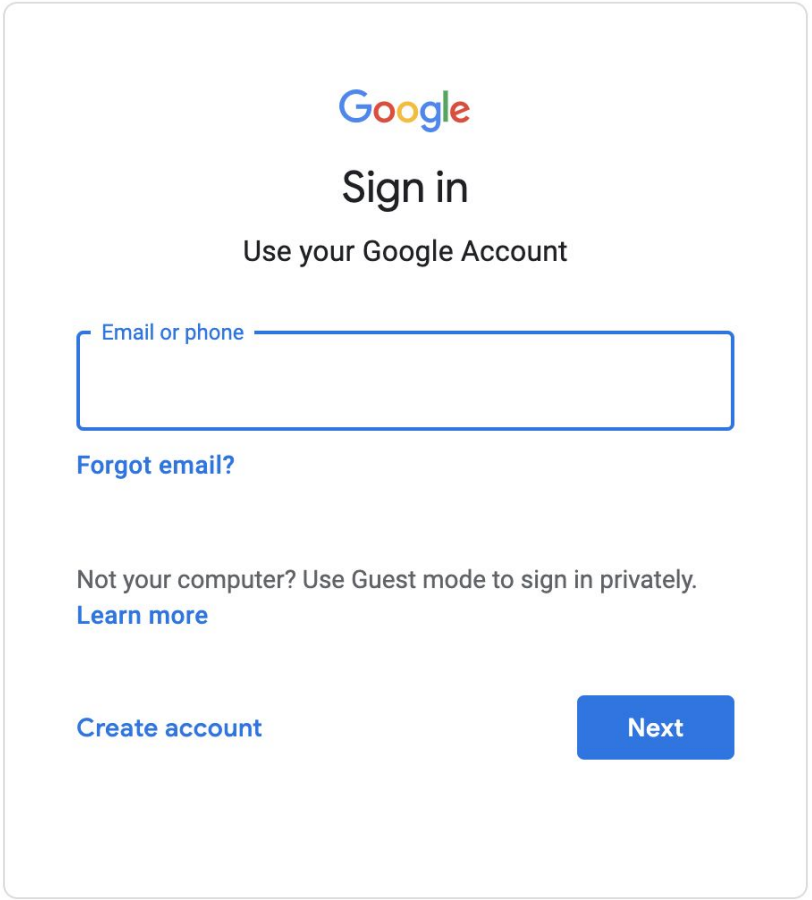
Google

Getting Started with Site Reliability Engineering

74th HPC User Forum – Lugano

Ramón Medrano Llamas / @rmedranollamas


Hello
my name is
Ramón Medrano
niobium@google.com


Google
Sign in
Use your Google Account
Email or phone
Forgot email?
Not your computer? Use Guest mode to sign in privately.
Learn more
Create account
Next
English (United Kingdom)
Help
Privacy
Terms

Software's long-term cost

Software engineering as a discipline focuses on designing and building rather than operating and maintaining, despite estimates that 40%[1] to 90%[2] of the total costs are incurred after launch.

[1] Glass, R. (2002). Facts and Fallacies of Software Engineering, Addison-Wesley Professional; p. 115.
[2] Dehaghani, S. M. H., & Hajrahimi, N. (2013). Which Factors Affect Software Projects Maintenance Cost More? Acta Informatica Medica, 21(1), 63–66. http://doi.org/10.5455/AIM.2012.21.63-66

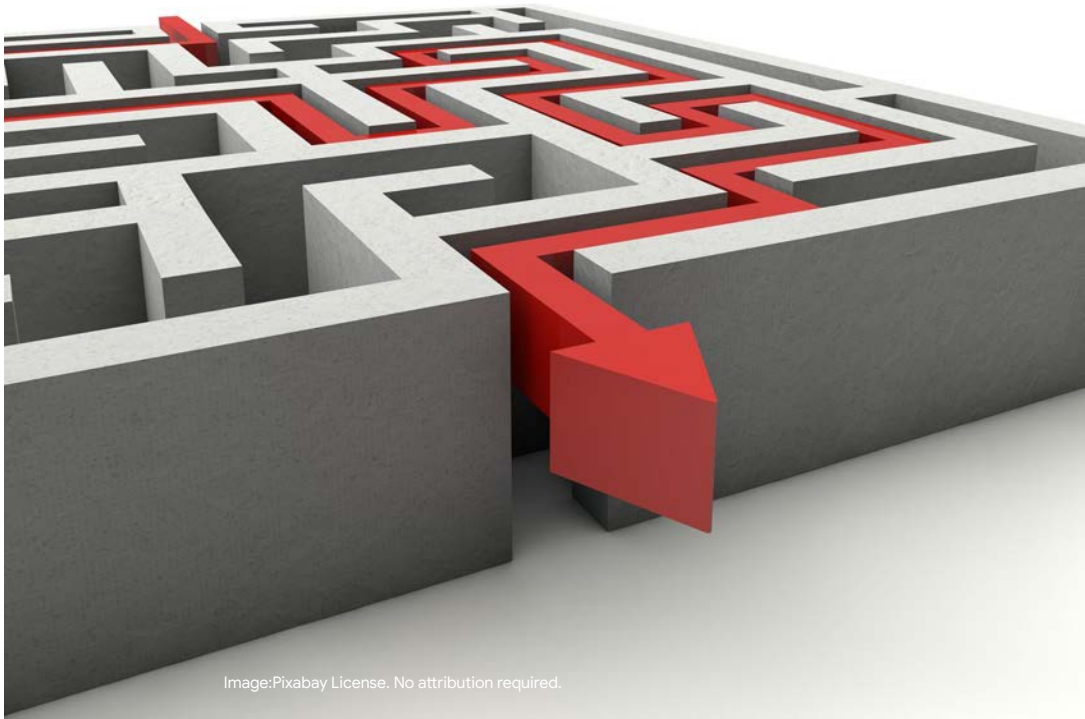
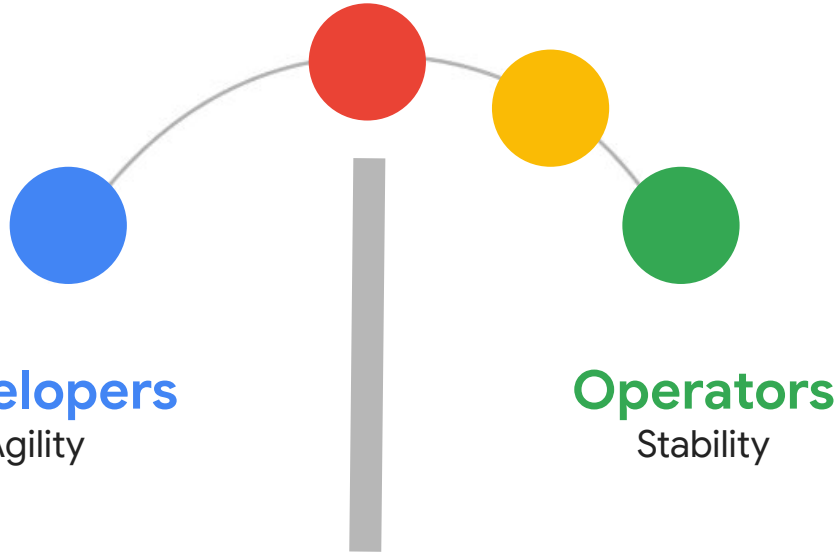


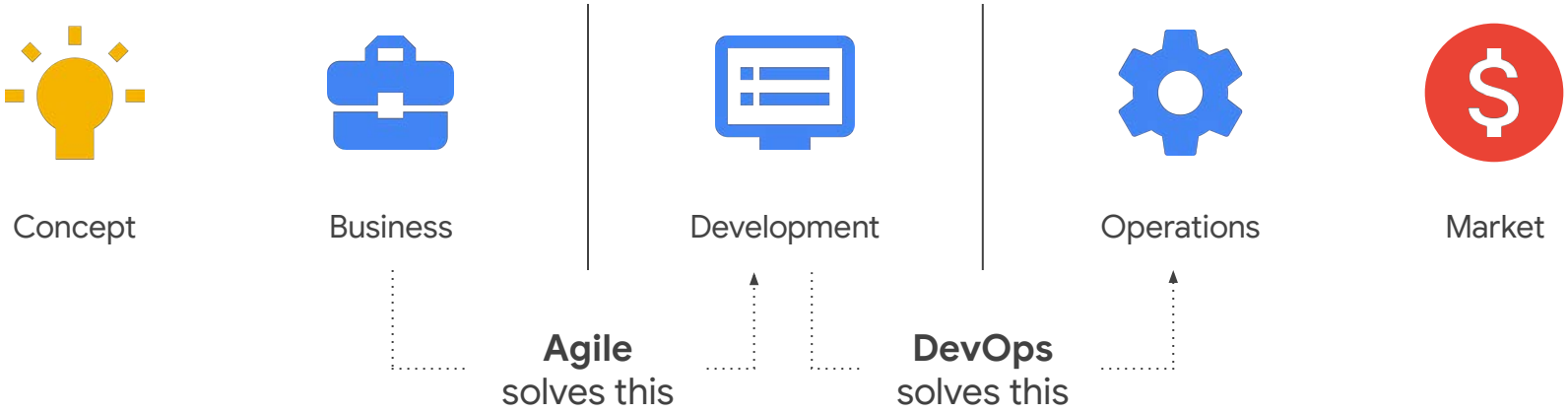
Image:Pixabay License. No attribution required.

Incentives aren't aligned.



Google

Reducing product lifecycle friction



Google

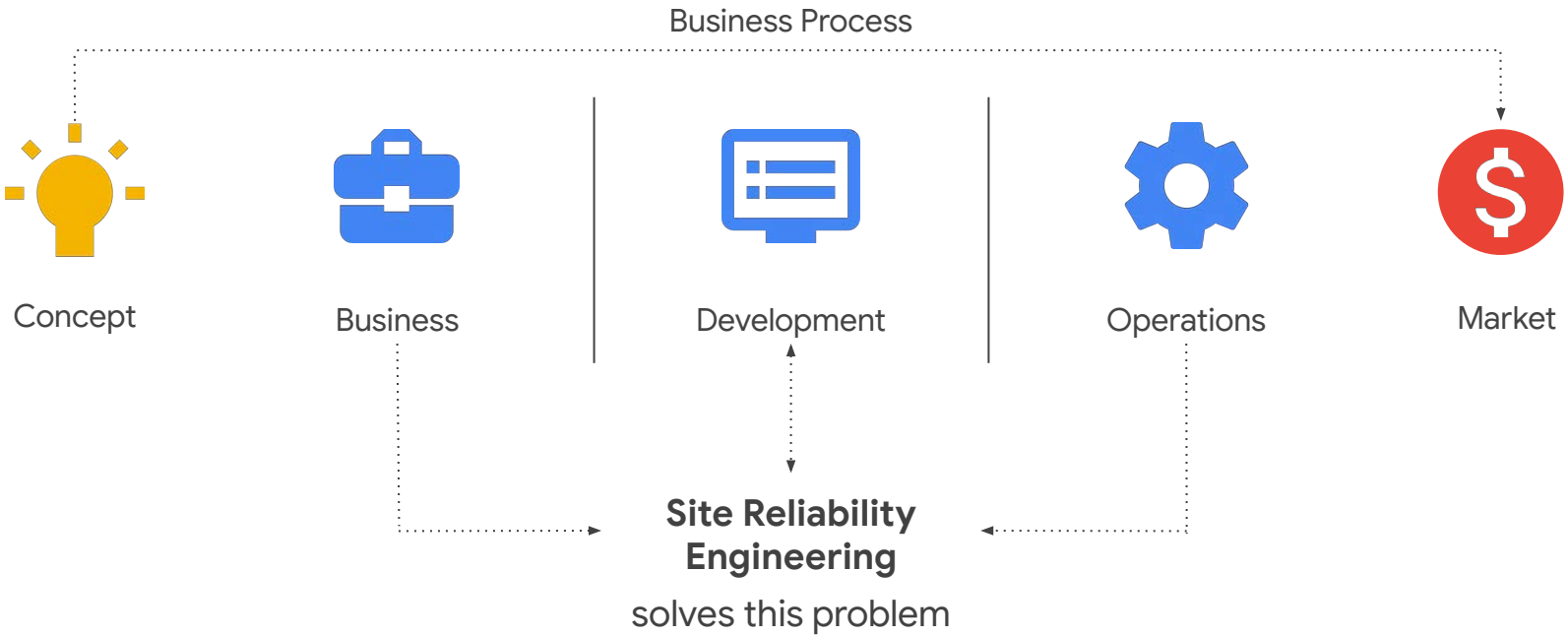
What is Site Reliability Engineering?

- Originated at Google in 2003
- Framework for operating large scale systems reliably
- "SRE is what happens when you ask a software engineer to design an operations function"
- Focuses on running systems in production

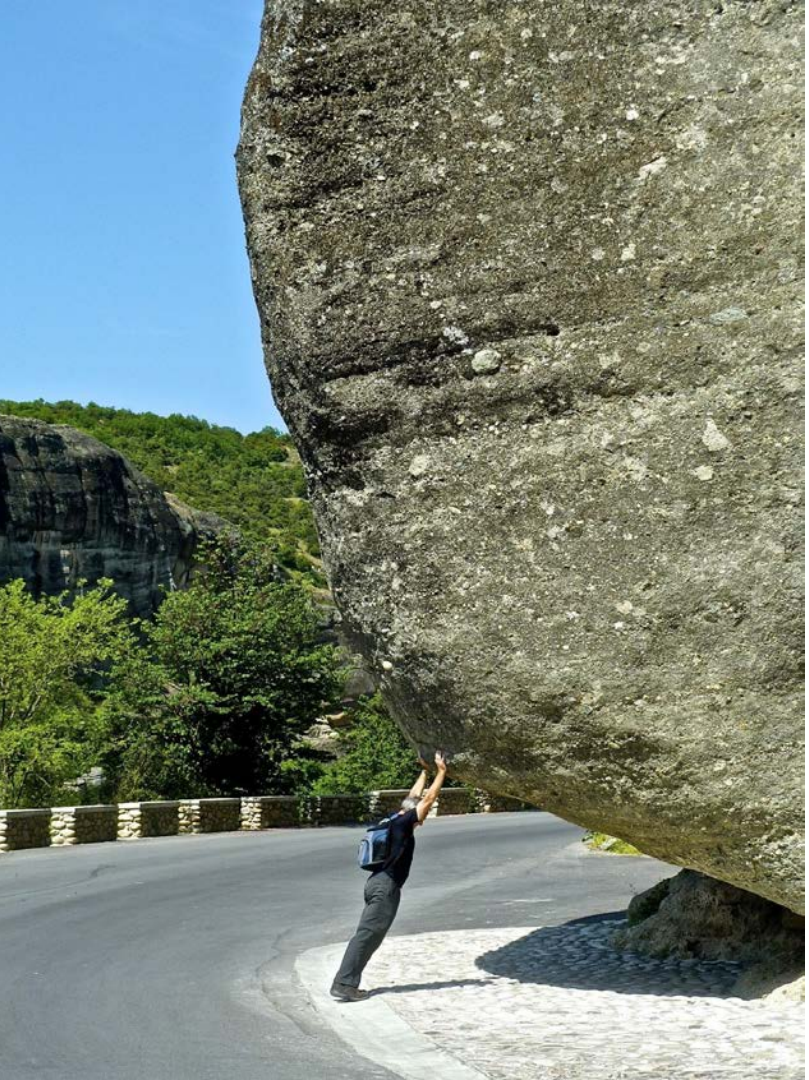
Site Reliability Engineering Principles

1. SRE needs Service Level Objectives (SLOs), with consequences.
2. SREs must have time to make tomorrow better than today.
3. SRE teams have the ability to regulate their workload.
4. Failure is an opportunity to improve.

Product lifecycle



Google



But getting started can feel daunting...

Image: CC0 license: https://pxhere.com/en/photo/739800

Google

Service Level Objectives

What is a Service Level Objective?

- Goal for how well the system should operate
- Tracks the customer experience
  - SLOs met = 😍🙂😀 Customers
  - 😟😱🤬 Customers = SLOs not met

Example SLOs

- 99.99% of HTTP requests per month succeed with 200 OK
- 90% of HTTP requests returned in under 300ms
- 99% of log entries processed in under 5 minutes

But What About SLAs?

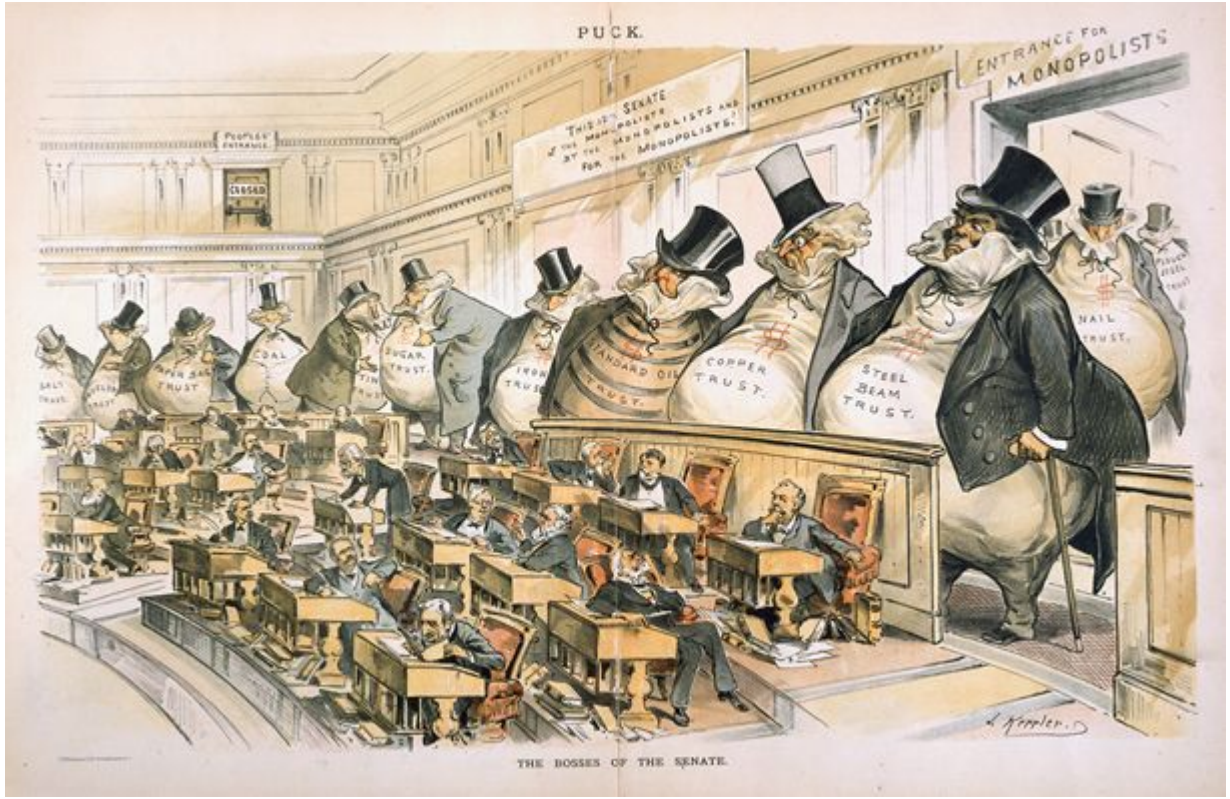
- Service Level Agreements = contractual guarantees
- SLAs met != 😍🙂😀 Customers

What Next?

- You could implement SLOs today for your application, but SLOs are only a foundation.
- You need **consequences**.

Error Budget Policy

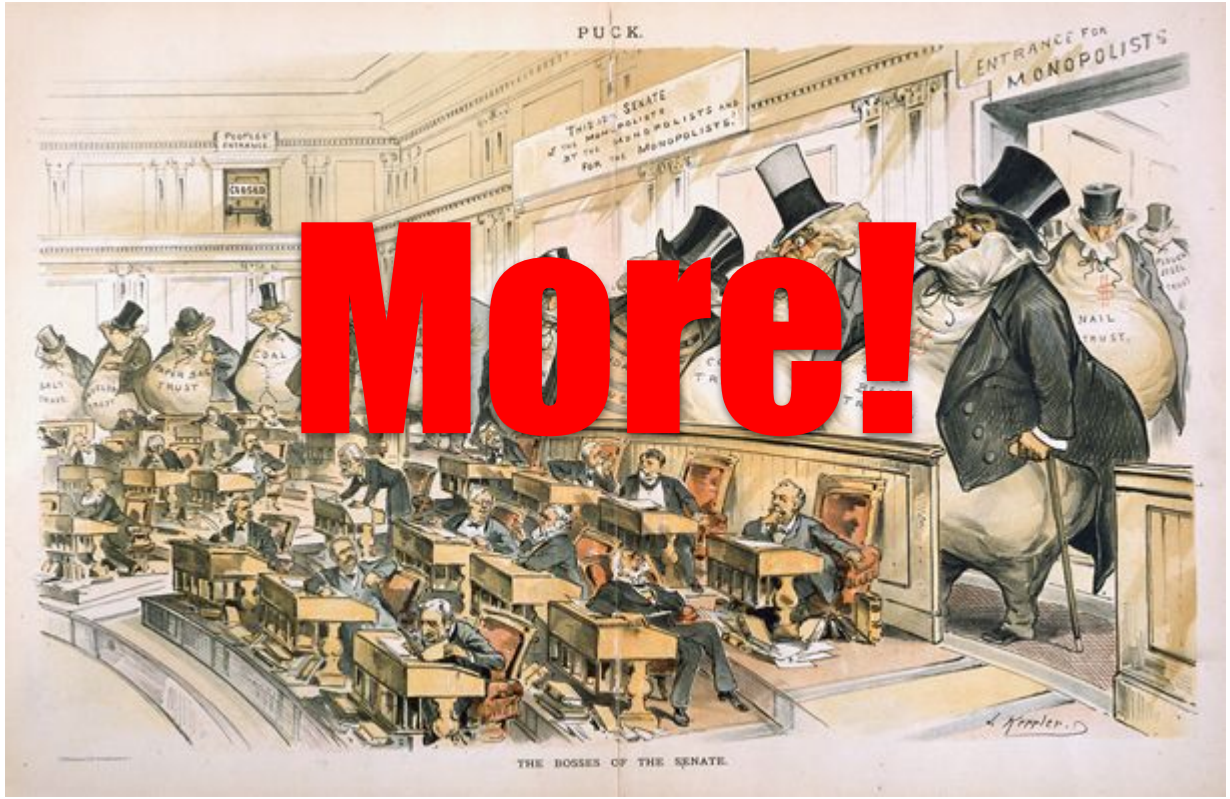
How Reliable Do You Want To Be?



The Bosses of the Senate (1889): Public Domain

Google

How Reliable Do You Want To Be?



The Bosses of the Senate (1889): Public Domain

Google

“

100% is the wrong reliability target for basically everything.

Benjamin Treynor Sloss
Vice President of 24x7 Engineering, Google

Google

SRE is About Balance

williamcho Pixabay License

Reliability

Engineering Time

Development Velocity

Cost

So we introduce a budget

Public Domain Image

Error Budgets

- Gap between perfect reliability and our SLO.
- This is a budget to be spent.
- Given an uptime SLO of 99.9%, after a 20 minute outage you still have 23 minutes of budget remaining for the month!

Error Budget Policy

- What you agree to do when the application exceeds its error budget.
- This is not "pay $$$".
- Must be something that will visibly improve reliability.

Error Budget Policy Examples

Until the application is again meeting its SLO and has some Error Budget:

- "No new feature launches allowed."
- "Sprint planning may only pull Postmortem Action Items from the backlog."
- "Software Development Team must meet with SRE Team daily to outline their improvements"

SRE Principle #1

SRE needs Service Level Objectives with Consequences.

SRE Principle #1

- Even without hiring a single SRE, you can have an Error Budget Policy.
- An error budget is a lever you can use to keep your customers from experiencing pain and sadness.
- You can implement this today: measure, account and act.

Making Tomorrow Better Than Today

Making Tomorrow Better Than Today

- SLOs and Error Budgets are the first step.
- The next step is staffing an SRE role...
- ...endowed with real responsibility.

Your First SRE

- Defines and refines Service Level Objectives.
- Enacts the Error Budget Policy when necessary.
- Makes sure that the application meets the reliability expectations of its users.

Toil

- A **bounded** part of the role.
- Recommend that less than 50% of the workload be operations.

Project Work

- Consulting on System Architecture and Design
- Authoring and iterating on Monitoring
- Automating repetitive work
- Coordinating implementation of Postmortem Action Items

SRE Principle #2

SREs have time to make tomorrow better than today.

SRE Principle #2

- An SRE's job is not to suffer under operational load, but to make each day brighter.
- "Brighter" might mean different things: it depends on what your SREs find most useful to do.
- Less toil, more meaningful system improvements.

Shared Responsibility Model

Dumping *all* production services on an SRE team cannot work.

Google

Photo By: Air Force Tech. Sgt. Jorge Intriago (Public Domain)

An overloaded team doesn't have time to make tomorrow better than today.

Used with permission of the image owner Jennifer Petoff, Sidewalk Safari Blog

Google

Implementing a mechanism to give back pressure to dev partners provides balance.

Google

Used with permission of the image owner Jennifer Petoff, Sidewalk Safari Blog

Regulating Workload

- Give 5% of the operational work to the developers.
- Track SRE team project work.
  - Not completing projects? ⟶ Something's wrong.
- Analyse and on-board new systems *only* if they can be operated safely.
- If every problem has to be escalated to its developer: why is SRE carrying the pager?

Leadership Buy-in

Without leadership buy-in, SRE cannot work.

Google

Image Credit: geralt Pixabay License

Leadership Buy-in

- When applications miss their SLOs and run out of Error Budget, it puts additional load on the SRE team. You need to either:
  - Devote more company resources to addressing reliability concerns
  - Loosen the SLO

Reliability & Consistency Up Front

- Fixing a product after launch is always more expensive.
- SRE teams can and should consult up-front on designs:
  - Architecting resilient systems
  - Maintaining consistency means fewer SREs can support more products

Automation

Three places SRE teams can benefit from Automation:

1. To eliminate their toil: Don't do things over and over!
2. To do capacity planning: Auto-scaling instead of manual forecasting!
3. To fix issues automatically: If you can write the fix in a playbook, you can make the computer do it!

SRE Principle #3

SRE teams have the ability to regulate their workload.

SRE Principle #3

- Teams need to be able to prioritise and do the work.
- Each new system to maintain has a human cost.
- Must be able to push-back on unreliable practices and systems.

A Culture of Blamelessness

Recognize the Antipattern

> I'm extremely angry right now. People should lose their jobs if this was an error.
>
> --Hawaii State Representative Matt Lopresti
> (in reference to the 2018 Hawaii nuclear alert false alarm)

Source: “How Hawaii Could Have Sent A False Nuclear Alarm”, Wired, Lapowski, January 13, 2018
https://www.wired.com/story/hawaii-nuclear-missile-alert-false-explanation/

Embrace Failure

- by setting SLOs less than 100%
- by modeling blamelessness at all levels
- by stamping out blame wherever it is found
- by celebrating cases of “I made a mistake” that lead to outages being resolved faster

Learn from Failure

- You've already paid the price in an outage.
- Write a blameless postmortem.
- Make postmortems widely available so others can learn, too.

“Human” errors are really systems problems.

Keep Asking Why

- The root cause of an outage is never a person.
- Ask “why” for as many iterations as it takes to identify system-related causes.
- Prioritize system fixes that support people to make the right choices.

SRE Principle #4

Failure is an opportunity to improve.

SRE Principle #4

Failure is an opportunity to improve.

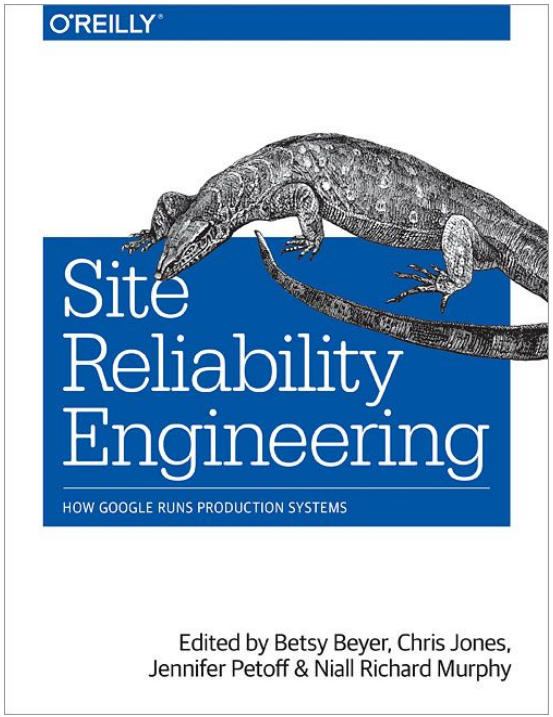
Not an excuse to brandish pitchforks

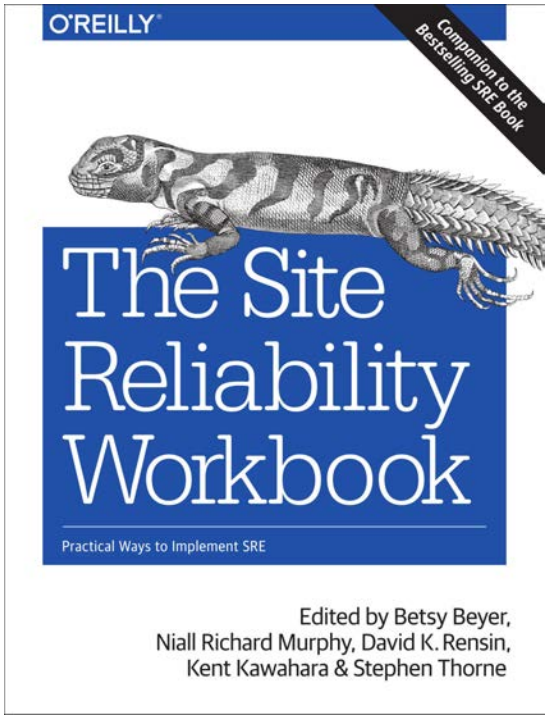
SRE Principle #4

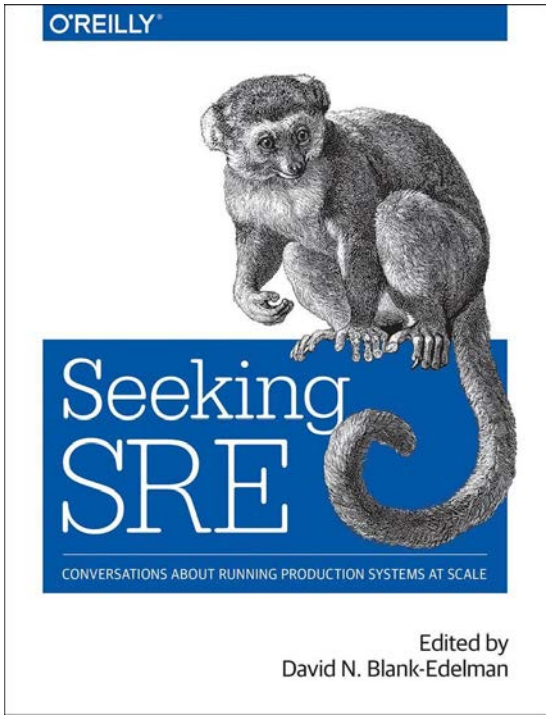
- Failure happens. There is no way around it.
- Stop pointing fingers.
- Embrace failure to improve MTTD and MTTR.
- Proactively addressing failure → more robust systems.

Site Reliability Engineering Principles

1. SRE needs Service Level Objectives, with consequences.
2. SREs must have time to make tomorrow better than today.
3. SRE teams have the ability to regulate their workload.
4. Failure is an opportunity to improve.







Google

Cover images used with permission. These books can be found on shop.oreilly.com
The full text of the Google SRE Books are available at www.google.com/sre