文章编号：2095-2163(2019)05-0341-03 中图分类号：TP315 文献标志码：A

Git 版本控制工具在团队协作项目中的应用

徐 娅
（温州商学院，浙江 温州 325035）

摘 要： 在大型团队协作项目中，对源代码和文档进行版本控制是有必要的，本文对版本控制以及 Git 如何在团队协作中使用进行了探讨。

关键词： Git；版本控制；团队协作

Application of Git version control tool in team collaboration project

XU Ya
（Wenzhou Business College, Wenzhou Zhejiang 325035, China）

【Abstract】 In large-scale team collaborative projects, it is necessary to realize version control of the source code and documents. This paper discusses version control and the use mode of Git in team collaboration.

【Key words】 Git; version control; team collaboration

0 引 言

随着社会发展与科技进步，软件的功能渐趋复杂，在一个软件项目开发的过程中，团队协作必不可少。因为团队的每个成员有着不同的兴趣和擅长的领域，所有的工作都需要分工并行，这种情况下就需要一种技术。该技术可以跟踪和记录每位团队成员对文档与代码的更改，辅助团队成员进行多个版本分支的协同开发及维护，最终合并到主分支进行部署和发布。此种技术就是版本控制技术，而 Git 就是一种版本控制工具。

1 版本控制

1.1 版本控制简介

版本的概念最早在印刷行业出现，由于编辑、印刷不同而衍生出不同样式和内容，产生了不同的成品，这就是版本。版本控制技术是一种可以记录一个或多个文件在修改过程中的变化，使得后续可以查阅文件的历史版本的技术。当一个项目的代码量达到一定规模时，对源代码的管理难度和重要性也会随即增加。使用版本控制可以有效地管理和维护软件的项目开发，因其保存了从项目研发开始到结束的所有历史提交和修改记录。

1.2 版本控制操作

在一个项目从开始需求分析起，历经设计、代码实现、测试和维护的所有环节的整个过程中，会产生不同版本的源代码和文档，此时可以对不同版本进行一系列操作，对此可做探讨分述如下。

（1）版本生成。在团队协作的过程中，每位成员可以将自己修改好的文件生成一个新的版本。

（2）版本提交。生成后的版本可以提交到远程，但需要进行合并冲突的审核。

（3）版本审核。团队成员可能对文件的同一处内容进行了修改，这个时候文件的合并就会产生冲突，因此要进行版本审核。

（4）版本修改。提交的版本若是没有通过审核，则要进行修改。

（5）版本提取。如果版本的修改发生了错误，就要恢复到之前的版本，此时可以使用版本提取，恢复为版本之前的状态。

（6）版本比较。在合并版本的过程中，常常需要对版本进行比较，来选取一个最佳版本进行恢复。比较的方式有 2 种。一种是通过时间进行比较，一种是通过空间进行比较。其中，通过时间进行比较是指用户可以对比不同时间发布的修改内容，通过空间进行比较是指比较版本的不同分支。

（7）版本合并。对于同一个文件，不同的团队成员可能会产生各自的修改，因而可以将处于不同分支的成员的修改进行合并。

1.3 版本控制的优势

（1）方便代码管理。版本控制技术可以避免代码被项目组其它成员无意间覆盖，还可以避免代码的遗失。版本控制技术可以对不同版本进行跟踪、记录和标识，也可以恢复到用户所需的版本。

作者简介：徐 娅（1997-），女，本科生，主要研究方向：大学生协作能力研究。

收稿日期：2019-05-20

(2)自动化。传统的版本管理可以采用如下方法:将自己的文件生成一个副本,将改动的时间点保存,并加上一个版本号,这种形式虽然方便,但是效率低下,因为不同文件的内容十分接近,不易找出两者之间的差异,如果通过人工记忆多个版本之间的差异,一段时间后也容易产生混淆,而版本控制可以做到版本之间差异的自动化比较,无需人工进行查找。另外,在解决冲突时,如果没有版本控制工具,每一位成员需要自己对冲突文档予以逐行查阅,这大大增加了工作量,并且在修改的过程中若是不慎还容易带来新的问题,因此版本控制在解决冲突时也更为方便。

(3)实现了分工和并行。空间上统一管理,时间上全程记录。版本控制技术对团队协作项目的开发有辅助作用,一个大型的软件项目的任务往往是分工并行协作的,需要多个人同时修改一个文件,而版本控制技术实现了团队成员任务在不同地域的同步开发,突破了时间和空间的局限,因此大大缩短了项目开发的周期,提高了项目研发的效率。

(4)访问控制。可以阻止未经授权的更改和查看。如果所有的成员都可以不受任何限制地进行代码的添加、修改和删除操作,没有明确的代码管理人,将会带来严重的后果。

1.4 版本控制系统分类

版本控制系统分为2类。一类是集中式版本控制系统,另一类是分布式版本控制系统。其中,集中式版本控制系统是指软件项目的所有代码的修改和提交都存储在一台中央服务器,本地的版本控制系统只保存最新的文件快照,所以每位开发人员在开始工作前,必须要从中央服务器同步最新的项目版本数据信息来避免版本冲突。集中式版本控制系统的缺点就是对中央服务器的依赖性很高,中央服务器的单点故障将会带来严重的后果。如果中央服务器出现宕机,所有人都无法进行文件更新的提交;如果中央服务器的硬盘出现损坏,那么就可能永久性地丢失所有的存储信息。与集中式版本控制系统相对应的是分布式版本控制系统,这是指每个参与者都在本地有自己的版本库,而且也没有中央服务器,对代码的修改信息除了保存在远程仓库,每个参与者在本地都有自己的一份完整的镜像文件,即使服务器出现了巨大的损坏,也可以通过镜像文件进行恢复,而且,同时也支持着离线工作。

1.5 Git

Git是一种典型的分布式版本控制系统,为了更好地管理Linux内核,在2005年由Linus Torvalds首次提出。Git具有使用简便、开源、速度快、保护数据完整性的特点,并且提供了对非线性开发模式的支持,有强大的分支管理能力,允许上千个分支同步并行开发,分支的开发不影响主分支。首先,软件项目的开发过程可能会经常出现功能和性能上的问题,漏洞是可追溯的,出现在某些版本中。使用Git的分支管理功能,通过检查不同版本,定位版本的漏洞的位置,再对这个版本的漏洞进行修复。其次,可以利用一个新的创建分支对软件进行测试性修改,用来验证本文设计的新算法是否可以让文中的软件项目给用户带来更好的使用体验,最后,可以利用Git的分支管理功能在不同的分支上给软件设计不同的功能,因此各个功能相互独立,如果后续研发中不再需要此项功能,可以回退到之前的版本,其它的功能分支也不会因此受到影响。Git采用快照模式,也就是说Git记录数据在某个时间点的映像,而不是记录提交前后数据的具体差异,并且采用校验和压缩,因此恢复起文件来要更为快速。Git再通过SHA-1算法对数据内容进行校验和计算,并将结果作为数据唯一的标识,所以,Git可以检测得到任何数据的修改和删除,这保证了数据的完整性。Git凭借自身的卓越性能,已广泛地应用于众多的著名开源项目,如Ruby on Rails、Perl、Eclipse等,是大型项目开发的首选工具。

1.6 Gitee

目前,随着版本控制系统的不断发展和完善,产生了品类众多的衍生品,其中之一就是代码托管平台。人们可以在代码托管平台上对代码进行下载,也可以将自己的代码交付给代码托管平台进行托管。比较著名的代码托管平台有GitHub和Gitee,Gitee又叫码云,是由开源中国社区在2013年推出的基于Git的代码免费托管服务。该服务基于GitLab、且又在其基础上做了大量的改进和定制开发,而与GitHub比起来则拥有更快的访问速度,当前的开发者超过300万,托管项目超过500万,汇聚了本土的几乎所有原创开源项目,并于2016年推出企业版,提供企业级代码托管服务,成为开发领域领先的SaaS服务提供商,也是目前国内最大的代码托管系统。Gitee除了提供最基础的代码托管之外,还提供代码在线查看、历史版本查看、Fork、Pull Request、打包下载任意版本、Issue、Wiki、保护分支、代码质量检测、PaaS项目演示等方便管理、开发、协作、共享的功能。对于一个开发者来说,一个代码托管平台必不可少,在使用Git进行团队协作项目的版本管理时,也需要和团队其它成员共同使用Gitee来对项目的仓库进行管理。

2　Git 在团队协作中的使用

2.1　使用过程

(1)每一位团队成员进入 Gitee 官网(https://www.gitee.com/)注册一个账号,并生成自己的 SSH Key。

(2)团队中一位成员 git init 在本地建立项目仓库,并使用 git remote add origin git 将远程仓库和本地仓库相关联。将其它成员的 SSH Key 添加到 Gitee。

(3)团队中的其它成员使用 git clone 命令将远程仓库的项目复制到本地,可使用多种网络协议,如 SSH 协议、https 协议。研究可知,SSH 协议的速度最快。

(4)对项目文件进行修改,将只会限定在自己的分支上,使用 git commit 在本地进行提交,这样就把修改保存到了暂存区,如果是暂存区还没有的文件,则需要使用 git add 将项目文件先添加到暂存区。

(5)使用 git push 命令将暂存区的文件推送到远程仓库。如果团队的其它成员在远程分支上有了更新的提交,则需要使用 git pull 将属于同一项目的远端仓库与同样属于同一项目的本地仓库进行合并,该过程包含了 2 个操作:从远端仓库中取出更新版本,再合并到本地仓库。如果合并发生了冲突,则先解决冲突并在本地提交。

(6)每位个体的工作应避免直接在主分支上展开,当每个人的项目有必要加以合并时进行分支合并,发布这个版本,这个版本是较为稳定的。

2.2　协作过程中的冲突解决和避免

代码冲突是多人协作开发过程中最常见的问题,一旦解决不当,会引发代码丢失或者业务功能异常等问题,所以在系统功能设计上,要做到功能解耦,并且在不同的文件中实现不同的功能。Git 会提示合并发生了冲突,此时可以得知发生冲突的文件,Git 还会通过不同的符号来标记不同分支的内容,故而用户将需要手动解决冲突。在开发过程中,也要培养良好的开发习惯,如果必须面临多位成员同时修改一个文件的情况,则每一位成员要定期执行 Git pull 命令,如果出现了冲突则要尽早解决,随着冲突解决时间的延长,冲突解决的难度也会随之增大。

2.3　不同团队人数的协作模式探讨

在不同的团队人数下,可以采用不同的协作模式。当团队人数为 2 时,如果 2 个人中有一个人 A 为主要开发者,另外一个人 B 为辅助者,假设 A 的机器地址为 a,B 的机器地址为 b,项目的仓库建立在 A 的机器上,2 个人的工作在 A 的机器上合并,则 B 可以通过 git clone 将远程仓库的项目复制到本地,而 A 执行 git pull b 将远程仓库合并到本地。但当团队人数为 3 及以上时,假设第三位开发者为 C,机器地址为 c,如果 C 和 B 采取一样的操作,则 A 需要执行 git pull c 和 git pull c 命令将仓库合并,此时 A 的工作量相对较大。如是 A 只负责本地仓库,而 B 和 C 先执行 git pull a,再执行 git push,则 B 和 C 的工作量较大,因此较为合适的方法就是将项目仓库建立在实验室的服务器上,A 通过 SSH 登录服务器后,建立一个目录,初始化一个空仓库,并将自己本地已经接受 git 管理的仓库推送到远程,B 和 C 再通过 git clone 将远程仓库的项目复制到本地,则服务器上的仓库会记录每一位用户涉及的版本更新提交,因此,提取和推送的工作并不会总是由一人或两个人来进行。综上可知,这种模式分担了团队其它成员的工作量。

3　结束语

在团队协作项目中,使用 Git 进行版本控制,减少了开发者大量不必要的工作,可以对项目进行高效的管理。如果没有版本控制工具,任何一种不可逆的操作都会给版本的回退带来困难。Git 保证了版本的可溯性,并且可以在某个版本的测试和使用的过程中出现问题时可以尽早定位问题发生的时间节点、原因和负责人。一个复杂的项目常常会遇到频繁地修改源代码的场景,使用 Git 进行版本控制可以有针对性地解决源代码混乱的问题。Git 分布式特性和分支管理能力使得大型软件项目尤为适合使用 Git,每个分支彼此独立,互不影响。Git 免费、开源的特点正吸引着来自世界各地的大量优秀开发人才。Git 有着可观的应用潜力,而且势必会不断地趋于成熟与完善。

参考文献

[1] 秦佳. 软件配置管理中版本控制的研究[J]. 软件,2019,40(3):137-139.

[2] 李志杰. 版本控制技术在团队协同开发中的应用研究[J]. 现代商贸工业,2012(14):168-169.

[3] 朱守园,王婷. 基于 GIT 的软件开发模式探究[J]. 信息通信,2017(3):117-118.

[4] 字凤芹. 基于 Git 的协作小组学习资源库的建设与研究[D]. 昆明:云南大学,2016.

[5] 王远. 基于 MFS 的校园安全同步网盘设计与实现[D]. 长沙:国防科学技术大学,2013.

[6] 诸葛文社. Git 仓库代码演化过程中的注解定位技术研究[D]. 南京:南京邮电大学,2018.