Reliability When Everything is a Platform

Why we need to SRE our customers

Dave Rensin
rensin@google.com

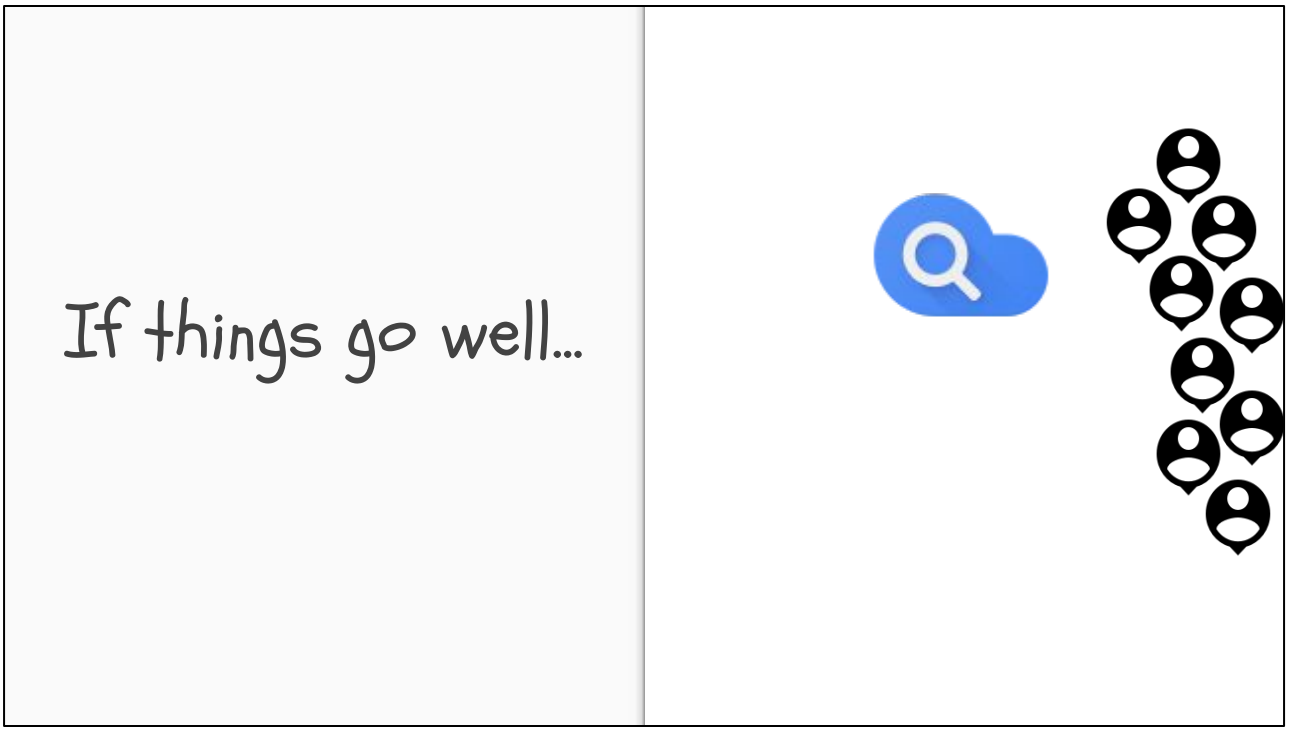
Disclaimer:

Reliability != Availability; but for this talk, let's pretend they are.

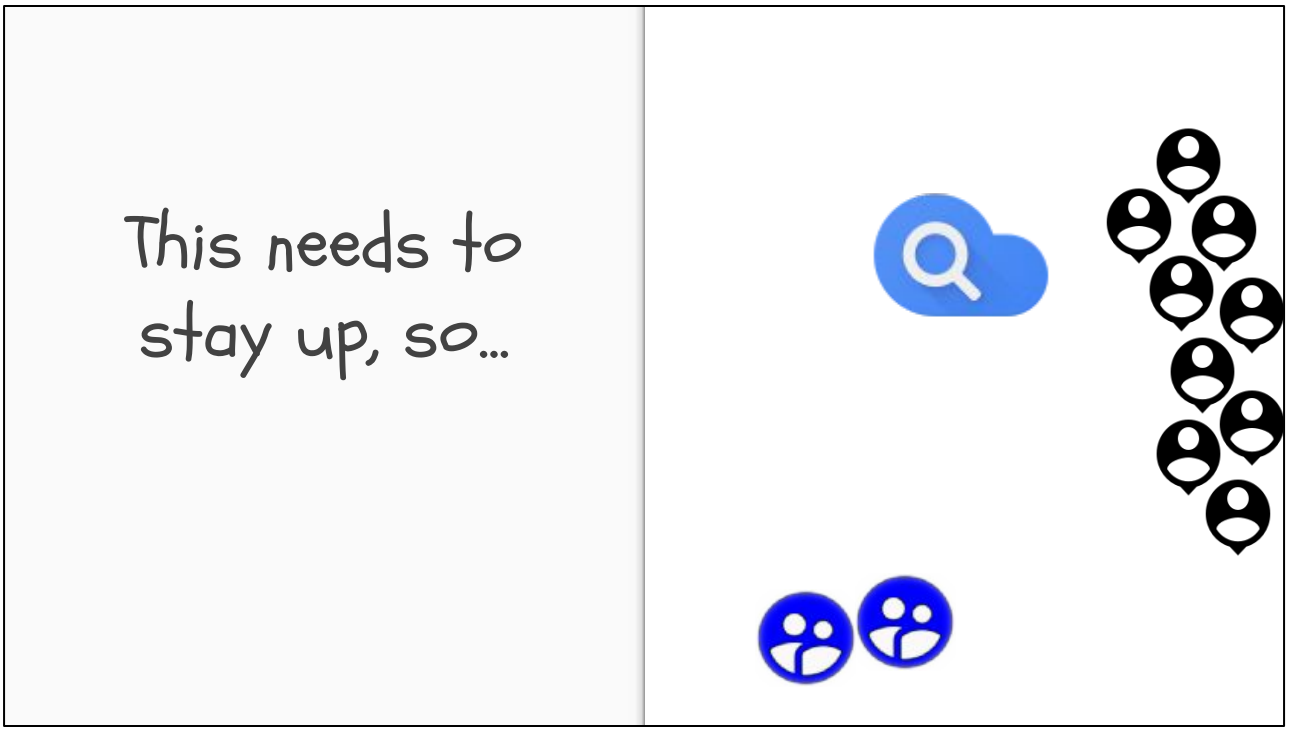
There is a well principled conversation to be had about the strict differences between "reliability" and "availability". I acknowledge they are often not the same thing, but for this talk we can treat them interchangeably. Please don't let the larger points get lost in an argument about MTBF vs MTTR

In the beginning...

In the beginning almost every company starts with just **one** application. Like.. I don't know… *Search*.

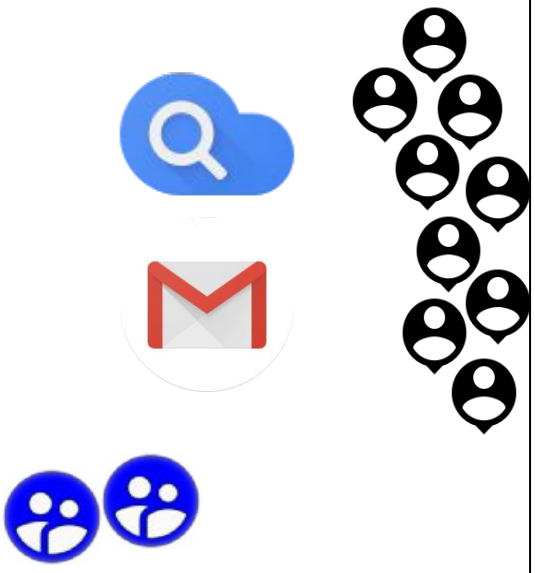


If things go well then this application gets lots of users.

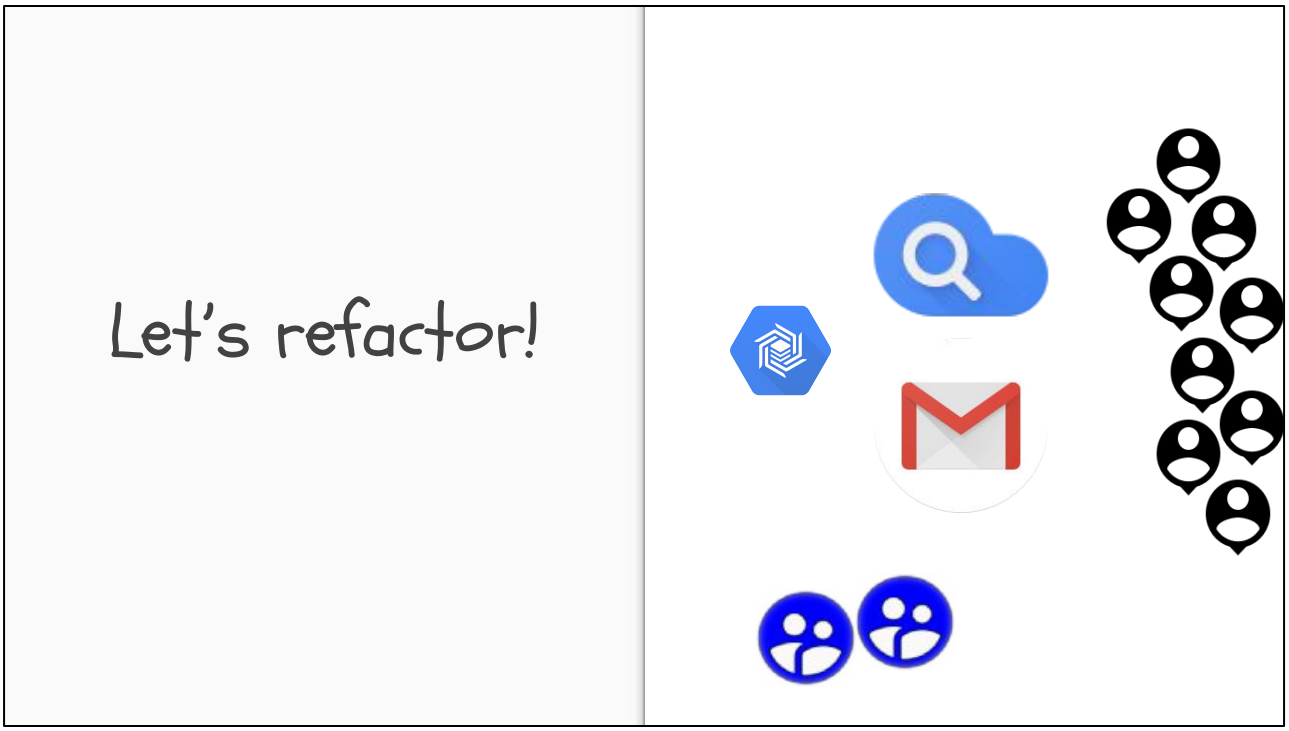


The more users it gets, the more important it is for it to be highly reliable. Eventually, specialists take over this responsibility. We call them *SREs*

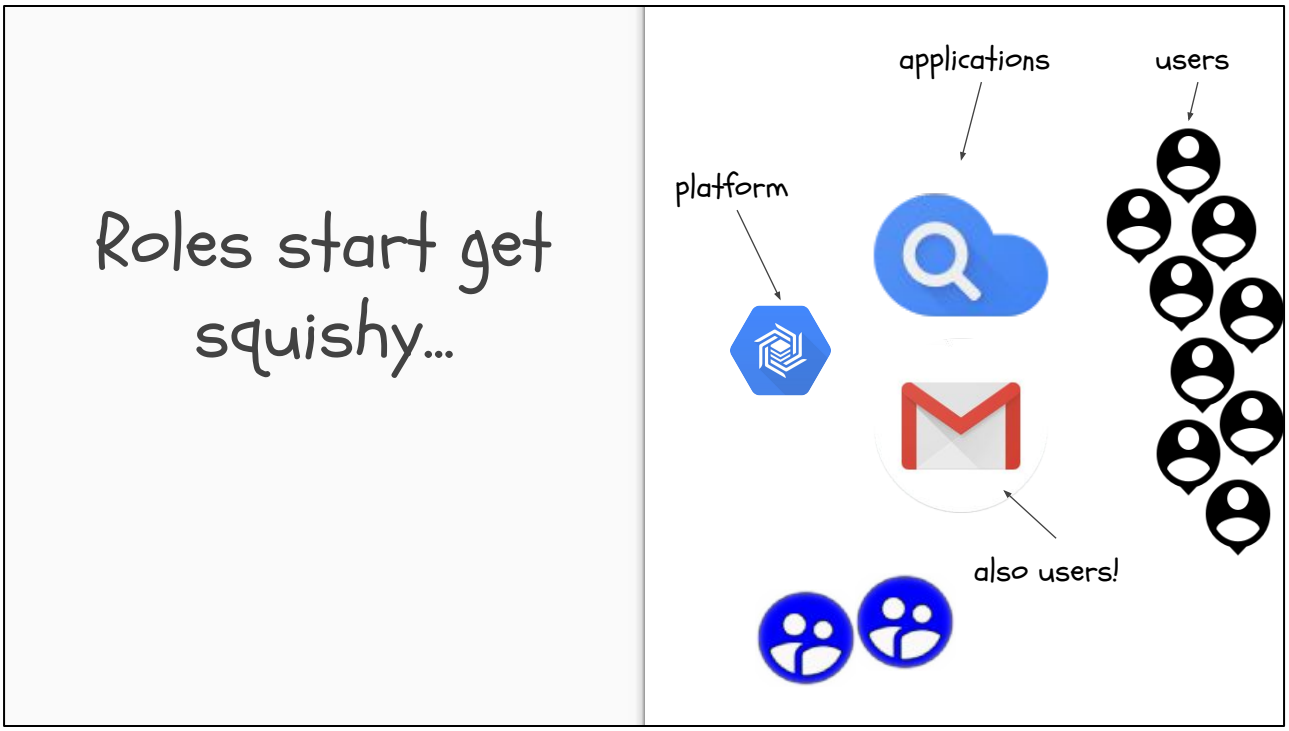
We did so well with one...



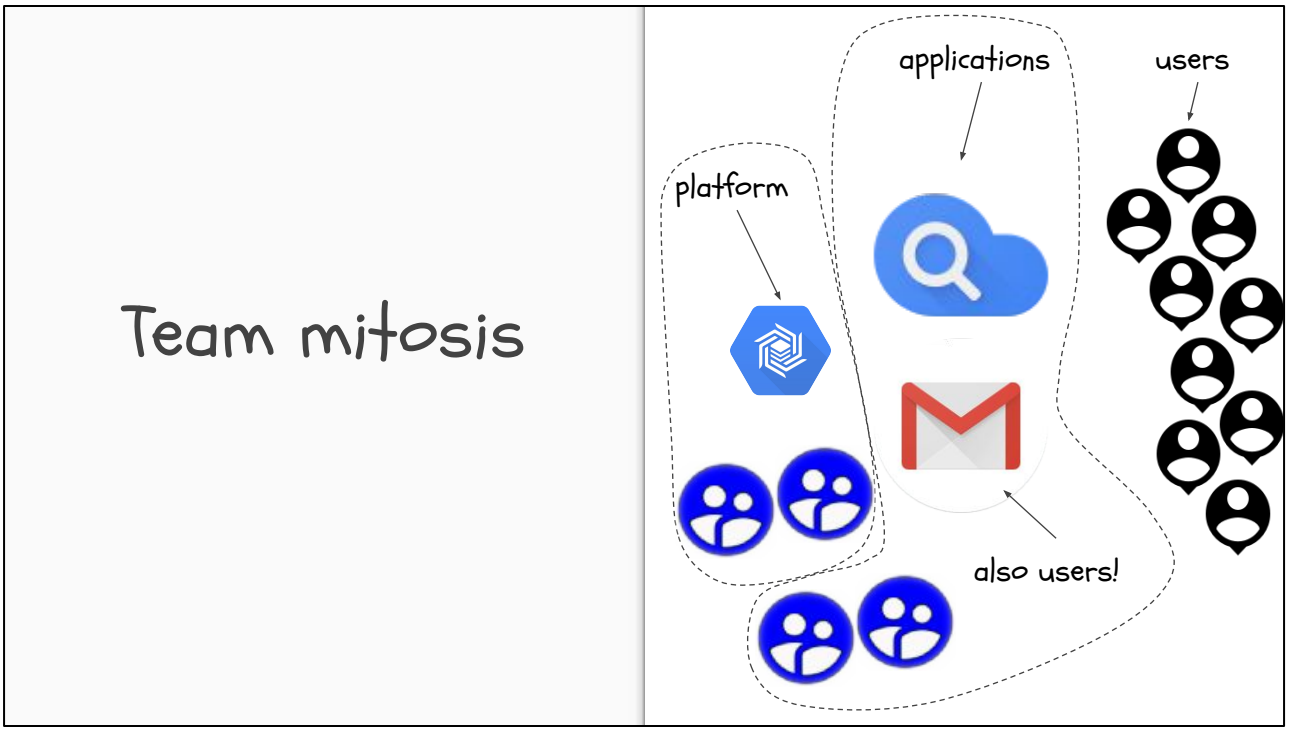
One killer app worked so well.. Let’s add a second!



Because we're engineers we pretty quickly realize that these systems share common requirements -- like storage. Let's refactor!

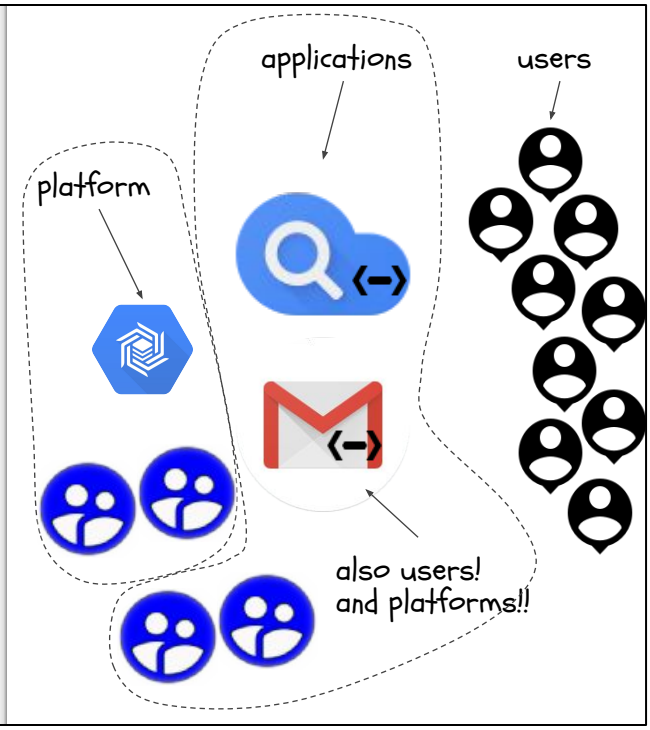


Before long, roles start to get a little squishy.. For example.. These are users and these are applications. But the applications are now *also* users. These other bits? They're now *platforms*



When this happens, it's time for *team mitosis*. We create new control boundaries and split SRE teams -- one group handles the applications and another group handles the platform.. This has been pretty much the state of the world for the last 10(ish) years...

APIs as far as the eye can see...



Network effects mean that our applications get more useful (and valuable!) when they interconnect with other popular applications.. So we add APIs.. But… Now our applications are also platforms! **Pro Tip:** a *platform* is any system whose interface is an API. An *application* is any system whose main interface is a UI. *Machines* talk to platforms. *Humans* talk to applications.

Principle #1

The Most Important Feature of Any System is its Reliability

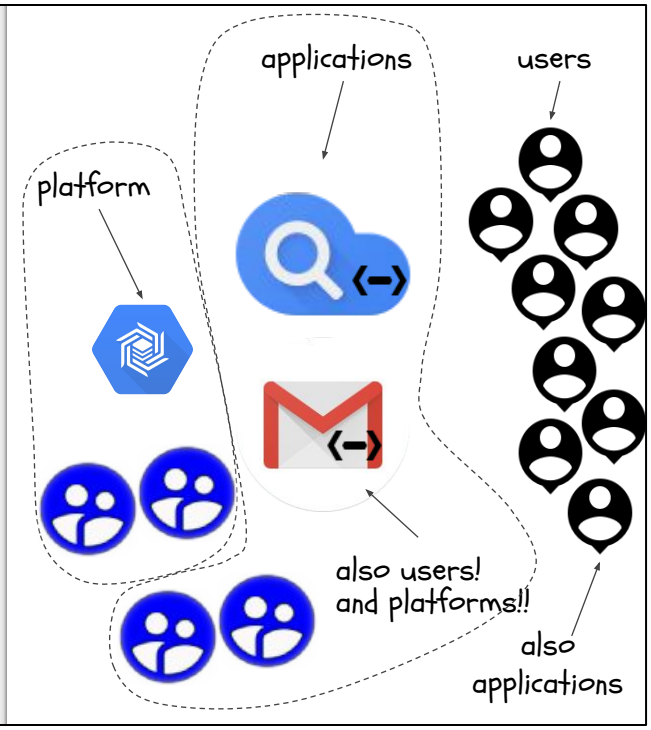
In this world we need some principles to reason about things. Everybody (I hope!) can agree on #1… The most important feature of any system is its reliability.

Principle #2

Our Monitoring Doesn’t Decide Our Reliability -- Our Users Do

There are lots of ways to measure reliability, but the **most** important one is how our users experience it in our systems. If **they** don’t perceive it as reliable, then it’s not -- no matter what the logs say...

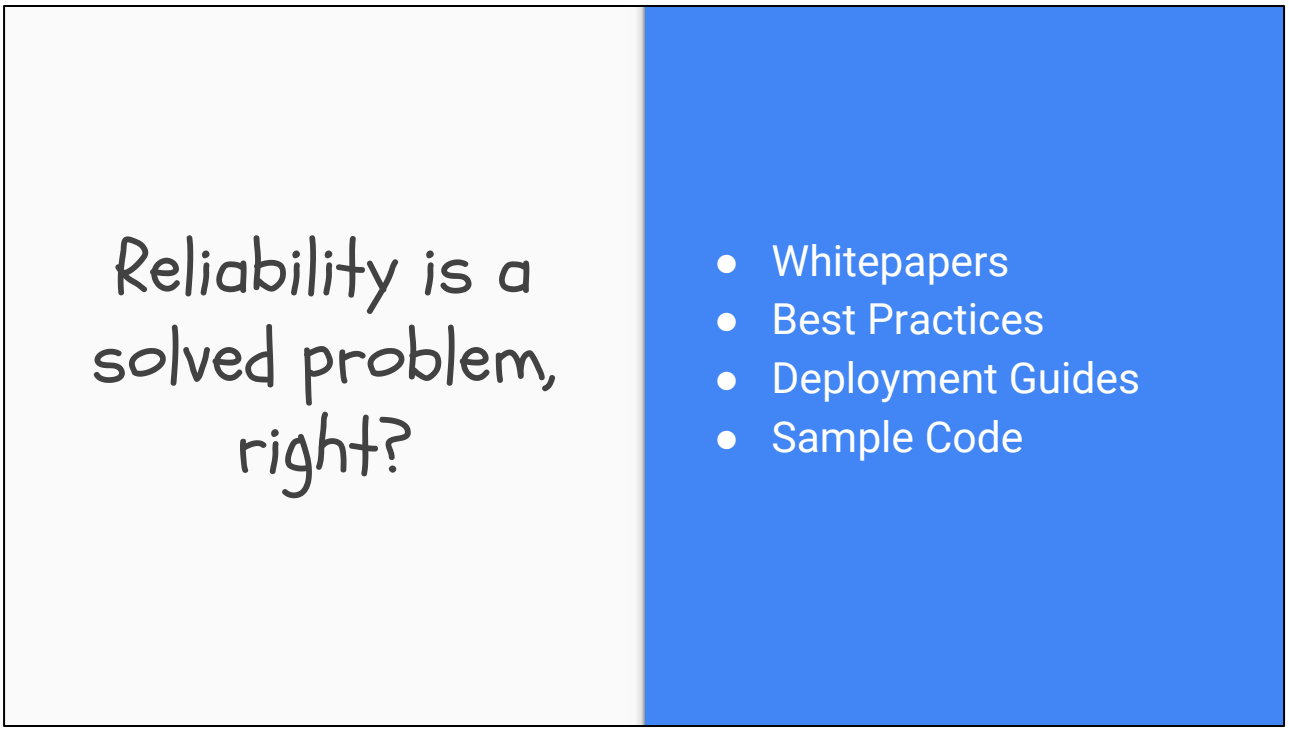
and here's where things get tricky...



Now… Some of our users are people, but some of them are also applications developed and maintained by other companies. And **they** have users, too.. Uh oh… Now the reliability perceived by our customers depends not just on the software **we** write, but on the software **they** write! All turtles, all the way down….

“I’m good... My application doesn’t have an API...”

No… Your application just doesn’t have an **official** API. In the age of bots, if you’re even a **little** bit popular then you’re getting scraped. Whether you like it or not, you have an API -- even if it’s just your web UI.

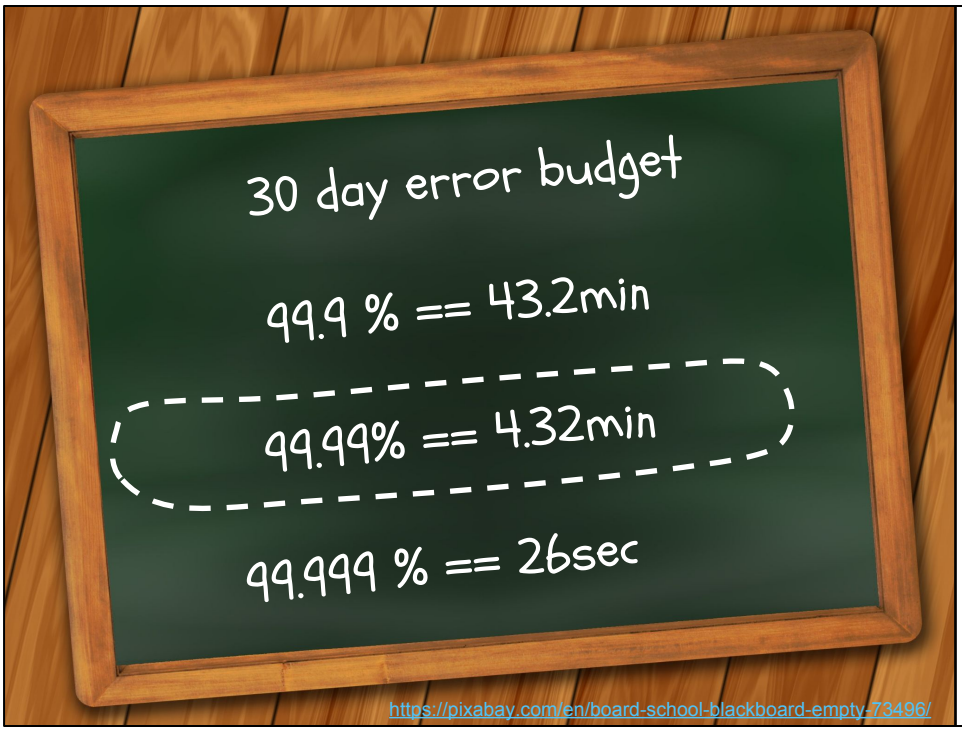


But surely this is a solved problem, right? We know how to do this.. We just need to produce useful artifacts like whitepapers, best practices, deployment guides, sample code, etc...

Principle #3

Well engineered software -> 99.9%; Well engineered operations -> 99.99%; Well engineered business -> 99.999%

It takes well engineered software to get 3 9's. That's as far as whitepapers, deployment guides, etc can get you. It takes well engineered operations -- including shared monitoring and fast rollbacks -- to get to 4 9's, and a well engineered business to get 5 9's. Usually around making hard choices about SLOs and SLAs.



SRE Math Review

NOTE: Image is CC0 - https://pixabay.com/en/board-school-blackboard-empty-73496/

Just a quick SRE math review.. What's the 30 day error budget at 3 9's? Right.. 43.2min. 4 9's? Yup. 4.32min. And for completeness… 5 9'2? Correct.. 26sec. Let's focus on the 4 9's target. If the application calling your platform is at all business critical, they'll usually be aiming for this target. The traditional approach of filing a support ticket pretty much guarantees that they can't rely on you **and** still build a 4 9's system.

Assertion

Your Platform Customers Can Only Get 99.99% by Luck if You Don’t Have Integrated Operations

The basic math and my observation while running Google’s Cloud Support is this: You cannot expect your customers to reliably achieve 4 9’s for their customers if you don’t have some kind of integrated operations… How much? What kind? Let’s talk about that...

We Need to SRE Our Customers

As SREs in our companies we're expected to be the experts and bar raisers for Reliability and scale. We need to do this for our customers, too. In the following slides I'll tell you how and why.

Stage 1

Perform periodic and rigorous application reliability reviews (ARRs) on their key components that rely on your platform.

What exactly is an ARR? This is the process of an SRE (or two) from your company deeply inspecting the design and operation of the customer's app. Your SREs know the failure modes of your platform, and common failure modes of apps on your platform - you know where to look for potential problems.

The Anatomy of an ARR (Key Questions to Ask)

- “What reliability is your application getting now? Can you prove it!”
  - Your customers are almost certainly measuring (and alerting!) on things that don’t much matter.
  - **Key:** Alert on symptoms -- not causes
- “What are your SLOs and how do they relate to your SLIs?”
  - We find that most customers have implicit (rather than explicit) SLOs that are only ‘intuitively’ tied to their business objectives. You need to sort that.
  - **Key:** They need to have a sane error budget and mechanisms to address misses or they’ll forever have unrealistic expectations of **your** platform.

Stage 2

Build Shared Monitoring and Alerting

- A common source of truth between teams
- They only page on metrics in the shared monitoring. No secret data
- System auto-cuts support tickets

NOTE: Shared Monitoring == The Greatest Black Box Probing Network EVER

Shared Monitoring is *Everybody's* Bestest Friend

- You can't realistically expect 99.99%+ without it.
  - 99.99% = 4.32min/mo. It's not practical to expect a human to (a) notice, (b) gather data, and (c) file a ticket in that time.
- Helps eliminate the blame game
- The greatest black-box probing network you can have because it perfectly tracks your customer's experience.

Stage 3

Practice Operational Rigor Between Teams

- Every postmortem is a **joint** postmortem
- Action items are assigned (and tracked) in **both** directions
- Open/closed items are reviewed at **least** quarterly, but probably monthly.

YOU are the incident commander for the joint postmortem. **YOU** drive it forward.

Some rules of the road..

1. **YOU** own and drive the postmortem process. **YOU** are the incident commander.
    a. This is a service you are providing to your customer (probably for $0 extra). That entitles you to have opinions. Postmortem follow-up is the vehicle to express those opinions and turn them into results.
2. If a postmortem doesn't have at least one AI for each party, then that's a red flag.
    a. We broke something? How could their app have been designed to be more resilient?
    b. They broke something? What can we do to our platform to make it harder for them to break it again in the same way?
    c. There's almost always ***something*** we/they can do to make it better for next time.

Stage 4

Joint On-call

- When they spin up a war room, you join and lead debugging from your side
- Regular DiRT/WoM exercises
- If your company permits it, joint projects -- including tooling development

DiRT == Disaster and Recovery Testing -- regularly scheduled production mayhem. Not a thought experiment!

Observation from our DiRT that **both** directions of escalation are important to exercise; how does your company contact them urgently, how do they contact you urgently?

WoM == Wheel of Misfortune -- periodic role playing of an outage/incident. **IS** a thought experiment, but goes as far as possible without actually intentionally causing problems.

“You can only fight the way you practice”

— Miyamoto Musashi, A Book of Five Rings: The Classic Guide to Strategy

We Started Doing This In July
(and announced it in October)

Google Cloud Platform Blog

Product updates, customer stories, and tips and tricks on Google Cloud Platform

Introducing Google Customer Reliability Engineering

Monday, October 10, 2016

Posted by Dave Rensin, Director of Google Customer Reliability Engineering (CRE)

In the 25 years that I've been in technology nearly everything has changed. Computers have moved out of the labs and into our pockets. They're connected together 24/7 and

What is CRE?

A team of Google SREs

- Pointed outwards, instead of inwards
- Comfortable talking to customers
- Software development (esp. externalizing internal SRE tools)
- Conduct ARRs
- Do ongoing design review
- Build and maintain shared monitoring

“So.. This is a kind of professional services, right?”

No. ProServ has its place, but this is not it. This is a partnership of equals where you **expect** to bring strong opinions to the table. We ensure this dynamic in CRE by not charging extra $’s for it. That way we can “hand the pagers back” anytime we get the idea that the customer isn’t interested in putting in their half of the effort. If you make this a work-for-hire arrangement you erode that common ground.

It’s early days, but we’re seeing promising results...

https://goo.gl/KBys08

TL;DR

- Everything is a platform -- we are **all** platform SREs!
- Reliability is the most important feature but is contingent on customer decisions. Your reputation (and success) depends in part on their choices.
- *Spes Consilium Non Est* - Hope is not a strategy. Don't **hope** your customers make good design choices. Help them do it!
- This is not theory. We do this everyday at Google. You can, too!

Q&A