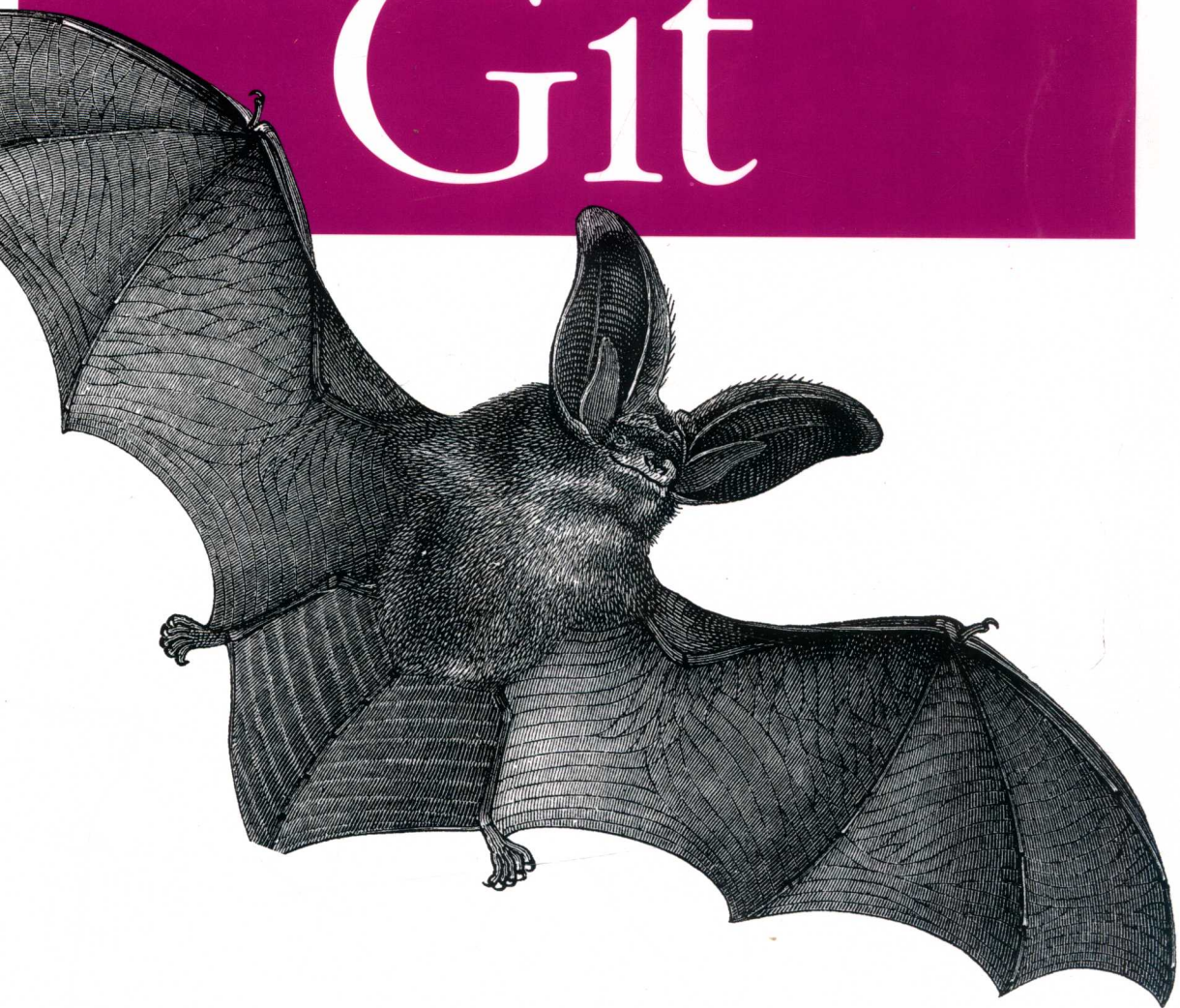


Git版本控制 (影印版)

第二版  
涵盖 GitHub

*Version Control with*

Git



O'REILLY®

东南大学出版社

*Jon Loeliger &  
Matthew McCullough 著*

第二版

# Git版本控制 (影印版)

## Version Control with Git

*Jon Loeliger & Matthew McCullough* 著

江苏理工学院图书馆  
藏书章

江苏理工学院图书馆



21294281

O'REILLY®

• Farnham • Köln • Sebastopol • Tokyo

a, 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

Git 版本控制: 第2版: 英文/(美)罗力格 (Loeliger, J.),  
(美)麦克库罗 (McCullough, M.)著. —影印本. —南京: 东  
南大学出版社, 2013.5

书名原文: Version Control with Git, 2E

ISBN 978-7-5641-4196-7

I. ① G… II. ① 罗… ② 麦… III. ① 软件工具—程序  
设计—英文 IV. ① TP311.56

中国版本图书馆 CIP 数据核字 (2013) 第 090231 号

江苏省版权局著作权合同登记

图字: 10-2013-122 号

©2012 by O'Reilly Media, Inc.

Reprint of the English Edition, jointly published by O'Reilly Media, Inc. and Southeast University Press,  
2013. Authorized reprint of the original English edition, 2013 O'Reilly Media, Inc., the owner of all rights  
to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 O'Reilly Media, Inc. 出版 2012。

英文影印版由东南大学出版社出版 2013。此影印版的出版和销售得到出版权和销售权的所有者——O'Reilly  
Media, Inc. 的许可。

版权所有, 未得书面许可, 本书的任何部分和全部不得以任何形式复制。

## Git 版本控制 第二版 (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](mailto:press@seupress.com)

印 刷: 扬中市印刷有限公司

开 本: 787 毫米 × 980 毫米 16 开本

印 张: 28.5

字 数: 558 千字

版 次: 2013 年 5 月第 1 版

印 次: 2013 年 5 月第 1 次印刷

书 号: ISBN 978-7-5641-4196-7

定 价: 62.00 元 (册)

---

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

## Audience

Although some familiarity with revision control systems will be good background material, a reader who is not familiar with any other system will still be able to learn enough about basic Git operations to be productive in a short while. More advanced readers should be able to gain insight into some of Git's internal design and thus master some of its more powerful techniques.

The main intended audience of this book should be familiar and comfortable with the Unix shell, basic shell commands, and general programming concepts.

## Assumed Framework

Almost all examples and discussions in this book assume the reader has a Unix-like system with a command-line interface. The author developed these examples on Debian and Ubuntu Linux environments. The examples should work under other environments, such as Mac OS X or Solaris, but the reader can expect slight variations.

A few examples require root access on machines where system operations are needed. Naturally, in such situations, you should have a clear understanding of the responsibilities of root access.

## Book Layout and Omissions

This book is organized as a progressive series of topics, each designed to build upon concepts introduced earlier. The first 11 chapters focus on concepts and operations that pertain to one repository. They form the foundation for more complex operations on multiple repositories covered in the final 10 chapters.

If you already have Git installed or have even used it briefly, then you may not need the introductory and installation information in the first two chapters, nor even the quick tour presented in the third chapter.



The concepts covered in Chapter 4 are essential for a firm grasp on Git's object model. They set the stage and prepare the reader for a clearer understanding of many of Git's more complex operations.

Chapters 5 through 11 cover various topics in more detail. Chapter 5 describes the index and file management. Chapters 6 and 10 discuss the fundamentals of making commits and working with them to form a solid line of development. Chapter 7 introduces branches so that you may manipulate several different lines of development from your one local repository. Chapter 8 explains how Git derives and presents "diffs."

Git provides a rich and powerful ability to join different branches of development. The basics of branch merging and resolving merge conflicts are covered in Chapter 9. A key insight into Git's model is to realize that all merging performed by Git happens in your local repository in the context of your current working directory. Chapters 10 and 11 expose some operations for altering, storing, tracking, and recovering daily development within your development repository.

The fundamentals of naming and exchanging data with another, remote repository are covered in Chapter 12. Once the basics of merging have been mastered, interacting with multiple repositories is shown to be a simple combination of an exchange step plus a merge step. The exchange step is the new concept covered in this chapter and the merge step is covered in Chapter 9.

Chapter 13 provides a more philosophical and abstract coverage of repository management "in the large." It also establishes a context for Chapter 14 to cover patch handling when direct exchange of repository information isn't possible using Git's native transfer protocols.

The next four chapters cover advanced topics of interest: the use of hooks (Chapter 15), combining projects and multiple repositories into a superproject (Chapter 16), and interacting with Subversion repositories (Chapter 17).

Chapters 19 and 20 provide some advanced examples and clever tips, tricks, and techniques that may help transform you into a true Git guru.

Finally, Chapter 21 introduces GitHub and explains how Git has enabled a creative, social development process around version control.

Git is still evolving rapidly because there is an active developer base. It's not that Git is so immature that you cannot use it for development; rather, ongoing refinements and user interface issues are being enhanced regularly. Even as this book was being written, Git evolved. Apologies if I was unable to keep up accurately.

I do not give the command `gitk` the complete coverage that it deserves. If you like graphical representations of the history within a repository, you should explore `gitk`. Other history visualization tools exist as well, but they are not covered here either. Nor am I able to cover a rapidly evolving and growing host of other Git-related tools. I'm not even able to cover all of Git's own core commands and options thoroughly in this book. Again, my apologies.

Perhaps, though, enough pointers, tips, and direction can be found here to inspire readers to do some of their own research and exploration!

## 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 icon signifies a useful hint or a tip.



This icon indicates a warning or caution.



This icon indicates a general note.

Furthermore, you should be familiar with basic shell commands to manipulate files and directories. Many examples will contain commands such as these to add or remove directories, copy files, or create simple files:

```
$ cp file.txt copy-of-file.txt
$ mkdir newdirectory
$ rm file
$ rmdir somedir
$ echo "Test line" > file
$ echo "Another line" >> file
```

Commands that need to be executed with root permissions appear as a `sudo` operation:

```
# Install the Git core package  
  
$ sudo apt-get install git-core
```

How you edit files or effect changes within your working directory is pretty much up to you. You should be familiar with a text editor. In this book, I'll denote the process of editing a file by either a direct comment or a pseudocommand:

```
# edit file.c to have some new text  
  
$ edit index.html
```

## Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book 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: "*Version Control with Git* by Jon Loeliger and Matthew McCullough. Copyright 2012 Jon Loeliger, 978-1-449-31638-9."

If you feel your use of code examples falls outside fair use or the permission given previously, feel free to contact us at [permissions@oreilly.com](mailto:permissions@oreilly.com).

## Safari® Books Online

**Safari®** Books Online ([www.safaribooksonline.com](http://www.safaribooksonline.com)) 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 product mixes and pricing programs for organizations, government agencies, and individuals. Subscribers 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 dozens 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://oreil.ly/VCWG2e>

To comment or ask technical questions about this book, send email to:

[bookquestions@oreilly.com](mailto: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

This work would not have been possible without the help of many other people. I'd like to thank Avery Pennarun for contributing substantial material to Chapters 15, 16, and 18. He also contributed some material to Chapters 4 and 9. His help was appreciated. I'd like to thank Matthew McCullough for the material in Chapters 17 and 21, assorted suggestions, and general advice. Martin Langhoff is paraphrased with permission for some repository publishing advice in Chapter 13, and Bart Massey's tip on keeping a file without tracking is also used with permission. I'd like to publicly thank those who took time to review the book at various stages: Robert P. J. Day, Alan Hasty, Paul Jimenez, Barton Massey, Tom Rix, Jamey Sharp, Sarah Sharp, Larry Streepy, Andy Wilcox, and Andy Wingo. Robert P. J. Day, thankfully, took the time to review both editions of the book front to back.

Also, I'd like to thank my wife Rhonda, and daughters Brandi and Heather, who provided moral support, gentle nudging, Pinot Noir, and the occasional grammar tip. And



Finally, I would like to thank the staff at O'Reilly as well as my editors, Andy Oram and Martin Streicher.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

---

# Table of Contents

<b>Preface .....</b>	<b>xi</b>
<b>1. Introduction .....</b>	<b>1</b>
Background .....	1
The Birth of Git .....	2
Precedents .....	4
Timeline .....	6
What's in a Name? .....	7
<b>2. Installing Git .....</b>	<b>9</b>
Using Linux Binary Distributions .....	9
Debian/Ubuntu .....	9
Other Binary Distributions .....	10
Obtaining a Source Release .....	11
Building and Installing .....	12
Installing Git on Windows .....	13
Installing the Cygwin Git Package .....	14
Installing Standalone Git (msysGit) .....	15
<b>3. Getting Started .....</b>	<b>19</b>
The Git Command Line .....	19
Quick Introduction to Using Git .....	21
Creating an Initial Repository .....	21
Adding a File to Your Repository .....	22
Configuring the Commit Author .....	24
Making Another Commit .....	24
Viewing Your Commits .....	25
Viewing Commit Differences .....	26
Removing and Renaming Files in Your Repository .....	26
Making a Copy of Your Repository .....	27
Configuration Files .....	28

Configuring an Alias	30
Inquiry	30
<b>4. Basic Git Concepts .....</b>	<b>31</b>
Basic Concepts	31
Repositories	31
Git Object Types	32
Index	33
Content-Addressable Names	33
Git Tracks Content	34
Pathname Versus Content	35
Pack Files	36
Object Store Pictures	36
Git Concepts at Work	39
Inside the .git Directory	39
Objects, Hashes, and Blobs	40
Files and Trees	41
A Note on Git's Use of SHA1	42
Tree Hierarchies	43
Commits	44
Tags	46
<b>5. File Management and the Index .....</b>	<b>47</b>
It's All About the Index	48
File Classifications in Git	48
Using git add	50
Some Notes on Using git commit	52
Using git commit --all	52
Writing Commit Log Messages	54
Using git rm	54
Using git mv	56
A Note on Tracking Renames	57
The .gitignore File	58
A Detailed View of Git's Object Model and Files	60
<b>6. Commits .....</b>	<b>65</b>
Atomic Changesets	66
Identifying Commits	67
Absolute Commit Names	67
refs and symrefs	68
Relative Commit Names	69
Commit History	72
Viewing Old Commits	72

Commit Graphs	74
Commit Ranges	78
Finding Commits	83
Using git bisect	83
Using git blame	87
Using Pickaxe	88
<b>7. Branches</b>	<b>89</b>
Reasons for Using Branches	89
Branch Names	90
Dos and Don'ts in Branch Names	91
Using Branches	91
Creating Branches	93
Listing Branch Names	94
Viewing Branches	94
Checking out Branches	97
A Basic Example of Checking out a Branch	97
Checking out When You Have Uncommitted Changes	98
Merging Changes into a Different Branch	99
Creating and Checking out a New Branch	101
Detached HEAD Branches	102
Deleting Branches	103
<b>8. Diffs</b>	<b>107</b>
Forms of the git diff Command	108
Simple git diff Example	112
git diff and Commit Ranges	115
git diff with Path Limiting	117
Comparing How Subversion and Git Derive diffs	119
<b>9. Merges</b>	<b>121</b>
Merge Examples	121
Preparing for a Merge	122
Merging Two Branches	122
A Merge with a Conflict	124
Working with Merge Conflicts	128
Locating Conflicted Files	129
Inspecting Conflicts	129
How Git Keeps Track of Conflicts	134
Finishing Up a Conflict Resolution	135
Aborting or Restarting a Merge	137
Merge Strategies	137
Degenerate Merges	140



Normal Merges	142
Specialty Merges	143
Applying Merge Strategies	144
Merge Drivers	145
How Git Thinks About Merges	146
Merges and Git's Object Model	146
Squash Merges	147
Why Not Just Merge Each Change One by One?	148
<b>10. Altering Commits</b>	<b>151</b>
Caution About Altering History	152
Using git reset	154
Using git cherry-pick	161
Using git revert	163
reset, revert, and checkout	164
Changing the Top Commit	165
Rebasing Commits	167
Using git rebase -i	170
rebase Versus merge	174
<b>11. The Stash and the Reflog</b>	<b>181</b>
The Stash	181
The Reflog	189
<b>12. Remote Repositories</b>	<b>195</b>
Repository Concepts	196
Bare and Development Repositories	196
Repository Clones	197
Remotes	198
Tracking Branches	199
Referencing Other Repositories	200
Referring to Remote Repositories	200
The refsPEC	202
Example Using Remote Repositories	204
Creating an Authoritative Repository	205
Make Your Own Origin Remote	206
Developing in Your Repository	208
Pushing Your Changes	209
Adding a New Developer	210
Getting Repository Updates	212
Remote Repository Development Cycle in Pictures	217
Cloning a Repository	217
Alternate Histories	218

Non-Fast-Forward Pushes	219
Fetching the Alternate History	221
Merging Histories	222
Merge Conflicts	223
Pushing a Merged History	223
Remote Configuration	223
Using git remote	224
Using git config	225
Using Manual Editing	226
Working with Tracking Branches	227
Creating Tracking Branches	227
Ahead and Behind	230
Adding and Deleting Remote Branches	231
Bare Repositories and git push	232
<b>13. Repository Management .....</b>	<b>235</b>
A Word About Servers	235
Publishing Repositories	236
Repositories with Controlled Access	236
Repositories with Anonymous Read Access	238
Repositories with Anonymous Write Access	242
Publishing Your Repository to GitHub	242
Repository Publishing Advice	243
Repository Structure	244
The Shared Repository Structure	244
Distributed Repository Structure	244
Repository Structure Examples	246
Living with Distributed Development	248
Changing Public History	248
Separate Commit and Publish Steps	249
No One True History	249
Knowing Your Place	250
Upstream and Downstream Flows	251
The Maintainer and Developer Roles	251
Maintainer-Developer Interaction	252
Role Duality	253
Working with Multiple Repositories	254
Your Own Workspace	254
Where to Start Your Repository	255
Converting to a Different Upstream Repository	256
Using Multiple Upstream Repositories	257
Forking Projects	259

<b>14. Patches .....</b>	<b>263</b>
Why Use Patches?	264
Generating Patches	265
Patches and Topological Sorts	272
Mailing Patches	273
Applying Patches	276
Bad Patches	283
Patching Versus Merging	283
<b>15. Hooks .....</b>	<b>285</b>
Installing Hooks	287
Example Hooks	287
Creating Your First Hook	288
Available Hooks	290
Commit-Related Hooks	290
Patch-Related Hooks	291
Push-Related Hooks	292
Other Local Repository Hooks	294
<b>16. Combining Projects .....</b>	<b>295</b>
The Old Solution: Partial Checkouts	296
The Obvious Solution: Import the Code into Your Project	297
Importing Subprojects by Copying	299
Importing Subprojects with <code>git pull -s subtree</code>	299
Submitting Your Changes Upstream	303
The Automated Solution: Checking out Subprojects Using Custom Scripts	304
The Native Solution: <code>gitlinks</code> and <code>git submodule</code>	305
Gitlinks	306
The <code>git submodule</code> Command	308
<b>17. Submodule Best Practices .....</b>	<b>313</b>
Submodule Commands	314
Why Submodules?	315
Submodules Preparation	316
Why Read Only?	316
Why Not Read Only?	317
Examining the Hashes of Submodule Commits	317
Credential Reuse	318
Use Cases	318
Multilevel Nesting of Repos	319
Submodules on the Horizon	320

<b>18. Using Git with Subversion Repositories .....</b>	<b>321</b>
Example: A Shallow Clone of a Single Branch	321
Making Your Changes in Git	324
Fetching Before Committing	325
Committing Through git svn rebase	326
Pushing, Pulling, Branching, and Merging with git svn	327
Keeping Your Commit IDs Straight	328
Cloning All the Branches	329
Sharing Your Repository	331
Merging Back into Subversion	332
Miscellaneous Notes on Working with Subversion	334
svn:ignore Versus .gitignore	334
Reconstructing the git-svn Cache	334
<b>19. Advanced Manipulations .....</b>	<b>337</b>
Using git filter-branch	337
Examples Using git filter-branch	339
filter-branch Pitfalls	344
How I Learned to Love git rev-list	345
Date-Based Checkout	345
Retrieve Old Version of a File	348
Interactive Hunk Staging	350
Recovering a Lost Commit	360
The git fsck Command	361
Reconnecting a Lost Commit	365
<b>20. Tips, Tricks, and Techniques .....</b>	<b>367</b>
Interactive Rebase with a Dirty Working Directory	367
Remove Left-Over Editor Files	368
Garbage Collection	368
Split a Repository	370
Tips for Recovering Commits	371
Subversion Conversion Tips	372
General Advice	372
Remove a Trunk After an SVN Import	372
Removing SVN Commit IDs	373
Manipulating Branches from Two Repositories	374
Recovering from an Upstream Rebase	374
Make Your Own Git Command	376
Quick Overview of Changes	376
Cleaning Up	377
Using git-grep to Search a Repository	378
Updating and Deleting refs	380



Following Files that Moved	380
Keep, But Don't Track, This File	381
Have You Been Here Before?	382
<b>21. Git and GitHub</b>	<b>385</b>
Repo for Public Code	385
Creating a GitHub Repository	388
Social Coding on Open Source	390
Watchers	391
News Feed	392
Forks	392
Creating Pull Requests	394
Managing Pull Requests	396
Notifications	398
Finding Users, Projects, and Code	401
Wikis	402
GitHub Pages (Git for Websites)	403
In-Page Code Editor	405
Subversion Bridge	407
Tags Automatically Becoming Archives	408
Organizations	409
REST API	410
Social Coding on Closed Source	411
Eventual Open Sourcing	411
Coding Models	412
GitHub Enterprise	414
GitHub in Sum	416
<b>Index</b>	<b>417</b>