哈尔滨工业大学 计算学部

2024 年秋季学期《开源软件开发实践》

Lab 1：Git 实战

姓名	学号	联系方式
王博涵	2022111580	1169052656@qq.com

目 录

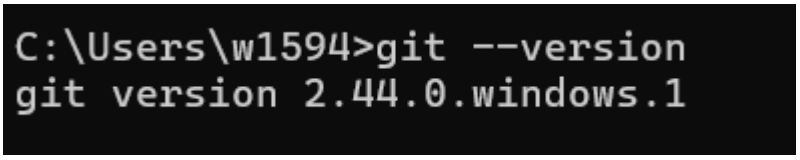
1 实验要求

- 熟练掌握 Git 的基本指令和分支管理指令；
- 掌握 Git 支持软件配置管理的核心机理；
- 使用 Git/Github 管理自己的项目源代码。

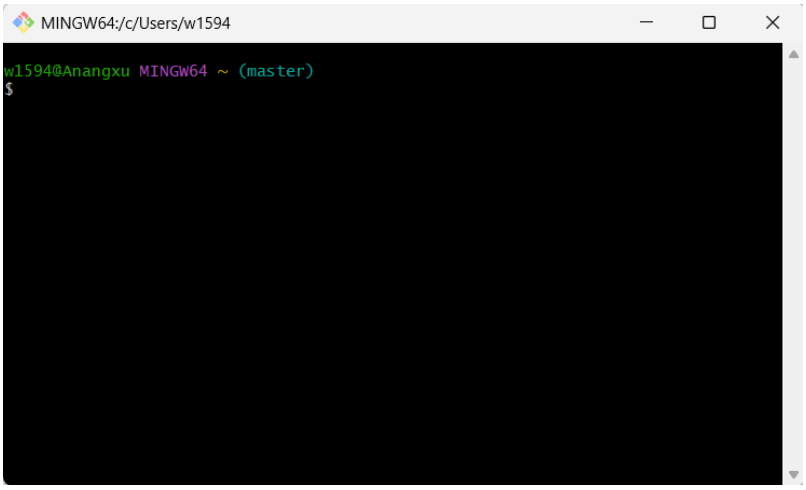
2 安装 Git

2.1 本地机器上安装 git

git 版本号：

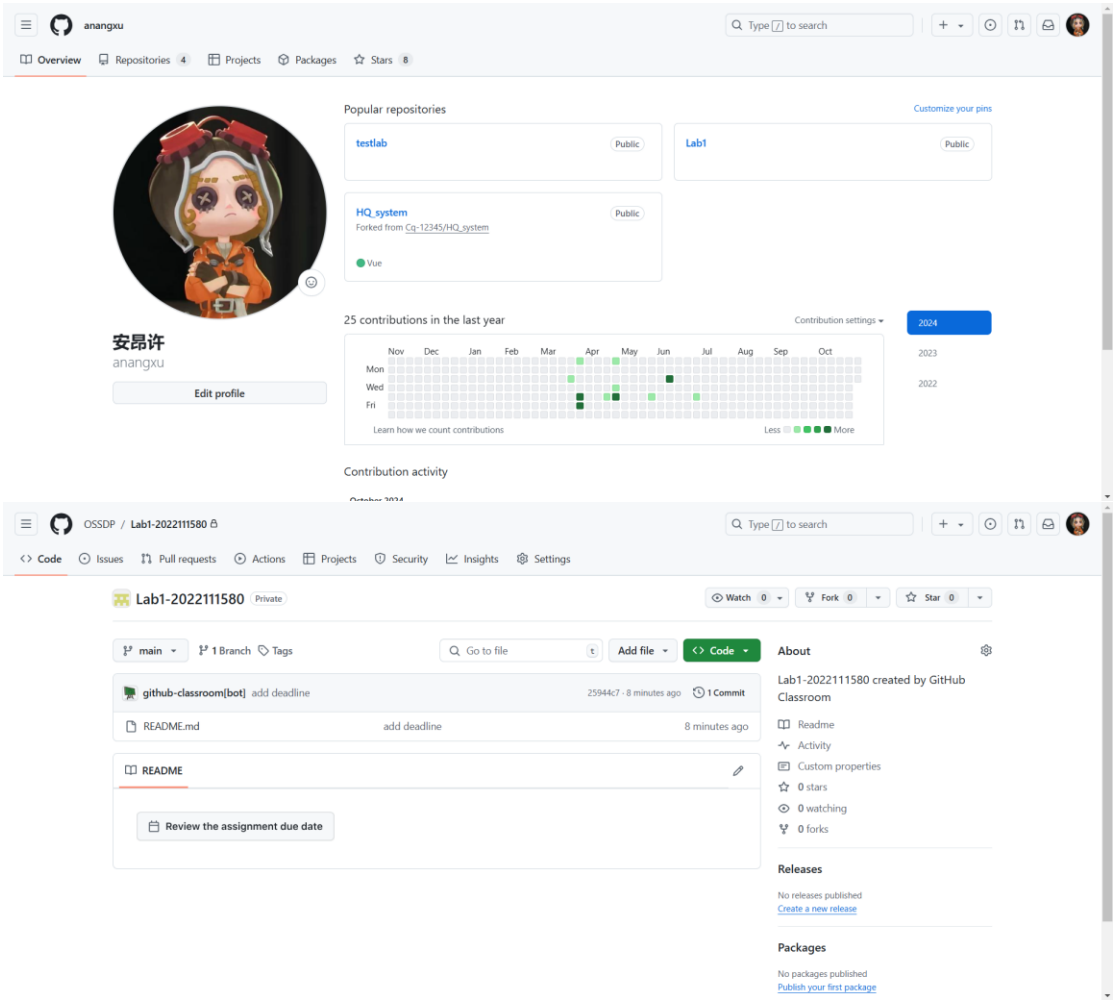


git 运行界面：



2.2 申请 github 帐号

github 上申请的帐号名称：anangxu
本次实验中创建仓库的 URL 地址：https://github.com/OSSDP/Lab1-2022111580.git
github 网站上账号信息和项目信息的截图：



3 Git 操作过程

3.1 实验场景(1)：仓库创建与提交

R0：查看工作区、暂存区、Git 仓库的状态

使用以下命令查看当前状态，了解文件的状态和更改：

`git status`

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1
$ git status
fatal: not a git repository (or any of the parent directories): .git
```

R1：本地初始化一个 Git 仓库并将项目文件纳入管理

1. 初始化一个新的 Git 仓库：

`git init`

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1
$ git init
Initialized empty Git repository in D:/0workspace/rlab1/.git/
```

2. 将项目的所有源代码文件添加到暂存区：

git add .

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (master)
$ git add .
```

R2：提交

提交到本地仓库：

git commit -m "Initial commit with project files"

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (master)
$ git commit -m "Initial commit with project files"
[master (root-commit) 10d9dfd] Initial commit with project files
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt
```

对某些文件进行修改

在代码编辑器中编辑文件并保存更改。

R3：查看上次提交后文件的修改及具体内容

1. 使用 **git status** 查看哪些文件已被修改。
2. 使用 **git diff** 查看修改的具体内容：

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (master)
$ git diff
diff --git a/test.txt b/test.txt
index 77356c3..2545e90 100644
--- a/test.txt
+++ b/test.txt
@@ -1 +1,2 @@
 test
+test^M
```

R4：重新提交

1. 将修改后的文件添加到暂存区：

git add .

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (master)
$ git add .
```

2. 再次提交：

git commit -m "Update files with new changes"

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (master)
$ git commit -m "Update files with new changes"
[master 7f8c5fa] Update files with new changes
 1 file changed, 1 insertion(+)
```

再次对某些文件进行修改

编辑文件并保存更改。

R5：重新提交

1. 将修改后的文件添加到暂存区：

git add .

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (master)
$ git add .
```

2. 提交修改：

git commit -m "Further update on files"

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (master)
$ git commit -m "Further update on files"
[master 05ccb32] Further update on files
 1 file changed, 1 insertion(+)
 create mode 100644 test1.txt
```

R6：撤销最后一次提交

保留修改但撤销提交：

git reset --soft HEAD~1

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (master)
$ git reset --soft HEAD~1
```

R7：查看全部提交记录

使用以下命令查看项目的全部提交历史：

git log

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (master)
$ git log
commit 7f8c5fa18f136d2dc964949120554a7f06a8fa75 (HEAD -> master)
Author: 安昂许 <1169052656@qq.com>
Date:   Tue Oct 29 09:41:23 2024 +0800

    Update files with new changes

commit 10d9dfd960bd4f239f9082bfc9685418d7007462
Author: 安昂许 <1169052656@qq.com>
Date:   Tue Oct 29 09:39:31 2024 +0800

    Initial commit with project files
```

R8：在本地仓库建立与远程仓库的关联

将本地仓库与远程仓库关联：

在本地 Git Bash 中进入到该仓库的目录，然后执行以下命令，将远程仓库地址添加为 origin：

git remote add origin https://github.com/OSSDP/Lab1-2022111580.git

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (master)
$ git remote add origin https://github.com/OSSDP/Lab1-2022111580.git
```

可以用以下命令确认远程仓库是否已添加成功：

git remote -v

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (master)
$ git remote -v
origin  https://github.com/OSSDP/Lab1-2022111580.git (fetch)
origin  https://github.com/OSSDP/Lab1-2022111580.git (push)
```

如果显示远程仓库的地址，说明关联成功。

R9：将本地仓库的内容推送到 GitHub 远程仓库

1. **推送内容到远程仓库：**

使用以下命令将所有内容推送到远程仓库的 main 分支（或 master 分支，取决于仓库默认分支名称）：

git push -u origin main

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (master)
$ git push -u origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 20 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 481 bytes | 481.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/OSSDP/Lab1-2022111580/pull/new/master
remote:
To https://github.com/OSSDP/Lab1-2022111580.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

首次推送时使用 -u 参数，目的是将本地的 main 分支与远程的 main 分支关联，以便之后直接使用 git push 进行推送。

2. **后续更新推送：**

每次修改和提交后，可以使用 git push 简单推送更新内容。

完成后，GitHub 仓库页面上应会显示本地仓库的全部提交记录和文件内容。

3.2 实验场景(2)：分支管理

R1：获得本地仓库的全部分支，切换至分支 master

 # 查看本地所有分支

git branch

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (main)
$ git branch
* main
```

 # 切换到 master 分支

git checkout master

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (main)
$ git checkout master
Switched to branch 'master'
A       test1.txt
```

R2：在 master 基础上建立两个分支 B1、B2

 # 创建并切换到 B1 分支

```
git checkout -b B1
```

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (master)
$ git checkout -b B1
Switched to a new branch 'B1'
```

 # 切换回 master 再创建 B2

```
git checkout master
```

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (B1)
$ git checkout master
Switched to branch 'master'
A       test1.txt
```

```
git checkout -b B2
```

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (master)
$ git checkout -b B2
Switched to a new branch 'B2'
```

R3：在 B2 分支基础上创建一个新分支 C4

 # 确保在 B2 分支上

```
git checkout B2
```

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (B2)
$ git checkout B2
Already on 'B2'
A       test1.txt
```

 # 创建并切换到 C4 分支

```
git checkout -b C4
```

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (B2)
$ git checkout -b C4
Switched to a new branch 'C4'
```

R4：在 C4 上，对某个文件进行修改并提交

1. 修改目标文件后，执行以下命令：

```
git add test.txt
```

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (C4)
$ git add test.txt
```

```
git commit -m "Modify file in C4"
```

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (C4)
$ git commit -m "Modify file in C4"
[C4 8bc806c] Modify file in C4
 2 files changed, 2 insertions(+)
 create mode 100644 test1.txt
```

R5：在 B1 分支上对同样的文件做不同修改并提交

1. 切换到 B1 分支，修改相同的文件，然后提交更改：

```
git checkout B1
```

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (C4)
$ git checkout B1
Switched to branch 'B1'
```

git add test.txt

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (B1)
$ git add test.txt
```

git commit -m "Different modification in B1"

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (B1)
$ git commit -m "Different modification in B1"
[B1 76f382e] Different modification in B1
 1 file changed, 1 insertion(+)
```

R6：将 C4 合并到 B1 分支，若有冲突，手工消解

1. **切换到 B1 分支，合并 C4 分支：**

git checkout B1

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (B1)
$ git checkout B1
Already on 'B1'
```

git merge C4

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (B1)
$ git merge C4
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.
```

2. **若遇到冲突，手动编辑文件解决冲突。然后执行以下命令完成合并：**

git add test.txt

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (B1|MERGING)
$ git add test.txt
```

git commit -m "Resolve merge conflict between B1 and C4"

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (B1|MERGING)
$ git commit -m "Resolve merge conflict between B1 and C4"
[B1 5ea5a66] Resolve merge conflict between B1 and C4
```

R7：在 B2 分支上对某个文件做修改并提交

1. **切换到 B2 分支，修改文件并提交：**

git checkout B2

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (B1)
$ git checkout B2
Switched to branch 'B2'
```

git add test.txt

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (B2)
$ git add test.txt
```

git commit -m "Modify file in B2"

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (B2)
$ git commit -m "Modify file in B2"
[B2 9f90ea7] Modify file in B2
 1 file changed, 1 insertion(+)
```

R8：查看目前哪些分支已经合并、哪些分支尚未合并

1. 查看已合并的分支：

git branch –merged

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (B2)
$ git branch --merged
* B2
  main
  master
```

2. 查看未合并的分支：

git branch --no-merged

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (B2)
$ git branch --no-merged
  B1
  C4
```

R9：将已合并的分支删除，尚未合并的分支合并到一个新分支上（以学号命名）

1. 删除已合并的分支：

git branch -D B2

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (main)
$ git branch -D B2
Deleted branch B2 (was 9f90ea7).
```

2. 合并未合并的分支：

创建以学号命名的新分支：

git checkout -b 2022111580

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (main)
$ git checkout -b 2022111580
Switched to a new branch '2022111580'
```

将未合并的分支合并到该新分支上：

git merge <unmerged_branch_name>

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (2022111580)
$ git merge B1
Updating 7f8c5fa..5ea5a66
Fast-forward
 test.txt  | 3 +++
 test1.txt | 1 +
 2 files changed, 4 insertions(+)
 create mode 100644 test1.txt

w1594@Anangxu MINGW64 /d/0workspace/rlab1 (2022111580)
$ git merge C4
Already up to date.

w1594@Anangxu MINGW64 /d/0workspace/rlab1 (2022111580)
$ git merge main
Already up to date.

w1594@Anangxu MINGW64 /d/0workspace/rlab1 (2022111580)
$ git merge master
Already up to date.
```

R10：将本地以学号命名的分支推送到 GitHub 仓库

`git push -u origin 2022111580`

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (2022111580)
$ git push -u origin 2022111580
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (10/10), 789 bytes | 789.00 KiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for '2022111580' on GitHub by visiting:
remote:      https://github.com/OSSDP/Lab1-2022111580/pull/new/2022111580
remote:
To https://github.com/OSSDP/Lab1-2022111580.git
 * [new branch]      2022111580 -> 2022111580
branch '2022111580' set up to track 'origin/2022111580'.
```

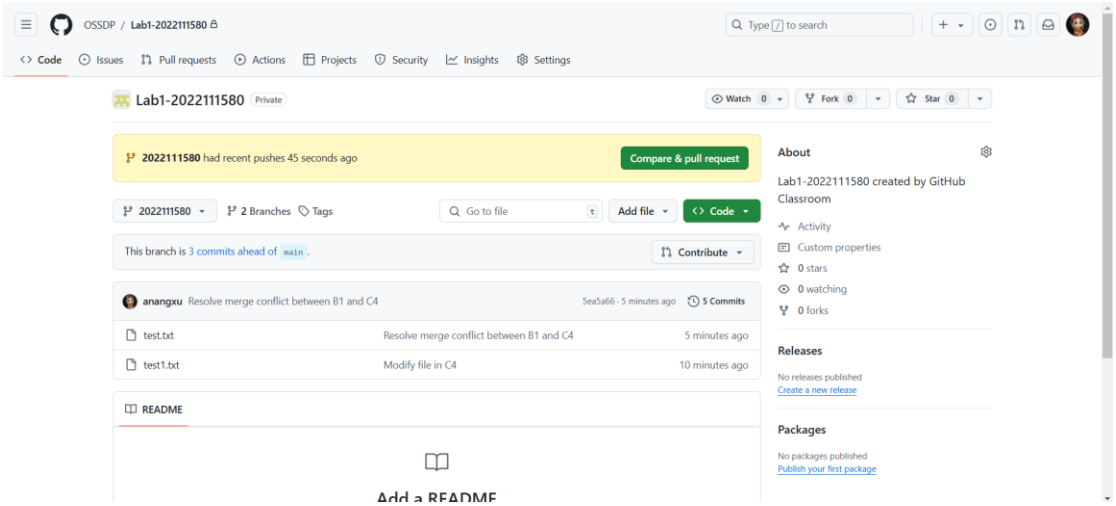
R11：查看完整的版本变迁树

`git log --graph --oneline --all -decorate`

```
w1594@Anangxu MINGW64 /d/0workspace/rlab1 (2022111580)
$ git log --graph --oneline --all --decorate
*   5ea5a66 (HEAD -> 2022111580, origin/2022111580, B1) Resolve merge conflict between B1 and
 C4
|\
| * 8bc806c (C4) Modify file in C4
* | 76f382e Different modification in B1
|/
* 7f8c5fa (origin/main, master, main) Update files with new changes
* 10d9dfd Initial commit with project files
```

R12：在 GitHub 上查看 Lab1 仓库的当前状态

1. 打开 GitHub，在你的仓库页面（https://github.com/OSSDP/Lab1-2022111580）。
2. 查看文件、提交历史、分支等内容，以确认仓库的最新状态。



3.3 实验场景(3)：在线 Git 练习

（一）主要页面-基础篇

任务 1：

操作命令集

```
git commit
git commit
```

任务 2：

操作命令集

```
git checkout -b bugFix
```

任务 3：

操作命令集

```
git checkout -b bugFix
git commit
git checkout main
git commit
git merge bugFix
```

任务 4：

操作命令集

```
git checkout -b bugFix
git commit
git checkout main
git commit
git checkout bugFix
git rebase main
```

（二）主要页面-高级篇

任务 1：

操作命令集

```
git checkout C4
```

任务 2：

操作命令集

```
git checkout bugFix^
```

任务 3：

操作命令集

```
git branch -f main C6
git branch -f bugFix bugFix~3
git checkout HEAD^
```

任务 4：

操作命令集

```
git reset local^
git checkout pushed
git revert pushed
```

（三）主要页面-移动提交记录

任务 1：

操作命令集

```
git cherry-pick C3 C4 C7
```

任务 2：

操作命令集

```
git rebase -i HEAD~4
```

（四）主要页面-杂项

任务 1：

操作命令集

```
git checkout main
git cherry-pick C4
```

任务 2：

操作命令集

```
git rebase -i HEAD~2
git commit --amend
git rebase -i HEAD~2
git branch -f main C3’’
```

任务 3：

操作命令集

```
git checkout newImage
git commit --amend
git checkout main
git cherry-pick C2’ C3
```

任务 4：

操作命令集

```
git tag v0 C1
git tag v1 C2
git checkout v1
```

任务 5：

操作命令集

```
git commit
```

（五）主要页面-高级话题*

任务 1：

操作命令集

```
git rebase main bugFix
git rebase bugFix side
git rebase side another
git branch -f main another
```

任务 2：

操作命令集

```
git branch bugWork main~^2~
```

任务 3：

操作命令集

```
git checkout one
git cherry-pick C4 C2 C3
git checkout two
git cherry-pick C5 C4 C3 C2
git branch -f three C2
```

（六）远程页面-Git 远程仓库

任务 1：

操作命令集

```
git clone
```

任务 2：

操作命令集

```
git commit
git checkout o/main
git commit
```

任务 3：

操作命令集

```
git fetch
```

任务 4：

操作命令集

```
git pull
```

任务 5：

操作命令集

```
git fakeTeamwork main 2
git commit
git pull
```

任务 6：

操作命令集

```
git commit
git commit
git push
```

任务 7：

操作命令集

```
git clone
git fakeTeamwork
git commit
git pull –rebase
git push
```

任务 8：

操作命令集

```
git checkout -b feature
git push
git branch -f main C1
```

（七）远程页面-Git 远程仓库高级操作

任务 1：

操作命令集

```
git checkout main
git cherry-pick C2 C3 C4 C5 C6 C7
git pull --rebase
git push
```

任务 2：

操作命令集

```
git checkout main
git pull
git merge side1
git merge side2
git merge side3
git push
```

任务 3：

操作命令集

```
git checkout -b side o/main
git commit
git pull --rebase
git push
```

任务 4：

操作命令集

```
git push origin main
git push origin foo
```

任务 5：

操作命令集

```
git push origin main^:foo
git push origin foo:main
```

任务 6：

操作命令集

```
git fetch origin C3:foo
git fetch origin C6:main
git checkout foo
```

```
git merge main
```

任务 7：

操作命令集

```
git push origin :foo
git fetch origin :bar
```

任务 8：

操作命令集

```
git pull origin C3:foo
git pull origin C2:side
```

（八）通关后的主界面截图

4 小结

本次实验让我学会了绝大多数 git 操作指令，让我收获很多。