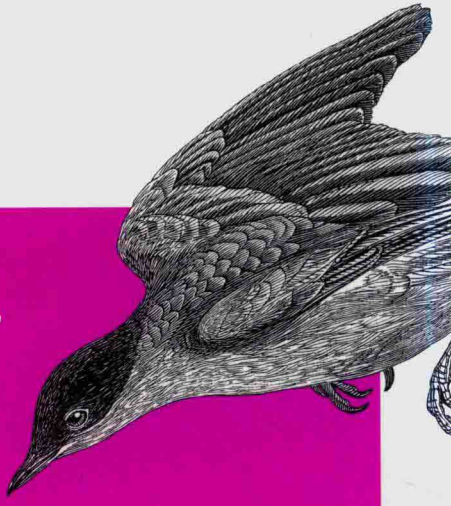
O'REILLY®



Git for Teams

用于团队协作的Git（影印版）

Emma Jane Hogbin Westby 著

東南大學出版社

用于团队协作的Git（影印版）

Git for Teams

Emma Jane Hogbin Westby 著

Beijing • Cambridge • Farnham • Köln • 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

用于团队协作的 Git:英文/(美)艾玛·简·霍格宾·韦斯特比(Emma Jane Hogbin Westby)著. —影印本. —南京:东南大学出版社,2017.1

书名原文:Git for Teams

ISBN 978-7-5641-6867-4

Ⅰ.①用… Ⅱ.①艾… Ⅲ.①软件工具—程序设计—英文 Ⅳ.①TP311.561

中国版本图书馆 CIP 数据核字(2016)第 294339 号

图字:10-2015-255 号

用于团队协作的 Git(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:22.25
字　　数:436 千字
版　　次:2017 年 1 月第 1 版
印　　次:2017 年 1 月第 1 次印刷
书　　号:ISBN 978-7-5641-6867-4
定　　价:82.00 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

Foreword

At the time of Git's inception, the Linux kernel development had used the proprietary version control system BitKeeper for several years, with great success. But there was one problem: some Linux developers took exception with the proprietary nature of their version control system and what ensued was an epic flame war. Out of this conflict, the free BitKeeper license for Linux developers was revoked, and Git was born. Linus Torvalds himself took two weeks off from working on Linux, originally to search for a replacement for BitKeeper. Failing to find any that met his criteria, he instead wrote the first, very rudimentary version of what we now call Git: tiny programs cobbled together with shell scripts, Unix style. An ironic twist is that the distributed nature of Git was implemented using rsync, a tool which in turn had been developed by the very Linux developer who triggered the fallout with BitKeeper.

As to myself, I was fascinated by the simplicity of Git's data structures and got drawn in early on, first by working on Git's portability, then on more and more general improvements, including the invention of the "interactive rebase" (sorry for the name!), and ultimately maintaining the Windows port of Git. For the past 10 years, I used Git almost daily as a life science researcher, as part of different teams ranging from being the designated coder in interdisciplinary projects to leading highly distributed Open Source projects.

My first contact with Emma was at the Git Merge conference in Paris celebrating Git's 10th birthday, where she gave a compelling talk titled "Teaching People Git" (*http://bit.ly/teaching-people-git*). This talk left quite the impression on me, reflecting Emma's broad skill set and experience in teaching and project management.

Reading *Git for Teams*, I learned a lot from its unique perspective that emphasizes how Git can facilitate teamwork. It sounds so simple, but all those years, I had been focusing on technical details, and I had been teaching Git in what must be one of the most frustrating ways: from the ground up. By focusing on workflows and interactions between roles, *Git for Teams* guides you, the reader, to understand your exact

needs within your particular projects. Equipped with this knowledge, you will then learn the fun part: how to use Git to best support your needs.

Just like her talk, Emma's writing style is very enjoyable, making this book both educative and fun to read. It gave me valuable insights into my daily work. Whatever your role in *your* daily work, let this book be more than just a manual. Explore the different ways teams can work together, the ways a modern version control system can help moving projects forward, and let it inspire you to unleash the full power of Git to support you in what you want to do.

—Dr. Johannes Schindelin
Git for Windows maintainer
August 2015
Cologne, Germany

Foreword

It is difficult to overstate the importance of version control.

I believe that it is as important as the inventions of the chalkboard and of the book for multiplying the power of people to create together.

Over my career, I have watched the approach to version control systems in software development advance from resistance to ubiquity, and have watched the underlying technology make quantum jumps, each jump accelerating the value of the work we create together and the speed at which we create it. We are doing more, faster, with more people.

The latest jump, exemplified by Git, imposes almost no arbitrary constraints on a workflow. Thus, we have to discover and share the workflows that suit our people and our organizations, instead of living with past awkward workflows that suited our machines. Some of those workflows are explored in this book. I'm sure that more will be discovered in the future.

It is also difficult to overstate the importance and difficulty of education. Not merely memorizing facts or merely training tasks, but the deeper kind of education: how to think a certain way, to understand why to think that way, and how to share those thoughts with someone else.

Using a version control system properly is a way to think: to structure, remember, and share thoughts, at the level of depth and rigor demanded by the exhausting craft of writing software. Without that understanding, using Git will be, at best, "magical incantations", used by rote, and full of unknown dangers. With that understanding, Git can become almost invisible, leaving instead the patterns of working up the intricate spells of symbols that are the magic of software.

This book will help you to educate yourself, to gain that understanding, and to do that work.

—Mark Atwood
Director of Open Source Engagement,
Hewlett-Packard Company
August 2015
Seattle, WA

Preface

For nearly two decades, I've been working on teams of one or more in a distributed fashion. My first paid job as a web developer was in the mid-'90s. At the time, I maintained versions of my files by simply changing the names to denote a new version. My workspace was littered with files that had unusual extensions; *v4.old-er.bak* was an all too common sight. I wasn't able to easily track my work. On one project, which was a particularly challenging one for me, I resorted to the copyediting techniques I used for my essays: I'd print out the Perl scripts I was working on, and put the pages into a ring binder. I'd then mark up my scripts with different colors of pen and transcribe the changes back into my text editor. (I *wish* I had photos to share.) I tracked versions by flipping through the binder to find previous versions of the script. I had no idea how to set up an actual version control system (VCS), but I was obsessive about not losing good work if a refactoring failed.

When I started working with other developers, either for open source projects or client work, I was never the first developer on the scene and there was always some kind of version control in place by the time I got there—typically Concurrent Versions System (CVS). It wasn't the easiest thing to use, but compared to my ring binder of changes, it was definitely more scalable for the distributed teams that I worked with. Very quickly I came to value the commit messages, and the ease of being able to review the work others were doing. It motivated me to watch others commit their work to the repository. I didn't want others to think I was slacking off!

Meanwhile, I'd been teaching web development at a couple of different community colleges. In 2004, I had my first opportunity to teach version control in a year-long program designed by Bernie Monette, at Humber College. The class was split into several groups. In the first semester, the students sketched out a development plan for a website. In the second semester, the teams were mixed up, and the new teams were asked to build the site described by the previous team. In the third and final semester, the groups were shuffled again, and the final task was to do bug fixing and quality assurance on the built site. Each team was forced to use version control to track their work. The students, who had no prior programming experience, weren't thrilled with

having to use version control because they felt it got in the way of doing work. But it also made it easier because they never accidentally overwrote their classmates' work. It taught me a lot about how to motivate people to use a tool that didn't feel like it was core to the job at hand.

In the decade since that class, I've learned a lot about how to teach version control, and a lot about best practices in adult education. This book is the culmination of what I've learned about how to work efficiently with others when using version control. I encourage you throughout the book to do whatever is best for your team. There are no Git police who will show up at your door and tell you "you're doing it wrong." That said, wherever I can, I explain to you "the Git way" of doing things so that you have some guidance on where you might want to start with your team, or what you might want to grow into. Using "common" ways of working will help you onboard others who've previously used similar techniques.

This book won't be for everyone. This book is for people who love to plan a route, and then follow the clearly defined road ahead. My hope is that, if nothing else, this book helps to fill the gaps that have been missing in Git resources to date. It's not so much a manual for the software as a manual for how teams collaborate. If your team of one (or more) finds bits of this book confusing, I hope you'll let me know (*emma@gitforteams.com*); and if you find it useful, I hope you'll let the world know.

Acknowledgments

Several years ago, in a little bar off the side of a graveyard in Prague, Carl Wiedemann indulged my questions about Git. Thank you, Carl. Your enthusiasm motivated me to convert my frustration with Git into resources to help others avoid the painful process I'd experienced when learning Git.

I had the wonderful fortune to work with Joe Shindelar at my first job-job after a decade of self-employment. Joe, your passion for excellence has raised the bar for my own work. I am grateful for your patience and leadership. This book was born out of the conversations we had about leadership, team structures, and the Git documentation we created for the Drupalize.Me team. Thank you.

O'Reilly found the excellent Christophe Portneuve to serve as one of my tech reviewers. Christophe, thank you for your patience as I worked through the first few chapters. Your feedback was invaluable. I am grateful for the conversation we had at Git Merge, which helped me to clarify the concepts I use in this book—I had lofty goals of transforming the way people learn Git. I hope this book has become a resource you will be proud to have been a part of.

Bernie Monette, Martin Poole, Drew McLelland: you gave me a platform to refine my understanding of version control through your own projects.

Lorna Jane Mitchell, your cheerleading is tireless. Thank you for sharing your own work on Git. It has inspired me to raise the bar even higher.

Much of this book was fueled by 200 Degrees Coffee, a Nottingham-based roaster. My beverage of choice is a flat white served from 200 Degrees Café, or Divine Coffee at the Galleries of Justice. Thanks for providing an escape and letting me stay as long as I needed to.

To the O'Reilly family: you have been superb at handling all of my requests (and missed deadlines). Thank you Rachel, Heather, Robert, Colleen, Brian, Josh, Rebecca, Kim, and the countless others who worked behind the scenes to make this book happen.

To the core Git community: thank you for welcoming me with open arms at Git Merge in 2015. You embraced my rant from the stage about exploring new ways of teaching Git. You took my suggestions to heart, and made improvements to the Git experience. I am looking forward to participating more in the wonderful community you have been quietly nurturing.

Thank you also to my community of reviewers: Diane Tani, Novella Chiechi, Amy Brown, Blake Winton, Stuart Langridge, Stewart Russell, Dave Hammond, John Wynstra, Chris Tankersley, Mike Anello, Piotr Sipika, Nancy Deschenes, Robert Day, Dave Hammond, Sébastien Simard, Tobias Hiep, Nick Gard, Christopher Maneu, Johannes Schindelin, Edward Thomson, matt j. sorenson, Douwe Maan, Sytse Sijbrandij, Rob Allen, Steven Pears, Laura Lemay. Your feedback was invaluable.

To my partner, James Westby: thank you for patiently waiting as I finish *just one last thing*. This book would not exist without your support and encouragement.

Introduction

The book takes a people-first approach to version control. I don't start with a history of Git; instead, I begin with a 10,000-foot view of how teams can work together. Then we will circle our way into the commands, ensuring you always know the *why* behind the command you're about to type. Sometimes you can save your future self time (and confusion) by adopting specific routines or workflows. These explanations give you a holistic understanding of how your work today affects your work tomorrow—and hopefully make sense out of the near-religious insistence by some people on why they use Git the way they do.

Part I will be most useful to managers, technical team leads, chief technology officers, project managers, and technical project managers who need to outline a workflow for their team.

Good technology comes from great teams. In Chapter 1, you will learn about the dynamics of creating a great team. By the end of this chapter, you will be able to identify roles within a team; plan highly effective meetings; recognize key phrases from people who are out of sync with what your team needs; and apply strategies that will help you to cultivate empathy and trust within your team.

Set the expectations early for the type of project you are running. In Chapter 2, you will learn about different permissions strategies used to grant and deny access to a Git repository. Should team members be allowed to save their work to the repository without a review, or is it more of a trust and be trusted scenario? Both systems have their merits, and you'll learn about them in this chapter.

Make the intentions of your work clear. In Git, you will separate streams of work with branches. Chapter 3 shows you how to separate each of the ideas your team is working on through the use of these branches. Of course, you will also need to know how to bring these disparate pieces of work into a unified piece of software. This chapter covers some of the more common branching strategies, including GitFlow.

Write the documentation today that will help you work more efficiently tomorrow. Chapter 4 is the culmination of all the ideas in Part I. You will learn how to create your own documentation and walk through the process of creating and deploying a simple software product.

Part II will be most useful for developers. This is where (finally!) you will get to learn how all those Git commands are actually supposed to work. If you're impatient and want to get your hands on code, you'll do well to skip ahead to Part II and then once you've completed it, go back and read Part I.

Ground yourself in practical skills. Chapter 5 covers the basics of distributed version control. In this chapter you will learn how to create repositories, and track your changes to files locally through commits, branches, and tags.

Learn to recover from your mistakes. Chapter 6 allows you to explore history revisionism. This chapter covers how to amend commits, remove commits from your time line, and rebase your work.

Expand your team to be inclusive of others. Now that you're a master of history in your own repository, it's time to begin collaborating with others. Chapter 7 will show you how to track remote changes, upload your code to a shared repository, and update your local repository with the updates from others.

Through peer review, share the glory and the responsibility of a job well done. In Chapter 8, you will learn about the process for conducting code reviews with your team. We'll also cover the commands for a common reviewing methodology, along with suggestions on how to customize it for your team.

Investigate history; it holds the answer to the problem you're facing. In Chapter 9, you will learn some advanced methods to track down bugs using Git. Don't be scared, though! The commands we'll be using are no more difficult than anything else you've done to date.

Finally, Part III gives the how-to for a few of the popular code hosting systems on the market today. It is aimed at both managers and developers.

Through open collaboration we grow our community. Chapter 10 covers the mechanics of starting and maintaining an open source project on GitHub.

A team must have a repository of their own if they are to write good code. In Chapter 11, you will learn how to collaborate on private repositories. This chapter will be especially useful for those who want to set up a private repository but have extremely limited funds to pay for private teams on GitHub.

Good fences sometimes do make better neighbors. In Chapter 12, you will learn how to host your own instance of GitLab, and run projects through it. This is particularly useful for developers who are inside a firewall and cannot access public repositories on the Internet.

This book won't be for everyone. It will be especially frustrating for people who learn by poking at things and tinkering and exploring. This book, rather, is written for people who are a little afraid of things that go bump in the night.

Additional resources and larger versions of several of the flowcharts are available from the book's companion site (*http://gitforteams.com*).

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.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *http://gitforteams.com*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Git for Teams* by Emma Jane Hogbin Westby (O'Reilly). Copyright 2015 Emma Jane Hogbin Westby, 978-1-491-91118-1."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

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/git-for-teams*.

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*

Table of Contents