# 《开源软件设计与开发》课程总结

姓名：刘映君　　学号：51194507010

## 1. Git 工具的理解

Git 本质是一个分布式版本控制系统，客户端可以完成的 Clone 整个仓库，然后进行修改和提交。这个好处是任何一个仓库出现问题都有其他的镜像来进行回复。每次提取操作都是对仓库的完备备份。

早期的 Linux 开发者使用的是一个叫做 BitKeeper 的工具来进行代码维护和管理。但是 2005 年 BitKeeper 和 Linux 开源社区的合作关系结束，收回了 BitKeeper 的使用权利，让 Linux 开源社区受到重大影响，所以为了防止类似情况，天才们开发了 Git，并且制定了一定的规范和功能标准。Git 适合分布式开发，强调个体；公共服务器压力和数据量都不会太大；它的速度快，十分灵活；可以很容易地在任意两个开发者之间解决冲突；可以离线工作。但是 Git 在模式上比 SVN 更加复杂；它不符合常规思维；代码的保密性差，一旦开发者把整个库克隆下来就可以完全公开所有代码和版本信息。

Git 的本质思想是直接记录快照而不是差异比较。也就是说 git 只关心文件数据的整体是否发生变化。Git 把整个项目看作是一个完整的文件系统或者说是包，只有当这个整体发生变化的时候，git 才会进行整体的 cover 或者 clone 等操作。基于这种思想，Git 的优点如下。

Git 上几乎所有的操作都是本地执行。基本除了推送和拉取以外的其他所有版本操作都是离线完成，也就是我们可以在任何离线的平台上进行相应的版本管理，只要在未来连接网络的时候整体推送手头这个版本就可以。

Git 可以时刻保证数据的完整性。因为 Git 把整个 Repo 看成一个整体，任何时候 Git 只要算出本地文件整体的 SHA-1 哈希值，就可以确认文件的异同。而所有的 Git 工作也是依赖于这类指纹子串。所以 Git 可以很好地差异验证和文件系统结构内容的分离。

Git 任何时候的操作仅仅是把数据添加到数据库而已，只要有定时的推送和拉取，根本无所谓数据丢失不丢。

在 Git 内部，文件只有三种状态，Committed、Modified、Staged。Committed 标识已经被安全保存在了本地数据库中，Modified 标识修改了某个文件，Staged 标识已经把改好的文件放在下次提交保存的清单中。

Git 主要使用四种协议：本地传输协议、SSH 传输协议、GIT 传输协议和 HTTP 协议。

本地协议：例如一群人在同一台服务器上维护编辑一个项目，则可以使用“git clone”命令，此时就是用了本地传输协议。SSH 协议是非常常用的，我们通过“git clone [ssh://user@server/project.git](ssh://user@server/project.git)”指令远程 clone 仓库。GIT 协议是包含在 git 安装包里面的一个特殊协议，使用一个特殊端口 9418。repo 要支持 git 协议，需要创建一个 git-daemon-export-ok 文件，但是这个协议一旦开放，任何人可以用任何 URL 进行推送，所以非常不安全。HTTP/S 协议：这个协议非常优雅，直接把 repo 放在 http 根目录下，配一个特定的 post-update 挂钩 hook 就可以。然后每个有权限访问服务器上 web 服务器的用户都可以进行 clone，是一个安全高效轻量的协议。

对于生产环境来说，Git 服务器架设最困难的部分在于账户管理。因为一个项目或者一个企业的生产环境对于不同用户的权限和 repo 权限的分别管理非常重要。如果能和 Amazon 的权限设定功能一样完备就再好不过了。这个时候如果是小型企业，一般推荐直接启用 SSH 白名单模式。具体方法参考知识库 SSH 的内容即可。

## 2. 开源贡献

本人参加的项目是“TiDB”，在课程学习阶段，提交了一个 PR 并且成功合并，PR 截图如图 1 所示，链接为：https://github.com/pingcap/tidb/pull/13316

本人首先在给出的 issue 内选择了感兴趣的函数并告诉大家我会完成它，然后为该函数实现了 vecEvalXType()和 vectorized()的方法。接着在向量化测试框架内添加了对该函数的测试。然后运行 make dev，保证所有 test 都能通过。最后发起 Pull Request 并完成 Merge 到主分支。

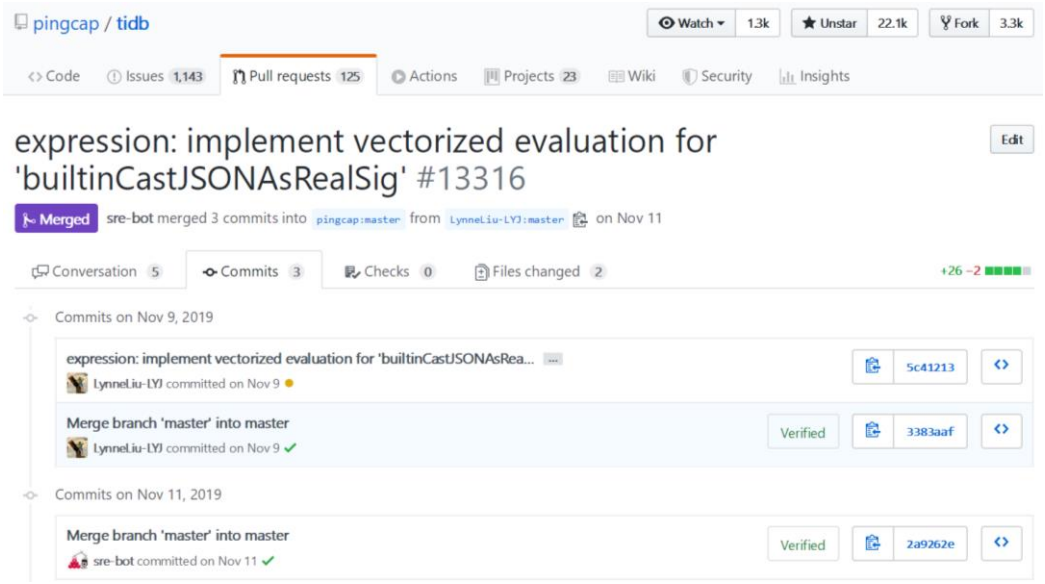


图 1 PR 截图

在这个过程中，我首先学会了 Git 的使用，包括 Git 的安装、配置等一系列操作。还学会了使用 Github 这个工具，学会了使用 Git 命令创建本地或者远端服务器上仓库的克隆、提交改动到远端仓库、创建新的分支、更新与合并等操作。

除了学习上述 Github 的知识，还学习了关于 TiDB 的知识。首先是 Go 语言，虽然没有非常的精通，但至少能读懂代码了，还了解到 Goroutine、Channel、Sync 等组件的使用。接着学习了数据库基础知识，了解一个单机数据库由哪些功能、哪些组件。还有 SQL 的基础知识，知道基本的 DDL、DML 语句，事务的基本常识。最后还了解到基本的后端服务知识，比如如何启动一个后台进程、RPC 是如何工作的等 TiDB 的基本原理。

本人首先在给出的 issue 内选择了感兴趣的函数并告诉大家我会完成它，然后为该函数实现了 vecEvalXType()和 vectorized()的方法。接着在向量化测试框架内添加了对该函数的测试。然后运行 make dev，保证所有 test 都能通过。最后发起 Pull Request 并完成 Merge 到主分支。

## 3. 课程反馈

这门课带给我很多知识，但是对我来说最重要的是让我了解了“开源”，在我的人生中种下了“开源”的种子。我在课程刚开始的时候感觉比较困难、吃力，老师们可以更换一下讲授内容的顺序，在前期多讲一些实际操作的东西，会让我们更容易入门。

## 4. 参考文献

[1] PingCAP 博客：十分钟成为 Contributor 系列|助力 TiDB 表达式计算性能提升 10 倍 https://pingcap.com/blog-cn/10mins-become-contributor-of-tidb-20190916/

[2] PingCAP 博客：十分钟成为 Contributor 系列|助力 TiDB 表达式计算性能提升 10 倍 https://pingcap.com/blog-cn/10mins-become-contributor-of-tidb-20190916/