# Getting Started with Site Reliability Engineering

DevOps Enterprise Summit London 2018

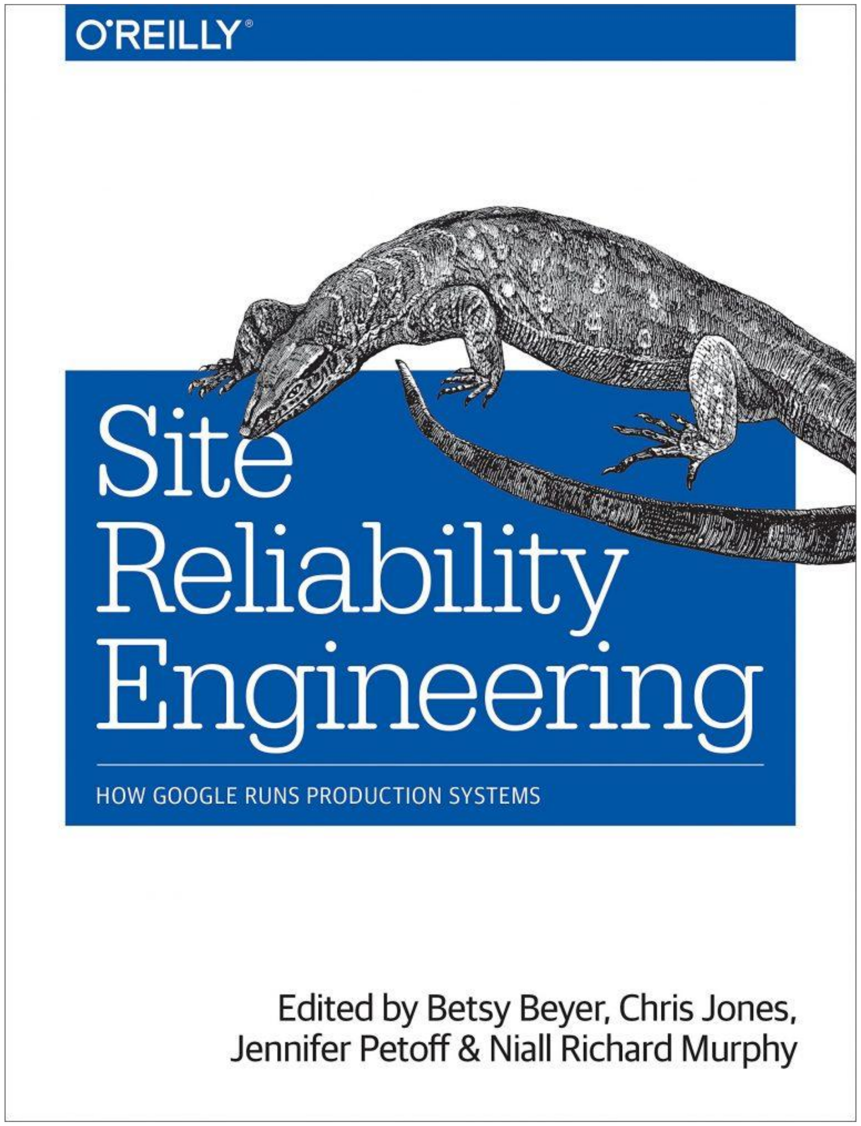
Stephen Thorne / @Jerub

## Agenda

## Introduction

## What is SRE?

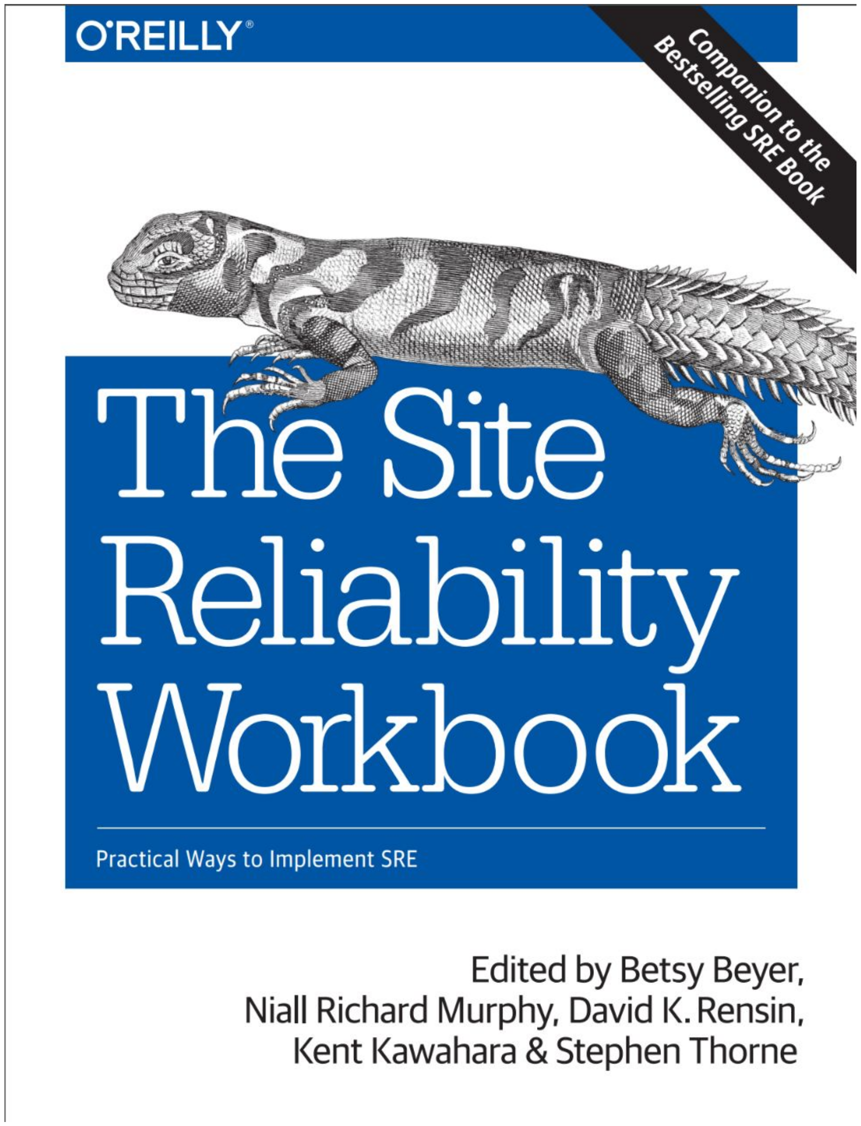
- Site Reliability Engineering originated at Google in 2003
- A framework for operating large scale systems reliably.
- "What happens when you make Engineers do Operations."
- SRE own running the system in production at an operational level.


O'REILLY®

Site Reliability Engineering

HOW GOOGLE RUNS PRODUCTION SYSTEMS

Edited by Betsy Beyer, Chris Jones, Jennifer Petoff & Niall Richard Murphy


O'REILLY®

Companion to the Bestselling SRE Book

The Site Reliability Workbook

Practical Ways to Implement SRE

Edited by Betsy Beyer, Niall Richard Murphy, David K. Rensin, Kent Kawahara & Stephen Thorne

Google

## Who am I?

- Stephen Thorne
- Spent 6 years working as a Google SRE in London. On both Ads and Google App Engine.
- The last year has been a new adventure: Starting Customer Reliability Engineering in London.
- CRE is the concept of taking SRE principles and helping our Cloud Customers operate Reliably at Scale.
- Editor of the Site Reliability Workbook

## The Site Reliability Principles

1. SRE needs Service Level Objectives, with consequences.
2. SREs have time to make tomorrow better than today.
3. SRE teams have the ability to regulate their workload.

## Service Level Objectives

## What is an SLO?

Service Level Objectives set a goal for how well the system should behave.

Specifically tracking customer experience.

If customers are happy, then the SLO is being met.

## Typical SLOs

- Uptime of 99.9% a month (i.e. 43 minutes of downtime a month)
- 99.99% of HTTP requests in a month succeed with a 200 OK
- 50% of HTTP requests returned in under 300ms
- 99% of log entries processed in under 5 minutes.

## But What About SLAs?

Service Level Agreements are typically guarantees with penalties for not meeting them.

A system can still be within its SLA and the customers can also feel very unhappy with the experience.

## What Next?

- You could implement this today in your organisation, but this is only a foundation.
- You need **consequences**.

## Error Budget Policy

## How Reliable Do You Want To Be?

The answer is always more!

We know the key to SRE is to be able to balance your error budget against engineering time, development velocity and money.

So we introduce a Budget.

## Error Budgets

- The error budget is the 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

- The Error Budget Policy is what you agree to do when the application exceeds it's 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 features 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.

Google

## SRE Principle #1

- Any organisation, even without hiring a single SRE team can have an Error Budget Policy.
- This is any lever you can use to keep your customers from experiencing pain using your application.
- 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.
- Real responsibility.

## Your First SRE

- Defining and refining the Service Level Objectives.
- Best placed person to see that the Error Budget Policy is enacted when necessary.
- Is now responsible for making sure that the application meets the reliability expectations of its users.

## Toil

- Operating systems in production.
- Must be a part of the role.
- On-call and incident management.
- A **bounded** part of the role: Recommend less than 50% of the workload be operations.

## Project Work

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

SRE Principle #2

## SREs have time to make tomorrow better than today.

## SRE Principle #2

- Once you have SREs: make sure they know that their 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.

## Shared Responsibility Model

## Shared Responsibility

- Dumping all production services on an SRE team cannot work.
- If a team gets overloaded with operational toil, they cannot make tomorrow better than today.
- Providing an SRE team some way of giving back-pressure to their dev partners provides balance.

## Regulating Workload

- Give 5% of the operational work to the developers: On-call shifts, rollout management, ops tasks.
- Track the project work of the SRE team: If it's not delivering completed projects: there's something wrong.
- Analyse new production systems and only on-board them if they can be operated safely.
- If every problem with a system has to be escalated to its developer: Give the pager to the developer instead!

## Leadership Buy-in

- An SRE organisation within a company needs a mandate.
- Without leadership buy-in, it can not work.
- When applications miss their SLOs and run out of Error Budget: it puts additional load on the SRE team:
    - Need to devote more company resources to addressing reliability concerns.
    - or: Loosen  the SLO.

## Reliability and 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:
    - To eliminate their toil - don't do things over and over!
    - To do capacity planning - auto-scaling instead of manual forecasting!
    - 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

- Your 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.

## Summary

Google

## The Site Reliability Principles

1 SRE needs Service Level Objectives, with consequences.

2 SREs have time to make tomorrow better than today.

3 SRE teams have the ability to regulate their workload.

Thank You