配置管理说明

项目名：07 学习平台

课程名：软件需求分析原理与实践

小组：G07

成员：31901237徐过

31901239许罗阳宁

31901240余浩凯

31901238徐晟

31903093邵云飞

教师：杨枨



标准 GB8567-88

2022/2/27

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [ ] 正式发布  [√] 正在修改 | 文件标识： | SRA2022-G07-配置管理说明 |
| 当前版本： | 0.1 |
| 作者： | 徐过、许罗阳宁、余浩凯、徐晟、邵云飞 |
| 完成日期： |  |

**配置管理说明(SRS)**

说明：

配置管理是软件需求工程开发中的重要一环，由于需求开发工程会产生各种文档，且各种文档会因各种缺陷和不足需要不停的迭代修订，为了方便文档的管理和小组项目的开发，需要配置管理工具对文档的编写、开发、修订进行管理，通过组内管理配置的形式，按照一定的文档编写和更新的规则，对文档进行版本管理，对小组各个成员的提交项目进行分支和审核管理，便于团队高效的工作，同时也能在文档出现重大失误和缺陷时，能够进行回退，以此减少软件需求开发的成本。

**版本记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 修订日期 | 版本/状态 | 修订人 | 发布日期 | 备注 |
| 01 | 2022/5/16 | 0.1 | 徐过 | 2022/5/18 | 配置管理说明 |

**目录**

[修 订 记 录 2](#_Toc99349409)

[目录 4](#_Toc99349410)

[第1章 引言 5](#_Toc99349411)

[1.1 编写目的 5](#_Toc99349412)

[1.2 术语定义 5](#_Toc99349413)

[1.3 参考资料 6](#_Toc99349414)

[第2章 软件配置 7](#_Toc99349415)

[2.1 软件配置环境 7](#_Toc99349416)

[2.2 配置管理员 7](#_Toc99349417)

[第3章 软件配置管理计划 8](#_Toc99349418)

[3.1 建立示例配置库 8](#_Toc99349419)

[3.2 配置标识管理 8](#_Toc99349420)

[3.3 配置库控制 9](#_Toc99349421)

[3.4 配置的检查和评审 9](#_Toc99349422)

[3.5 配置库的备份 10](#_Toc99349423)

[第4章 里程碑 12](#_Toc99349424)

[4.1 各阶段里程碑 12](#_Toc99349425)

[第5章 文档命名规则 13](#_Toc99349426)

[5.1 配置分支命名规则 13](#_Toc99349427)

[5.2 配置文件夹命名规则 13](#_Toc99349428)

[5.3 配置文档命名规则 13](#_Toc99349429)

[5.4 提交版本号规则 14](#_Toc99349430)

[第6章 配置提交流程 15](#_Toc99349431)

[6.1 git命令 15](#_Toc99349432)

[6.2 配置提交流程 17](#_Toc99349433)

**引言**

**编写目的**

配置管理是软件需求工程开发中的重要一环，由于需求开发工程会产生各种文档，且各种文档会因各种缺陷和不足需要不停的迭代修订，为了方便文档的管理和小组项目的开发，需要配置管理工具对文档的编写、开发、修订进行管理，通过组内管理配置的形式，按照一定的文档编写和更新的规则，对文档进行版本管理，对小组各个成员的提交项目进行分支和审核管理，便于团队高效的工作，同时也能在文档出现重大失误和缺陷时，能够进行回退，以此减少软件需求开发的成本。

**术语定义**

**要求（requirement）**：明示的，通常隐含的或必需履行的要求或期望。

**管理（management）**：指挥和控制组织的协调的活动。

**配置管理（configuration management）**：标识和记录配置项的功能与物理特性；标识持续发展的系统中的所有组件及其关系，以维护其完整性、可跟踪性以及对整个生命周期内所有变更的可控性

**配置管理员（Configuration administrator）**：项目组中负责配置管理工作的角色，该角色可以兼职。在某一开发阶段通过评审或某一质量检查点通过审核后，配置管理员负责统一添加或修改相关文档的最新有效版本以及审批人签字。

**基线（BaseLine）**：基线是项目储存库中每个工件版本在特定时期的一个“快照”。它提供一个正式标准，随后的工作基于此标准，并且只有经过授权后才能变更这个标准。建立一个初始基线后，以后每次对其进行的变更都将记录为一个差值，直到建成下一个基线。

**软件配置管理（Software Configuration Management）**：软件配置环境是在项目开发中，标识、控制和管理软件变更的一种管理。配置管理的使用取决于项目规模和复杂性以及风险水平。软件的规模越大，配置管理就显得越重要。

**配置标识（Configuration Identification）**：对软件项目在开发过程中的资源进行标识，以便识别。

**里程碑（milepost）：**重大标志性事件。

**参考资料**

1.CSDN作者 halaoda https://blog.csdn.net/halaoda/article/details/78661334 2022-03-26

2.https://wenku.baidu.com/view/2df3b80ba1116c175f0e7cd184254b35eefd1afe.html 2022-03-2-22

**软件配置**

**软件配置环境**

|  |  |
| --- | --- |
| 软件名称 | 作用 |
| Windows7/Windows10/ Windows XP | 操作系统。 |
| 浏览器（谷歌浏览器、火狐浏览器等） | 支持运行网页版Github。 |
| Github Desktop | 配置管理工具，提交文档，更新文档，使本地库和远程仓库链接的工具，通过github desktop桌面软件使用命令实现各种版本管理工作。 |

硬件环境无特殊要求。

配置管理工具说明：项目组成员需要在各自的计算机安装Github Desktop工具，下载网址：https://desktop.github.com，选择Download for Windows（64 bit）进行下载。

**配置管理员**

负责配置管理仓库的管理：

1. 负责对main分支的管理和其它子分支的创建。
2. 指导成员对Gihub Desktop工具的使用。
3. 定期检查仓库是否更新。
4. 协助项目经理对各项交付文档进行检查和评审。

**软件配置管理计划**

**建立示例配置库**

配置管理员依据小组项目需要，建立符合本项目的配置管理库。

在Githube创建小组仓库，只有组内成员有权限进入和使用相应功能。配置仓库，建立主分支和成员各自的分支，管理员依据不同阶段的需要，依据需求工程计划，建立各子系统配置目录。

建立完配置管理库后，小组成员注册各自的Github账号，配置管理员将小组各个成员邀请进仓库，分配各小组成员相应的权限。

示例库分支如下：

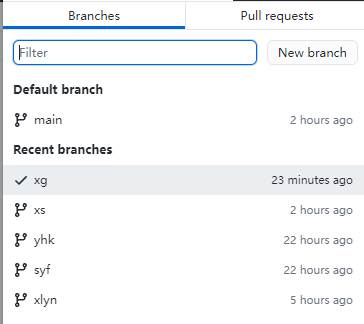
main（主分支）

xg（成员分支）

xs（成员分支）

yhk（成员分支）

syf（成员分支）

xlyn（成员分支）

分支状态规则：

1. 常规分支：仓库成员（开发者权限及以上）可推送内容至常规分支。
2. 保护分支：可自定义保护策略，此处采用默认策略，只有仓库管理员才能推送内容至保护分支，开发者权限只能读取内容，不能修改推送。

**配置标识管理**

文档：根据需求工程计划中的文档清单，配置管理员要检查需要提交的文档是否都按时提交，文档格式是否合规，文档修订记录是否及时更新，版本号是否符合程序规定。

基线：所有本项目和各子系统的各类基线需要根据需求工程计划确定。文档基线，当项目文档经过评审通过，版本号更新为1.0时，增加相关文档的基线。

**配置库控制**

权限控制：

|  |  |  |
| --- | --- | --- |
| 成员 | 仓库权限 | 职能 |
| 徐过 | 管理员/开发人员 | 负责项目评审和main分支交付物的更新。按时提交各阶段需要提交的文档等、图形等交付物。 |
| 徐晟 | 开发人员 | 按时提交各阶段需要提交的文档等、图形等交付物。 |
| 余浩凯 | 开发人员 | 按时提交各阶段需要提交的文档等、图形等交付物。 |
| 邵云飞 | 开发人员 | 按时提交各阶段需要提交的文档等、图形等交付物。 |
| 许罗阳宁 | 开发人员 | 按时提交各阶段需要提交的文档等、图形等交付物。 |

**配置的检查和评审**

配置的检查和评审依据相应的管理制度进行，相关评审内容如下：

|  |  |
| --- | --- |
| 审核分类 | 审核内容 |
| 初稿审核 | 文档是否寻找模板，是否参考相应标准。 |
| 是否有修订记录、封面以及目录，修订和目录是否各占一页。 |
| 文档目录定义是否合规范，是否参照课本内容进行制作。 |
| 修订记录是否更新，记录时间是否准确，是否符合规范。 |
| 修订审核 | 文档各项内容是否进行制作，是否符合书上要求和评审要求。 |
| 对已知存在的缺陷是否进行及时的修改。 |
| 版本号是否及时更行，是否符合更新的规范。 |
| 修订记录是否更新，记录时间是否准确，是否符合规范。 |
| 修订内容是否按时提交，是否以及项目计划执行 |
| 是否按照提出的新需求进行修订合提交 |
| 发布审核 | 文档是否经过课上评审或者周末评审。 |
| 是否根据课上评审或者周末评审中发现的问题对文档进行修改更正。 |
| 修订记录是否更新，记录时间是否准确，是否符合规范。 |
| 审核后是否确定基线。 |
| 版本号是否及时更行，是否符合更新的规范。 |
| 文档内容是否充分足够，是否上传分支和备份。 |

配置管理员应每周定期与组员核定提交项，检查交付物是否符合规定，内容是否充分。此处该小组定期检查时间为周一和周四的例会。

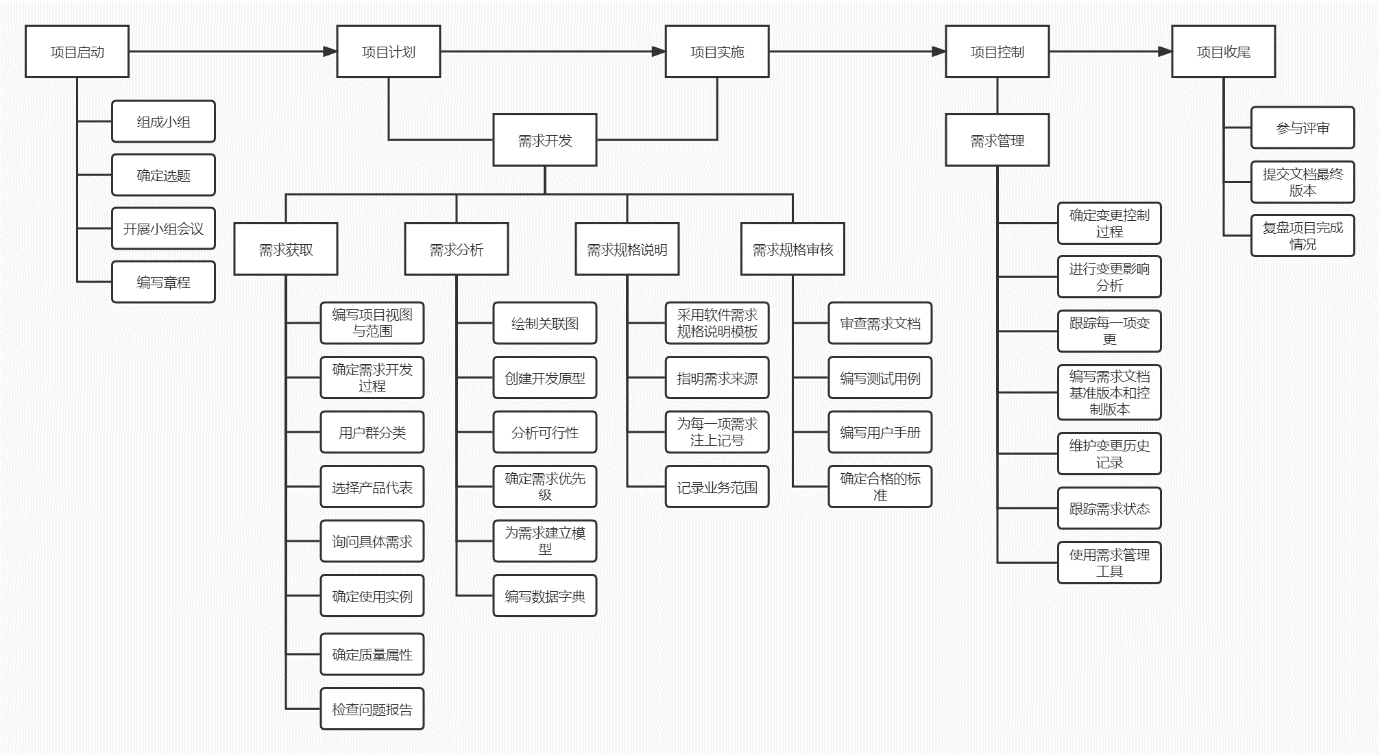
**配置库的备份**

在项目开发实施的各个阶段，配置管理员应定期做好软件配置库的备份，防止因特殊情况给整个小组带来严重损失，也应提醒小组成员做好备份，规避风险，在每周日完成各项工作和组内评审后，及时进行更新和备份。

1. 小组每个成员计算机上应留有配置库的备份，并且要有readme文件进行说明。
2. github保存历史记录，能够回滚至任意阶段或者下载任意版本的文档内容。

**里程碑**

**各阶段里程碑**

1. WBS
2. 课程里程碑

第一周：学生分组信息

第二周：软件需求工程项目计划

第四周至第七周：UML作业

第十一周：软件需求规格说明SRS

第十三周：UML基础知识测试

第十四周：软件需求变更文档

第十五周：文档改进

第十六周： 项目收尾 - - 课程作业评审

**文档命名规则**

**配置分支命名规则**

仓库分支分为主分支和成员分支，主分支用main命名，成员分支用项目各成员的姓名缩写命名，具体规则如下：小组成员名字拼音首字母；

其中所有英文字母均为小写：

xg（成员分支）

xs（成员分支）

yhk（成员分支）

syf（成员分支）

xlyn（成员分支）

**配置文件夹命名规则**

配置管理员依据项目内容新建文件夹并命名，文件夹命名使用英文小写

例如：

项目计划：projectplan

需求规格：requirement

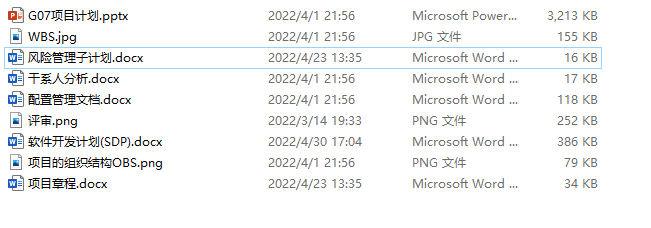
其他更多：more

**配置文档命名规则**

文档用各个报告中文命名，命名的文档命要体现出该文档编写的主题。

例如：

风险管理子计划.docx

软件开发计划(SDP).docx

**提交版本号规则**

相应提交版本号由文件名+版本号组成，文档首次编写时，版本号规定为0.1，若对某个内容部分进行编写或者修改，版本号在第二个数字上增加1，更新为1.0。

示例：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 修订日期 | 版本/状态 | 修订人 | 发布日期 | 备注 |
| 01 | 2022/2/23 | 0.1 | 徐过 | 2022/2/27 | 项目计划书 |
| 02 | 2022/2/28 | 0.2 | 徐过 | 2022/2/28 | 项目计划书 |
| 03 | 2022/3/5 | 0.3 | 徐过 | 2022/3/11 | 项目计划书 |
| 04 | 2022/3/12 | 1.0 | 徐过 | 2022/3/13 | 项目计划书 |
| 05 | 2022/3/21 | 1.1 | 徐过 | 2022/3/21 | 项目计划书 |
| 06 | 2022/3/27 | 1.2 | 徐过 | 2022/3/27 | 项目计划书 |
| 07 | 2022/3/30 | 1.3 | 徐过 | 2022/4/2 | 项目计划书 |
| 08 | 2022/4/22 | 1.4 | 徐过 | 2022/4/24 | 项目计划书 |

**配置提交流程**

**git命令**

|  |  |
| --- | --- |
| 合并与变基 | |
| git merge <分支名称> | 把指定的分支合并到当前所在分支下。 |
| git merge (--continue| --about| --quit) | 合并冲突，选择继续或者静止。 |
| git rebase <分支名称> | 变基：把当前分支改动在新的基线上重放。 |
| git rebase –i –autosquash <分支名称> | 交互式变基，通常用于整理提交。 |
| git rebase (--continue| --skip| --about| --quit| --edit -todo) | 变基暂停后，选择继续、跳过、终止或者编辑变基任务。 |

|  |  |
| --- | --- |
| 远程仓库操作 | |
| git remote add <别名> <URL> | 添加远程仓库，别名表示远程主机名，URL为远程仓库地址。 |
| git remote -v | 查看远程仓库的详细信息。 |
| git remote remove <别名> | 删除指定名称的远程仓库。 |
| git remote set-url <别名> <新URL> | 修改远程仓库的URL地址 |
| git fetch [--prune] <远程仓库的别名> | 获取远程仓库引用，使—prune会清理本地缓存的已由远程仓库删除的引用。 |
| git fetch <远程仓库别名> <分支名> | 将远程仓库指定的分支的最新版本取回到本地。 |
| git pull <远程仓库别名> <分支名> | 拉取远程仓库某个分支的更新，再与本地的指定分支合并。 |
| git push <远程仓库的别名> <本地分支名> <远程分支名> | 把本地仓库分支推送到远程仓库的指定分支。 |
| git push <远程仓库的别名> --delete <远程分支名> | 删除指定远程仓库的分支。 |
| git push <远程仓库的别名> <标签名称> | 将指定的标签提交到远程仓库。 |
| git push <远程仓库别名> --tags | 将本地所有的标签全部提交到远程仓库。 |

|  |  |
| --- | --- |
| 分支与标签 | |
| git branch [-a] | 查看本地分支 （参数 –a显示远程分支）。 |
| git branch <新分支> | 在本地创建新分支。 |
| git branch –d <本地分支名> | 删除指定本地分支。 |
| git checkout <本地分支名> | 切换到指定分支。 |
| git checkout –b <新分支> | 创建并切换到新分支。 |
| git checkout <版本> --<文件> | 用指定版本的文件覆盖本地文件。 |
| git tag | 查看本地所有标签。 |
| git tag –m <标签描述信息> <标签名称> [<版本>] | 基于指定提交创建新标签， -m参数用于说明。 |
| git tag -s –u <key-id> -m <标签描述信息> <标签名称> [<版本>] | 创建带有PGP签名的标签。 |
| git tag -d <标签名称> | 删除指定的标签。 |

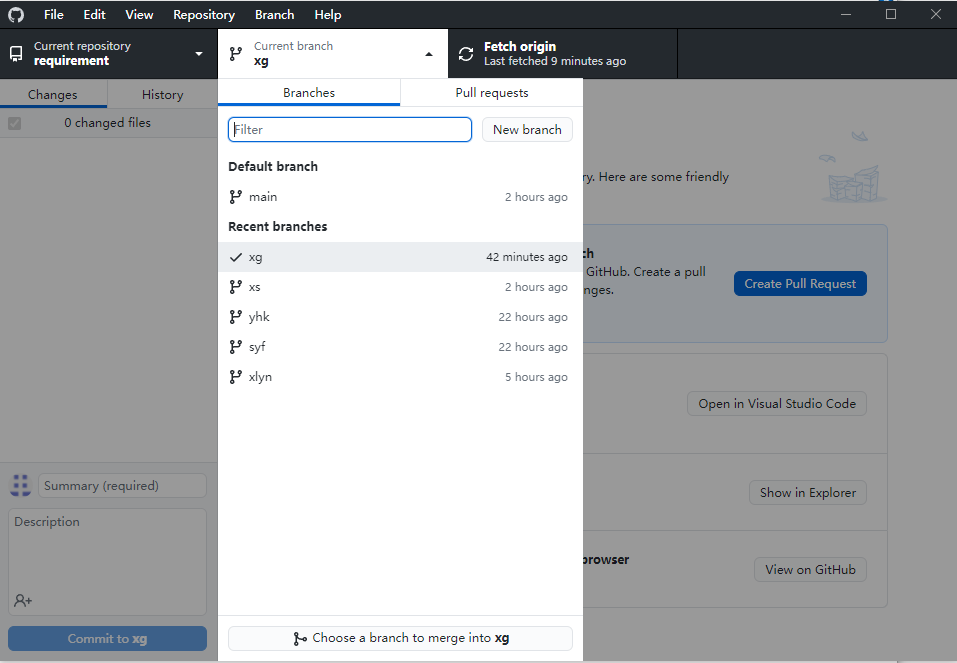
|  |  |
| --- | --- |
| 仓库初始化操作 | |
| git init | 初始化仓库 |
| git commit –m <修改文档描述> | 对添加的所有改动文档进行说明。 |
| git log | 查看当前分支所有提交历史。 |
| git status | 查看本地仓库当前状态，查看是否由文件改动。 |
| git config --global user.name <代表使用者的签名> | 确定使用者的签名，起到标签作用。 |
| git config --global user.email <代表使用者的邮箱> | 确定使用者的邮箱，起到标签作用。 |

|  |  |
| --- | --- |
| 撤销操作 | |
| git reset | 撤销暂存区改动 |
| git reset --soft <版本> | 软重置：仅修改HEAD指向，暂存区和本地文件均不修改。 |
| git reset [--mixed] <版本> | 混合重置：本地文件不修改，但HEAD指向和暂存区均修改。 |
| git reset --hard <版本> | 硬重置：HEAD指向、暂存区、本地文件均修改。 |
| git revert [-m 1] <版本> | 生成一个新的提交来撤销某次历史提交（-m参数用于撤销合并提交）。 |

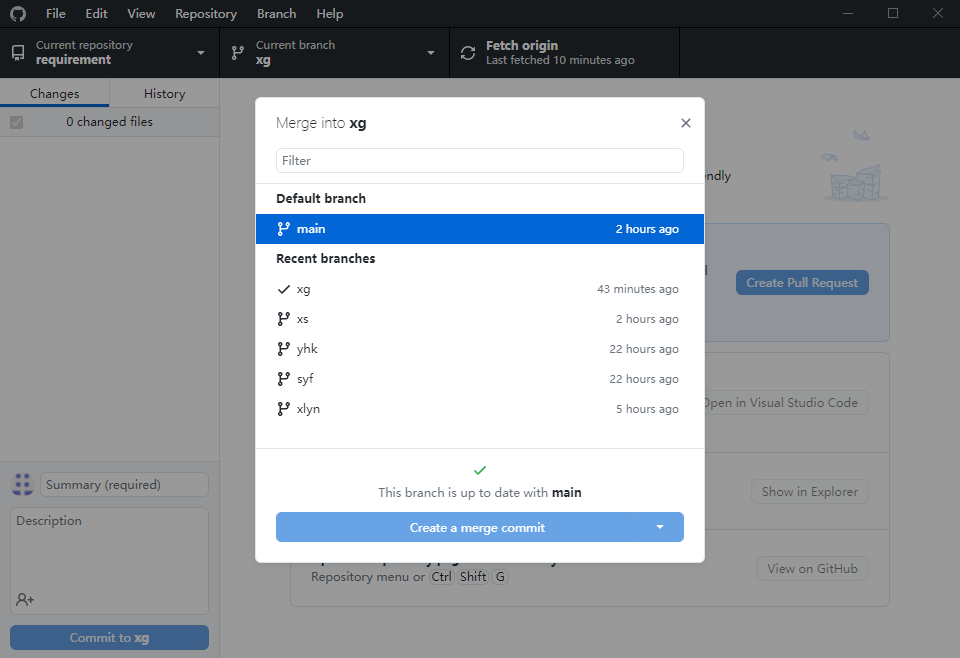
**配置提交流程**

文档提交流程步骤。

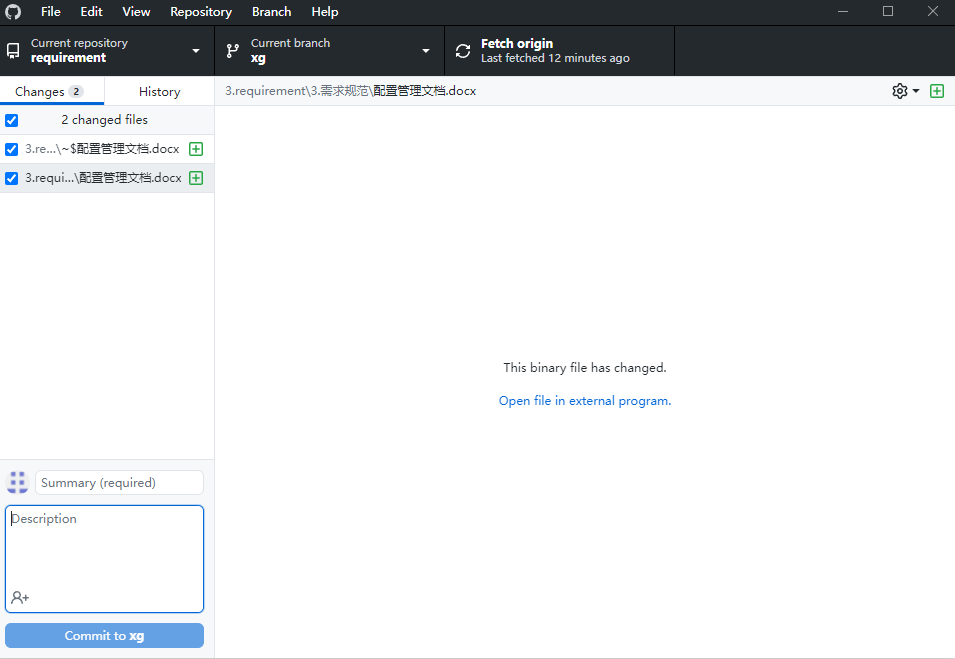
1.切换到个人分支



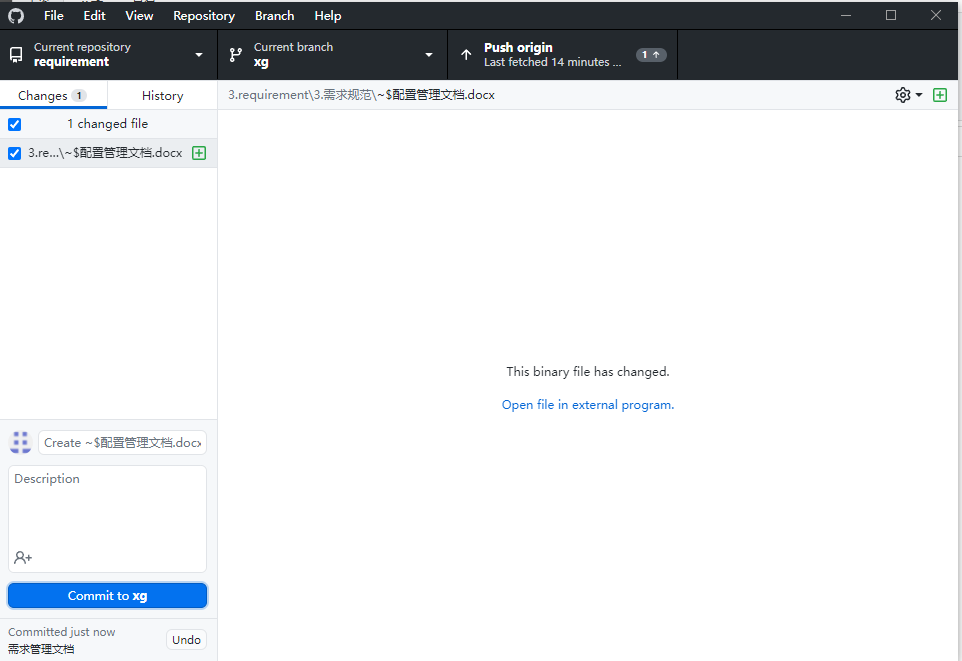
2.点击 Choose a branch to merge into xx将main分支内容复刻到个人分支



3.当有文件修改时填写summary和describtion，点击commit to xx



4.上传后点击Push oigin将变更上传到github端



5(管理员).在main分支点击Choose a branch to merge into main将各个开发人员检查完的分支改动点击Create a merge commit合并到main

