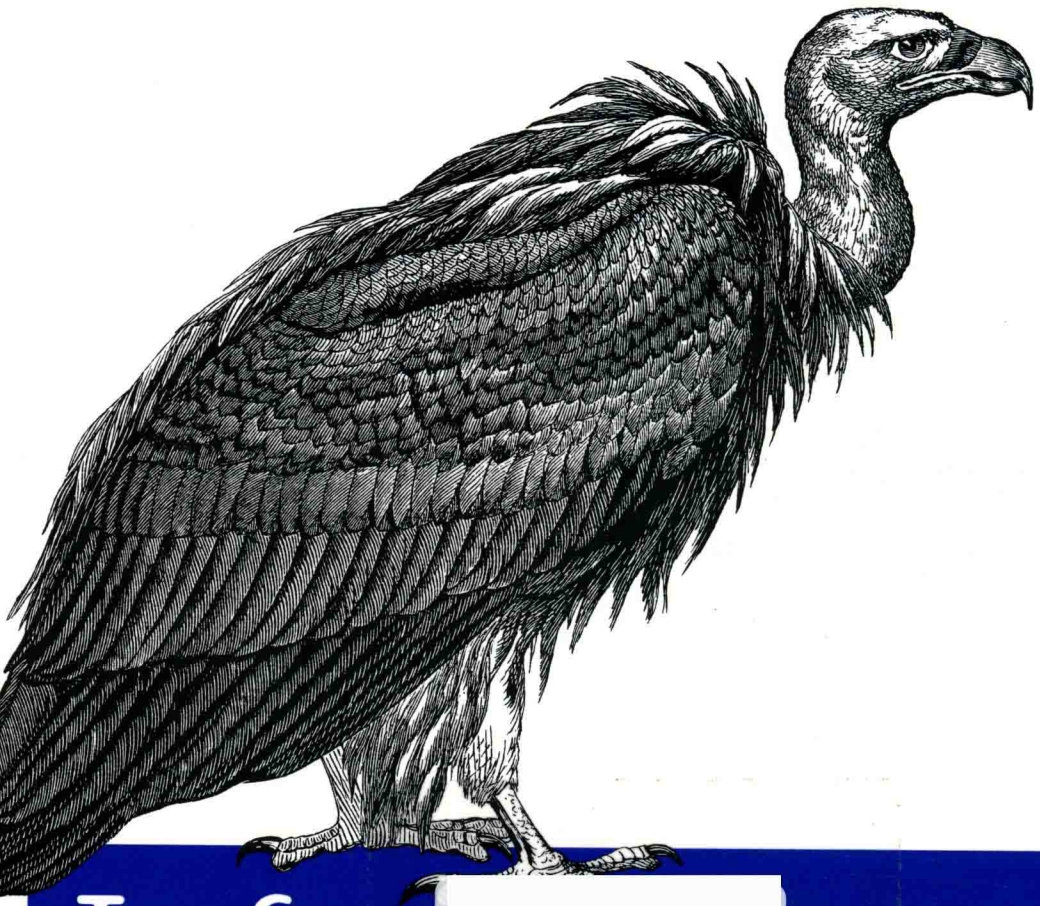
O'REILLY



Infrastructure as Code

基础设施即代码（影印版）

東南大學出版社

Kief Morris 著

基础设施即代码（影印版）

Infrastructure as Code

Kief Morris 著

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

O'Reilly Media, Inc.授权东南大学出版社出版

南京 东南大学出版社

图书在版编目(CIP)数据

基础设施即代码:英文/(美)基弗·莫里斯(Kief Morris)著. —影印本. —南京:东南大学出版社,2018.1

书名原文:Infrastructure as Code

ISBN 978-7-5641-7295-4

Ⅰ.①基… Ⅱ.①基… Ⅲ.①软件开发—英文 Ⅳ.①TP311.52

中国版本图书馆 CIP 数据核字(2017)248389 号

图字:10-2017-349 号

基础设施即代码(影印版)

出版发行:东南大学出版社

地　　址:南京四牌楼 2 号　　邮编:210096

出 版 人:江建中

网　　址:http://www.seupress.com

电子邮件:press@seupress.com

印　　刷:常州市武进第三印刷有限公司

开　　本:787 毫米×980 毫米　　16 开本

印　　张:22.75

字　　数:446 千字

版　　次:2018 年 1 月第 1 版

印　　次:2018 年 1 月第 1 次印刷

书　　号:ISBN 978-7-5641-7295-4

定　　价:94.00 元

本社图书若有印装质量问题,请直接与营销部联系。电话(传真):025-83791830

Preface

Infrastructure and software development teams are increasingly building and managing infrastructure using automated tools that have been described as "infrastructure as code." These tools expect users to define their servers, networking, and other elements of an infrastructure in files modeled after software source code. The tools then compile and interpret these files to decide what action to take.

This class of tool has grown naturally with the DevOps movement.[1] The DevOps movement is mainly about culture and collaboration between software developers and software operations people. Tooling that manages infrastructure based on a software development paradigm has helped bring these communities together.

Managing infrastructure as code is very different from classic infrastructure management. I've met many teams who have struggled to work out how to make this shift. But ideas, patterns, and practices for using these tools effectively have been scattered across conference talks, blog posts, and articles. I've been waiting for someone to write a book to pull these ideas together into a single place. I haven't seen any sign of this, so finally took matters into my own hands. You're now reading the results of this effort!

How I Learned to Stop Worrying and to Love the Cloud

I set up my first server, a dialup BBS,[2] in 1992. This led to Unix system administration and then to building and running hosted software systems (before we called it SaaS, aka "Software as a Service") for various companies, from startups to enterprises.

1 Andrew Clay Shafer and Patrick Debois triggered the DevOps movement with a talk at the Agile 2008 conference (*http://www.jedi.be/presentations/agile-infrastructure-agile-2008.pdf*). The movement grew, mainly driven by the series of DevOpsDays (*http://www.devopsdays.org/*) conferences organized by Debois.

2 A BBS is a bulletin board system (*https://en.wikipedia.org/wiki/Bulletin_board_system*).

I've been on a journey to infrastructure as code the entire time, before I'd ever heard the term.

Things came to a head with virtualization. The story of my stumbling adoption of virtualization and the cloud may be familiar, and it illustrates the role that infrastructure as code has to play in modern IT operations.

My First Virtual Server Farm

I was thrilled when my team got the budget to buy a pair of beefy HP rack servers and licenses for VMware ESX Server back in 2007.

We had in our office's server racks around 20 1U and 2U servers named after fruits (Linux servers) and berries (Windows database servers) running test environments for our development teams. Stretching these servers to test various releases, branches, and high-priority, proof-of-concept applications was a way of life. Network services like DNS, file servers, and email were crammed onto servers running multiple application instances, web servers, and database servers.

So we were sure these new virtual servers would change our lives. We could cleanly split each of these services onto its own virtual machine (VM), and the ESX hypervisor software would help us to squeeze the most out of the multicore server machines and gobs of RAM we'd allocated. We could easily duplicate servers to create new environments and archive those servers that weren't needed onto disk, confident they could be restored in the future if needed.

Those servers did change our lives. But although many of our old problems went away, we discovered new ones, and we had to learn completely different ways of thinking about our infrastructure.

Virtualization made creating and managing servers much easier. The flip side of this was that we ended up creating far more servers than we could have imagined. The product and marketing people were delighted that we could give them a new environment to demo things in well under a day, rather than need them to find money in the budget and then wait a few weeks for us to order and set up hardware servers.

The Sorcerer's Apprentice

A year later, we were running well over 100 VMs and counting. We were well underway with virtualizing our production servers and experimenting with Amazon's new cloud hosting service. The benefits virtualization had brought to the business people meant we had money for more ESX servers and for shiny SAN devices to feed the surprising appetite our infrastructure had for storage.

But we found ourselves a bit like Mickey Mouse in "The Sorcerer's Apprentice" from *Fantasia*. We spawned virtual servers, then more, then even more. They over-

whelmed us. When something broke, we tracked down the VM and fixed whatever was wrong with it, but we couldn't keep track of what changes we'd made where.

> Well, a perfect hit!
> See how he is split!
> Now there's hope for me,
> and I can breathe free!
>
> Woe is me! Both pieces
> come to life anew,
> now, to do my bidding
> I have servants two!
> Help me, O great powers!
> Please, I'm begging you!
>
> —Excerpted from Brigitte Dubiel's translation of "Der Zauberlehrling" ("The Sorcerer's Apprentice") by Johann Wolfgang von Goethe

As new updates to operating systems, web servers, app servers, database servers, JVMs, and various other software packages came out, we would struggle to install them across all of our systems. We would apply them successfully to some servers, but on others the upgrades broke things, and we didn't have time to stomp out every incompatibility. Over time, we ended up with many combinations of versions of things strewn across hundreds of servers.

We had been using configuration automation software even before we virtualized, which should have helped with these issues. I had used CFEngine in previous companies, and when I started this team, I tried a new tool called Puppet. Later, when spiking out ideas for an AWS infrastructure, my colleague Andrew introduced Chef. All of these tools were useful, but particularly in the early days, they didn't get us out of the quagmire of wildly different servers.

The problem was that, although Puppet (and Chef and the others) should have been set up and left running unattended across all of our servers, we couldn't trust it. Our servers were just too different. We would write manifests to configure and manage a particular application server. But when we ran it against another, theoretically similar app server, we found that different versions of Java, application software, and OS components would cause the Puppet run to fail, or worse, break the application server.

So we ended up using Puppet ad hoc. We could safely run it against new VMs, although we might need to make some tweaks after it ran. We would write manifests for a specific task and then run them against servers one at a time, carefully checking the result and making fixes as needed.

So configuration automation was a useful aid, somewhat better than shell scripts, but the way we used it didn't save us from our sprawl of inconsistent servers.

Cloud from Scratch

Things changed when we began moving things onto the cloud. The technology itself wasn't what improved things; we could have done the same thing with our own VMware servers. But because we were starting fresh, we adopted new ways of managing servers based on what we had learned with our virtualized farm and on what we were reading and hearing from IT Ops teams at companies like Flickr, Etsy, and Netflix. We baked these new ideas into the way we managed services as we migrated them onto the cloud.

The key idea of our new approach was that every server could be automatically rebuilt from scratch, and our configuration tooling would run continuously, not ad hoc. Every server added into our new infrastructure would fall under this approach. If automation broke on some edge case, we would either change the automation to handle it, or else fix the design of the service so it was no longer an edge case.

The new regime wasn't painless. We had to learn new habits, and we had to find ways of coping with the challenges of a highly automated infrastructure. As the members of the team moved on to other organizations and got involved with communities such as DevOpsDays, we learned and grew. Over time, we reached the point where we were habitually working with automated infrastructures with hundreds of servers, with much less effort and headache than we had been in our "Sorcerer's Apprentice" days.

Joining ThoughtWorks was an eye-opener for me. The development teams I worked with were passionate about using XP engineering practices like test-driven development (*http://martinfowler.com/bliki/TestDrivenDevelopment.html*) (TDD), continuous integration (*http://www.martinfowler.com/articles/continuousIntegration.html*) (CI) and continuous delivery (*http://martinfowler.com/books/continuousDeliv ery.html*) (CD). Because I had already learned to manage infrastructure scripts and configuration files in source control systems, it was natural to apply these rigorous development and testing approaches to them.

Working with ThoughtWorks has also brought me into contact with many IT operations teams, most of whom are using virtualization, cloud, and automation tools to handle a variety of challenges. Working with them to share and learn new ideas and techniques has been a fantastic experience.

Why I'm Writing This Book

I've run across many teams who are in the same place I was a few years ago: people who are using cloud, virtualization, and automation tools but haven't got it all running as smoothly as they know they could.

Much of the challenge is time. Day-to-day life for system administrators is coping with a never-ending flow of critical work. Fighting fires, fixing problems, and setting up new business-critical projects doesn't leave much time to work on the fundamental improvements that will make the routine work easier.

My hope is that this book provides a practical vision for how to manage IT infrastructure, with techniques and patterns that teams can try and use. I will avoid the details of configuring and using specific tools so that the content will be useful for working with different tools, including ones that may not exist yet. Meanwhile, I will use examples from existing tools to illustrate points I make.

The infrastructure-as-code approach is essential for managing cloud infrastructure of any real scale or complexity, but it's not exclusive to organizations using public cloud providers. The techniques and practices in this book have proven effective in virtualized environments and even for bare-metal servers that aren't virtualized.

Infrastructure as Code is one of the cornerstones of DevOps. It is the "A" in "CAMS" (*http://itrevolution.com/devops-culture-part-1/*): culture, automation, measurement, and sharing.

Who This Book Is For

This book is for people who work with IT infrastructure, particularly at the level of managing servers and collections of servers. You may be a system administrator, infrastructure engineer, team lead, architect, or a manager with technical interest. You might also be a software developer who wants to build and use infrastructure.

I'm assuming you have some exposure to virtualization or IaaS (Infrastructure as a Service) cloud, so you know how to create a server, and the concepts of configuring operating systems. You've probably at least played with configuration automation software like Ansible, Chef, or Puppet.

While this book may introduce some readers to infrastructure as code, I hope it will also be interesting to people who work this way already and a vehicle through which to share ideas and start conversations about how to do it even better.

What Tools Are Covered

This book doesn't offer instructions in using specific scripting languages or tools. There are code examples from specific tools, but these are intended to illustrate concepts and approaches, rather than to provide instruction. This book should be helpful to you regardless of whether you use Chef on OpenStack, Puppet on AWS, Ansible on bare metal, or a completely different stack.

The specific tools that I do mention are ones which I'm aware of, and which seem to have a certain amount of traction in the field. But this is a constantly changing landscape, and there are plenty of other relevant tools.

The tools I use in examples tend to be ones with which I am familiar enough to write examples that demonstrate the point I'm trying to make. For example, I use Terraform for examples of infrastructure definitions because it has a nice, clean syntax, and I've used it on multiple projects. Many of my examples use Amazon's AWS cloud platform because it is likely to be the most familiar to readers.

How to Read This Book

Read Chapter 1, or at least skim it, to understand the terms this book uses and the principles this book advocates. You can then use this to decide which parts of the book to focus on.

If you're new to this kind of automation, cloud, and infrastructure orchestration tooling, then you'll want to focus on Part I, and then move on to Part II. Get comfortable with those topics before proceeding to Part III.

If you've been using the types of automation tools described here, but don't feel like you're using them the way they're intended after reading Chapter 1, then you may want to skip or skim the rest of Part I. Focus on Part II, which describes ways of using dynamic and automated infrastructure that align with the principles outlined in Chapter 1.

If you're comfortable with the dynamic infrastructure and automation approaches described in Chapter 1, then you may want to skim Parts I and II and focus on Part III, which gets more deeply into the infrastructure management regime: architectural approaches as well as team workflow.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

Constant width italic
Shows text that should be replaced with user-supplied values or by values determined by context.

This element signifies a tip or suggestion.

This element signifies a general note.

This element indicates a warning or caution.

Safari® Books Online

Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals.

Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/infrastructureAsCode_1e*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

When I started working on this book, I assumed the result would be a product that was entirely my own. But in the end, it's just the opposite: this book is the product of the ideas, thoughts, opinions, and experiences of far more people than I could have imagined. I have probably mangled, oversimplified, and misrepresented this input. But this book would not exist as it is without these contributions.

The people I worked with on the University of Tennessee's computer science lab staff taught me my Unix chops and inducted me into the culture. Chad Mynhier was particularly responsible for hooking me into the Unix world. He explained why I could no longer `cd` into my own home area after I had experimented with the `chmod` command.

Working with a series of companies, including Syzygy, Vizyon, Cellectivity, and the Map of Medicine, gave me the arena to develop my understanding and to learn how to apply infrastructure automation to real-world business and user problems. I owe much to the many good people of those organizations. I'll specifically call out Jonathan Waywell and Ketan Patel for their unending support and encouragement,

Andrew Fulcher for quickly learning what I had to teach and then teaching me even more, and Nat Billington for his inspiration.

This book would truly never have happened without my current home, ThoughtWorks. I've learned much about the ideas in this book and how to think about and explain them to other people, thanks to more than five years of exposure to a dizzying number of organizations of various sizes, sectors, and technologies. The endless curiosity of my colleagues, past and present, and their heartfelt drive to improve our industry and the experiences of people in it, continually challenges me.

Generous support and encouragement from ThoughtWorks as an organization has been vital for this book, especially as my energy flagged with the finish line coming into view. Chris Murphy, Dave Elliman, Maneesh Subherwal, and Suzi Edwards-Alexander are a few among many who have made this more than a personal project for me.

An incomplete list of past and present ThoughtWorks colleagues who have gone out of their way to contribute suggestions, feedback, and other support include: Abigail Bangser, Ashok Subramanian, Barry O'Reilly, Ben Butler-Cole, Chris Bird (DevOops!), Chris Ford, David Farley, Gurpreet Luthra, Inny So, Jason Yip, Jim Gumbley, Kesha Stickland, Marco Abis, Nassos Antoniou, Paul Hammant, Peter Gillard-Moss, Peter Staples, Philip Potter, Rafael Gomes, Sam Newman, Simon Brunning, Tom Duckering, Venu Murthy, and Vijay Raghavan Aravamudhan.

Martin Fowler has given me tremendous encouragement and practical support in writing this book. He gave me far more time than I could have asked, thoroughly reviewing the manuscript several times. Martin gave me detailed, useful feedback and advice based on his considerable experience of organizing and conveying technical concepts. He has been a true champion of this book.

My colleague Rong Tang created the images for this book. She was extremely patient with my muddled explanations of what I wanted. Any failings of clarity or consistency is down to me, but the great look is a credit to her.

The folks behind the long-dormant Infrastructures.org (*http://www.infrastructures.org/index.shtml*) exposed me to the ideas of Infrastructure as Code before the term existed.[3]

I owe a great debt to the people of the DevOpsDays community, who are collectively responsible for bringing the ideas of DevOps and Infrastructure as Code to prominence. Regardless of who actually coined the term "Infrastructure as Code," people like Adam Jacob, Andrew Clay-Shafer, John Allspaw, John Willis, Luke Kaines, Mark

3 Sadly, as of early 2016, Infrastructures.org hasn't been updated since 2007.

Burgess, and of course, Patrick Debois ("The Godfather of DevOps") have given me inspiration and many great ideas.

A number of other people have given me feedback and advice on earlier drafts of this book, including Axel Fontaine, Jon Cowie, Jose Maria San Jose Juarez, Marcos Hermida, and Matt Jones. I also want to thank Kent Spillner, although I don't recall why. Ovine and dairy. Putney Bridge.

Last, but the furthest from least, everlasting love to Ozlem and Erel, who endured my obsession with this book.

Table of Contents