GOTOP

完全学会 Git GitHub Git Server 的 24 堂课

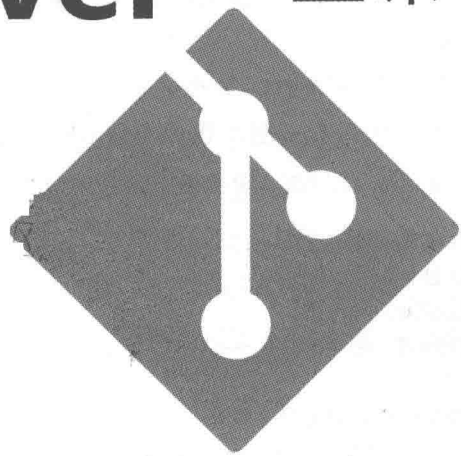
- 针对项目开发人员：以最有效率的方式，引导你在实践中完整地学习Git的操作技巧和应用。
- 针对项目管理者：在实例中学习如何查看项目开发的各种统计图，以便随时掌握项目开发的状况。
- 针对MIS人员：提供建立Git Server的多种方法，自行选用和规划最适合自己实际运用的方法。

清華大學出版社

孙宏明 著

完全学会 Git GitHub Git Server 的 24 堂课

孙宏明 著



清華大學出版社
北京

北京市版权局著作权合同登记号：图字：01-2016-2268
本书为碁峰资讯股份有限公司授权出版发行的中文简体字版本。

内 容 简 介

本书以教科书的方式，循序渐进地向软件开发人员或软件开发的项目管理人员，讲述如何快速掌握分布式版本控制系统 Git 的应用方法，让繁琐复杂的项目开发的版本控制变得高效且轻松自如。书中细致入微地介绍了如何使用设置文件和操作 Git 文档库，以及建立项目开发的分支、合并分支和解决冲突的操作；在具备扎实的基础之后，介绍了 Git GUI 程序用法，以及 Git 在团队开发模式中应用的问题；讲述如何与全球的项目开发人员进行交流，提升程序开发的经验和能力；介绍了 GitHub、Bitbucket、GitLab 等提供 Git 服务的网站；本书最后的重点为介绍架设 Git Server 的几种方法，让每一个项目开发团队，可以按照自己的软硬件环境，选择适合的方式来控管项目开发。

图书在版编目（CIP）数据

完全学会 Git, GitHub, Git Server 的 24 堂课/孙宏明著. —北京：清华大学出版社，2016

ISBN 978-7-302-43779-6

Ⅰ. ①完… Ⅱ. ①孙… Ⅲ. ①软件工具－程序设计Ⅳ. ①TP311.56

中国版本图书馆 CIP 数据核字（2016）第 100121 号

责任编辑：夏毓彦
封面设计：王　翔
责任校对：闫秀华
责任印制：何　芊

出版发行：清华大学出版社
网　址：http://www.tup.com.cn，http://www.wqbook.com
地　址：北京清华大学学研大厦 A 座　　**邮　编**：100084
社总机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：三河市春园印刷有限公司
经　　销：全国新华书店
开　　本：170mm×230mm　　**印　张**：17　　**字　数**：237 千字
版　　次：2016 年 6 月第 1 版　　**印　次**：2016 年 6 月第1 次印刷
印　　数：1～3000
定　　价：49.00 元

产品编号：069428-01

目 录

第一部分 Git 的基本概念和操作

第二部分　建立分支、合并和解决冲突

第三部分　Git 程序的图形操作介面

第一部分　Git 的基本概念和操作

第 1 课

Git、谁与争锋

“版本控制系统”（Version Control System, VCS）是程序代码管理软件的通称，是用来保存程序文件的修改记录以及历史版本，以便日后查看或是使用。VCS 已经有数十年的发展历史，最早期的系统是采用集中管理和控制的方式。如果要修改某一个程序文件，必须先将它锁定，然后取出修改。在完成修改和回传之前，其他人都不能更改这个程序文件。这种方式可以有效避免“冲突”（conflict）的发生（也就是防止不同人同时修改同一段程序代码所造成混淆的情况），但是付出的代价是效率降低。因为如果想要修改的文件正好有别人正在修改，就必须等程序文件回传之后才能动手。如果要修改的人很多，因为互相等待所造成的时间浪费将非常可观。为了解决这个问题，新的 VCS 改成采用分布式的方法。每一个人随时都可以获取任何一个程序文件来进行修改，等到送回 VCS 的时候，再根据需要进行“合并”（merge），Git 就是采用这种分布式技术的系统。

当前市场上占有率最高的 VCS 有二套：Subversion（简称 SVN）和 Git。在 2013 年以前，SVN 的市场占有率维持在 40%以上。但是之后，Git 的使用率开始接近 SVN，至 2014 年以后就超越了 SVN。现在越来越多的软件公司开始使用 Git 管理程序项目，网络上也有专门提供 Git Server 服务的网站，像是 GitHub 和 GitLab。许多开放源码的程序项目，例如 Linux 和 Android，也都是使用 Git 管理程序代码的，在职场上要求具备 Git 应用能力的情况也越来越普遍。如果你的 Git 功力深厚，甚至具备架设和管理 Git Server 的能力，在职场上必定更受青睐。不过在开始学习 Git 之前，让我们先了解一下 Git 是如何诞生的。

Git 的开发者其实就是 Linux 的作者，也就是鼎鼎大名的 Linus Torvalds。一开始的时侯，Torvalds 是使用 BitKeeper 软件来管理 Linux 的程序代码，该软件原来是免费使用的，但是后来却要收费。于是 Torvalds 想要更换 VCS，却苦于寻找不到其他的替代方案可以符合开发 Linux 的需求。因为参与开发 Linux 的人数高达上千人，而且分布在世界各地。最后 Torvalds 决定自己打造一个 VCS。经过短短几个星期，第一版的 Git 就正式诞生。Linux 项目从此用 Git 管理，当时有将近七百万行的程序代码！

由于 Git 管理 Linux 项目非常成功，许多公司于是纷纷效仿，开始使用 Git 来管理和控制软件研发项目。这些成功的经验，让 Git 的影响力越来越高，使用率也快速提升。

1-1 安装和使用 Git

每一个操作系统平台都可以找到适合的 Git 安装程序，以下是官方网站的下载网址：

http://git-scm.com/downloads

除了官方的版本，也有其他软件公司将 Git 包装成 GUI 操作界面类型的程序（官方版本主要是使用指令模式进行操作）。虽然 GUI 的图形用户操作画面看起来比较吸引人，但是建议初学者还是先从 Git 的指令模式开始学习，这样才能够清楚地了解 Git 的工作细节。等到具备完整的应用基础以后，再转换到 GUI 界面，自然就能够水到渠成。本书将带领读者从基础的指令操作开始，一直到 GUI 的操作模式以及各种情况的应用，最后还会介绍 Git Server 的架设和管理，让读者能够熟练使用 Git 来管理程序代码，从而提高程序项目的开发效率，现在就让我们开始动手吧。

1 步骤 安装 Git。如果是 Windows 平台，直接运行下载的安装程序。在安装选项设置页面中勾选 Git Bash Here 和 Git GUI Here（参见图 1-1），然后按照说明完成安装。其他操作系统平台请参考网页上的说明来完成安装。

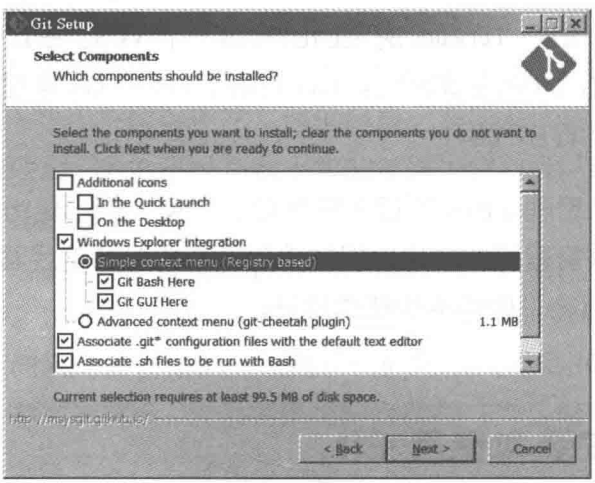


图 1-1　Windows 平台安装 Git 时的选项

2 步骤 安装完毕之后，我们就来体验一下 Git 的操作方式。先创建一个新的文件夹，文件夹名称可以使用中文或是英文。

3 步骤 启动 Git Bash 程序，使用 cd 指令切换到前一个步骤创建的文件夹：

```
cd '文件夹路径'
```

如果文件夹路径中包含空格，必须用单引号或是双引号将路径括起来，如以上指令格式。完成之后就会看到如下信息：

```
用户名称@计算机名称 文件夹路径
$
```

4 步骤 现在，我们要让 Git 开始管理这个文件夹，请执行以下指令：

```
git init
```

Git 会响应消息“Initialized empty Git repository in 文件夹路径/.git/”，表示已经完成准备工作。

补充说明

1. Git 可以管理任何一个文件夹中的文件和子文件夹，只要在该文件夹中执行“git init”，就可以让 Git 完成管理前的准备工作。Git 会在这个文件夹创建所谓的 repository（我们把它称为“文档库”），里面存储被管理的文件和文件夹内容，包括所有曾经被加入的历史版本。
2. “文档库”其实是名称叫作“.git”的子文件夹，默认它会被隐藏起来，我们可以改变文件夹的查看选项，让隐藏的文件和文件夹也能够正常显示。
3. Git 可以正确处理中文名称的文件和文件夹，只是 Git Bash 程序在显示中文的时候会出现乱码，后续我们会介绍 gitk 图形查看模式，它可以正确显示中文。
4. 单独执行“git”指令会显示辅助说明；执行“git help - a”则显示完整的指令列表；执行“git 指令 - help”（例如“git ini t - help”）则会显示该指令的网页说明文件。
5. 如果指令太长，想要换到下一行继续输入，可以用反斜杠字符“\”结尾，然后按下 Enter 键，继续输入。

5 步骤 现在我们创建一个纯文本文件来测试 Git 如何管理文件，请将这个文本文件取名为 poem.txt，在里面输入一行文字如下（这是唐朝诗人“王维”的《山居秋暝》）：

```
空山新雨后
```

6 步骤 回到 Git Bash 程序，依序执行以下两个指令：

```
git add poem.txt
git status
```

Git 会响应以下信息：

```
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

      new file:   poem.txt
```

这些信息告诉我们，现在有一个名称叫作 poem.txt 的新文件将要被送进“文档库”存储。其实这是执行第一个指令，也就是“git add poem.txt”的结果，第二个指令只是显示当前 Git 的状态。

7 步骤 前一个步骤是准备把文件送进 Git 文档库，现在我们要确实把它送进去，请继续执行：

```
git commit -m '这次操作的说明' --author='操作者姓名 <email 邮箱>'
```

Git 会响应以下信息，显示是谁执行了这项工作，以及有几个文件被改动了：

```
[master (root-commit) 57d02a7] first commit
 Author: 操作者姓名 <email 邮箱>   ← 谁执行这项工作
 1 file changed, 1 insertion(+) ← 有一个文件被改动了，总共加入一行文字
 create mode 100644 poem.txt    ← 被改动的文件名
```

如果执行 commit 指令以后，想要修改操作说明或是操作者的资料，可以使用“--amend”选项，也就是：

```
git commit --amend -m '新的操作说明' --author='操作者姓名 <email 邮箱>'
```

8 步骤 接下来修改一下 poem.txt 文件的内容，请将它打开，加入第二行文字如下，再将它保存。

```
空山新雨后
天气晚来秋    ← 这是新加入的文字
```

9 步骤 回到 Git Bash 程序，按序执行下列指令，将修改后的 poem.txt 文件送进文档库存储。

```
git add poem.txt
git commit -m '这次操作的说明' --author='操作者姓名 <email 邮箱>'
```

10 步骤 现在让我们看一下当前文档库中有哪些文件，请执行以下指令，启动图形查看模式：

```
gitk
```

屏幕上会显示图 1-2 所示的画面，左上角会有不同颜色的小节点，每一个节点表示执行一次“git commit”指令。节点右边会显示我们输入的操作说明的第一行文字，左下角窗口会显示当前选定节点的完整信息。最上面的节点会多出一个 master 标签，表示这是文件的“主要分支（branch）”。节点会按照时间顺序从上往下排列，最上面是最近一次的 commit。节点窗口右边还有两个窗口，分别显示执行指令的人和时间。右下角的窗口是当前选定的 commit 节点的内容，它有两种查看模式：patch 和 tree。patch 模式会显示和前一个 commit 节点的差异，tree 模式则显示完整的节点内容。我们可以用鼠标单击要查看的文件，左边的窗口会显示它的内容。读者可以在左上角的窗口单击不同的节点，再使用 tree 模式观察 poem.txt 文件

的内容。到当前为止，我们已经把两个不同版本的 poem.txt 文件送进文档库存储。

补充说明

文件可以有一个以上的“分支”，假如有两个人同时想要修改同一个文件，可以在不同的“分支”上进行，稍后我们可以将不同的“分支”合并，后续我们会再详细介绍这种操作模式。

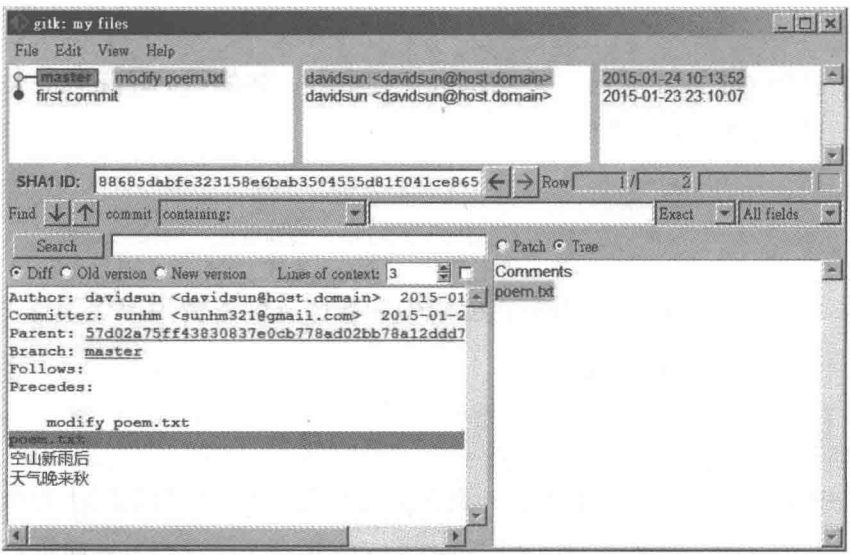


图 1-2　用图形查看模式查看文档库

11 步骤 查看完文档库之后，结束图形操作模式，回到 Git Bash 程序。然后执行以下指令离开 Git Bash。Git 会把当前的执行状态记录在文件夹内，下次再启动 Git Bash 程序，回到这个文件夹的时候，会自动恢复到离开时的状态。

```
exit
```

我们已经完成了第一次 Git 探索之旅，实际体验 Git 的操作方式，也学会了几个指令的用法，或许还有点似懂非懂的感觉，没关系，在下一个小节我们会进一步解释 Git 工作的方式。

1-2 了解 Git 的工作方式

操作 Git 的基本流程就是先修改文件，然后执行“git add”指令。“git add”指令会把文件内容加入 Git 系统的索引。接着就可以执行“git commit”指令，将文件内容存入文档库。于是文档库中就多出一份文件的新版本，我们可以把整个过程用图 1-3 来描述。如果同时要把多个文件送进文档库存储，可以在 git add 指令后面逐一列出这些文件的名称（用空格符隔开），也可以使用通配符“*”，或是执行多次“git add”指令，最后再执行“git commit”。“git add”指令后面也可以指定文件夹名称，这样该文件夹内的文件都会被处理。如果执行“git add .”指令（“.”表示当前这个文件夹），则全部的文件和子文件夹都会被处理。如果想要忽略特定文件，可以在文件夹中创建“.gitignore”文件，后续我们会介绍它的用法。

补充说明

有些 Git 指令执行的时候会显示英文信息，里面可能包含 Git 专用的术语，以下是这些专用术语的说明：

1. working tree：就是当前 Git 正在管理的这个文件夹，因为文件夹是一个树状结构，所以 Git 将它称为 working tree。
2. index：就是前面所说的“Git 系统索引”，把文件内容加入 Git 索引又称为 stage 或是 cache，把文件内容从索引中删除则称为 unstage。
3. repository：就是“Git 文档库”，也就是 Git 存储文件的地方。