



Lightning Talks w/ Women of Scribd Engineering: Migrations, Owning Quality, Instrumenting, and Finding the Right Culture

Network: WeWork Password: -

@WomenWhoCodeTO

Want to support our community? Spread the word, provide feedback about our events to improve them, join Slack, talk to your company about hosting or sponsoring, and invite women out.

@WomenWhoCodeTO

WWC Toronto Team



Steph



Betty



Stella



Rebecca



Bhavana



Aileen



Nahrin

@WomenWhoCodeTO

Our Sponsors



intelliware.ca
software development



SCRIBD



@WomenWhoCodeTO

HAVE YOU JOINED OUR SLACK GROUP?



Get updates first! Special giveaways on Slack

bit.ly/wwctoslack

@WomenWhoCodeTO



Words from our Host

@WomenWhoCodeTO



Green Field Projects, Or How I Learned to Love Migrations

@WomenWhoCodeTO



Paige Stone

Software Engineer, Notifications

@WomenWhoCodeTO

Overview Contents



Section One: CRM to CRM, Migration



Section Two: Flux, the Green field



Section Three: The big win!

Change is Inevitable

Overview

Continuous Improvement, Production
features, growth, economics

How we levered a Migration to green field
a new email/notification system

Paging Doctor Howard

Legacy Systems

- Overnight Updates
- Deep nested Triggers
- High message lag
- High dependency on Engineering
- Low Stability
- Low Velocity

On a legacy system, we tend to be surgical in our fixes - the original system was architected over 5 years ago, and we've been patching it ever since.

Enter: the Marketing Platform

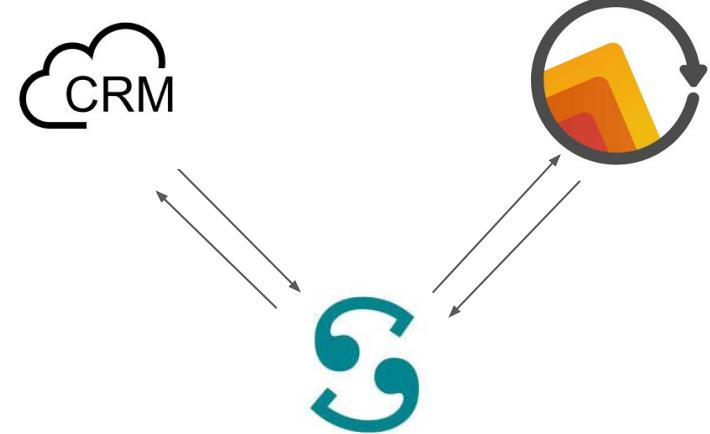
Contracts came up, we had a new opportunity, but do we simply re-create the old in the new platform?

- SQL
- REST, SOAP, FTP
- List Segmentation
- Heavy use of timed queries
- Limited data per users
- Elasticsearch
- REST
- Personal users flows
- Real Time Event driven
- Unlimited data per user
- Full featured Webhooks and Data Feeds



Help from an Unexpected Place

Why we need to have unsubscribe sync ie GDPR, CANSPAM, CSPAM, etc laws



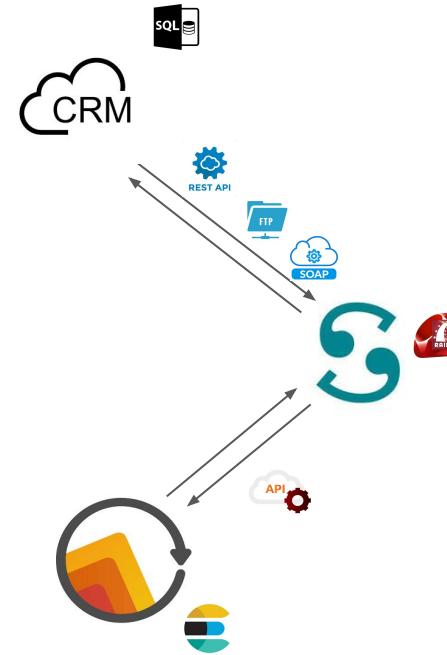
The Source of Truth

Moving the source of truth

CRM -> Scribd

Scribd -> Iterable

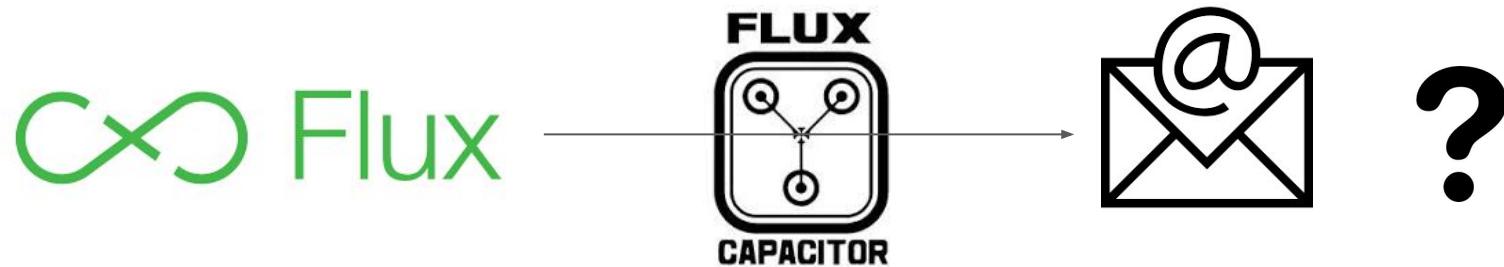
The platforms are based on
different paradigms



The Flux-Capacitor

Inspired by Flux Architecture

Inspired by Flux Architecture (Action -> Dispatch -> Store -> View)



Actions

Helper methods that facilitate passing data to the Dispatcher

Triggered Sends

- User forgot their password
- Send a referral
- Admin Emails
- Dunning

Triggered Events

- New sign up
- Send a gift card
- Cancellation
- Unsubscribe from email/push
- Hard bounce

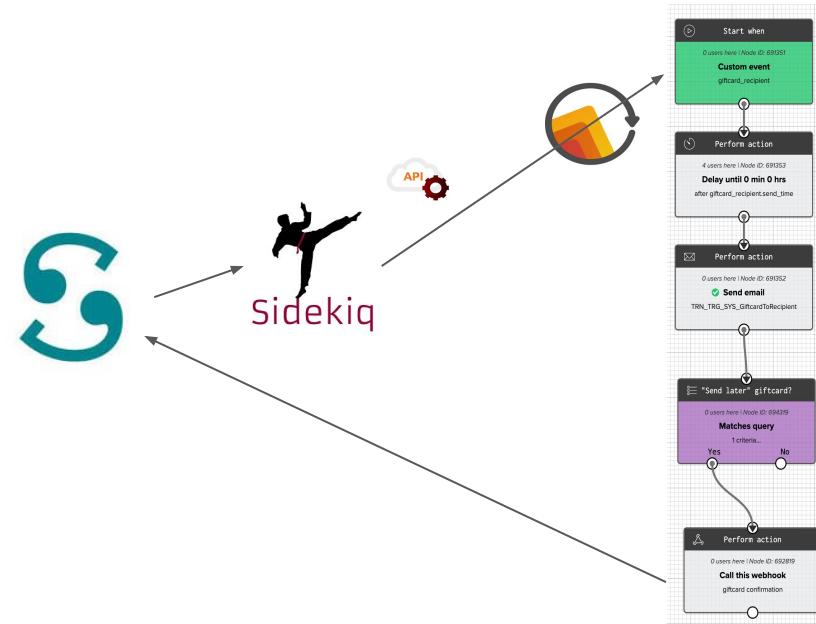
Dispatcher

-Receives actions and broadcasts payloads to registered callbacks

Sidekiq - Worker

Workflows

Call back via
- Webhooks



Store

Containers for application state & logic that have callbacks registered to the dispatcher

Iterable acts as the Store

- Object-Oriented database.
- Levering Iterable's System events and Webhooks as callbacks.
- Our store listens for events via System events
- Events can also be triggered as well

Views

React Components that grab the state from Stores and pass it down via props to the child components

Email as an extension of Frontend

-Treating email templates as components



88 Miles Per Hour!

- Realtime Updates
- Common Sense Triggers
- Time to get messages almost massively improved
- Reduced dependency on Engineering
- Improved Stability
- Improved Velocity

The Game of Marketing

The end user

Instead of doing User Segmentation, our marketers had to relearn how to develop user flows, journeys, paths through the product. We can talk about gameplay, user experience loops, and driving engagement.



Back to the Future

What up Next

Because everything is centralized components, it is now easier to switch components out. It's more generic. We can switch out components (Sidekiq to Kafka?) without reinventing the wheel.

Additional data is trivial, Additional Events and Triggers are trivial

Less maintenance, means more Features!

Push Notifications, Localization, Improved Testing, Brand-Refresh, Data Analysis



**Don't go chasing waterfalls: owning quality
as a whole team effort**

@WomenWhoCodeTO



Nikki Hernandez

Web QA Engineer

@WomenWhoCodeTO

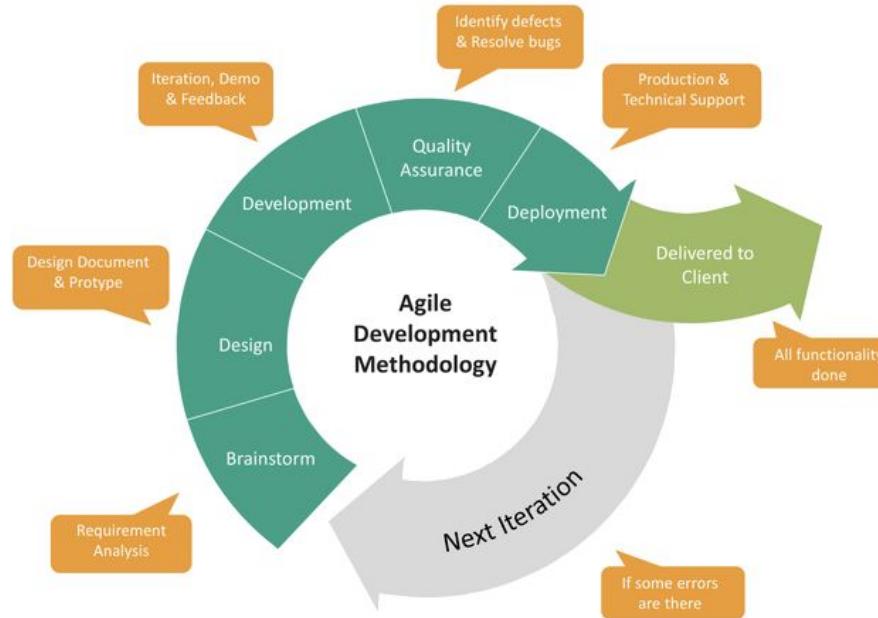
\$whoami

- Tester / professional guinea pig
- Agile tester, Scrummaster certified
- Self taught techie
- Mother of CAT, Tyler Purrdon

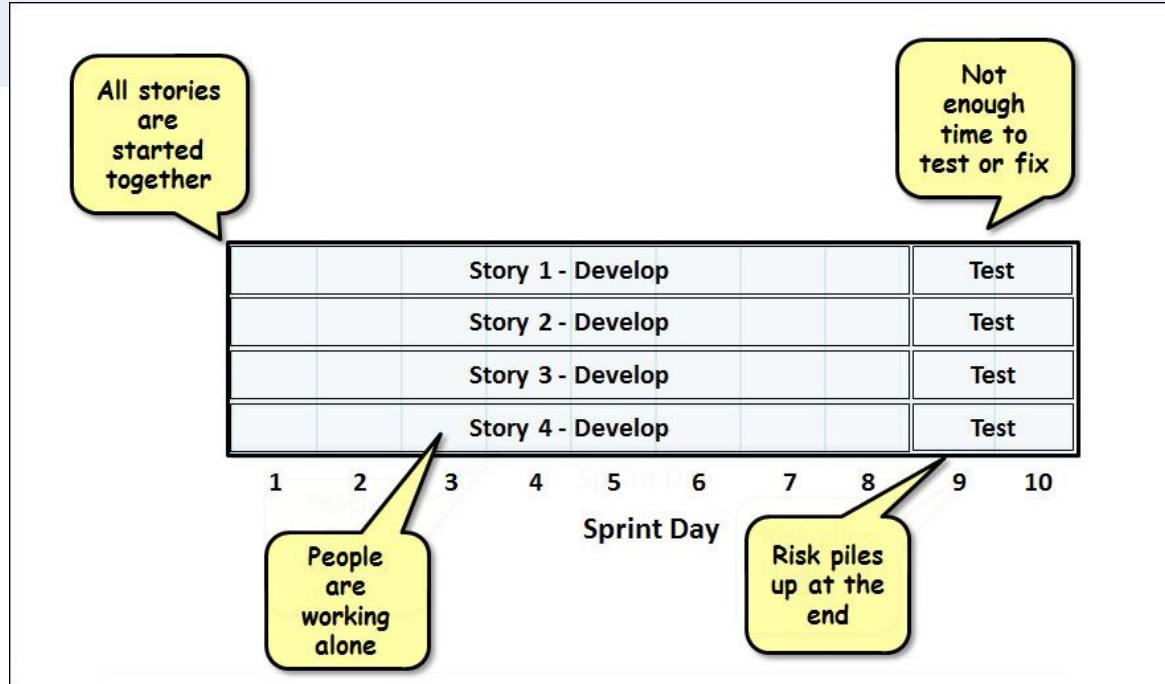


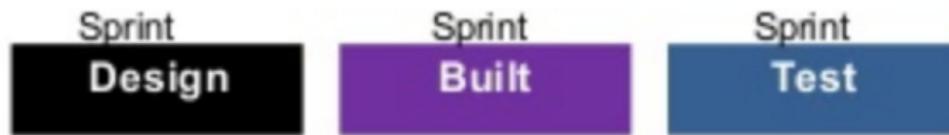
Context

Agile

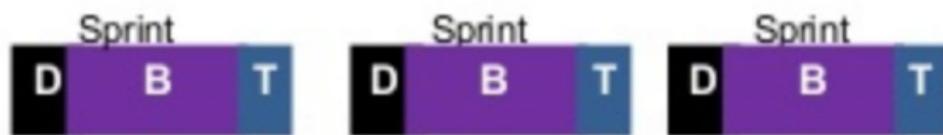


Mini Waterfalls look like this





No!



*Little better, but No!
High risk of non-delivery*



Yes!

Why is this a problem?

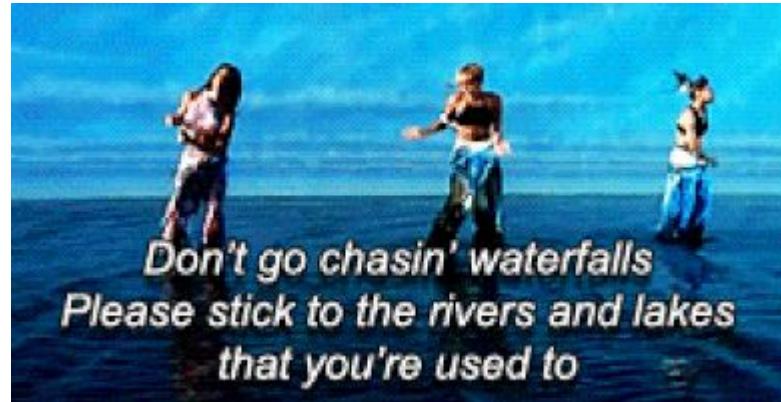
- Lots of risk at the end of the cycle
- Rushed testing (or building) inherently degrades quality
- “Feast or Famine” cycles of work
- Being isolate/silo’d leads to tunnel vision or feature blindness





Preventing mini-waterfalls: Own quality!

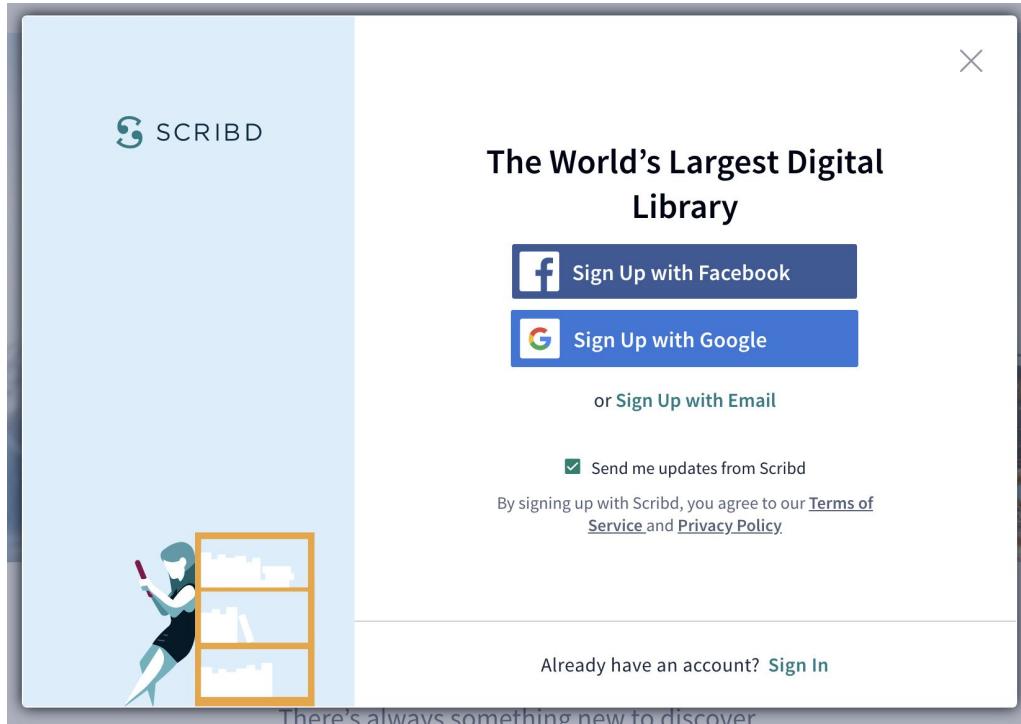
- Build in quality from day #1
 - Break down work meaningfully - MVP, “Shippable Pieces”
 - Limit tasks to 1-2 days
 - Write small user stories
 - Storypoint for non-dev work, external dependencies, third parties
 - Write and agree on Acceptance Criteria
 - Agile teams work *together*
 - Pair test, Pair dev, Pair design



Quality means more than just “testing”



Breaking down tasks (an example)



Back X

Sign Up with Email

Name

Email

Password
(at least 6 characters) Show

Send me updates from Scribd

Sign Up

Already have an account? [Sign In](#)

Acceptance Criteria

- State the intent clearly, concisely - not the full flow
 - Good: “Copy is translated in all supported languages”
 - Bad: “User can select Spanish, view new copy on Saved page”
- Include happy path, and any other important considerations (accessibility, locale, geolocation, exclusions, application security, etc)
- Not a replacement for full regression tests, unit tests, or other tests but may include some aspect of it
 - Eg: “Performance: Page load time remains the same”
- A ticket isn’t done until the AC is

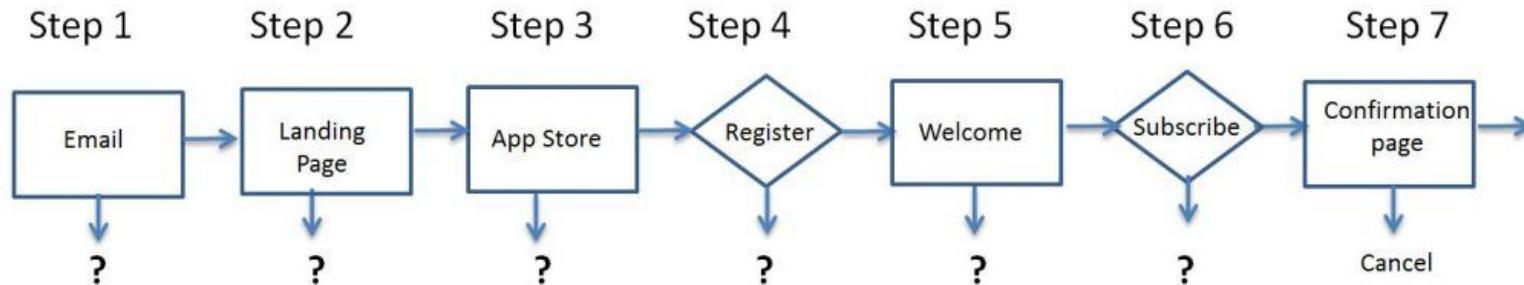


Writing quality code

- Keep changes as isolated as possible
- Anticipate shared code or dependencies
- Keep 'em separated - separate branches
- Quality code is:
 - Clear, concise, stable
 - Minimal in change and scope
 - Is manageable, easy to code review
 - Can be easily reverted
 - Is testable in some way
 - Communicated clearly! (MRs, to team, in tickets, etc)

“Panning for gold”

- Identify the happy/golden path
 - What do we want users to do?
- Identify negative path testing
 - What will users actually do?



Testing 101: Personas

- What?
 - Real people with hopes and dreams (of using our products)
 - Everyone has a unique motivation for doing things the way they do
 - A way to “put yourself in \$X shoes”
- Why?
 - Because we have our own biases. Utilizing personas allows us to break out of that and try to mimic real user behavior.
 - As a tool to support ad-hoc, exploratory, usability, happy path, and edge case testing
 - To see the site with new eyes
 - To be able to test the full life cycle of a user as they cycle through our products
- How?
 - Build a real person. Give them a name, a job title, what's up in their life right now? What are their motivations for using Scribd? What brings them value?
 - <https://personagenerator.com/>

Testing 101: Some tips/tricks

- Testing user stories:

- Work with QA to prepare some generic test scenarios
- Setup staging environments, sandboxes, etc
- Test against your acceptance criteria
- Utilize personas to guide you
- Exploratory / ad-hoc tests
- Involve the whole team

Some generic testing considerations:

- Is it accessible
- Responsive
- Localized
- Check various platforms (desktop vs mobile, mac vs windows, ios vs android)
- Check various browsers: FF, Chrome, IE, Safari
- Check various user states (if applicable)
- Check various payment flows or methods
- Check the happy path
- Check for regressions where code was touched
- Check metrics / analytic tracking events

Conclusion: Everyone can own quality

- Owning quality means:
 - Planning for it from day 1
 - Work breakdown, sandboxes, how to test, etc
 - Writing quality specs, code, tests
 - Writing acceptance criteria
 - Communication / collaboration
 - Working small pieces at a time
 - Advocating for the customer
 - Working together!
- Testing for quality means:
 - Happy paths
 - Unit tests
 - Functional QA
 - Design / spec review
 - Pair testing
 - Exploratory / Ad-hoc usability testing
 - Personas
 - Acceptance Criteria
 - Identifying high risk areas and testing against them

Conclusion: Quality prevents mini-waterfalls

By baking in quality from the beginning
you'll:

- Mitigate risk or deadline slippage
- Build quality products
- Ship stable code often
- Be a lot less stressed
- Avoid mini-waterfalls
- Stay agile, my friends!





Instrumenting your Code for Great Justice

@WomenWhoCodeTO



Katerina Hanson

Software Development Manager,
Payments & Notifications

@WomenWhoCodeTO

About Me

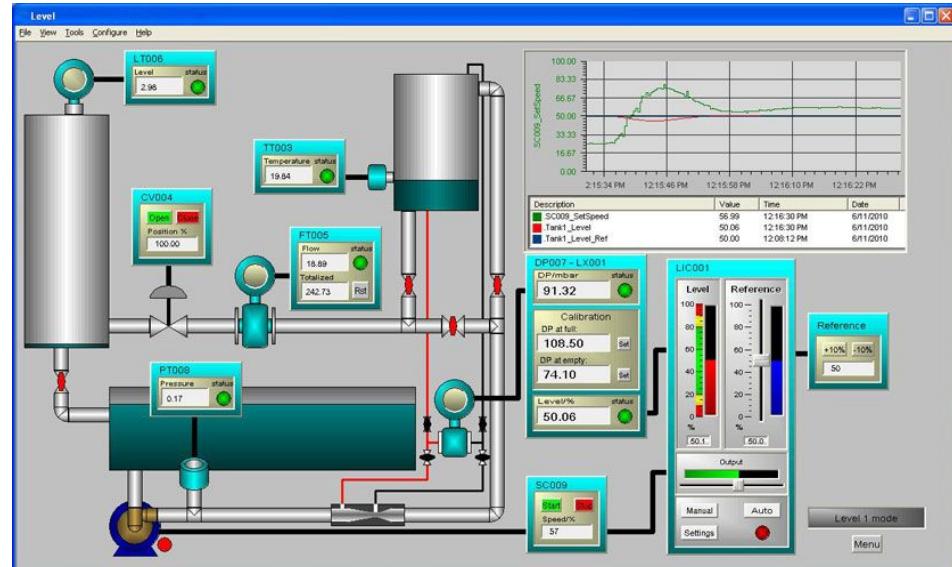
- Self-taught Software Engineer
- Previous career as a chemical engineer
- Love going and getting the \$\$
- Old Millennial - my memes are old-school



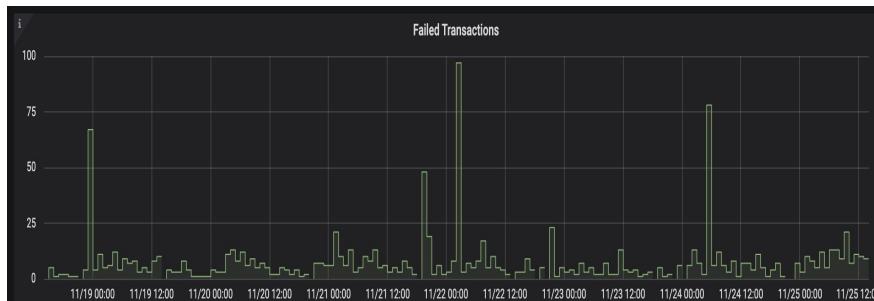
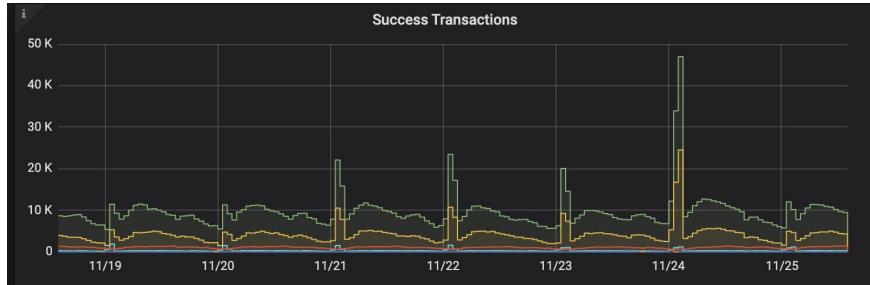
Instrumentation

What are Metrics?

- Measurements
- Quantitative data points
- Track the operation of complex systems



Why Have Metrics?



- Visibility into the Health of your system
- Give you insight into performance
- Easy to read and understand
- Alerts you to new problems
- Helps with diagnosing
- Prevents flying blind, wasting time

Hits to API Endpoints

At minimum success vs failures



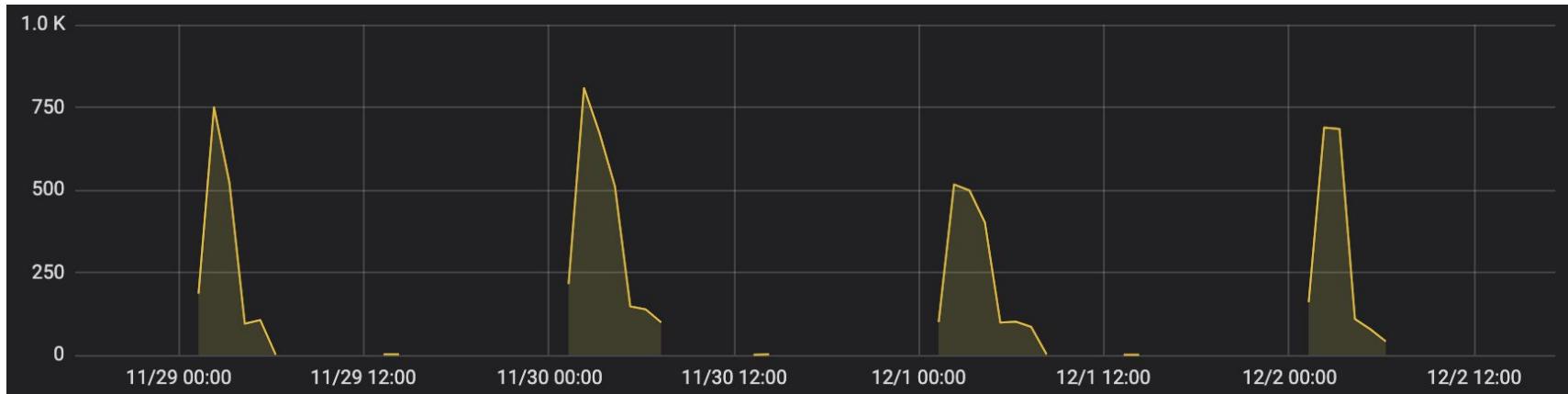
Timings on API Endpoints

No math necessary, just send times



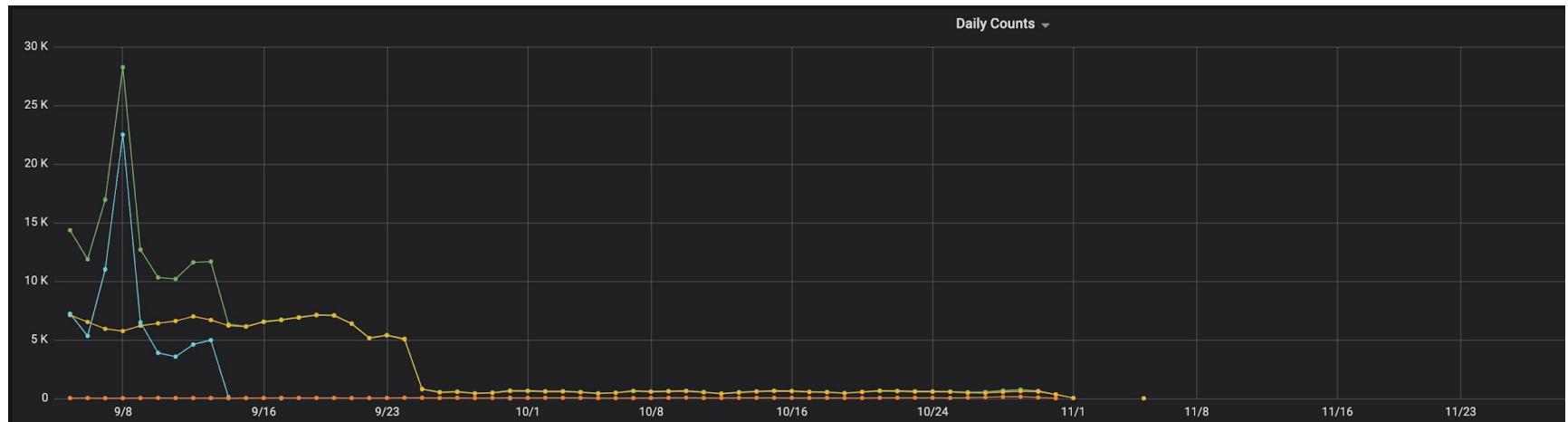
Workers

Queue Sizes, Rates, Success / Fails



Deprecated Code

Wave goodbye to your old features



Other - Be Creative

More Suggestions

- 1) Database reads / writes
- 2) Calls to 3rd Parties (and response types)
- 3) Any important changes of state
- 4) Anything that may run out of space
- 5) Cyclomatic Complexity (if / else)

Context is Important!

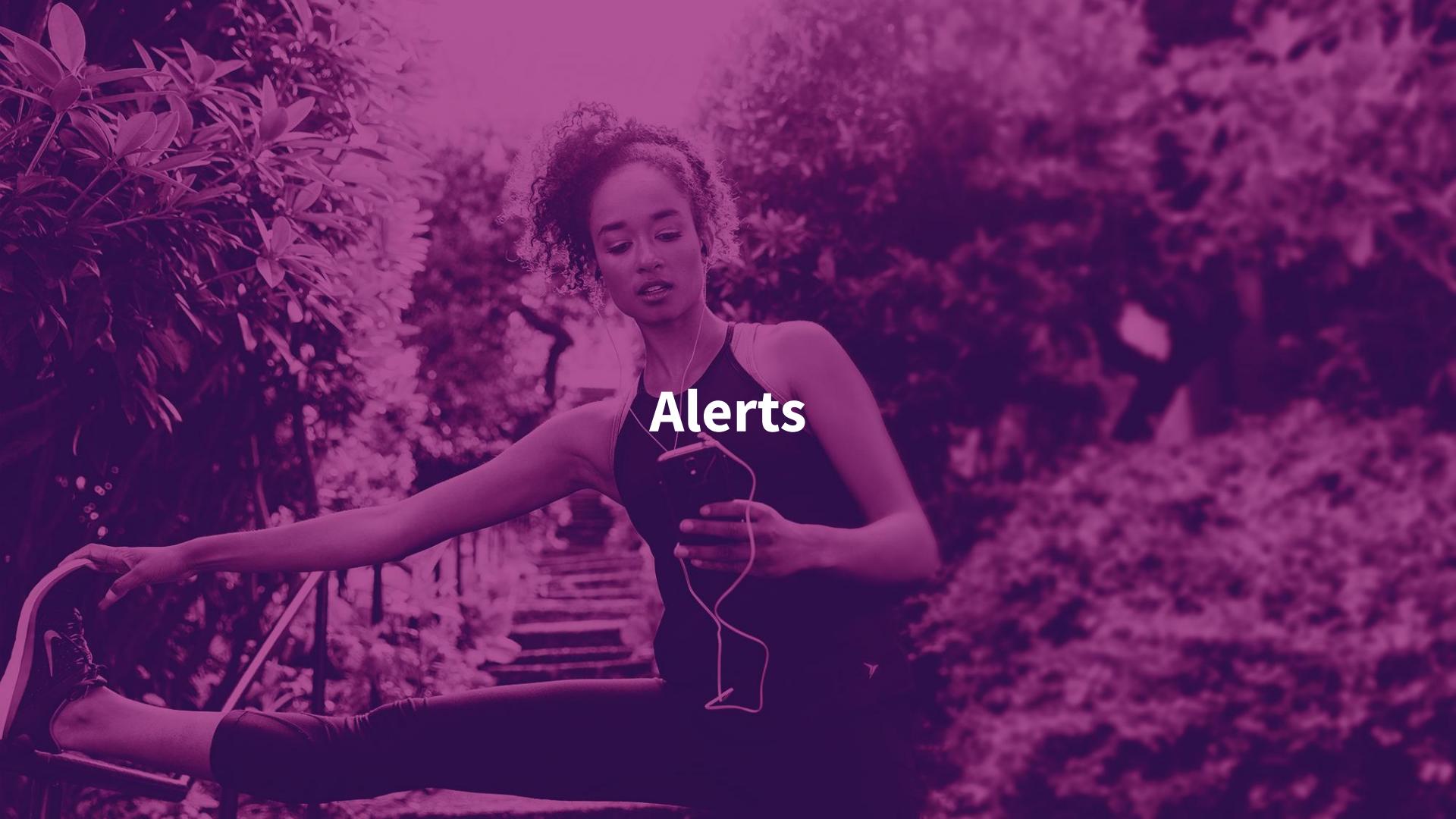


Is this users-over-time?



Or is potentially runaway queue?

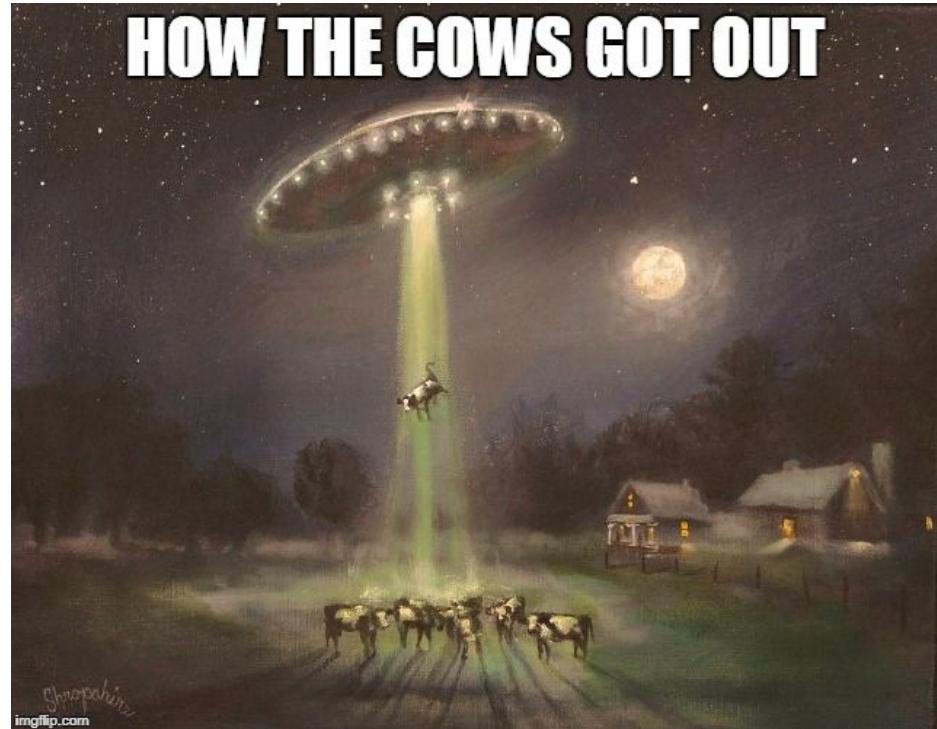


A woman with curly hair, wearing a black tank top and leggings, is sitting on a park bench. She is stretching her right leg forward and holding it with both hands. She is wearing white headphones and a black smartphone around her neck. The background is filled with green bushes.

Alerts

Why Alert?

- Emergency
- Time-sensitive
- Actionable
- Demands Immediate Attention



Problems with Alerting

It's Too Easy to set up Bad Alerts

Nuisance Alerts

If Everything is an Emergency - Nothing Is

- Limited Attention
- We get Desensitized, start ignoring everything
- Lose the valuable alerts in the Noise



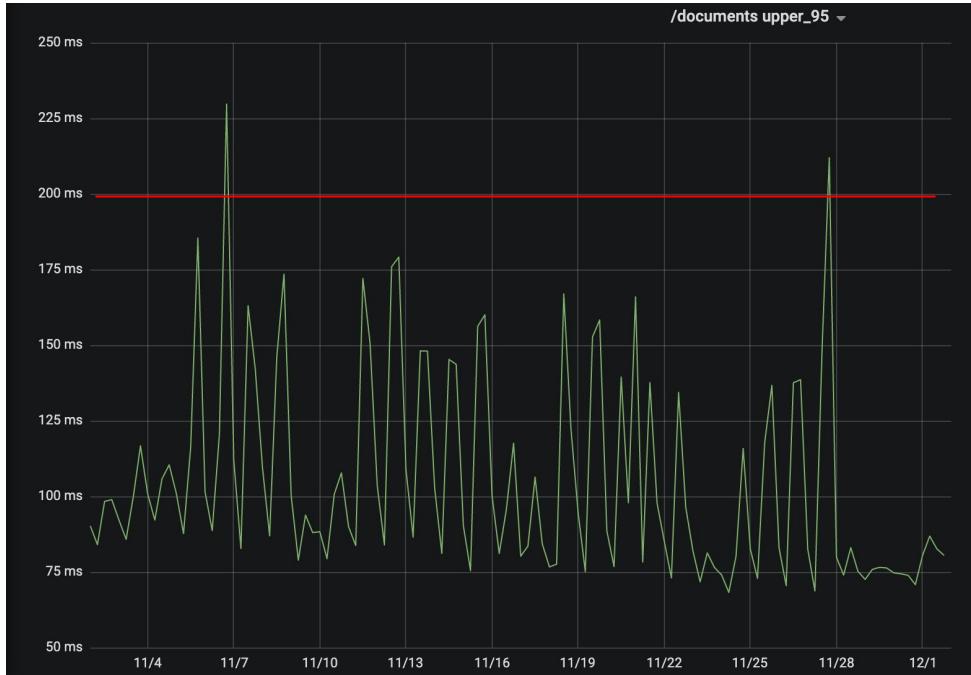
BE PICKY

Alerting - Thresholds

- #1 cause of Nuisance Alerts
- Outliers and drift will happen over time.
- Alert only when the situation is serious and actionable.
- Review periodically and adjust

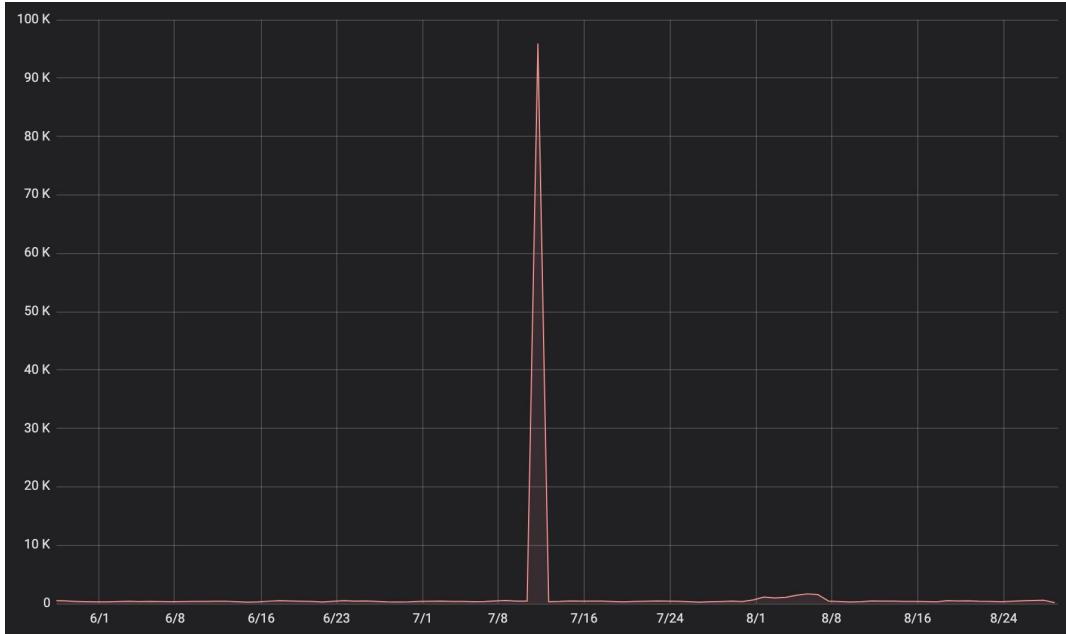


Alerting - SLA Based



- SLA's can be interpreted strictly
- Make sure they include tolerances.
- These should be evaluated for trends, and the work to address scheduled and prioritize over the long term, before it develops into an emergency

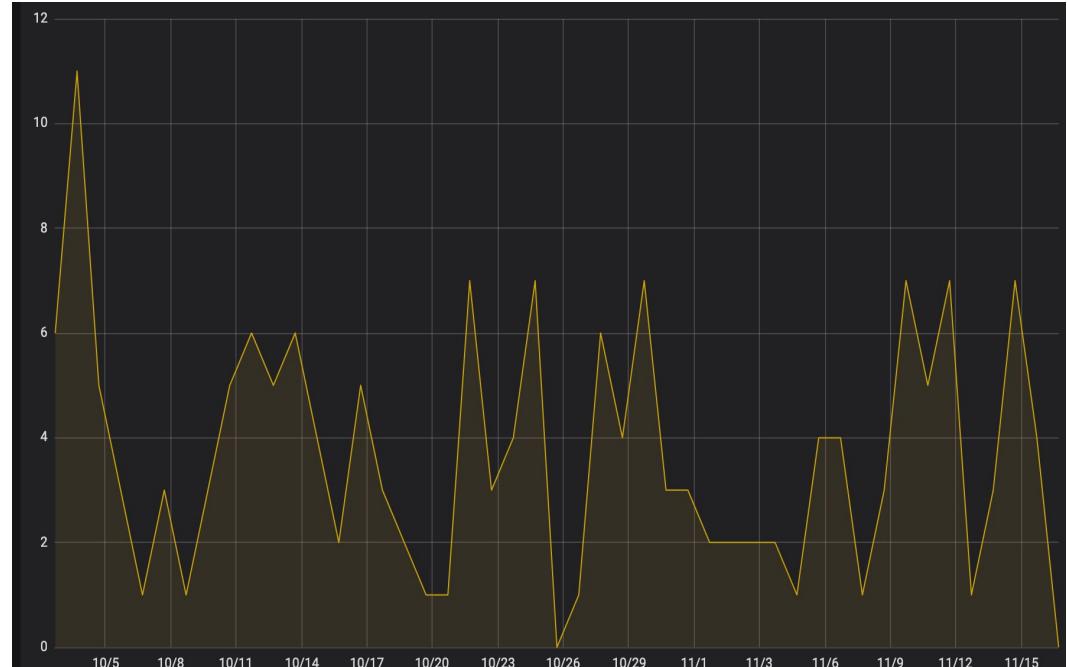
Worst-Case Scenario



- This is what an attack on an endpoint looks like.
- Ideally, we'd have a circuit breaker in the code that would prevent this
- But a high level threshold alert would be an ok backup

In Short: Live with your Data

- Review your charts regularly.
- View over different time scales, see if you need more info, or less.
- Make sure you have logs to support deeper dives.
- Look for things you would have liked to be warned / alerted on, and consider how to define your rules.





**What you seek is seeking you : Find the
company and culture that is right for you**

@WomenWhoCodeTO



Harini Iyer

Lead Software Engineer

LinkedIn: hariniiyer

Accidental Engineer

Point of No Direction

Self Image

Is being a good developer enough?

Turning Point

Should I quit?

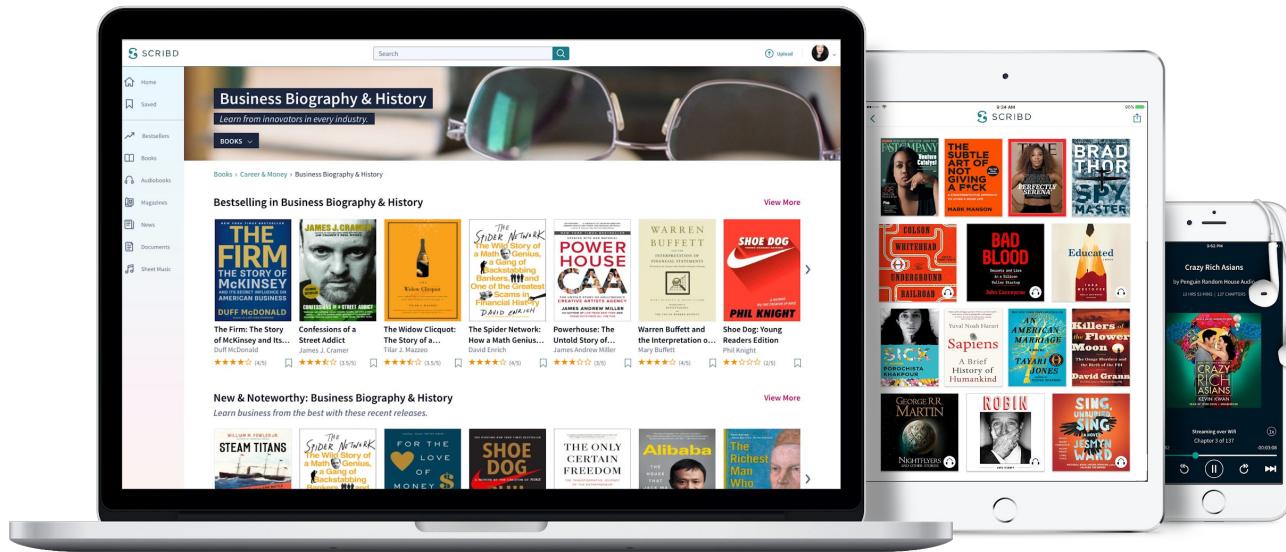
The Magic Words : I don't know (yet)!

Feedback

Culture can be influenced

But I don't understand the product!

Hello Scribd!



Company values



People



Accountability and Ownership



Love of Reading

Engineering values



Refactor and Rebuild



Quality is everyone's responsibility



Velocity

Building by Un-building

The Future

- AWS Migration
- MicroService Architecture
- Redesign systems

Scribd is Hiring!



WOMEN WHO
CODE[®]
TORONTO

Thank you



@WomenWhoCodeTO



See you next time!

@WomenWhoCodeTO