HZ BOOKS
华章科技

Git领域的集大成之作，在广度、深度和实战性上均史无前例
国内顶级Git专家亲自撰写，Git官方维护者等数位专家联袂推荐

實戰

蒋鑫 著

Got Git: The Definitive Guide of Git

Git 权威指南

机械工业出版社
China Machine Press

實戰

Got Git: The Definitive Guide of Git

Git权威指南

蒋鑫 著

机械工业出版社
China Machine Press

本书是Git领域的集大成之作，是一本关于Git的百科全书，在广度、深度和实战性上让同类作品望尘莫及。作者是国内顶尖的版本控制专家和咨询顾问之一，本书得到了Git官方维护者Junio C Hamano和ITeye创始人范凯（Robbin）先生等数位专家的高度认可和极力推荐，权威性毋庸置疑。

全书一共9篇，共41章和4个附录，内容几乎涵盖了Git的所有方面。第1篇介绍了版本控制工具的演变历史、Git的各种优点，以及它在3种主流操作系统中的安装与配置。第2篇和第3篇既是本书的基础，又是本书的核心，不仅介绍了Git的操作和使用，而且还讲解了Git的原理。第2篇详细讲解了个人用户如何使用Git，包括Git初始化、日常操作、暂存区、对象、重置、检出、恢复进度、历史变更、克隆、库管理等；第3篇详细讲解了Git协议和团队如何使用Git，包括Git支持的协议、冲突解决、里程碑、分支、远程版本库和补丁文件交互等。第4篇全面介绍了Git的协同模型，即它在实际工作中的使用模式，包括各种经典的Git协同模型、Topgit协同模型、子模组协同模型、子树合并、Android多版本库协同、Git与SVN协同模型等。第5篇介绍了Git服务器的架设，首先讲解了HTTP协议、Git协议、SSH协议的使用，然后讲解了Gitolite、Gitosis、Gerrit等服务器的架设方法，最后还讲解了Git版本库的托管。第6篇介绍了版本库的迁移，包括如何从CVS、SVN、Hg等版本库迁移到Git，以及Git版本库整理等方面的内容。第7篇讲解了Git的其他应用，包括etckeeper、Gistore等的安装、配置和使用，以及补丁中的二进制文件和云存储等内容。第8篇介绍了Git的跨平台操作，以及它的钩子和模板、稀疏检出和浅克隆、嫁接和替换等重要特性。第9篇是附录，详细给出了Git的命令索引，以及CVS、SVN和Hg与Git的比较与命令对照，方便读者查阅。

图书在版编目（CIP）数据

Git权威指南 / 蒋鑫著. —北京：机械工业出版社，2011.6

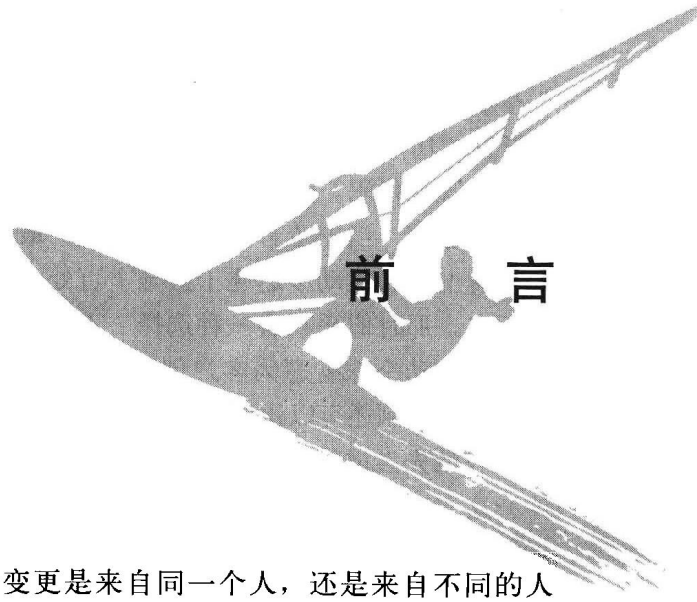
ISBN 978-7-111-34967-9

I. G…　II. 蒋…　III. 软件工具—程序设计　IV. TP311.56

中国版本图书馆CIP数据核字（2011）第103019号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：陈佳媛
北京京北印刷有限公司印刷
2011年6月第1版第1次印刷
186mm×240mm • 39印张
标准书号：ISBN 978-7-111-34967-9
定价：89.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991；88361066
购书热线：（010）68326294；88379649；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com



前 言

版本控制是管理数据变更的艺术，无论数据变更是来自同一个人，还是来自不同的人（一个团队）。版本控制系统不但要忠实地记录数据的每一次变更，还要能够帮助还原任何一次历史变更，以及实现团队的协同工作等。Git 就是版本控制系统中的佼佼者。

我对版本控制系统的兴趣源自于我的个人知识管理实践，其核心就是撰写可维护的文档，并保存于版本控制系统中。可维护文档的格式可以是 DocBook、FreeMind、reStructuredText 等。我甚至还对 FreeMind 加以改造以便让其文档格式更适合于版本控制系统，这就是我的第一个开源实践：托管于 SourceForge 上的 FreeMind-MMX 项目[①]。文档书写格式的问题解决之后，就是文档的存储问题了。通过版本控制系统，很自然地就可以实现对文档历史版本的保存，但是如何避免因为版本控制系统瘫痪而导致数据丢失呢？Git 用其崭新的分布式的版本控制设计提供了最好的解决方案。使用 Git，我的知识库不再只有唯一的版本库与之对应，而是可以通过克隆操作分发到不同的磁盘或主机上，克隆的版本库之间通过推送（PUSH）和拉回（PULL）等操作进行同步，数据安全得到了极大的提升。在版本控制系统的忠实呵护下，我的知识库中关于 Git 的 FreeMind 脑图在日积月累中变得越来越翔实，越来越清晰，最终成为本书的雏形。

版本控制能决定项目的成败，甚至是公司的生死，此言不虚。我在推广开源项目管理工具和为企业提供咨询服务的过程中看到，有很多团队因为版本控制系统管理的混乱导致项目延期、修正的 Bug 重现、客户的问题不能在代码中定位……无论他们使用的是什么版本控制系统（开源的或是商业的）都是如此。这是因为传统的集中式版本控制系统不能有效地管理分支和进行分支间合并。集中管理的版本库只有唯一的分支命名空间，需要专人管理，从而造成分支创建的不自由；分支间的合并要么因为缺乏追踪导致重复合并、引发严重冲突，要

① *http://freemind-mmx.sourceforge.net/*

么因为版本控制系统本身蹩脚的设计导致分支合并时效率低下和陷阱重重。Git 凭借其灵活的设计让项目摆脱分支管理的梦魇。

我的公司也经历过代码管理的生死考验。因为公司的开发模式主要是基于开源软件的二次开发，所以最早在使用 SVN（Subversion）做版本控制时，很自然地使用了 SVN 卖主分支模型来管理代码。随着增加和修改的代码越来越多，我们开发的软件与开源软件上游的偏离也越来越远，当上游有新版本发布时，最早可能只用几个小时就可以将改动迁移过去，但是如果对上游的改动多达几十甚至上百处时，迁移的过程就会异常痛苦，基本上和重新做一遍差不多。那时似乎只有一种选择：不再与上游合并，不再追踪上游的改动，而这与公司的价值观“发动全球智慧为客户创造价值”相违背。迷茫之中，分布式版本控制系统飘然而至，原来版本控制还可以这么做。

我最先尝试的分布式版本控制系统是 Hg（Mercurial），当发现 Hg 和 MQ（Hg 的一个插件）这一对宝贝儿的时候，我如获至宝。逐渐地，公司的版本库都迁移到了 Hg 上。但随着新的开发人员的加入，问题又出现了，一个人使用 Hg 和 MQ 很好，但多个人使用时则会出现难以协同的问题。于是我们大胆地采用了 Git，并在实践中结合 Topgit 等工具进行代码的管理。再一次，也许是最后一次，我们的代码库迁移到了 Git。

最早认识分布式版本控制，源自于我们看到了众多开源项目的版本控制系统大迁移，这场迁移还在进行中。

- MoinMoin 是我们关注的一个开源的维基软件，2006 年，它的代码库从 SVN 迁移到了 Hg[①]。
- Mailman 同样是我们关注的一个开源邮件列表软件。2007 年，它的代码库从 SVN 迁移到了 Bazaar[②]。
- Linux 采用 Git 作为版本控制系统（一点都不奇怪，因为 Git 就是 Linus Torvalds 开发的）。
- Android 是目前最为流行的开源项目之一，因为潜在市场巨大，已经吸引了越来越多的开发者进入这个市场，而 Android 就是用 Git 维护的。

当开源软件纷纷倒向分布式版本控制系统大旗（尤其是 Git）的时候，很多商业公司也在行动了，尤其是涉及异地团队协同和 Android 核心代码定制开发的公司。对于那些因保守而不敢向 Git 靠拢的公司，Git 也可以派上用场，因为 Git 可以与现在大多数公司部署的 SVN 很好地协同，即公司的服务器是 SVN，开发者的客户端则使用 Git。相信随着 Git 的普及，以及公司在代码管理观念上的改进，会有更多的公司拥抱 Git。

---

① *http://moinmo.in/NewVCS*

② *http://wiki.list.org/display/DEV/Home*

本书的组织

本书共分为 9 篇，前 8 篇是正文，一共 41 章，第 9 篇是附录。

第 1 篇讲解了 Git 的相关概念，以及安装和配置的方法，共 3 章。第 1 章介绍了版本控制的历史。第 2 章用十几个小例子介绍了 Git 的一些闪亮特性，期待这些特性能够让你爱上 Git。第 3 章则介绍了 Git 在三种主要操作系统平台上的安装和使用。在本书的写作过程中，我 70% 的时间使用的是 Debian Linux 操作系统，Linux 用户可以毫无障碍地完成本书列举的所有实践操作。在 2010 年年底，当得知有出版社愿意出版这本书后，我向妻子阿巧预支了未来的部分稿费购买了我的第一台 MacBook Pro，于是本书就有了较为翔实的如何在 Mac OS X 下安装和使用 Git 的内容，以及在本书第 22 章中介绍的关于 Topgit 在 Mac OS X 上的部署和改进相关的内容。在本书的编辑和校对过程中因为要使用 Word 格式的文稿，所以本书后期的很多工作是在运行于 VirtualBox 下的 Windows 虚拟机中完成的，即使是使用运行于资源受限的虚拟机中的 Cygwin，Git 依然完美地完成了工作。

第 2 篇和第 3 篇详细讲解了 Git 的使用方法，是本书的基础和核心，大约占据了全书 40% 的篇幅。这两篇的内容架构方式是我在进行 SVN 培训时就已经形成的习惯，即以“独奏”指代一个人的版本控制所要讲述的知识点，以“和声”指代团队版本控制涉及的话题。在第 2 篇“Git 独奏”中，本书将 Git 的设计原理穿插在各章之中讲解，因为唯有了解真相（Git 原理），才有可能自由（掌握 Git）。在第 3 篇“Git 和声”中，本书讲解了团队版本控制必须掌握的里程碑和分支等概念，以及如何解决合并中遇到的冲突。

第 4 篇细致地讲解了 Git 在实际工作中的使用模式。除了传统的集中式和分布式使用模式之外，第 22 章还介绍了 Topgit 在定制开发中的应用，这也是我公司在使用 Git 时采用的最主要的模式。这一章还讲解了我对 Topgit 所做的部分改进，相关的具体介绍最早出现在我公司的博客上[①]。第 23 ～ 25 章介绍了多版本库协同的不同方法，其中第 25 章介绍的一个独辟蹊径的解决方案是由 Android 项目引入的名为 repo 的工具实现的，我对其进行改造后可以让这个工具脱离 Gerrit 代码审核服务器，直接操作 Git 服务器。第 26 章介绍了 git-svn 这一工具，该工具不但可以实现从 SVN 版本库到 Git 版本库的迁移，还可以实现以 Git 作为客户端向 SVN 提交。

第 5 篇介绍了 Git 服务器的架设。本篇是全书最早开始撰写的部分，这是因为我给客户做的 Git 培训讲义的相关内容不够详细，于是应客户要求针对 Gitolite 等服务器的架设撰写了详细的管理员手册，即本书的第 30 章。第 32 章介绍了 Android 项目在 Git 管理上的又一大创造，即 Gerrit，它实现了一个独特的集中式 Git 版本库管理模型。

第 6 篇讲解了 Git 版本库的迁移。其中第 34 章详细介绍了从 CVS 版本库到 Git 版本库

① *http://blog.ossxp.com/*

的迁移，其迁移过程也可以作为从 CVS 到 SVN 迁移的借鉴。本篇还介绍了从 SVN 和 Hg 版本库到 Git 的迁移。对于其他类型的版本库，介绍了一个通用的需要编程来实现的方法。在本篇的最后还介绍了一个 Git 版本库整理的利器，可以理解为一个 Git 库转换为另外一个 Git 库的方法。

第 7 篇是关于 Git 的其他应用，其主要内容介绍了我在 etckeeper 启发下开发的一款备份工具 Gistore，该工具可以运行于 Linux 和 Mac OS X 下。

第 8 篇是 Git 杂谈。其中第 40 章的内容可供跨平台的项目组借鉴。第 41 章介绍了一些在前面没有涉及的 Git 的相关功能和特性。

第 9 篇是附录。首先介绍了完整的 Git 命令索引，然后分别介绍了 CVS、SVN、Hg 与 Git 之间的比较和命令对照，对于有其他版本控制系统使用经验的用户而言，这一部分内容颇具参考价值。

适用读者

本书适合所有翻开它的人，因为我知道这本书在书店里一定是放在计算机图书专柜。本书尤其适合以下几类读者阅读。

1. 被数据同步困扰的“电脑人”

困扰“电脑人”的一个常见问题是，有太多的数据需要长久保存，有太多的电脑设备需要数据同步。可能有的人会说：“像 Dropbox 一样的网盘可以帮助我呀”。是的，云存储就是在技术逐渐成熟之后应运而生的产品，但是依然解决不了如下几个问题：多个设备上同时修改造成的冲突；冗余数据传输造成的带宽瓶颈；没有实现真正的、完全的历史变更数据备份。具体请参见本书第 7 篇第 39 章的内容。

Git 可以在数据同步方面做得更好，甚至只需借助小小的 U 盘就可以实现多台电脑的数据同步，并且支持自动的冲突解决。只要阅读本书第 1 篇和第 2 篇，就能轻易掌握相关的操作，实现数据的版本控制和同步。

2. 学习计算机课程的学生

我非常后悔没有在学习编程的第一天就开始使用版本控制，在学校时写的很多小程序和函数库都丢失了。直到使用了 CVS 和 SVN 对个人数据进行版本控制之后，才开始把每一天的变更历史都保留了下来。Git 在这方面可以比 CVS 和 SVN 等做得更好。

在阅读完本书的前 3 篇掌握了 Git 的基础知识之后，可以阅读第 5 篇第 33 章的内容，通过 Github 或类似的服务提供商建立自己的版本库托管，为自己的数据找一个安全的家。

3. 程序员

使用 Git 会让程序员有更多的时间休息，因为可以更快地完成工作。分布式版本控制让每一个程序员都能在本地拥有一个完整的版本库，所以几乎所有操作都能够脱离网络执行而不受带宽的限制。加之使用了智能协议，版本库间的同步不但减少了数据传输量，还能显示完成进度。

Git 帮助程序员打开了进入开源世界的大门，进而开阔视野，提升水平，增加择业的砝码。看看使用 Git 作为版本控制的开源软件吧：Linux kernel、Android、Debian、Fedora、GNOME、KDevelop、jQuery、Prototype、PostgreSQL、Ruby on Rails……不胜枚举。还有，不要忘了所有的 SVN 版本库都可以通过 Git 方式更好地访问。

作为一个程序员，必须具备团队协同能力，本书第 3 篇应该作为学习的重点。

4.Android 程序员

如果你是谷歌 Android 项目的参与者，尤其是驱动开发和核心开发的参与者，必然会接触 Git、repo 和 Gerrit。对于只是偶尔参考一下 Android 核心代码的 Android 应用开发人员而言，也需要对 repo 有深入的理解，这样才不至于每次为同步代码而耗费一天的时间。

repo 是 Android 为了解决 Git 多版本库管理问题而设计的工具，在本书第 4 篇第 25 章有详细介绍。

Gerrit 是谷歌为了避免因分布式开发造成项目分裂而开发的工具，打造了 Android 独具一格的集中式管理模式，在本书第 5 篇第 32 章有详细介绍。

即使是非 Android 项目，也可以使用这两款工具为自己的项目服务。我还为 repo 写了几个新的子命令以实现脱离 Gerrit 提交，让 repo 拥有更广泛的应用领域。

5. 定制开发程序员

当一个公司的软件产品需要针对不同的用户进行定制开发时，就需要在一个版本库中建立大量的特性分支，使用 SVN 的分支管理远不如使用 Git 的分支管理那么自然和方便。还有一个应用领域就是对第三方代码进行维护。当使用 SVN 进行版本控制时，最自然的选择是卖主分支，但随着定制开发的逐渐深入，与上游的偏离也会越大，于是与上游代码的合并也将越来越令人痛苦。

第 4 篇第 22 章介绍 Topgit 这一杀手级的工具，这是这个领域最佳的解决方案。

6.SVN 用户

商业软件的研发团队因为需要精细的代码授权，所以不会轻易更换现有的 SVN 版本控制系统，这种情况下 Git 依然大有作为。无论是出差在外，或是在家办公，或是开发团队分处异地，都会遇到 SVN 版本控制服务器无法访问或速度较慢的情况。这时 git-svn 这一工具

会将 Git 和 SVN 完美地结合在一起，既严格遵守 SVN 的授权规定，又可以自如地进行本地提交，当能够连接到 SVN 服务器时，可以在悠闲地喝着绿茶的同时，等待一次性批量提交的完成。

我有几个项目（pySvnManager、Freemind-MMX）托管在 SourceForge 的 SVN 服务器上，现在都是先通过 git-svn 将其转化为本地的 Git 库，然后再使用的。以这样的方式访问历史数据、比较代码或提交代码，再也不会因为网速太慢而望眼欲穿了。

本书第 4 篇第 26 章详细介绍了 Git 和 SVN 的互操作。

7. 管理员

Git 在很大程度上减轻了管理员的负担：分支的创建和删除不再需要管理员统一管理，因为作为分布式版本控制系统，每一个克隆就是一个分支，每一个克隆都拥有独立的分支命名空间；管理员也不再需要为版本库的备份操心，因为每一个项目成员都拥有一个备份；管理员也不必担心有人在服务器上篡改版本库，因为 Git 版本库的每一个对象（提交和文件等）都使用 SHA1 哈希值进行完整性校验，任何对历史数据的篡改都会因为对后续提交产生的连锁反应而原形毕露。

本书第 7 篇第 37 章介绍了一款我开发的基于 Git 的备份工具，它使得 Linux 系统的数据备份易如反掌。本书第 5 篇介绍的 Git 服务器搭建，以及第 6 篇介绍的版本库迁移方面的知识会为版本控制管理员的日常维护工作提供指引。

8. 开发经理

作为开发经理，你一定要对代码分支有深刻的理解，不知本书第 18 章中的“代码管理之殇”是否能引起你的共鸣。为了能在各种情况下恰当地管理开发团队，第 4 篇“Git 协同模型”是项目经理应该关注的重点。你的团队是否存在着跨平台开发，或者潜在着跨平台开发的可能？本书第 8 篇第 40 章也是开发经理应当关注的内容。

排版约定

本书使用的排版格式约定如下：

1. 命令输出及示例代码

执行一条 Git 命令及其输出的示例如下：

```
$ git --version
git version 1.7.4
```

2. 提示符（$）

命令前面的 $ 符号代表命令提示符。

3. 等宽字体（Constant width）

用于标示屏幕输出的字符、示例代码，以及正文中出现的命令、参数、文件名和函数名等。

4. 等宽粗体（**Constant width bold**）

用于表示由用户手工输入的内容。

5. 占位符（*<Constant width>*）

用尖括号扩起来的内容，表示命令中或代码中的占位符，读者应当用实际值将其替换。

在线资源

- 官方网站：*http://www.ossxp.com/doc/gotgit/*

  在本书的官方网站上，大家可以了解到与本书相关的最新信息，查看本书的勘误，以及下载与本书相关的资源。官网是以 Git 方式维护的，人人都可以参与其中。

- 新浪微博：*http://weibo.com/GotGit*

  欢迎大家通过新浪微博与作者交流，也欢迎大家通过新浪微博将你们的宝贵意见和建议反馈给作者。

致谢

感谢 Linus Torvalds、Junio C Hamano 和 Git 项目的所有贡献者，是他们带给我们崭新的版本控制体验。

本书能够出版要感谢机械工业出版社华章公司，华章公司对中文原创计算机图书的信任让中国的每一个计算机从业者都有可能圆自己出书的梦想。作为一个新人，拿着一个新的选题，遇到同样充满激情的编辑，我无疑是幸运的。这个充满激情的编辑，就是华章公司的杨福川编辑。甚至没有向我索要样章，在看过目录之后就“冒险”和我签约，他的激情让我不敢懈怠。同样要感谢王晓菲编辑，她的耐心和细致让我吃惊，也正是因为她的工作本书的行文才能更加流畅，本书也才能够更快问世。还有张少波编辑，感谢她在接到我的电话后帮我分析选题并推荐给杨福川编辑。

本书的部分内容是由我的 Git 培训讲义扩展而来的，在此感谢朝歌数码的蒋宗贵，是他的鼓励和鞭策让我完善了本书中的与服务器架设的相关章节。还要感谢王彦宁，正是通过她的团队我才认识了 Android，才有了本书关于 repo 和 Gerrit 的相关章节。

感谢群英汇的同事们，尤其要感谢王胜，正是因为我们在使用 Topgit 0.7 版本时遇到了严重的冲突，才使我下定决心研究 Git。

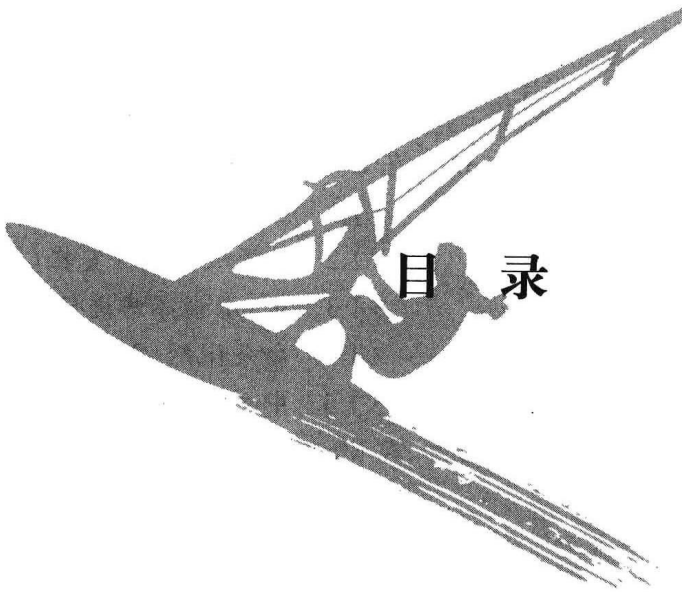
感谢上海爱立信研发中心的高级技术专家蔡煜，他对全书尤其是 git-svn 和 Gitolite 相关章节做了重点评审，他的意见和建议修正了本书的很多不当之处。因为时间的关系，他的一些非常好的观点没有机会在这一版中体现，争取在改版时弥补遗憾。

中国科学院软件研究所的张先轶、比蒙科技的宋伯润和杨致伟、摩博科技的熊军、共致开源的秦红胜，以及王胜等人为本书的技术审校提供了帮助，感谢他们的宝贵意见和建议。来自中国台湾的 PyLabs 团队纠正了本书在对 Hg 的认识上的偏颇，让本书附录中的相关内容更加准确和客观，在此向他们表示感谢。

因为写书亏欠家人很多，直到最近才发现女儿小雪是多么希望拥有一台儿童自行车。感谢妻子阿巧对我的耐心和为家庭的付出。感谢岳父、岳母这几年来对小雪和我们整个家庭的照顾，让我没有后顾之忧。还要感谢我的父母和妹妹，他们对我事业的支持和鼓励是我前进的动力。在我写作本书的同时，老爸正在富春江畔代表哈尔滨电机厂监督发电机组的制造，而且也在写一本监造手册方面的书，抱歉老爸，我先完成了。 :)

蒋鑫（http://www.ossxp.com/）

2011 年 4 月



目 录

第 2 篇　Git 独奏