

Git版本控制管理（第2版）



[Git版本控制管理（第2版）_下载链接1](#)

著者:[美]乔恩 罗力格（Jon Loeliger）

出版者:人民邮电出版社

出版时间:2015-3

装帧:

isbn:9787115382436

Git是一款免费、开源的分布式版本控制系统，最早由LinilusTorvalds创建，用于管理Linux内核开发，现已成为分布式版本控制的主流工具。

《Git版本控制管理（第2版）》是学习掌握Git的最佳教程，总共分为21章，其内容涵盖了如何在多种真实开发环境中使用Git；洞察Git的常用案例、初始任务和基本功能；如何在集中和分布式版本控制中使用Git；使用Git管理合并、冲突、补丁和差异；获得诸如重新定义变基（rebasing）、钩子（hook）以及处理子模块（子项目）等的高级技巧；Git如何与SVN版本库交互（包括SVN向Git的转换）；通过GitHub导航、使用开源项目，并对开源项目做贡献。

《Git版本控制管理（第2版）》适合需要进行版本控制的开发团队成员阅读，对Git感兴趣的开发人员也可以从中获益。

作者介绍:

Jon Loeliger是一位自由的软件开发工程师，对Linux、U-Boot和Git等开源项目颇有贡献。他在许多会议上（比如Linux World）发表过Git教程相关的演讲，并为Linux Magazine撰写了多篇Git相关的稿件。在成为自由的软件开发工程师之前，他花费了多年的时间来开发高度优化的编译器、路由器协议、Linux移植，还偶尔编写过游戏软件。Jon持有普度大学计算机科学学位。在闲暇之余，他还会在家里自行酿酒。

Matthew McCullough, Github.com的培训副总裁，在企业软件开发领域有15年的从业经历，还是一名经常往返于世界各地的开源教育家，以及一家美国咨询公司的联合创始人。这所有的经历使得 he 可以与大家分享利用Git和GitHub来取得成功的相关故事。Matthew是Gradle and Jenkins O'Reilly图书的特约作者，以及O'Reilly Git Master Class系列图书的创作者。Matthew经常在No Fluff Just Stuff巡回会议上发表演讲，还是DZone Git RefCard的作者，同时还是Denver Open Source Users Groups的主席。

目录: 第1章 介绍 1

1.1 背景 1

1.2 Git的诞生 2

1.3 先例 4

1.4 时间线 5

1.5 名字有何含义 6

第2章 安装Git 7

2.1 使用Linux上的二进制发行版 7

2.1.1 Debian/Ubuntu 7

2.1.2 其他发行版 8

2.2 获取源代码 9

2.3 构建和安装 10

2.4 在Windows上安装Git 11

2.4.1 安装Cygwin版本的Git 12

2.4.2 安装独立的Git (msysGit) 13

第3章 起步 16

3.1 Git命令行 16

3.2 Git使用快速入门 18

3.2.1 创建初始版本库 18

3.2.2 将文件添加到版本库中 19

3.2.3 配置提交作者 21

3.2.4 再次提交 21

3.2.5 查看提交 21

3.2.6 查看提交差异 23

3.2.7 版本库内文件的删除和重命名	23
3.2.8 创建版本库副本	24
3.3 配置文件	25
3.4 疑问	27
第4章 基本的Git概念	28
4.1 基本概念	28
4.1.1 版本库	28
4.1.2 Git对象类型	29
4.1.3 索引	30
4.1.4 可寻址内容名称	30
4.1.5 Git追踪内容	31
4.1.6 路径名与内容	31
4.1.7 打包文件	32
4.2 对象库图示	33
4.3 Git在工作时的概念	35
4.3.1 进入.git目录	35
4.3.2 对象、散列和blob	36
4.3.3 文件和树	37
4.3.4 对Git使用SHA1的一点说明	38
4.3.5 树层次结构	40
4.3.6 提交	40
4.3.7 标签	41
第5章 文件管理和索引	43
5.1 关于索引的一切	44
5.2 Git中的文件分类	44
5.3 使用git add	46
5.4 使用git commit的一些注意事项	48
5.4.1 使用git commit --all	48
5.4.2 编写提交日志消息	50
5.5 使用git rm	50
5.6 使用git mv	52
5.7 追踪重命名注解	54
5.8 .gitignore文件	55
5.9 Git中对象模型和文件的详细视图	56
第6章 提交	61
6.1 原子变更集	62
6.2 识别提交	62
6.2.1 绝对提交名	63
6.2.2 引用和符号引用	64
6.2.3 相对提交名	65
6.3 提交历史记录	67
6.3.1 查看旧提交	67
6.3.2 提交图	70
6.3.3 提交范围	73
6.4 查找提交	77
6.4.1 使用git bisect	78
6.4.2 使用git blame	82
6.4.3 使用Pickaxe	83
第7章 分支	84
7.1 使用分支的原因	84
7.2 分支名	85
7.3 使用分支	86
7.4 创建分支	88
7.5 列出分支名	89

- 7.6 查看分支 89
- 7.7 检出分支 91
 - 7.7.1 检出分支的一个简单例子 91
 - 7.7.2 有未提交的更改时进行检出 92
 - 7.7.3 合并变更到不同分支 94
 - 7.7.4 创建并检出新分支 95
 - 7.7.5 分离HEAD分支 96
- 7.8 删除分支 97
- 第8章 diff 100
 - 8.1 git diff命令的格式 101
 - 8.2 简单的git diff例子 104
 - 8.3 git diff和提交范围 108
 - 8.4 路径限制的git diff 110
 - 8.5 比较SVN和Git如何产生diff 112
- 第9章 合并 114
 - 9.1 合并的例子 114
 - 9.1.1 为合并做准备 115
 - 9.1.2 合并两个分支 115
 - 9.1.3 有冲突的合并 117
 - 9.2 处理合并冲突 121
 - 9.2.1 定位冲突的文件 122
 - 9.2.2 检查冲突 122
 - 9.2.3 Git是如何追踪冲突的 126
 - 9.2.4 结束解决冲突 128
 - 9.2.5 中止或重新启动合并 129
 - 9.3 合并策略 130
 - 9.3.1 退化合并 132
 - 9.3.2 常规合并 134
 - 9.3.3 特殊提交 135
 - 9.3.4 应用合并策略 136
 - 9.3.5 合并驱动程序 137
 - 9.4 Git怎么看待合并 138
 - 9.4.1 合并和Git的对象模型 138
 - 9.4.2 压制合并 139
 - 9.4.3 为什么不一个接一个地合并每个变更 140
- 第10章 更改提交 142
 - 10.1 关于修改历史记录的注意事项 143
 - 10.2 使用git reset 144
 - 10.3 使用git cherry-pick 152
 - 10.4 使用git revert 154
 - 10.5 reset、revert和checkout 154
 - 10.6 修改最新提交 155
 - 10.7 变基提交 158
 - 10.7.1 使用git rebase -i 160
 - 10.7.2 变基与合并 164
- 第11章 储藏和引用日志 170
 - 11.1 储藏 170
 - 11.2 引用日志 178
- 第12章 远程版本库 183
 - 12.1 版本库概念 184
 - 12.1.1 裸版本库和开发版本库 184
 - 12.1.2 版本库克隆 185
 - 12.1.3 远程版本库 186
 - 12.1.4 追踪分支 186

- 12.2 引用其他版本库 187
 - 12.2.1 引用远程版本库 188
 - 12.2.2 refspec 189
- 12.3 使用远程版本库的示例 191
 - 12.3.1 创建权威版本库 192
 - 12.3.2 制作你自己的origin远程版本库 193
 - 12.3.3 在版本库中进行开发 195
 - 12.3.4 推送变更 196
 - 12.3.5 添加新开发人员 197
 - 12.3.6 获取版本库更新 199
- 12.4 图解远程版本库开发周期 203
 - 12.4.1 克隆版本库 204
 - 12.4.2 交替的历史记录 205
 - 12.4.3 非快进推送 205
 - 12.4.4 获取交替历史记录 207
 - 12.4.5 合并历史记录 208
 - 12.4.6 合并冲突 208
 - 12.4.7 推送合并后的历史记录 209
- 12.5 远程版本库配置 209
 - 12.5.1 使用git remote 210
 - 12.5.2 使用git config 211
 - 12.5.3 使用手动编辑 212
- 12.6 使用追踪分支 212
 - 12.6.1 创建追踪分支 212
 - 12.6.2 领先和落后 215
- 12.7 添加和删除远程分支 216
- 12.8 裸版本库和git推送 217
- 第13章 版本库管理 219
 - 13.1 谈谈服务器 219
 - 13.2 发布版本库 220
 - 13.2.1 带访问控制的版本库 220
 - 13.2.2 允许匿名读取访问的版本库 221
 - 13.2.3 允许匿名写入权限的版本库 225
 - 13.2.4 在GitHub上发布版本库 225
 - 13.3 有关发布版本库的建议 227
 - 13.4 版本库结构 228
 - 13.4.1 共享的版本库结构 228
 - 13.4.2 分布式版本库结构 228
 - 13.4.3 版本库结构示例 229
 - 13.5 分布式开发指南 231
 - 13.5.1 修改公共历史记录 231
 - 13.5.2 分离提交和发布的步骤 232
 - 13.5.3 没有唯一正确的历史记录 232
 - 13.6 清楚你的位置 233
 - 13.6.1 上下游工作流 233
 - 13.6.2 维护者和开发人员的角色 234
 - 13.6.3 维护者-开发人员的交互 234
 - 13.6.4 角色的两面性 235
 - 13.7 多版本库协作 236
 - 13.7.1 属于你自己的工作区 236
 - 13.7.2 从哪里开始你的版本库 237
 - 13.7.3 转换到不同的上游版本库 238
 - 13.7.4 使用多个上游版本库 239
 - 13.7.5 复刻项目 241

- 第14章 补丁 244
 - 14.1 为什么要使用补丁 245
 - 14.2 生成补丁 246
 - 14.3 邮递补丁 254
 - 14.4 应用补丁 256
 - 14.5 坏补丁 264
 - 14.6 补丁与合并 264
- 第15章 钩子 265
 - 15.1 安装钩子 267
 - 15.1.1 钩子示例 267
 - 15.1.2 创建第一个钩子 268
 - 15.2 可用的钩子 270
 - 15.2.1 与提交相关的钩子 270
 - 15.2.2 与补丁相关的钩子 271
 - 15.2.3 与推送相关的钩子 272
 - 15.2.4 其他本地版本库的钩子 273
- 第16章 合并项目 274
 - 16.1 旧解决方案：部分检出 275
 - 16.2 显而易见的解决方案：将代码导入项目 276
 - 16.2.1 手动复制导入子项目 277
 - 16.2.2 通过git pull -s subtree导入子项目 278
 - 16.2.3 将更改提交到上游 282
 - 16.3 自动化解决方案：使用自定义脚本检出子项目 283
 - 16.4 原生解决方案：gitlink和git submodule 284
 - 16.4.1 gitlink 284
 - 16.4.2 git submodule命令 287
- 第17章 子模块最佳实践 290
 - 17.1 子模块命令 291
 - 17.2 为什么要使用子模块 291
 - 17.3 子模块准备 292
 - 17.4 为什么是只读的 293
 - 17.5 为什么不用只读的 293
 - 17.6 检查子模块提交的散列 293
 - 17.7 凭据重用 294
 - 17.8 用例 295
 - 17.9 版本库的多级嵌套 296
 - 17.10 子模块的未来 296
- 第18章 结合SVN版本库使用Git 297
 - 18.1 例子：对单一分支的浅克隆 297
 - 18.1.1 在Git中进行修改 300
 - 18.1.2 在提交前进行抓取操作 301
 - 18.1.3 通过git svn rebase提交 302
 - 18.2 在git svn中使用推送、拉取、分支和合并 303
 - 18.2.1 直接使用提交ID 304
 - 18.2.2 克隆所有分支 305
 - 18.2.3 分享版本库 307
 - 18.2.4 合并回SVN 308
 - 18.3 在和SVN一起使用时的一些注意事项 310
 - 18.3.1 svn:ignore与.gitignore 310
 - 18.3.2 重建git-svn的缓存 310
- 第19章 高级操作 312
 - 19.1 使用git filter-branch 312
 - 19.1.1 使用git filter-branch的例子 314
 - 19.1.2 filter-branch的诱惑 319

19.2 我如何学会喜欢上git rev-list	320
19.2.1 基于日期的检出	320
19.2.2 获取文件的旧版本	323
19.3 数据块的交互式暂存	325
19.4 恢复遗失的提交	336
19.4.1 git fsck命令	336
19.4.2 重新连接遗失的提交	340
第20章 提示、技巧和技术	341
20.1 对脏的工作目录进行交互式变基	341
20.2 删除剩余的编辑器文件	342
20.3 垃圾回收	342
20.4 拆分版本库	344
20.5 恢复提交的小贴士	345
20.6 转换Subversion的技巧	346
20.6.1 普适建议	346
20.6.2 删除SVN导入后的trunk	346
20.6.3 删除SVN提交ID	347
20.7 操作来自两个版本库的分支	348
20.8 从上游变基中恢复	348
20.9 定制Git命令	349
20.10 快速查看变更	350
20.11 清理	351
20.12 使用git-grep来搜索版本库	352
20.13 更新和删除ref	354
20.14 跟踪移动的文件	354
20.15 保留但不追踪文件	355
20.16 你来过这里吗	356
第21章 Git和GitHub	357
21.1 为开源代码提供版本库	358
21.2 创建GitHub的版本库	360
21.3 开源代码的社会化编程	362
21.4 关注者	363
21.5 新闻源	363
21.6 复刻	364
21.7 创建合并请求	365
21.8 管理合并请求	367
21.9 通知	369
21.10 查找用户、项目和代码	371
21.11 维基	373
21.12 GitHub页面（用于网站的Git）	373
21.13 页内代码编辑器	375
21.14 对接SVN	377
21.15 标签自动归档	378
21.16 组织	379
21.17 REST风格的API	379
21.18 闭源的社会化编程	381
21.19 最终开放源代码	381
21.20 开发模型	381
21.21 GitHub企业版	383
21.22 关于GitHub的总结	384
• • • • •	(收起)

标签

Git

版本控制

软件工程

编程

软件开发

计算机

github

GitHub

评论

手上接触到了超大型的项目，原本的git知识就显得捉襟见肘了。但是这本书让人读下去，大概翻了2/3，翻译问题，理解起来更绕了，弃。

有一些技术细节，只是细节的地方，翻译的感觉怪怪的。

一般般吧，其实多数技巧网上一查就出来了，比书上更详细，看书主要是为了查漏补缺

详略得当，实用

读完前13章，后面的估计很长一段时间内用不到，暂时不读，日后再说。

虽然有部分机翻痕迹，但是内容很全很细，例子也很容易懂。

180829 0901 看到第七章分支啦！ 0914 commit --amend 和 rebase
对保持清爽的提交记录很有帮助啊。 190113 要尝试下 git add -p。
总的来说，讲解基础那几章节还是很实用的。后面几章讲解高级用法的略读一遍（看不懂）。20.4 讲拆分版本库的已经有更好的做法了。 0122

git的相关知识介绍非常到位，基本能理解git的原理以及一些高级的应用。
后面的章节就属于稍微高级的应用，包括和svn的结合，在工作中不常用，但是也可以进行了解。

“项目的状态可以用一个提交和一棵目录树表示，它可以来自项目历史中的任意时刻，或者它可以是你正在开发的未来状态”，从这个角度看，git的设计与平行宇宙的概念多么相似呀。

有些地方翻译不通畅略遗憾，不过书籍本身解答了自己的很多疑惑，之后值得再读

内容很不错，但译本真的就是机译的水平啊，完全是在骗钱

讲的还是蛮使用的了。

翻译的不好。内容排布也有问题，它按模块来的（虽然这种方法容易写清楚所有操作），而不是按工作流的顺序

看了好几本，包括官方文档，此书最好！git是程序员的强力工具，早日掌握早日受用。

git感觉还是要从底层学起，这本书讲的还是比较清楚的。

翻译真是太差了。

读的不是原本，是迷你的，不好。

查漏补缺git技能~~~

对于git讲得很深，建议先看完pro git再去看这边书

比较详细的说明了GIT操作的详细内容，作为工具书还是可以的。

[Git版本控制管理（第2版）_下载链接1](#)

书评

2017.8.26

对于入门Git，并且想要了解Git底层原理的人，这本书可以加入你的可读目录。
唯一的缺点是，翻译，当然这是普遍存在的问题，不过这本书的翻译有点太像机器翻译的了，读起来很硬，没有一点主观语气。
如果能接受翻译，这本书很不错，不能接受的话，可以看原著，或者看...

Git permits practically an infinite variety of methods for development and collaboration, but its flexibility also means that some users don't understand how to use it to best advantage. This book offers tutorials on ways to use it, as well as friendly yet ...

用Git还是有必要深入了解它的细节;和其它几本Git的书比起来,这本对细节讲得都比较细,而且思路也比较清晰,看起来比较轻松;

原理的核心是 git 是 track content 而不是简单 track version, 每个 version 是与一系列 content objects 绑定的, 这里有一个简单介绍:
<http://stackoverflow.com/questions/31995829/how-does-git-track-changes-to-files>

这书一定是请非洲黑猩猩翻译的。
书本身的内容很不错,讲解也很深入,但人民邮电很显然雇佣了两个非洲黑猩猩来翻译这本书,如果有谁打算买这本书,请一定慎重,最好去买pro git,或者下载电子版去看,这本书就绕过吧,免得浪费钱财,破坏了心情。

这本书对于Git版本控制系统的原理讲的很清晰透彻,对于正常的开发项目中Git的使用,这本书完全够用了。
用了大概一星期时间看完了前11个章节,文章用了很多图表的形式来讲解git内部的各个概念的逻辑关系和组成结构,直观易懂,读起来并不是很费力。

使用GIT前应该看的第一本书,从头到尾没有什么多余的章节,每章都值得一看。这本书看起来还是很舒服的,偶尔出现几个生僻词汇,基本上不卡。而且尤其喜欢用简图的方式介绍GIT的各种对象和动作,贴心的。其实从另一方面说来,GIT本身就是建立在DAG上的分布式VCS,当然用图解最爽...

[Git版本控制管理（第2版）_下载链接1](#)