Git权威指南

作者：蒋鑫

Git权威指南

蒋鑫　著

ISBN：978-7-111-34967-9

本书纸版由机械工业出版社于2011年出版，电子版由华章分社（北京华章图文信息有限公司）全球范围内制作与发行。

客服热线：+ 86-10-68995265

客服信箱：service@bbbvip.com

官方网址：www.hzmedia.com.cn

新浪微博 @研发书局

腾讯微博 @yanfabook

目　录

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[2]。

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[3]。

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。