TURING 图灵程序设计丛书

Apress®

Pro Git, Second Edition

精通Git（第2版）

- GitHub联合创始人倾心之作
- 没有版本控制概念的读者也可轻松入门
- 涵盖Git常见工作场景
- 有效帮助程序员提升软技能

[美] Scott Chacon Ben Straub 著

门佳 刘梓懿 译

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Pro Git, Second Edition

精通Git（第2版）

[美] Scott Chacon Ben Straub 著
门佳 刘梓懿 译

人民邮电出版社
北京

图书在版编目（CIP）数据

精通Git : 第2版 / (美) 斯科特·查康
(Scott Chacon) , (美) 本·斯特劳布 (Ben Straub) 著;
门佳, 刘梓懿译. -- 北京 : 人民邮电出版社, 2017.9
(图灵程序设计丛书)
ISBN 978-7-115-46306-7

Ⅰ. ①精… Ⅱ. ①斯… ②本… ③门… ④刘… Ⅲ.
①软件工具－程序设计 Ⅳ. ①TP311.561

中国版本图书馆CIP数据核字(2017)第165692号

内 容 提 要

Git仅用了几年时间就一跃成为了几乎一统商业及开源领域的版本控制系统。本书全面介绍Git进行版本管理的基础和进阶知识。全书共10章，内容由浅入深，展现了普通程序员和项目经理如何有效利用Git提高工作效率，掌握分支概念，灵活地将Git用于服务器和分布式工作流，如何将开发项目迁移到Git，以及如何高效利用GitHub。

本书适合所有软件开发人员阅读。

◆ 著 [美] Scott Chacon Ben Straub
译 门 佳 刘梓懿
责任编辑 谢婷婷
执行编辑 赵瑞琳
责任印制 彭志环

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
三河市海波印务有限公司印刷

◆ 开本：800×1000 1/16
印张：26
字数：614千字 2017年9月第1版
印数：1－3 500册 2017年9月河北第1次印刷
著作权合同登记号 图字：01-2015-1055号

定价：89.00元

读者服务热线：(010)51095186转600 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

版权声明

致我的爱妻Becky，没有她的支持，我的这次冒险之旅断然无法成行。

——Ben

将本书的此版献给我的爱妻和女儿。致我的爱妻Jessica，感谢你多年来对我的支持；致我的女儿Josephine，感谢你在我落伍得已经不知道周遭变化的时候仍然力挺我。

——Scott

前　言

欢迎阅读《精通Git（第2版）》。本书第1版出版至今已经4年有余。自那时起，太多的东西都已改变，但很多重要的事情依然如故。由于Git核心团队在保持向后兼容性方面不可思议的表现，大多数核心命令和概念在今天仍旧有效，但Git社区中已经出现了一些重大的变化，注入了新鲜的血液。本书第2版旨在讨论这些变化并更新相关内容，以便能够更好地帮助新用户。

在我编写本书的第1版时，就算是对于黑客中的老手，Git也是一件相对难使的工具，极少有人使用。尽管当时它在一些社区中已经开始迅速发展，但远不像如今这样无处不在。从那之后，几乎所有的开源社区都用起了Git。无论是在Windows操作系统上，还是在各个平台的图形用户界面版本数量、IDE支持方面以及商业应用中，Git都取得了令人不可思议的进步。4年前的《精通Git》可没预料到这些。新版本的主要目标之一就是介绍Git社区中所有这些新潮内容。

采用Git的开源社区也呈爆发之势。我差不多是在5年前（有一些时间用在了第1版的出版上）开始编写这本书的，当时我正在一家名不见经传的公司开发一个叫作GitHub的Git托管网站。网站上线的时候，大概也就几千个用户，工作人员只有我们4个。而在我写这篇前言之时，第1000万个托管项目已经落户在GitHub上了，注册的开发者账户接近500万，雇员超过230人。爱也罢，恨也罢，GitHub已经深深地改变了一大部分开源社区，而这种改变方式是我当初编写本书时难以想象的。

我编写了《精通Git》中关于GitHub的一小节内容，但一直不是特别喜欢Git托管方面的东西。我并不太愿意去写一些我觉得其实就是社区资源的内容，也不愿意在其中谈及自己的公司。尽管我仍旧接受不了这种兴趣上的冲突，但GitHub在Git社区中的重要性是无法视而不见的。因此我不打算再写Git托管了，而是选择在这部分内容中更加深入地讲述GitHub是什么以及如何有效地使用它。如果你打算学习如何使用Git，那么了解GitHub的使用方法有助于你参与到一个庞大的社区中，这才是最有价值的地方。至于你用什么Git主机托管代码，其实并不重要。

自本书上一版出版之后，另一个重大变化就是HTTP协议在Git网络事务方面的应用与发展。书中的大部分例子已经从SSH改为了HTTP，因为后者要简单得多。

目睹Git在几年间从一个相对艰涩难用的版本控制系统发展到基本上一统商业及开源领域，着实令人惊奇。很高兴《精通Git》一书能为读者带来不少帮助，另外这本书还是市场上为数不多的完全开源的技术类畅销书之一，这一点也让我颇感欣慰。

希望你能够喜欢这本《精通Git》的升级版。

——Scott Chacon

本书的第一版让我迷恋上了Git。这是我第一次见识到这种打造软件的方式，比我之前碰到过的都要更自然。那时我已经是一名具有多年从业经验的开发人员了，但就是这次正确的转折，把我带上了一条比以往更有乐趣的道路。

多年之后，如今的我是一个大型Git项目的贡献者，曾供职于全球最大的Git托管公司，曾环游世界各地为人们传授Git的知识。当Scott问我有没有兴趣参与本书第二版的编写工作时，我想都没想就答应了。

> 能够成为这本书的作者之一，于我而言是莫大的喜悦和荣幸。希望它能够像帮助我那样对你有所裨益。
>
> ——Ben Straub

对本书做出贡献的读者

作为一本开源图书，几年来我们收到了一些勘误和义务更新的内容。以下是为开源项目Pro Git的英文版做出贡献的所有人员。感谢为提高本书质量给予过帮助的每一个人。

```
 2  Aaron Schumacher
 4  Aggelos Orfanakos
 4  Alec Clews
 1  Alex Moundalexis
 2  Alexander Harkness
 1  Alexander Kahn
 1  Andrew McCarthy
 1  AntonioK
 1  Benjamin Bergman
 1  Brennon Bortz
 2  Brian P O'Rourke
 1  Bryan Goines
 1  Cameron Wright
 1  Chris Down
 1  Christian Kluge
 1  Christoph Korn
 2  Ciro Santilli
 2  Cor
 1  Dan Croak
 1  Dan Johnson
 1  Daniel Kay
 2  Daniel Rosen
 1  DanielWeber
 1  Dave Dash
10  Davide Fiorentino lo Regio
 2  Dilip M
 1  Dimitar Bonev
 1  Emmanuel Trillaud
 1  Eric-Paul Lecluse
 1  Eugene Serkin
 1  Fernando Dobladez
 2  Gordon McCreight
 1  Helmut K. C. Tessarek
31  Igor Murzov
 1  Ilya Kuznetsov
 1  Jason St. John
 1  Jay Taggart
 1  Jean Jordaan
51  Jean-Noël Avila
```

1 Jean-Noël Rouvignac
1 Jed Hartman
1 Jeffrey Forman
1 John DeStefano
1 Junior
1 Kieran Spear
1 Larry Shatzer, Jr
1 Linquize
1 Markus
7 Matt Deacalion Stevens
1 Matthew McCullough
1 Matthieu Moy
1 Max F. Albrecht
1 Michael Schneider
8 Mike D. Smith
1 Mike Limansky
1 Olivier Trichet
1 Ondrej Novy
6 Ori Avtalion
1 Paul Baumgart
1 Peter Vojtek
1 Philipp Kempgen
2 Philippe Lhoste
1 PowerKiKi
1 Radek Simko
1 Rasmus Abrahamsen
1 Reinhard Holler
1 Ross Light
1 Ryuichi Okumura
1 Sebastian Wiesinger
1 Severyn Kozak
1 Shane
2 Shannen
8 Sitaram Chamarty
5 Soon Van
4 Sven Axelsson
2 Tim Court
1 Tuomas Suutari
1 Vlad Gorodetsky
3 W. Trevor King
1 Wyatt Carss
1 Włodzimierz Gajda
1 Xue Fuqiao
1 Yue Lin Ho
2 adelcambre
1 anaran
1 bdukes
1 burningTyger
1 cor
1 iosias
7 nicesw123
1 onovy
2 pcasaretto
1 sampablokuper

引　言

接下来你将花几小时来了解Git。先来说说我们为你准备了哪些内容。下面是对本书所包含的10章正文以及3个附录的简要总结。

第1章介绍了版本控制系统（Version Control System，VCS）以及Git的基础知识，其中并不涉及技术细节，仅仅讲了Git是什么，为什么在已经有了各种版本控制系统的情况下还会出现Git，Git有哪些不同之处，以及为什么有那么多人使用Git。然后讲解了如何下载Git，如何完成初次使用时的设置。

第2章讲述了Git的基本用法：如何使用Git来处理你最常碰到的80%的工作场景。阅读完这一章之后，你应该能够完成仓库克隆、查看项目历史、修改文件以及贡献变更等操作了。如果学完这一章时这本书开始自燃，那么光是去换本书的工夫就够你熟练掌握Git了。

第3章分析了Git的分支模型，它经常被描绘成Git的杀手级特性。在这一章中，你会明白究竟是什么使得Git与众不同。学完这一章后，你可能需要花点时间好好想想之前没有Git分支的时候自己是怎么过来的。

第4章探讨了Git服务器。这一章的目标读者是那些想在组织内部或是个人服务器上搭建Git以展开协作的用户。如果你倾向于让别人代劳，我们也探究了各种托管做法。

第5章详述了各种分布式工作流以及如何使用Git实现这些流程。读完这一章之后，你应该能够熟练地处理多个远程仓库，利用电子邮件使用Git，以及熟练使用各式远程分支和补丁。

第6章深入叙述了GitHub托管服务以及工具的用法。这一章的内容包括：账户的注册和管理，Git仓库的创建和使用，为项目做贡献和接纳他人贡献的常见工作流，GitHub的可编程接口，以及可以让你的日子过得更轻松的大量小技巧。

第7章涵盖了Git的高级命令。在这里，你会学到多个主题，比如掌握让人提心吊胆的`reset`命令、利用二分搜索法确定bug、编辑历史记录、修正版本选择的细节等。这一章将为你的Git求知之旅画上一个句号，使你成为真正的高手。

第8章介绍了如何配置自定义的Git环境。主要内容包括编写钩子脚本来强制或推动自定义策略，以及利用环境配置选项建立适合个人的工作方式。另外还讲解了如何建立自己的脚本集，强制实施自定义的提交策略。

第9章探讨了Git以及其他的版本控制系统。内容包括在Subversion（SVN）下使用Git，以及将采用其他版本控制系统的项目转换到Git。很多组织仍旧在使用SVN，也并未打算做出改变，这时你就会发现Git难以置信的威力。如果你不得不使用SVN服务器，那么这一章会向你展示如

何应对这种情况。要是你打算说服大家尝试一下Git，我们也讲到了如何从其他系统中导入项目。

第10章深入讲解了晦涩难懂但又充满吸引力的Git内幕。到这个时候，你已经了解了Git的所有内容，能够充分、优雅地使用它了，因而可以学习下列内容了：Git存储对象的方式、对象模型是什么、包文件的细节、服务器协议等。在本书中，我们会提到这一章的各个部分，以便应对你深入了解某个知识点之需。但如果你像我们一样热衷于技术细节，可以先阅读这一章。具体的阅读方法由你自己决定。

在附录A中，我们介绍了Git在各种环境下的一些用例，其中涵盖了各种可以使用Git的GUI和IDE编程环境以及可用的工具。如果你对在shell、Visual Studio或Eclipse中使用Git感兴趣，那么可以阅读这个附录。

在附录B中，我们研究了如何利用Libgit2和JGit这类工具实现Git的脚本化及扩展。如果你对编写复杂高效的自定义工具感兴趣，并且需要对Git执行低层操作，那么你得看看这个附录。

最后，在附录C中，我们列出了所有重要的Git命令，回顾了书中涉及这些命令的章节以及命令的用法。如果你想知道某个命令出现在本书中的哪一部分，可以在这里查找。

好了，让我们开始这段Git学习之旅吧。

目　录

第1章 入 门

本章主要介绍Git的入门知识。我们首先会讲述版本控制工具的一些背景，然后介绍如何在你自己的系统上安装、配置和运行Git。学完本章，你将明白Git是怎么来的、为什么需要Git，并掌握使用Git的基础知识。

1.1 关于版本控制

什么是“版本控制”，为什么需要它？版本控制是一套系统，该系统按时间顺序记录某一个或一系列文件的变更，让你可以查看其以前的特定版本。本书以软件源代码文件为例讲解了版本控制的方法，但实际上这种方法对于计算机上几乎所有文件类型都适用。

如果你是一位平面或网页设计师，那么可能（几乎必然）想要保存一幅图片或一个布局的每一个版本，这时使用版本控制系统（VCS）就是非常明智的选择。使用版本控制系统，你可以将文件或整个项目恢复到先前的状态，还可以比对文件随时间的变更，查看什么人最后做出的更改可能会造成麻烦，谁在何时引入了一个问题，等等。使用版本控制系统通常意味着，如果你把事情搞砸了或是弄丢了文件，都可以轻而易举地恢复原状。而且，你要为所有这些福利付出的开销也很低。

1.1.1 本地版本控制系统

很多人控制版本的方法是将文件复制到另一个文件目录下（如果他们够聪明，还会给目录加上时间戳）。这种做法之所以常见，是因为它实现起来非常简单。然而它又非常容易导致错误，你很容易忘记当前所处的目录，不小心写入了错误的文件，或是把不该覆盖的文件给覆盖掉了。

为了解决这个问题，开发人员在很久以前就开发了一些本地版本控制系统，使用简单的数据库来保存文件的所有变更。

RCS是一个常用的VCS工具，至今还部署在很多计算机上。在流行的Mac OS X操作系统中，只要你安装了开发者工具，就会包含一个`rcs`命令。RCS会在磁盘上以一种特殊的格式保存补丁集（patch set，也就是文件之间的差异）。通过叠加补丁来将文件恢复到某个历史状态。

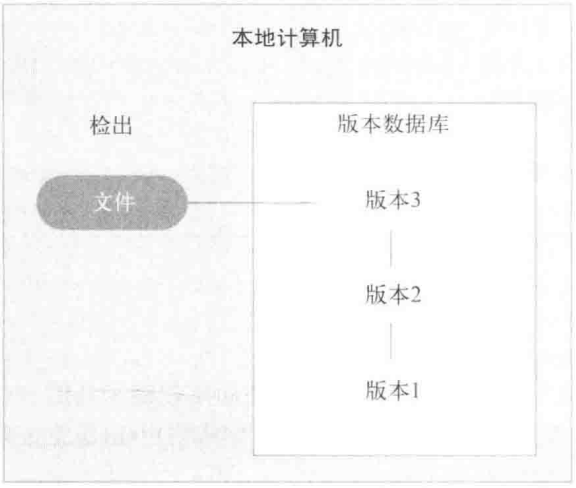

图1-1　本地版本控制

1.1.2　集中式版本控制系统

另一个主要的问题，是不同系统上开发人员之间的协作。为了解决这个难题，集中式版本控制系统（Centralized Version Control System，CVCS）应运而生。像CVS、Subversion以及Perforce这类系统，都有一个包含文件所有修订版本的单一服务器，多个客户端可以从这个中心位置检出文件。多年以来3，这已成为版本控制的标准。

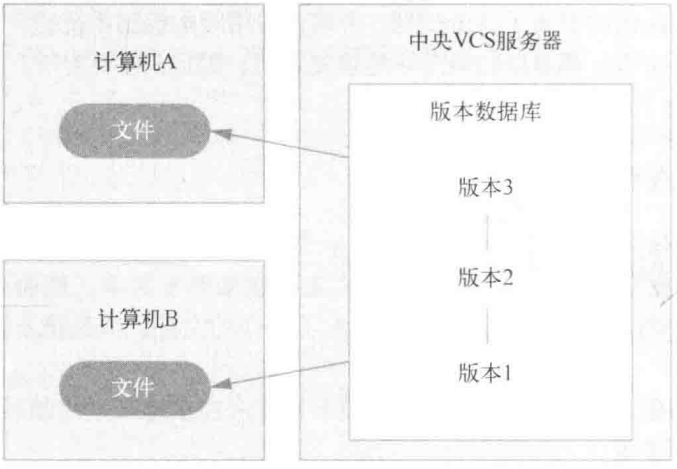

图1-2　集中式版本控制

这种方案有多方面的优势，尤其是与本地版本控制系统相比。例如，所有人都可以在一定程度上掌握其他人在项目中都做了什么，管理员可以精细地控制每个人的权限；同时，维护一个集