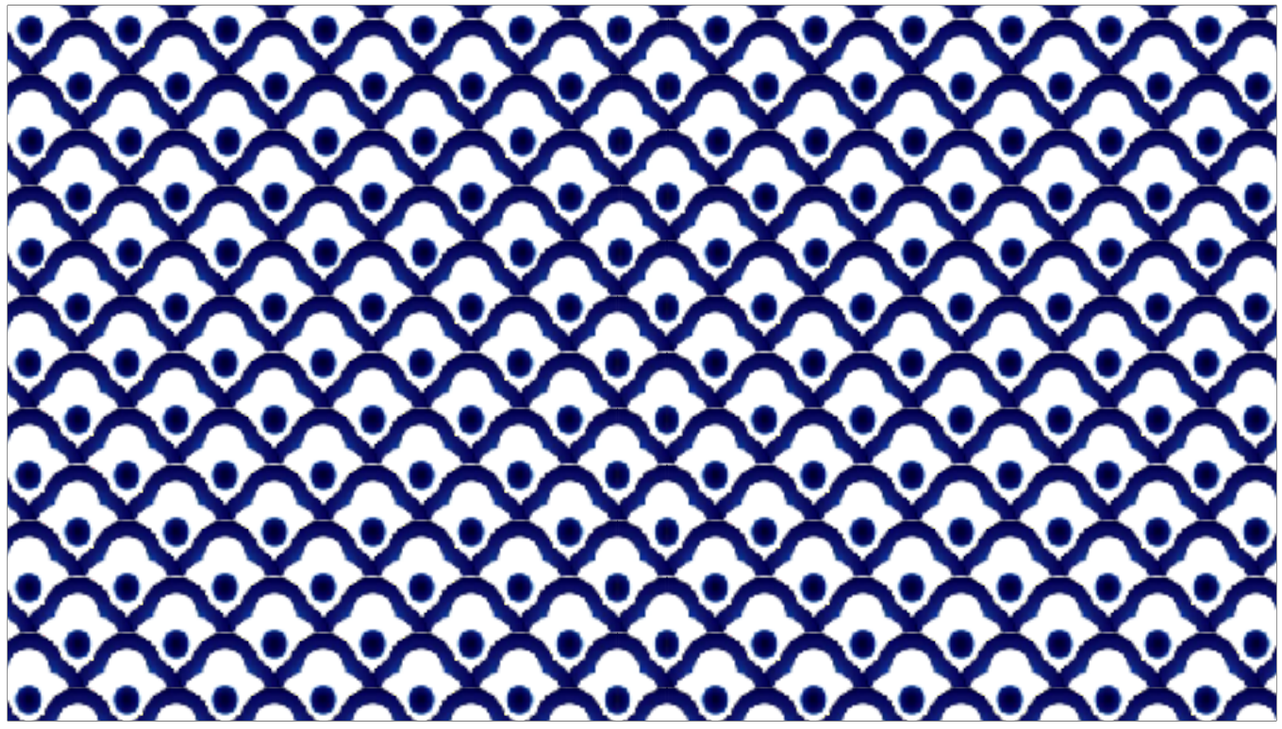
Sharing workflows also increases consistency. We avoid **costly** creativity, meaning when the same problem is solved multiple ways in multiple places. Divergent solutions require conditionals and complexity.

But most importantly, improvement becomes multiplicative. Mina finds an edge case we wouldn't have thought of, Belle's environment is slightly different and makes the script more robust, Brittany knew about a gotcha we didn't anticipate and catches the crash before it happens. As the team iterates together, improvement becomes almost incidental.

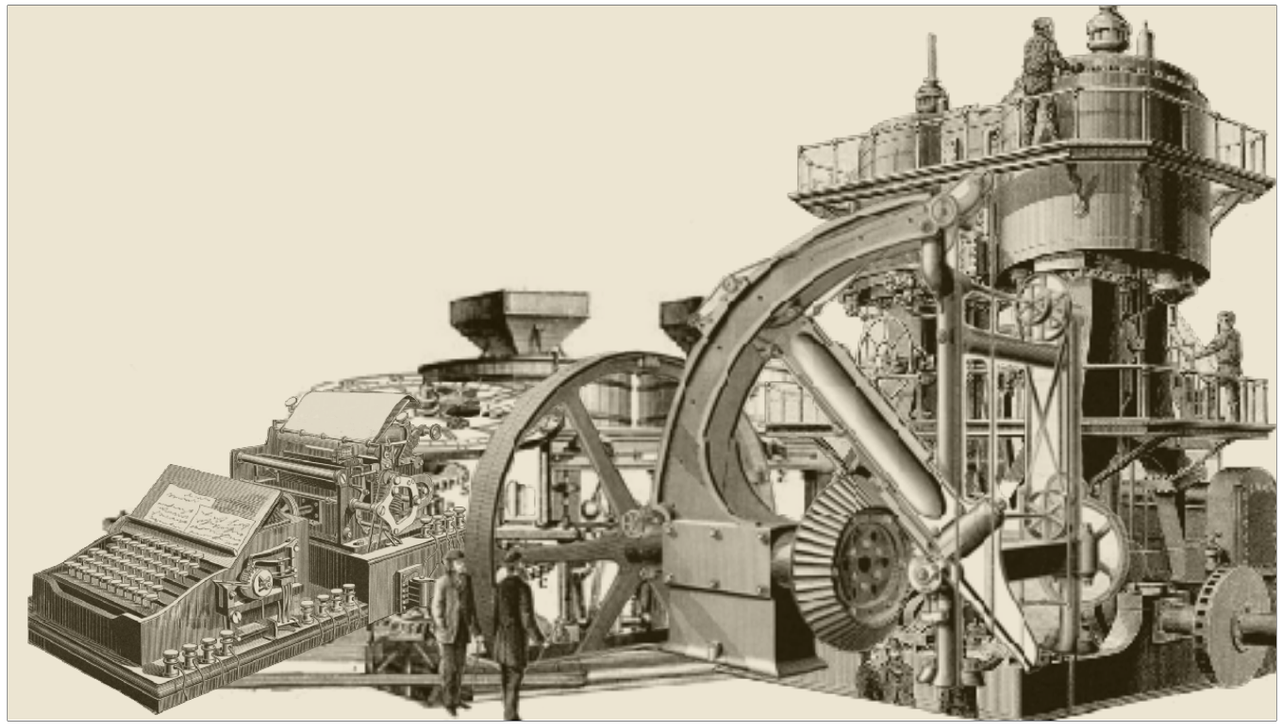
imagine walking away from a task and coming back three weeks later and having to do half the steps, in half the time, and it's been rolled into the project's CLI tool. Brilliant!



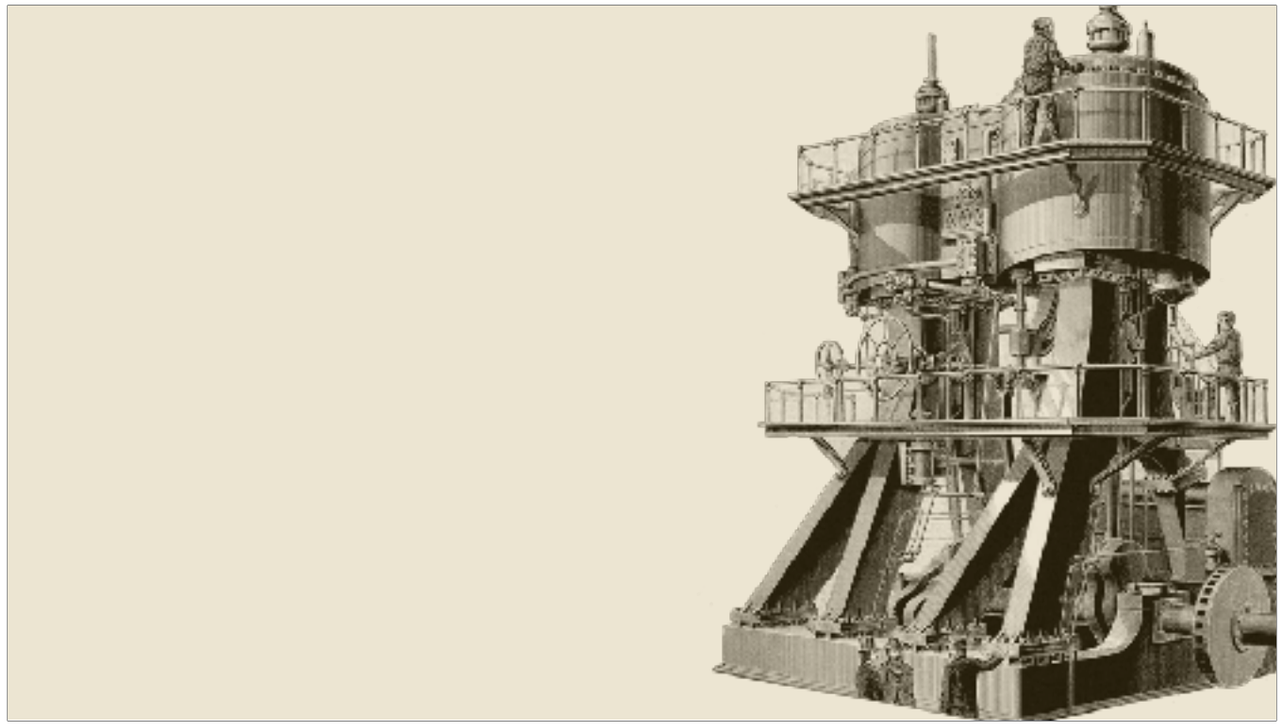
Soon CI for our teams will mean "continuous improvement" as we are 'always capturing,' and we 'refine to be fine' together.



new lists of locations came in to the <> team at least once a day between when the first ask happened and launch day. but because we were sharing the workload, no one person was dragged under by repetitive cognitive stress.



and my team's location upload documentation, and eventually script, was faster than the manual work, and getting faster all the time. more edge cases were covered, more of the outliers that used to be a manual escape hatch were covered by the machine



eventually, we landed on what is the secret killer app of automation. not a principle, because it's not something that'll happen every time, but maybe a north star.. something to keep in our back pockets, ready to pounce when the opportunity arises.

because the script had been improved upon over time, we were able to add its functionality directly into the application. Everything was in place, and it was essentially **already running** in production. The final step required was just to add a file upload input to the superuser dashboard. instead of emailing the dev team a new spreadsheet, users could do it themselves and see the results instantaneously.

By operating as we would have anyway, completing tasks, building out our application, we got a feature for free! it just... fell out of this process.

surprise!

if that's not a compelling side effect of this approach then i don't know what to tell you

I feel like I could stop right there

but i can't...

I haven't been

completely honest 🥺

because the story here is not about how cool I am for having followed the path of zen automation

the story is that I didn't do... any of that. my practical example, my rhetorical device, is a fiction.

Oh the task was real, but my response to it is a rose-colored reimagining in 4k hindsight of how I *wish* I had been, and am still working toward being.
but why didn't I act that way?

The Checklist Manifesto

Manual Work is a Bug

Atomic Habits

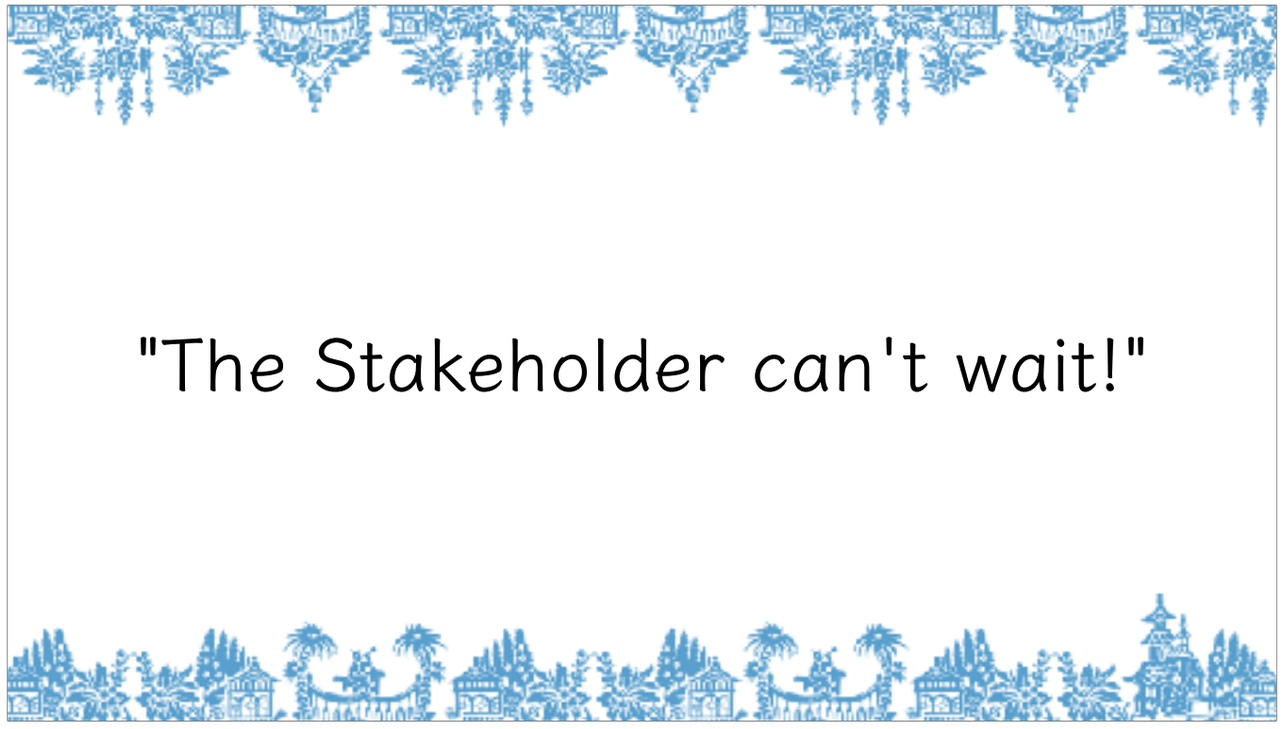
Do-Nothing Scripting

I had read all the same articles and books that I have now...

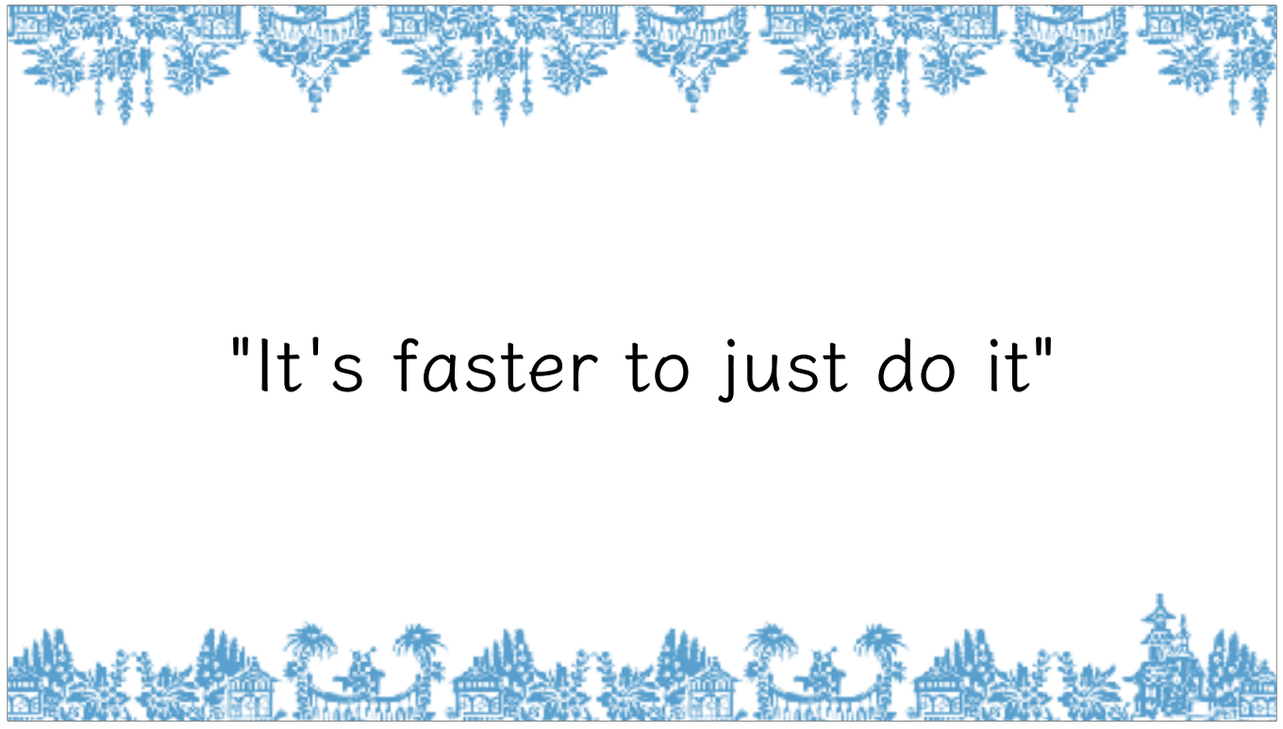
...I always had a reason not to automate it, and I expect we're all going to be familiar with most of them

"It's urgent!"

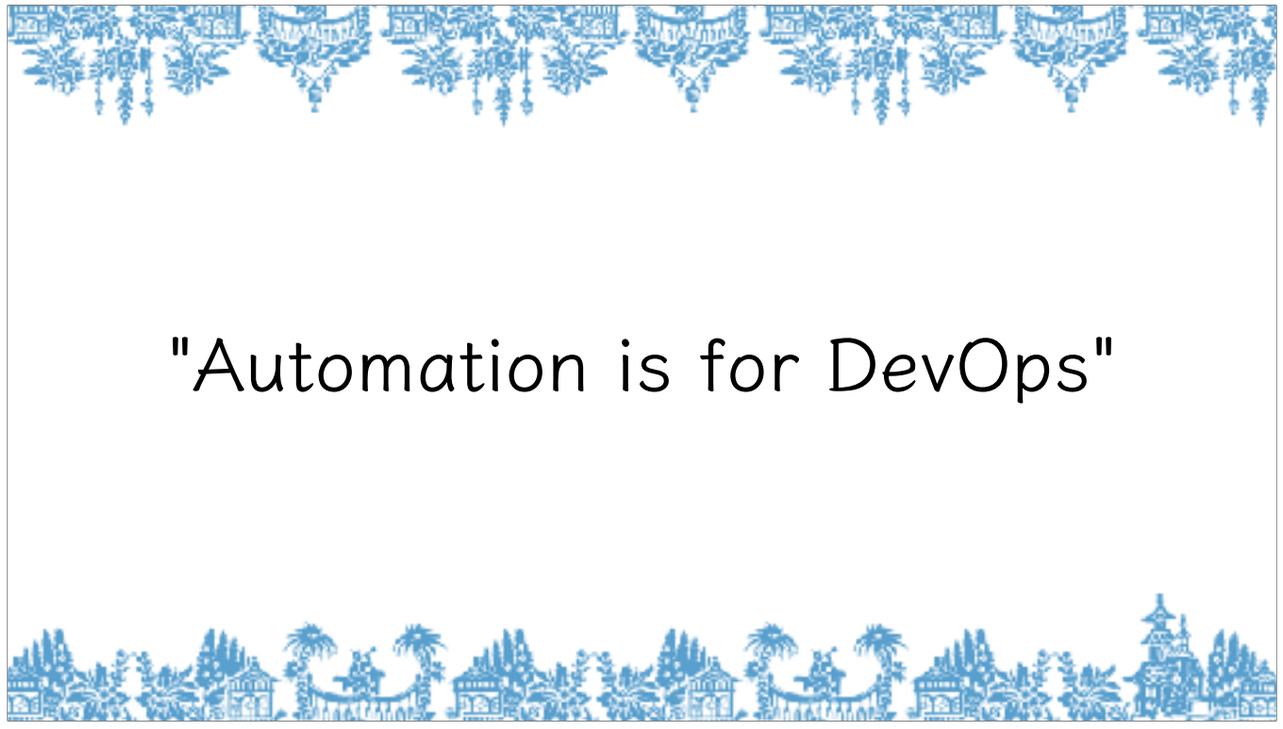
its urgent! aka



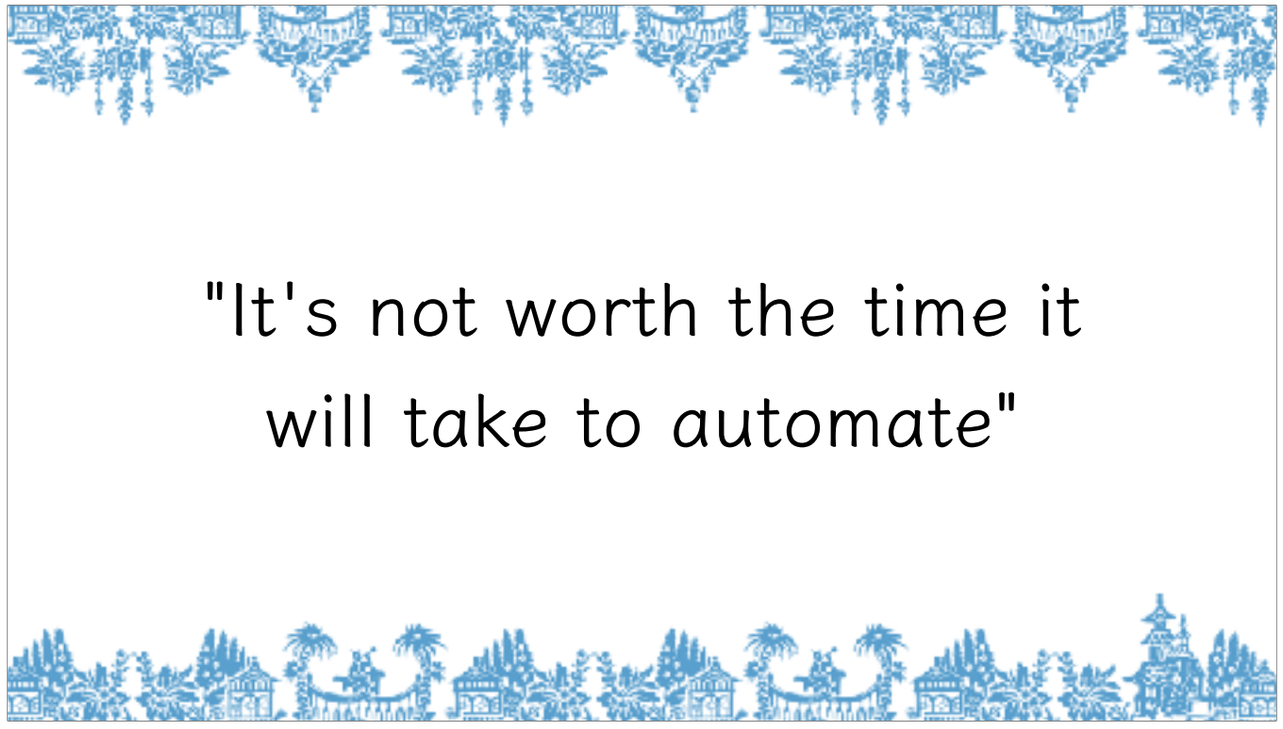
the stakeholder can't wait!



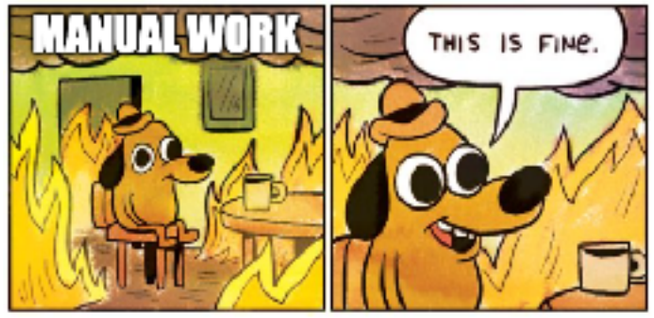
its faster to just do it



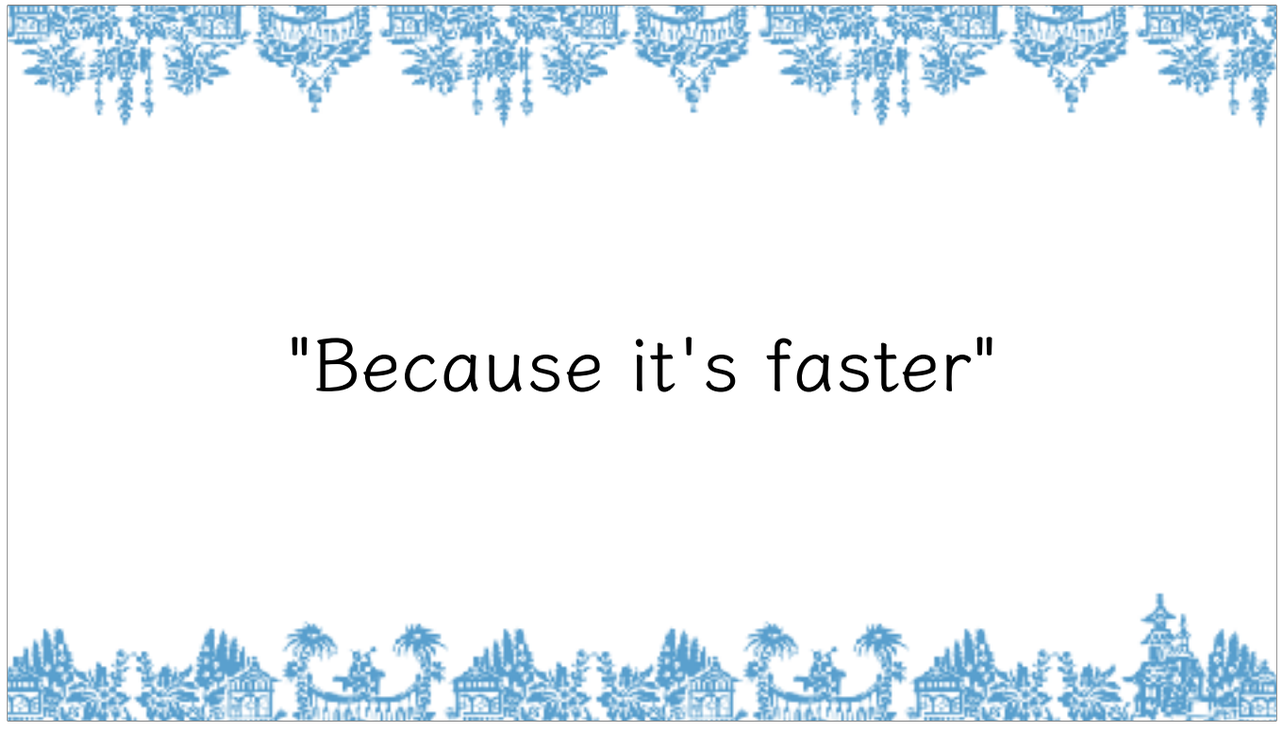
automation is for devops



its not worth the time it will take to automate



We've been this dog before, heard those reasons, thought those thoughts, or had them said to us.. and I listened, I didn't push back against the impatient stakeholder in my head.



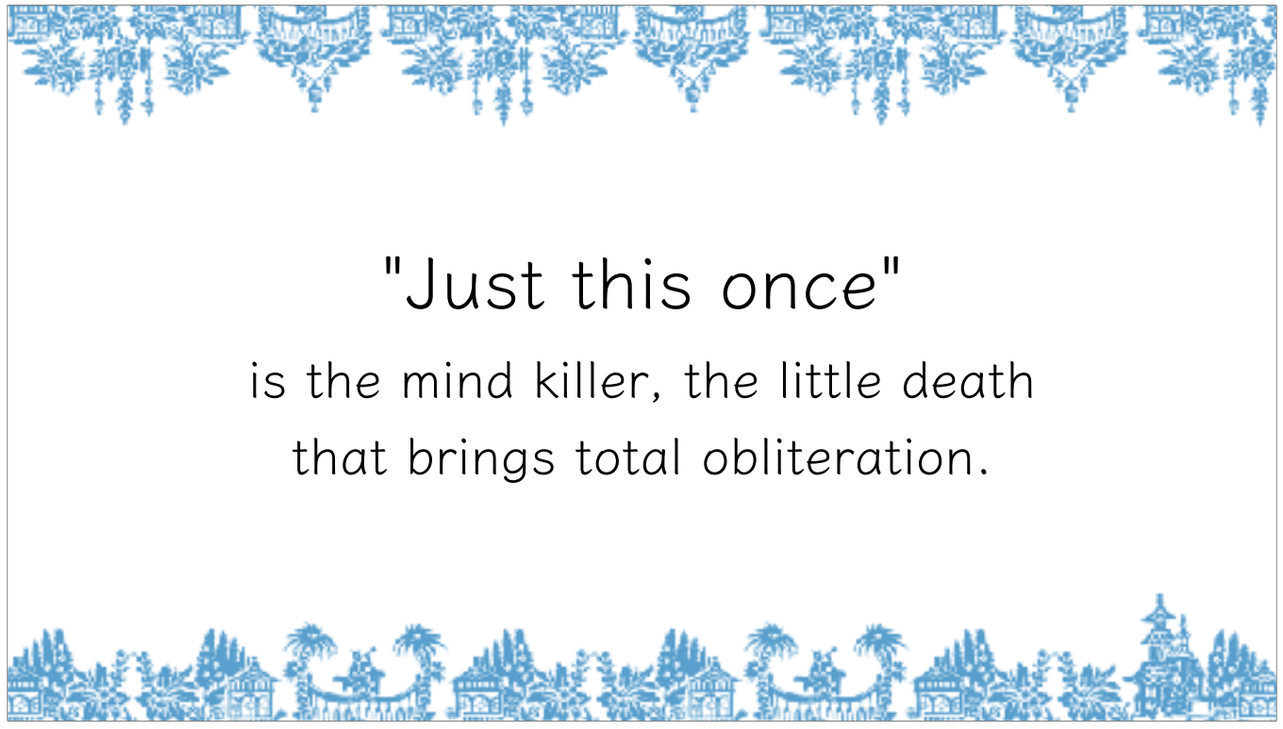
Doing it manually "because it's faster"... that might be true if you were going to charge on and write it to be automated from the start. But the approach of written notes that gradually evolve describes another way.

20 minutes to do it manually vs. a day and a half to write the automation is a substantial difference. 20 minutes to do it manually vs. 25 minutes to do it manually but record your terminal commands is not.

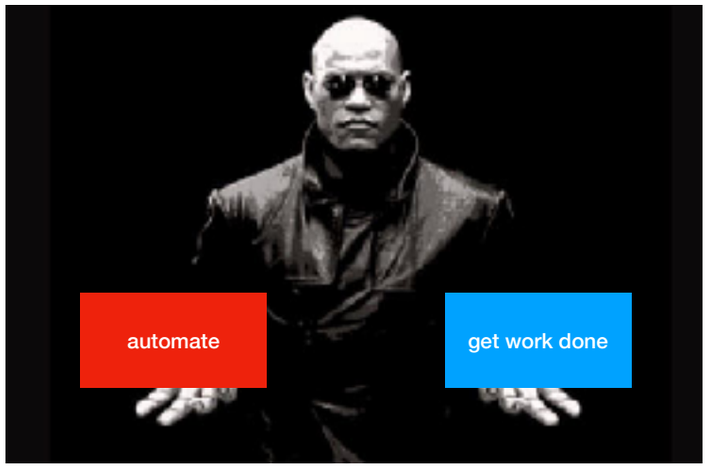
so what is "because its faster", really?

Technical Debt

it's accruing debt.



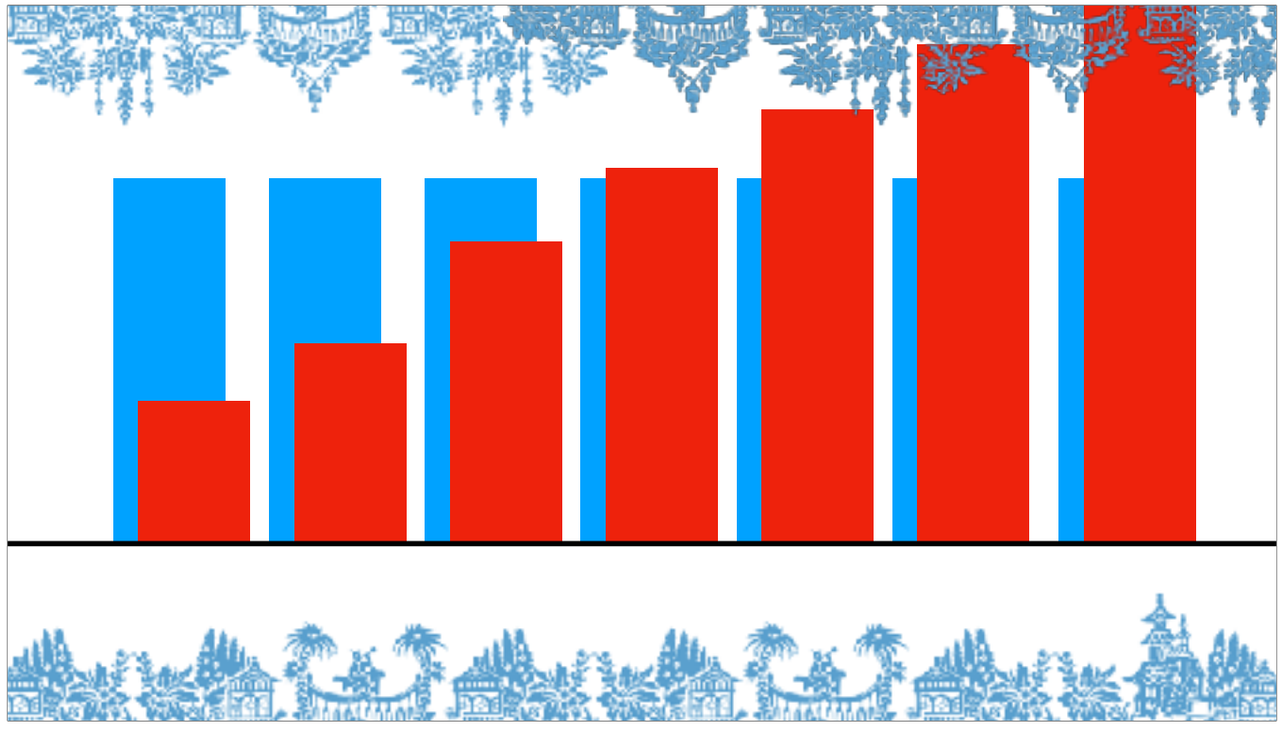
Resist "just this once" thinking by reminding ourselves of the cost of paying full price with manual work.
We're leveraging our future, and our team's future, for **exceptionally** small benefit.



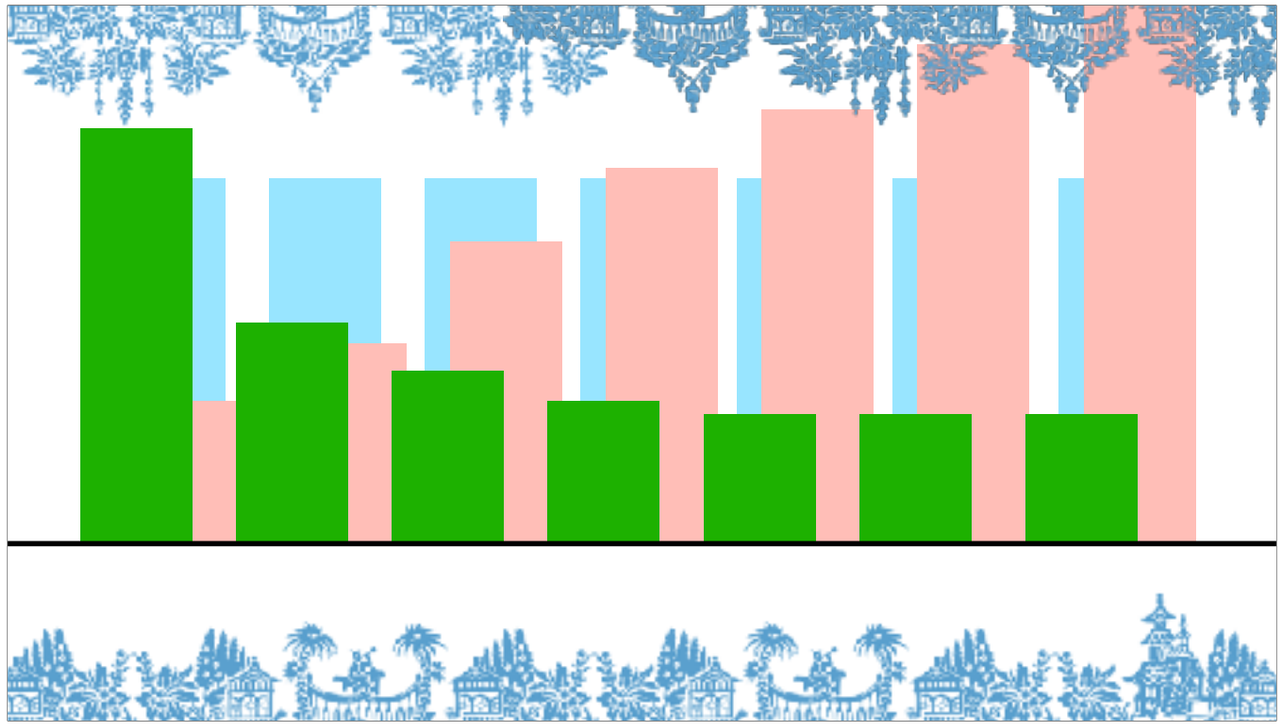
I bought into the false choice between
* 'halting work to write an unrequested feature' or
* getting things done

I must have done that task a dozen times before it was final.

It took roughly the same amount of time every time, a not-insignificant portion of my day, every morning.



the frustration with that task did not stay so constant



when of course I could have had this

With each manual execution, I buddied up to my excuses and lost all the compounding interest this practice can provide

Zen Automation Principles

I know that I've asked you all to think about taking a big shift in the way that you go about your work. A new mindset, built on habits that ask you to ignore present anxiety for future benefit. That's tough.

except you don't need to go back to work and instantly be a Zen Automation master, capturing every thought, refining to the last detail, and influencing your entire company to do the same, epic do-nothing-scripting in each pull request.

because just as automation doesn't have to be all or nothing, neither does this practice.

1. Always be Capturing
2. Refine to be Fine
3. Share Early & Often

In my example, the real one, not the fairytale... Just a fraction of Zen Automation would have saved me so much frustration.

So don't look at these steps and hold yourself up against the ideal example. Not every process is automatable. But Zen and the Art of Incremental automation isn't about the end result. It is, again, a practice.

The journey of 1000 miles begins with a single step. See! There's nothing there about where we're going to end up! The destination might be right back where we started, but after 1000 miles of adventure. There and back again.

千里之行，始於足下

A journey of 1000 miles
begins with a single step

So even if you never create a self service document upload feature... like I never did... I hope you will consider yourselves practitioners of Zen Automation and take those those first steps of your own 1000 mile journey.

Manual Work is a Bug - Thomas A Limoncelli

The Case for Incremental Automation - Jessica Abelson

Do-Nothing Scripting: the key to gradual automation - Dan Slimmon

Runbook: A Ruby DSL for Gradual System Automation - Patrick Blessi

Gradual Automation in Ruby - Tomasz Wrobel

thank you

So much I didn't get to talk about! If you want to geek out about personal automation and making your computer work FOR you find me online! or around the conf!

These links and more, Github/doodlingdev/incremental-automation



doodlingdev

@ruby.social