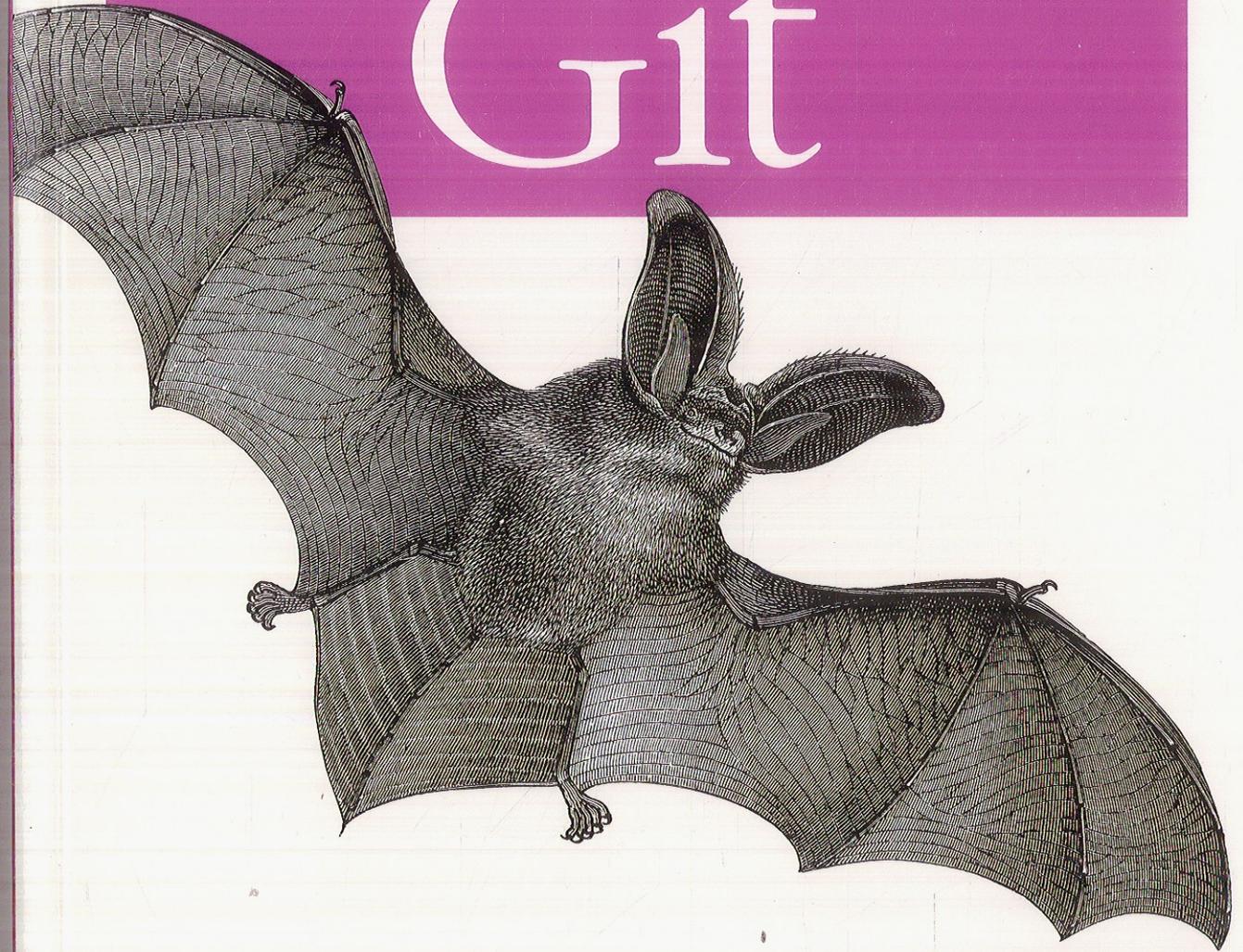


*Powerful Techniques for Centralized and  
Distributed Project Management*

*Version Control with*

# Git



**O'REILLY®**

*Jon Loeliger*

# 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
Time Line	5
What's in a Name?	6
<b>2. Installing Git</b> .....	<b>7</b>
Using Linux Binary Distributions	7
Debian/Ubuntu	7
Other Binary Distributions	8
Obtaining a Source Release	9
Building and Installing	9
Installing Git on Windows	11
Installing the Cygwin Git Package	12
Installing Standalone Git (msysGit)	13
<b>3. Getting Started</b> .....	<b>17</b>
The Git Command Line	17
Quick Introduction to Using Git	19
Creating an Initial Repository	19
Adding a File to Your Repository	20
Configuring the Commit Author	22
Making Another Commit	22
Viewing Your Commits	22
Viewing Commit Differences	24
Removing and Renaming Files in Your Repository	24
Making a Copy of Your Repository	25
Configuration Files	26
Configuring an Alias	28

Inquiry	28
<b>4. Basic Git Concepts</b> .....	<b>29</b>
Basic Concepts	29
Repositories	29
Git Object Types	30
Index	31
Content-Addressable Names	31
Git Tracks Content	32
Pathname Versus Content	33
Object Store Pictures	33
Git Concepts at Work	36
Inside the .git directory	36
Objects, Hashes, and Blobs	37
Files and Trees	38
A Note on Git's Use of SHA1	39
Tree Hierarchies	41
Commits	42
Tags	43
<b>5. File Management and the Index</b> .....	<b>45</b>
It's All About the Index	46
File Classifications in Git	46
Using git add	48
Some Notes on Using git commit	50
Using git commit --all	50
Writing Commit Log Messages	51
Using git rm	52
Using git mv	54
A Note on Tracking Renames	55
The .gitignore File	56
A Detailed View of Git's Object Model and Files	58
<b>6. Commits</b> .....	<b>63</b>
Atomic Changesets	64
Identifying Commits	65
Absolute Commit Names	65
refs and symrefs	66
Relative Commit Names	67
Commit History	69
Viewing Old Commits	69
Commit Graphs	72
Commit Ranges	76

Finding Commits	81
Using git bisect	81
Using git blame	85
Using Pickaxe	86
<b>7. Branches</b> .....	<b>87</b>
Reasons for Using Branches	87
Branch Names	88
Dos and Don'ts in Branch Names	89
Using Branches	89
Creating Branches	90
Listing Branch Names	92
Viewing Branches	92
Checking Out Branches	94
A Basic Example of Checking Out a Branch	95
Checking Out When You Have Uncommitted Changes	96
Merging Changes into a Different Branch	97
Creating and Checking Out a New Branch	99
Detached HEAD Branches	100
Deleting Branches	101
<b>8. Diffs</b> .....	<b>105</b>
Forms of the git diff Command	106
Simple git diff Example	110
git diff and Commit Ranges	113
git diff with Path Limiting	116
Comparing How Subversion and Git Derive diffs	118
<b>9. Merges</b> .....	<b>119</b>
Merge Examples	119
Preparing for a Merge	120
Merging Two Branches	120
A Merge with a Conflict	122
Working with Merge Conflicts	126
Locating Conflicted Files	126
Inspecting Conflicts	127
How Git Keeps Track of Conflicts	131
Finishing Up a Conflict Resolution	133
Aborting or Restarting a Merge	135
Merge Strategies	135
Degenerate Merges	138
Normal Merges	140
Specialty Merges	141

Applying Merge Strategies	142
Merge Drivers	144
How Git Thinks About Merges	144
Merges and Git's Object Model	144
Squash Merges	145
Why Not Just Merge Each Change One by One?	146
<b>10. Altering Commits .....</b>	<b>149</b>
Caution About Altering History	151
Using git reset	152
Using git cherry-pick	159
Using git revert	161
reset, revert, and checkout	161
Changing the Top Commit	163
Rebasing Commits	165
Using git rebase -i	167
rebase Versus merge	171
<b>11. Remote Repositories .....</b>	<b>177</b>
Repository Concepts	178
Bare and Development Repositories	178
Repository Clones	179
Remotes	180
Tracking Branches	180
Referencing Other Repositories	181
Referring to Remote Repositories	182
The refspec	183
Example Using Remote Repositories	185
Creating an Authoritative Repository	186
Make Your Own origin Remote	187
Developing in Your Repository	189
Pushing Your Changes	189
Adding a New Developer	190
Getting Repository Updates	192
Remote Repository Operations in Pictures	196
Cloning a Repository	197
Alternate Histories	198
Non-Fast-Forward Pushes	199
Fetching the Alternate History	200
Merging Histories	201
Merge Conflicts	202
Pushing a Merged History	203
Adding and Deleting Remote Branches	203

Remote Configuration	204
git remote	205
git config	205
Manual Editing	206
Bare Repositories and git push	206
Publishing Repositories	208
Repositories with Controlled Access	208
Repositories with Anonymous Read Access	210
Repositories with Anonymous Write Access	213
<b>12. Repository Management</b>	<b>215</b>
Repository Structure	215
The Shared Repository Structure	215
Distributed Repository Structure	216
Repository Structure Examples	217
Living with Distributed Development	219
Changing Public History	219
Separate Commit and Publish Steps	220
No One True History	220
Knowing Your Place	221
Upstream and Downstream Flows	222
The Maintainer and Developer Roles	222
Maintainer-Developer Interaction	223
Role Duality	224
Working with Multiple Repositories	225
Your Own Workspace	225
Where to Start Your Repository	226
Converting to a Different Upstream Repository	227
Using Multiple Upstream Repositories	229
Forking Projects	231
<b>13. Patches</b>	<b>233</b>
Why Use Patches?	234
Generating Patches	235
Patches and Topological Sorts	242
Mailing Patches	243
Applying Patches	246
Bad Patches	253
Patching Versus Merging	253
<b>14. Hooks</b>	<b>255</b>
Installing Hooks	257
Example Hooks	257

Creating Your First Hook	258
Available Hooks	260
Commit-Related Hooks	260
Patch-Related Hooks	261
Push-Related Hooks	262
Other Local Repository Hooks	263
<b>15. Combining Projects .....</b>	<b>265</b>
The Old Solution: Partial Checkouts	266
The Obvious Solution: Import the Code into Your Project	267
Importing Subprojects by Copying	269
Importing Subprojects with <code>git pull -s subtree</code>	269
Submitting Your Changes Upstream	273
The Automated Solution: Checking Out Subprojects Using Custom Scripts	274
The Native Solution: <code>gitlinks</code> and <code>git submodule</code>	275
<code>gitlinks</code>	276
The <code>git submodule</code> Command	278
<b>16. Using Git with Subversion Repositories .....</b>	<b>283</b>
Example: A Shallow Clone of a Single Branch	283
Making Your Changes in Git	286
Fetching Before Committing	287
Committing Through <code>git svn rebase</code>	288
Pushing, Pulling, Branching, and Merging with <code>git svn</code>	290
Keeping Your Commit IDs Straight	290
Cloning All the Branches	292
Sharing Your Repository	293
Merging Back into Subversion	294
Miscellaneous Notes on Working with Subversion	296
<code>svn:ignore</code> Versus <code>.gitignore</code>	296
Reconstructing the <code>git-svn</code> cache	297
<b>Index .....</b>	<b>299</b>