Release It!: Design and Deploy Production-Ready Software

Michael T. Nygard

Release It!: Design and Deploy Production-Ready Software

Michael T. Nygard

Release It!: Design and Deploy Production-Ready Software Michael T. Nygard
A single dramatic software failure can cost a company millions of dollars - but can be avoided with simple changes to design and architecture. This new edition of the best-selling industry standard shows you how to create systems that run longer, with fewer failures, and recover better when bad things happen. New coverage includes DevOps, microservices, and cloud-native architecture. Stability antipatterns have grown to include systemic problems in large-scale systems. This is a must-have pragmatic guide to engineering for production systems.

If you're a software developer, and you don't want to get alerts every night for the rest of your life, help is here. With a combination of case studies about huge losses - lost revenue, lost reputation, lost time, lost opportunity - and practical, down-to-earth advice that was all gained through painful experience, this book helps you avoid the pitfalls that cost companies millions of dollars in downtime and reputation. Eighty percent of project life-cycle cost is in production, yet few books address this topic.

This updated edition deals with the production of today's systems - larger, more complex, and heavily virtualized - and is the first book to cover chaos engineering, the discipline of applying randomness and deliberate stress to reveal systematic problems. Build systems that survive the real world, avoid downtime, implement zero-downtime upgrades and continuous delivery, and make cloud-native applications resilient. Examine ways to architect, design, and build software - particularly distributed systems - that stands up to the typhoon winds of a flash mob, a Slashdotting, or a link on Reddit. Take a hard look at software that failed the test and find ways to make sure your software survives.

To skip the pain and get the experience...get this book.

Release It!: Design and Deploy Production-Ready Software Details

Date	:	Published October 25th 2017 by Pragmatic Bookshelf (first published March 30th 2007)
ISBN	:	9781680502398
Author	:	Michael T. Nygard
Format	:	Paperback 356 pages
Genre	:	Computer Science, Programming, Science, Technology, Technical, Software

From Reader Review Release It!: Design and Deploy Production-Ready Software for online ebook

Alexander Yakushev says

A must-read for every software engineer who builds complex systems.

James Healy says

This book radically influenced the way I build and deploy software.

It's a whirlwind tour through designing code that behaves well in production, the many ways interaction between multiple systems can fail, deployment styles that avoid scheduled downtime, and case studies to demonstrate the surprises that happen in the real world.

For those new developing and deploying production software the pace might be hard to follow, but those with a bit of experience under their belt will find this triggers memories, provides a language and framework to understand the issues you've encountered in production, and patterns to help you manage those issues when they reoccur.

For those that haven read the first edition of Release it, the second edition is worth a revisit. A lot has changed in 10 years, and the book has been significantly updated to account for that. I like the logical progression of the new book outline too - Creating Stability, Designing for Production, Delivering your System.

Tom Purl says

I need to start by saying that this is one of the best technical books I have ever read. To me, it's easily as enjoyable and useful as Code Complete, The Pragmatic Programmer, or The Mythical Man Month. If you're a sysadmin, an architect, or a developer that works with medium-to-large-sized systems, then do the following:

1. Stop reading this post
2. Order this book from your library or buy it from The Pragmatic Programmer's web site
3. Owe me a pint :D

What The Book Is Really About

Actually, there is one thing that I don't like about this book, but it really has nothing to do with the book. The description of this book on the Pragmatic Programmer's web site sucks. It's vague, and it really gives the potential reader a tiny amount of insight into the book's contents.

What it should have said is that this book contains *tons* of great information on designing, deploying,

maintaining and *improving* medium-to-large-sized IT systems. It's filled with patterns, anti-patterns, and general best practices that should be part of the shared lexicon of every developer, administrator, and system architect. Also, it does a good job of giving you enough information to be useful without boring you to death. And finally, it's written very well and is a joy to read.

The Highlights

Thread Dumps & Garbage Collection Tuning

The internals of the Java Virtual Machine (JVM) have been a black box to me for the majority of my career in IT. Thankfully, this book has provided excellent examples of how you can troubleshoot and improve your system using tools that interrogate and manipulate a JVM at runtime.

For me, this was the most interesting and useful part of the book, and I am looking forward to seeing what can be gained by tuning and "poking at" the JVM's that are in the system that I maintain.

Patterns and Anti-Patterns

It's great to finally find a book that codifies some patterns that administrators and architects can use.

Transparency

I thought that I new a lot about monitoring and transparency before reading this book, but now I know better. I especially like the concept of a unified "OpsDB", and I am eager to build something like this myself for the system that I maintain.

Integration Point Risks

I always knew that integration points (e.g. data feeds, databases, LDAP providers, etc.) added risk to you system, but the author does a great job calculating the actual risk. Also, he shows you many ways in which you can avoid brittle integration points.

Caveats

I have one warning about this book, but it's half-hearted. This book is what I would all Java-centric. All of the case studies involve systems that are written in Java, and some of the sections will only apply directly to you if you are working with Java-based software.

But does that mean that you should avoid this book if you are working with Ruby, PHP, or .Net-based software? Absolutely not. Even though there are a few small sections of the book that won't directly apply to your line of work, most of them will apply in an indirect way, regardless of your platform. And the other 94% of the book will directly apply to medium-to-large systems of every stripe.

Simon Eskildsen says

This book is an incredible introduction to creating and maintaining resilient web applications in realistic, chaotic environments. This book has changed how I approach development more than any other. Every

developer with something in production should read it.

Rod Hilton says

You're wasting time reading this review, you could be reading this book instead.

Release It! is one of the most important books I think programmers can read, easily as important as the oft-cited classics like The Pragmatic Programmer or the GoF book. Release It! isn't about writing super-spiffy code, or object-oriented design, but it should drastically affect how professional programmers write their code. It focuses on what engineers need to do to get their software into a state where it can actually be deployed safely in a production environment. It covers patterns and antipatterns to support (or subvert) stability as well as capacity, and the section of the book covering this is simply excellent. But then it goes beyond that to also discuss Operational enablement. Even if you're not into DevOps, and don't want to really be involved in DevOps work, this book gives you the tools and tips to do what aspect of DevOps is the purview of pure developers.

Nobody who writes production enterprise software should write another line of code until they read this book. I honestly can't give it enough of a glowing endorsement. Like any other patterns book, a great deal of it will be familiar to people who have been in the industry for a while, and have come up with (or encountered) its principles on their own. But I guarantee, if there's a single thing in the book you haven't seen before, it'll be worth reading the entire book, pretty much every section is gold.

My only complaint about it is that Michael Nygard has a tendency to go on randomish tangents, especially when talking about "case studies," which generally come off as the author trying too hard to convince the reader he knows what he's talking about. A lot of these stories are about things he personally encountered in his career, and how he fixed them, and each one has a weird arrogant quality, it's a little offputting to be honest, like Nygard is almost bragging. I get that these sections underscore the value of the ideas in the book, but there's something about how they're written that comes off boastful, or irritating in a way I can't quite put into words. What's more, the book STARTS with a particularly long one of these kinds of stories, making it kind of tough to get into the book. I actually tried to read it years ago and gave up on it very early. Only recently did I decide to revisit it and keep pushing forward (based on a co-worker's recommendation) and I'm extremely glad I did.

I highly, highly recommend picking up this book and reading it cover to cover. It's a struggle at first due to the unfortunate decision to start it off with one of the most annoying sections of the book, but I implore you to power through it and keep reading. It's worth it.

Jelena K says

Perfect reading! Can't wait for the updated version of this book.

Andreea Lucau says

You can tell from the first use case the writer worked with big websites and using Java. Still, the book is full

of useful advice on how to design software projects in terms in scalability, transparency, adaptability and ease of troubleshoot. I enjoyed the style - the examples are well chosen and the level of details is not to deep, just enough to explain why some decisions are better than the others and how to apply good judgement when needed.

Roman Pichlík says

Even almost seven years after publishing the book is a source of inspiration in designing production friendly software. I wish i could read the book three years ago. It would safe few sleepless nights to me and my colleagues. The book would deserve to being extend about e.g. Cloud, Software As A Service and even DevOps since they are key change drivers in release/deployment process nowadays. Anyway worth reading and thumbs up.

Michael Koltsov says

There's a relatively short list of books I would like to keep on my desk. Most often those books are references and a composition of famous quotes. After I've read this chap I'd like to have it on my work desk at any moment.

This book is a perfect mix of lots of useful technical insights, practices and recommendations got from the author's hard-earned experience combined with some of the soft-skills you need to make your software and its maintenance (which as the author states costs more than the initial 1.0 version) as smooth as possible with as much of interrupted sleep as you could possibly get.

The book is definitely outdated, some of the references to particular technologies look odd and obvious (if not even funny). Nevertheless, I will put this book in one row with the "SRE book" & "Project Phoenix" as it combines them both.

My score is 5/5

Aleksey says

This book is a rare gem. It is full of valuable insights and is written in a very good language. Which makes this book not only valuable source of information but also a pleasure to read.
I would set 10 starts rating out of 5 possible if I could.

Definetely recommend it to any software developer or system engineer.

Sergey Shishkin says

This book is a battle proven must read for any software engineer. Even after 7 years since the book has been published, pretty much every advice in it remains valid and relevant. The idea of the operations database has

found market confirmation in products like Splunk. The Curcuit Breaker pattern has ever since been implemented numerous times in various OSS libraries.

A seasoned developer would have probably been learned some of the advice the hard way. Nonetheless I've picked up a lot of wisdom from the book, liked Michael's storytelling and appreciated his very broad perspective.

Rod Hilton says

This remains one of the most important books software engineers can read. The second edition is even better than the first, updated to fix a lot of the "outdated" criticisms the first book gets, incorporating the modern DevOps movement, microservices, and modern technologies used in software engineering.

I really just can't say enough about this book. It's required reading. If you're responsible for code that runs on networked production systems, failing to read this book should be a fireable offense. Skipping "Release It!" is professional negligence. Stop what you're doing and read this before shipping another line of code.

Release It! is all about how to build cynical software, and once you start down that path you find that you can no longer think any other way. This book changes you and your career, it's just phenomenal. Even if you think you know everything in it because the patterns and practices it describes have become widespread, it's still worth reading.

The second edition fixes - or at least somewhat improves - every minor complaint I had about the first edition, and reorganizes the information, adds a lot of great new sections, and removes outdated cruft. It's superior to the first edition in every way, and that was a 5-star book for me.

Sergey Teplyakov says

I've been working on project that heavily used clouds and high availability for relatively short period of time but even that experience helped me to appreciate this book.

The book predates all the dev-ops hype, but still gives you tons of suggestions how to build a robust, scalable and easy-to-understand-when-something-goes-wrong application: think about failure, every possible component WILL fail in production. Every possible 'joint' like external system interaction will be broken. Every possible and impossible situation will occur and you should be prepared for that: not by trying eliminate it, but by accepting that disaster will happen.

Some of the advises are a bit outdated (but look at the title, the book is from 2007!), and some of them are less clear that I wanted to, but overall the book is helpful.

Mitchell says

This was a discussion book at my current company. As a starting point for conversation, it worked well

enough. It definitely had some challenges. Writing a software related book almost implies specific technologies. But specifying these technologies almost immediately makes the book out of date, as in this one. But only dealing in concepts makes the book impenetrable. But just rambling your way through chapters just makes it annoying. This book was uneven, but it championed better logging and metrics, so how bad could it be?

Lino says

It's on overview of how systems break in production and how to avoid it. The author manages to take a pretty dry subject and produce a book that is very easy to read.

Now, it's pretty dated in a few places. It's from 2007, when the term DevOps wasn't even widely used yet. At the same time it's interesting to see how a lot of it still holds.

There are too many references to JVM-speficic issues. I don't think this ruins the experience for readers working outside the JVM, but it's strange seeing casual references (garbage collection tuning, permgen size, JSP, JMX, etc) in several places while the book title/subtitle/cover/description say nothing about it. Perhaps that's from a time when coding for enterprise meant exclusively Java.
