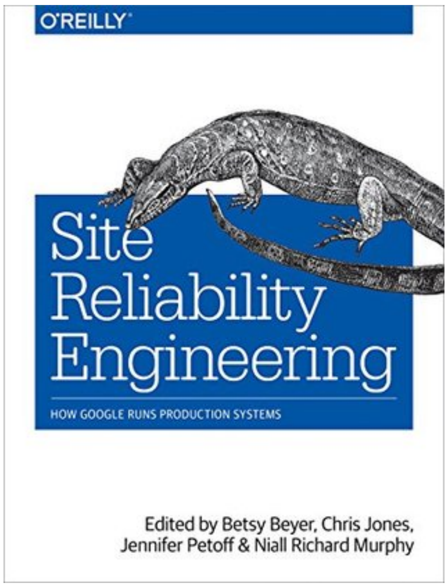
O'REILLY
Site Reliability Engineering
HOW GOOGLE RUNS PRODUCTION SYSTEMS
Edited by Betsy Beyer, Chris Jones, Jennifer Petoff & Niall Richard Murphy

Site Reliability Engineering: How Google Runs Production Systems

Betsy Beyer , Chris Jones , Jennifer Petoff , Niall Richard Murphy

Download now

Read Online

Site Reliability Engineering: How Google Runs Production Systems

Betsy Beyer , Chris Jones , Jennifer Petoff , Niall Richard Murphy

Site Reliability Engineering: How Google Runs Production Systems Betsy Beyer , Chris Jones , Jennifer Petoff , Niall Richard Murphy

The overwhelming majority of a software system’s lifespan is spent in use, not in design or implementation. So, why does conventional wisdom insist that software engineers focus primarily on the design and development of large-scale computing systems?

In this collection of essays and articles, key members of Google’s Site Reliability Team explain how and why their commitment to the entire lifecycle has enabled the company to successfully build, deploy, monitor, and maintain some of the largest software systems in the world. You’ll learn the principles and practices that enable Google engineers to make systems more scalable, reliable, and efficient—lessons directly applicable to your organization.

This book is divided into four sections:

Introduction—Learn what site reliability engineering is and why it differs from conventional IT industry practices **Principles**—Examine the patterns, behaviors, and areas of concern that influence the work of a site reliability engineer (SRE) **Practices**—Understand the theory and practice of an SRE’s day-to-day work: building and operating large distributed computing systems **Management**—Explore Google's best practices for training, communication, and meetings that your organization can use

Site Reliability Engineering: How Google Runs Production Systems Details

Date	:	Published March 23rd 2016 by O'Reilly Media
ISBN	:	
Author	:	Betsy Beyer , Chris Jones , Jennifer Petoff , Niall Richard Murphy
Format	:	Kindle Edition 554 pages
Genre	:	Science, Technology, Computer Science, Programming, Technical, Nonfiction

Download Site Reliability Engineering: How Google Runs Productio ...pdf

Read Online Site Reliability Engineering: How Google Runs Product ...pdf

Download and Read Free Online Site Reliability Engineering: How Google Runs Production Systems Betsy Beyer , Chris Jones , Jennifer Petoff , Niall Richard Murphy

From Reader Review Site Reliability Engineering: How Google Runs Production Systems for online ebook

Tomas Varaneckas says

This was a really hard read, in a bad sense. The first couple of dozen pages were really promising, but the book turned out to be unnecessarily long, incredibly boring, repetative and inconsistent gang bang of random blog posts and often trivial information. It has roughly 10% of valuable content, and would greatly benefit from being reduced to 50-pager. At it's current state it seems that it was a corporate collaborative ego-trip, to show potential employees how cool Google SRE is, and how majestic their scale happens to be. After reading this book, I am absolutely sure I would never ever want to work for Google.

Michael Koltsov says

I don’t normally buy paper books, which means that in the course of the last few years I’ve bought only one paper book even though I’ve read hundreds of books during that period of time. This book is the second one I’ve bought so far, which means a lot to me. Not mentioning that Google is providing it on the Internet free of charge.

For me, personally, this book is a basis on which a lot of my past assumptions could be argued as viable solutions with the scale of Google. This book is not revealing any Google’s secrets (do they really have any secrets?) But it’s a great start even if you don’t need the scale of Google but want to write robust and failure-resilient apps.

Technical solutions, dealing with the user facing issues, finding peers, on-call support, post-mortems, incident-tracking systems – this book has it all though, as chapters have been written by different people some aspects are more emphasized than the others. I wish some of the chapters had more gory production-based details than they do now.

My score is 5/5

Tadas Talaikis says

"Boring" (at least from the outside world perspective, ok with me), basically can be much shorter. Culture, automation of everything, load balancing, monitoring, like everywhere else, except maybe Borg thing.

Alexander Yakushev says

This book is great on multiple levels. First of all, it packs great content — the detailed explanation of how and why Google has internally established what we now call "the DevOps culture." Rationale coupled together with hands-on implementation guide provide incredible insight into creating and running SRE team in your own company.

The text quality is top-notch, the book is written with clarity in mind and thoroughly edited.
I'd rate the content itself at four stars. But the book deserves the fifth star because it is a superb example of a material that gives you the precise understanding of how some company (or its division) operates inside. Apparently, Google can afford to expose such secrets while not many other companies can, but we need more low-BS to-the-point books like this to share and exchange the experience of running the most complex systems (that is, human organizations) efficiently.

Alex Palcuie says

I think this is the best engineering book in the last decade.

Michael Scott says

Site Reliability Engineering, or *Google's claim to fame re: technology and concepts developed more than a decade ago by the grid computing community*, is a collection of essays on the design and operation of large-scale datacenters, with the goal of making them simultaneously scalable, robust, and efficient. Overall, despite (willing?) ignorance of the history of distributed systems and in particular (grid) datacenter technology, this is an excellent book that teaches us how Google thinks (or used to think, a few years back) about its datacenters. If you're interested in this topic, you have to read this book. Period.

Structure
The book is divided into four main parts, each comprised of several essays. Each essay is authored by what I assume is a Google engineer, and edited by one of Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy. (I just hope that what I didn't like about the book can be attributed to the editors, because I really didn't like some stuff in here.)

In Part I, Introduction, the authors introduce Google's Site Reliability Engineering (SRE) approach to managing global-scale IT services running in datacenters spread across the entire world. (Truly impressive achievement, no doubt about it!) After a discussion about how SRE is different from DevOps (another hot term of the day), this part introduces the core elements and requirements of SRE, which include the traditional Service Level Objectives (SLOs) and Service Level Agreements (SLAs), management of changing services and requirements, demand forecasting and capacity, provisioning and allocation, etc. Through a simple service, Shakespeare, the authors introduce the core concepts of running a workflow, which is essentially a collection of IT tasks that have inter-dependencies, in the datacenter.

In Part II, Principles, the book focuses on operational and reliability risks, SLO and SLA management, the notion of toil (mundane work that scales linearly (why not super-linearly as well?!?!) with services, yet can be automated) and the need to eliminate it (through automation), how to monitor the complex system that is a datacenter, a process for automation as seen at Google, the notion of engineering releases, and, last, an essay on the need for simplicity . This rather disparate collection of notions is very useful, explained for the laymen but still with enough technical content to be interesting even for the expert (practitioner or academic).

In Parts III and IV, Practices and Management, respectively, the book discusses a variety of topics, from time-series analysis for anomaly detection, to the practice and management of people on-call, to various ways to prevent and address incidents occurring in the datacenter, to postmortems and root-cause analysis that could help prevent future disasters, to testing for reliability (a notoriously difficult issue), to software engineering int he SRE team, to load-balancing and overload management (resource management and scheduling 101), communication between SRE engs, etc. etc. etc., until the predictable call for everyone to use SRE as early as possible and as often as possible. Overall, palatable material, but spread too thin and with too much much overlap with prior related work of a decade ago, especially academic, and not much new insight.

What I liked

I especially liked Part II, which in my view is one of the best introductions to datacenter management available today to the students of this and related topics (e.g., applied distributed systems, cloud computing, grid computing, etc.)
Some of the topics addressed, such as risk and team practices, are rather new for many in the business. I liked the approach proposed in this book, which seemed to me above and beyond the current state-of-the-art.
Topics in reliability (correlated failures, root-cause analysis) and scheduling (overload management, load balancing, architectural issues, etc.) are currently open in both practice and academia, and this book emphasizes in my view the dearth of good solutions but for the simplest of problems.
Many of the issues related to automated monitoring and incident detection could lead in the future to better technology and much innovation, so I liked the prominence given to these topics in this book.

What I didn't like
I thoroughly disliked the statements claiming by omission that Google has invented most of the concepts presented in the book, which of course in the academic world would have been promptly sent to the reject pile. As an anecdote, consider the sentence *Ben Treynor Sloss, Google's VP for 24/7 Operations, originator of the term SRE, claims that reliability is the most fundamental feature of any product: a system isn't very useful if nobody can use it!*. I'll skip the discussion about who is the originator of the term SRE, and focus on the meat of this statement. By omission, it makes the reader think that Google, through its Ben Treynor Sloss, is the first to understand the importance of reliability for datacenter-related systems. In fact, this has been long-known in the grid computing community. I found in just a few minutes explicit references from Geoffrey Fox (in 2005, on page 317 of yet another grid computing anthology, "service considers reliable delivery to be more important than timely delivery"), Alexandru Iosup (in 2007, on page 5 of this presentation, and again in 2009, in this course, "In today's grids, reliability is more important than performance!"). Of course, this notion has been explored for the general case of services much earlier... anyone familiar with air and especially space flight? The list of concepts actually not invented at Goog but about which the book implies to the contrary goes on and on...

I also did not like some of the exaggerated claims of having found solutions for the general problems. Much remains to be done, as hiring at Google in these areas continues unabated. (There's also something called computer science, whose state-of-the-art indicates the same.)

David says

The book seems largely to be a collection of essays written by disparate people within Google's SRE organization. It's as well-organized and coherent as that can be (and I think it's a good format for this -- far better than if they'd tried to create something with a more unified narrative). But it's very uneven: some chapters are terrific while some seem rather empty. I found the chapters on risk, load balancing, overload, distributed consensus, and (surprisingly) launches to be among the most useful. On the other hand, the chapter on simplicity was indeed simplistic, and the chapter on data integrity was (surprisingly) disappointing.

The good: there's a lot of excellent information in this book. It's a comprehensive, thoughtful overview for anybody entering the world of distributed systems, cloud infrastructure, or network services. Despite a few misgivings, I'm pretty on board with Google's approach to SRE. It's a very thoughtful approach to the problems of operating production services, covering topics ranging from time management, prioritization, onboarding, plus all the technical challenges in distributed systems.

The bad: The book gets religious (about Google) at times, and some of it's pretty smug. This isn't a big deal, but it's likely to turn off people who've seen from experience how frustrating and unproductive it can be when good ideas about building systems become religion.

Chris says

There's a ton of great information here, and we refer to it regularly as we're trying to change the culture at work. I gave it a 4 instead of a 5 because it does suffer a little from the style – think collection of essays rather than a unified arc – but it's really worth reading even if it requires some care to transfer to more usual environments.

Mircea ?ivadariu says

Boring as F. The main message is: oh look at us, we have super hard problems and like saying 99.999% a lot. And oh yeah... SREs are developers. We don't spend more than 50% on "toil" work. Pleeeease. Book has some interesting stories and if you are good at reading between the lines you might learn something. Everything else is BS. Does every chapter needs to start telling us who edited the chapter? I don't give a f. The book also seems to be the product of multiple individuals (a lot of them actually) whose sole connection is that they wrote a chapter for this book. F the reader, F structure, F focusing on the core of the issue. Let's just dump a stream of consciousness kind of junk and after that tell everyone how hard it is and how we care about work life balance. Again, boring and in general you're gonna waste your time reading this (unless you want to know what borg, chubby and bigtable are)

Tim O'Hearn says

“Perfect algorithms may not have perfect implementations.”

And perfect books may not have perfect writers. *Site Reliability Engineering* is an essay collection that can be rickety at times but is steadfast in its central thesis. Google can claim credit for inventing Site Reliability Engineering and, in this book, a bunch of noteworthy engineers share their wisdom from the trenches.

When it comes to software architecture and product development, I’ve found delight in reading about how startups’ products are built because the stories are digestible. It’s possible for a founder, lead engineer, or technical writer to lay down the blueprint of a small-scale product and even get into the nuts and bolts. When it comes to large tech companies, this is impossible from a technical point of view and improbable from a compliance standpoint.

This is beside the purpose of the book, but arrangements like this one help bridge the gap between one’s imagination and the inner-workings of tech giants. There are plenty of (good!) books that tell you all about how Google the business works, but this one happens to be the best insight into how the engineering side operates. Sure, you have to connect some dots and bring with you some experience, but the result is priceless--you start to feel like you get it.

The essays are almost all useful. If you haven’t spent at least an internship’s worth of time in the workforce, you should probably table this one until you have a bit more experience. I would have enjoyed this book as an undergraduate, no doubt, but most of it wouldn’t have clicked. The Practices section--really, the meat of the book--is where the uninitiated might struggle. When I emerged on the other side I had a list of at least twenty topics that I needed to explore in more detail if I was to become truly great at what I do.

I highly recommend this book to anyone on the SRE/DevOps spectrum as well as those trying to understand large-scale tech companies as a whole.

See this review and others on my blog

James Stewart says

Loads of interesting ideas and thoughts, but a bit of a slog to get through.

The approach of having different members of the team write different sections probably worked really well for engaging everyone, but it made for quite a bit of repetition. It also ends up feeling like a few books rolled into one, with one on distributed systems design, another on SRE culture and practices, and maybe another on management.

Dimitrios says

I have so many bookmarks in this book and consider it an invaluable read. While not every project / company needs to operate at Google scale, it helps streamlining the process to define SLO / SLAs for the occasion and establishing communication channels and practices to achieve them.

It helped me wrap my head around concepts for which I used to rely on intuition.
I've shaped processes and created template documents (postmortem / launch coordination checklist)
for work based on this book.

Ahmad hosseini says

What is SRE?
Site Reliability Engineering (SRE) is Google’s approach to service management.
An SRE team is responsible for the availability, latency, performance, efficiency, change management, monitoring, emergency response, and capacity planning of their service(s).
Typical SRE activities fall into the following approximate categories:
• Software engineering: Involves writing or modifying code, in addition to any associated design and documentation work.
• System engineering: Involves configuring production systems, modifying configuration, or documenting systems in a way that products lasting improvements from a one-time effort.
• Toil: work directly to running a service that is repetitive, manual, etc.
• Overhead: Administrative work not tied directly to running a service.

Quotes
“Be warned that being an expert is more than understanding how a system is supposed to work. Expertise is gained by investigating why a system doesn’t work.” – Brain Redman
“Ways in which things go right are special cases of the ways in which things go wrong.” – John Allspaw

About book
This book is a series of essays written by members and alumni of Google’s Site Reliability Engineering organization. It’s much more like conference proceedings than it is like a standard book by an author or a small number of authors. Each chapter is intended to be read as a part of a coherent whole, but a good deal can be gained by reading on whatever subject particularly interests you.
“Essential reading for anyone running highly available web services at scale.” – Adrian Cockcroft, Battery Ventures, former Netflix Cloud Architect

---

Simon Eskildsen says

Much of the information on running production systems effectively from Google has been extremely important to how I have changed my thinking about the SRE role over the years—finally, there's one piece that has all of what was previously something you'd had to look long and hard for in various talks, papers and abstracts: error budgets, the SRE role definition, scaling, etc. That said, this book suffers a classic problem from having too many authors write independent chapters. Much is repeated, and each chapter stands too much on its own—building from first principles each time, instead of leveraging the rest of the book. This makes the book much longer than it needs to be. Furthermore, it tries to be both technical and non-technical—this confuses the narrative of the book, and it ends up not excelling at either of them. I would love to see two books: SRE the technical parts, and SRE the non-technical parts. Overall, this book is still a goldmine of information to a 5/5—but it is exactly that, a goldmine that you'll have to put a fair amount of effort into dissecting to retrieve the most value from, because the book's structure doesn't hand it to you—that's why we land at a 3/5. When recommending this book to coworkers, which I will, it will be chapters from the book—not the book at large.

Daniël says

So, I can see how this book has been so influential in the industry. A lot of what's in here should be common sense for people who have been working in tech for over 5 years but sadly isn't. This book reminded me of much advice I've given over my career, that was then promptly discarded for multiple reasons. This book gives me some ammo in these kinds of discussions, so that's great. That's not to say that I didn't learn anything from the book, there's many things in here that made me think "Oh, that's a great idea" and other twists on things that I already knew that I will incorporate in my work. Furthermore, it's interesting to see how Google manages things and the details about their infrastructure are good reads. This book was quite a dry read though, as tech books tend to be, which explains why I took quite a bit longer to finish it.
