实验报告 1

伍紫涵

2025 年 8 月 30 日

摘要

本文是我学习的关于 git 的 20 个指示的实验报告，接下来我会详细地介绍我学习的内容，本文的 LaTeX 代码已经上传到了 github 仓库中，可以点击超链接进行查看。

目录

1 git init

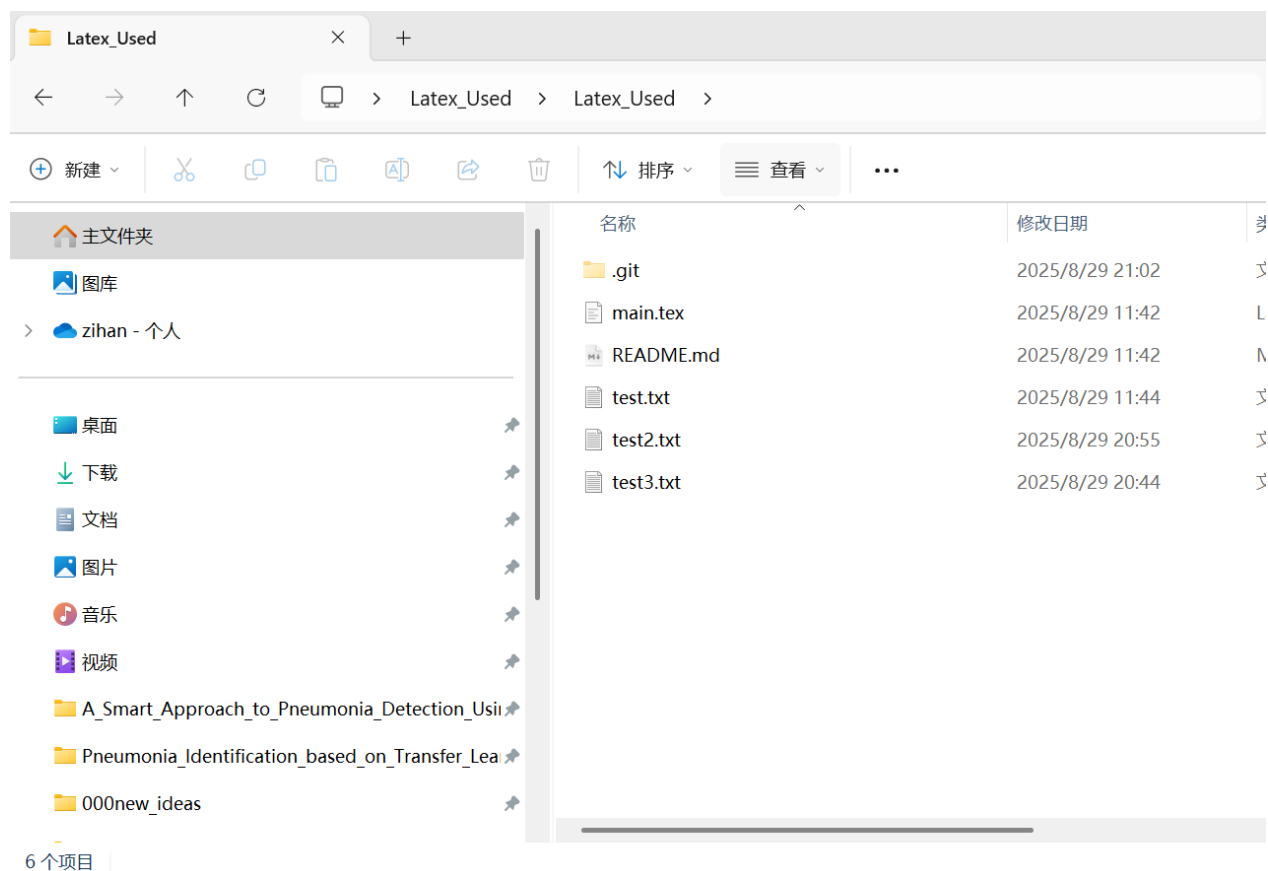git init 的作用是建立一个仓库，在本地直观的体现就是所在文件夹中建立一个.git，把文件夹中的其他文件也包含在了仓库中。

图 1: git init 运行图

图 2: 文件夹中增加的.git

2 git help

git help 会列出所有 git 指示以及其作用。

```
Administrator@vice-Gabby MINGW64 ~/Desktop/Latex_Used/Latex_Used (main)
$ git help
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--no-lazy
-fetch]
           [--no-optional-locks] [--no-advice] [--bare] [--git-dir=<path>]
           [--work-tree=<path>] [--namespace=<name>] [--config-env=<name>=<envva
r>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone     Clone a repository into a new directory
   init      Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add       Add file contents to the index
   mv        Move or rename a file, a directory, or a symlink
   restore   Restore working tree files
   rm        Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect    Use binary search to find the commit that introduced a bug
   diff      Show changes between commits, commit and working tree, etc
   grep      Print lines matching a pattern
   log       Show commit logs
   show      Show various types of objects
   status    Show the working tree status

grow, mark and tweak your common history
   backfill  Download missing objects in a partial clone
   branch    List, create, or delete branches
   commit    Record changes to the repository
   merge     Join two or more development histories together
   rebase    Reapply commits on top of another base tip
   reset     Reset current HEAD to the specified state
   switch    Switch branches
   tag       Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch     Download objects and refs from another repository
   pull      Fetch from and integrate with another repository or a local branch
   push      Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
```

图 3: git help 运行图

3   git clone <name>

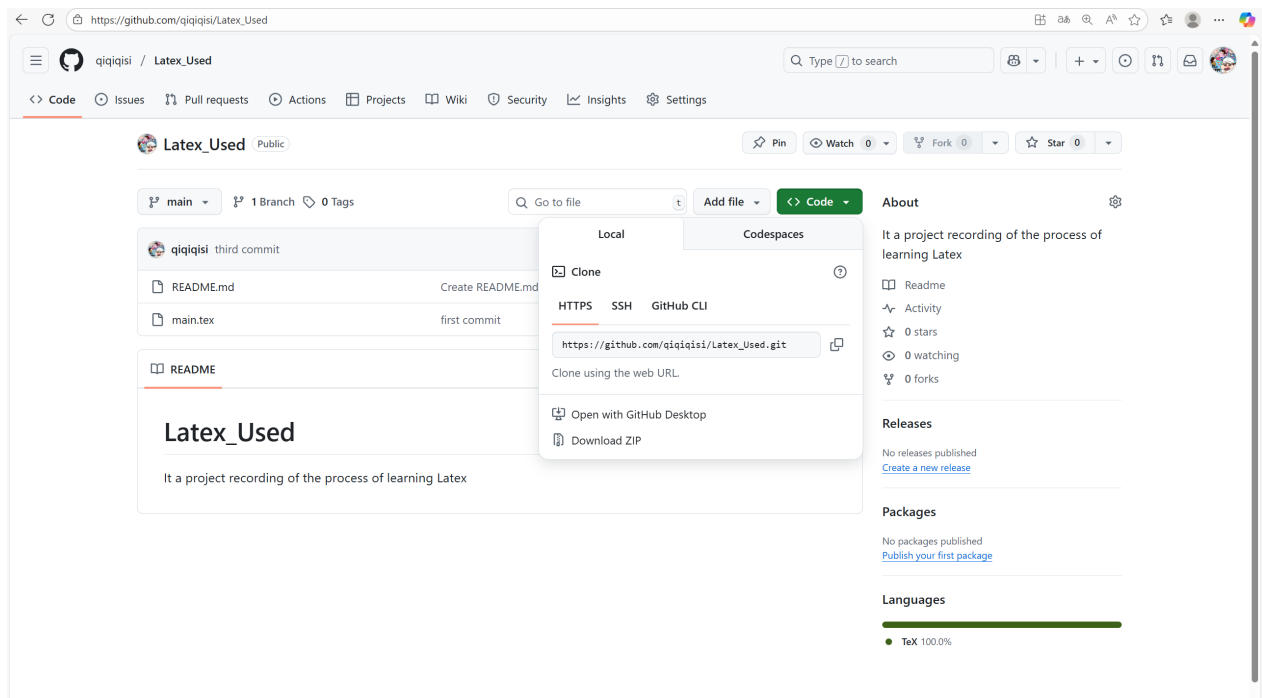git clone <name> 是把 github 上的仓库克隆到本地，首先需要找到 github 上的仓库链接然后再克隆。

3

图 4: github 仓库界面

图 5: git clone <name> 运行图

4 git status

git status 可以查看仓库状态。

图 6: git status 运行图

5 git add <name>

git add <name> 可以添加特定的文件到仓库中。

图 7: git add <name> 运行图

6 git add .

git add . 可以把所有被添加在文件夹中的文件添加到仓库中去。

图 8: git add . 运行图

7 git commit -m <information>

git commit -m <information> 使用于说明简短的 commit。

图 9: git commit -m <information> 运行图

8 git push

git push 可以提交本地内容到 github 上的仓库上。
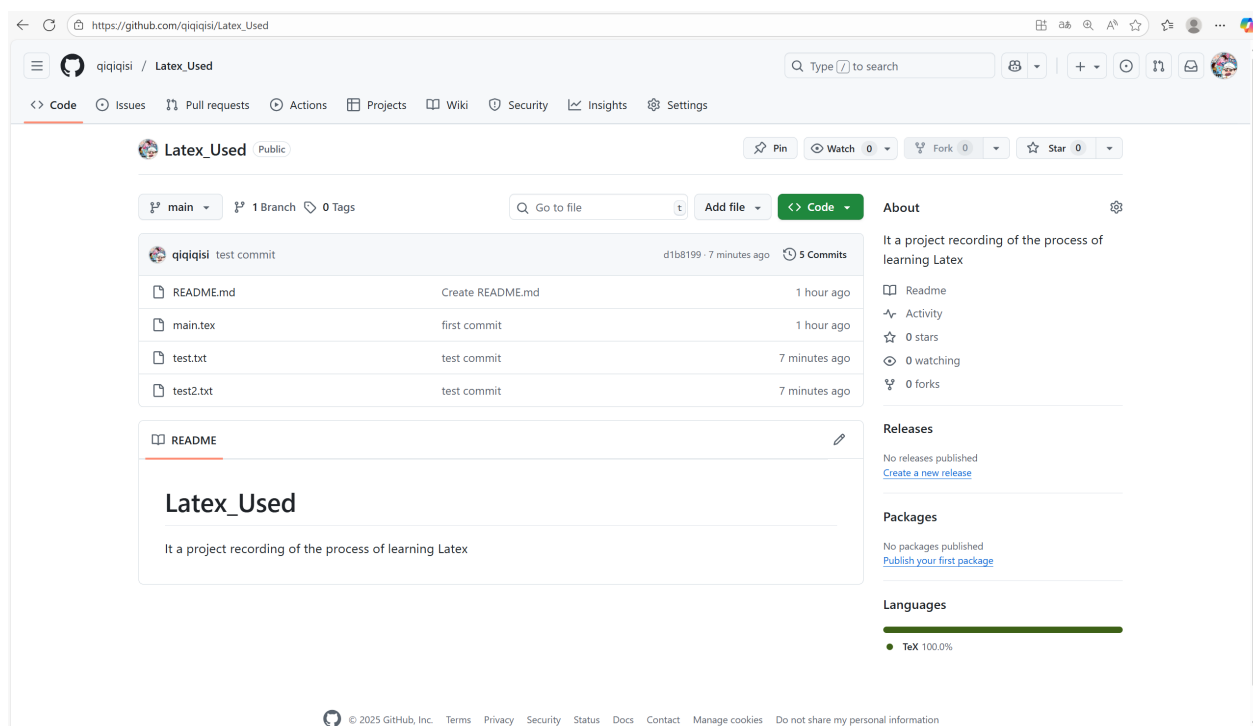
图 10: git push 运行图

图 11: git push 运行后的 github 仓库展示

9 git log

git log 可以查看日志。

图 12: git log 运行图

10　git log –all –graph –decorate

git log –all –graph –decorate 可以以图像化的方式显示所有历史提交。

图 13: git log –all –graph –decorate

11 git branch

git branch 可以查看分支。

图 14: git branch 运行图

12 git branch <name>

git branch <name> 可以创建分支。

图 15: git branch <name> 运行图

13 git push -u origin <name>

git push -u origin <name> 将本地分支推送到 github 仓库中。

图 16: git push -u origin <name> 运行图

14 git checkout <name>

git checkout <name> 可以转换分支。

图 17: git checkout <name> 运行图

15 git merge <name>

git merge <name> 可以和当前分支融合。

图 18: git merge <name> 运行图

16 git remote

git remote 可以查看远程仓库名称。

图 19: git remote 运行图

17 git remote <name> <url>

git remote <name> <url> 可以添加新的远程仓库。

图 20: git remote <name> <url> 运行图

18 git pull

git pull 可以在 github 仓库上有新的修改的时候在本地的文件同步修改。

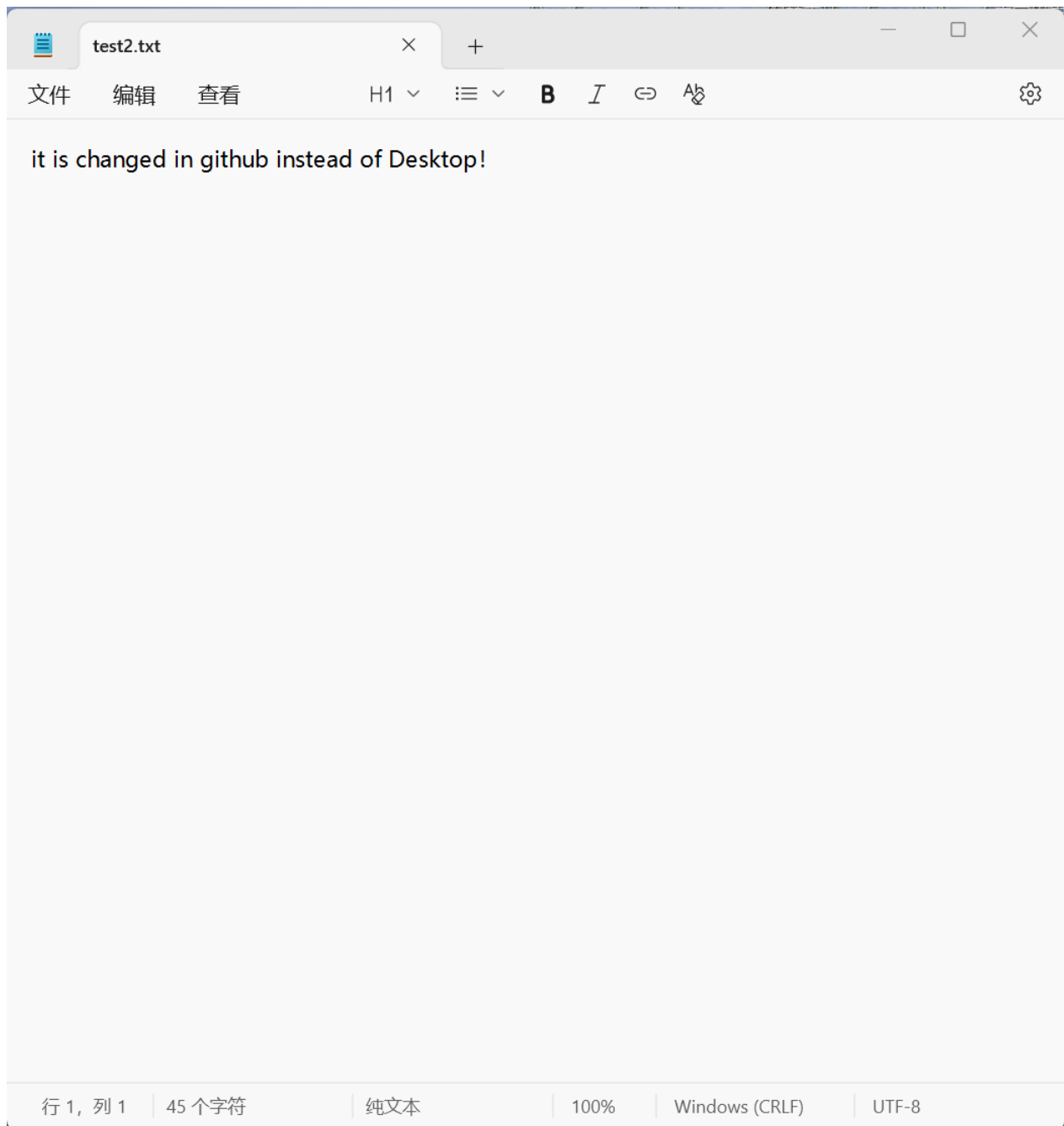图 21: 在 github 上修改文件内容

图 22: git pull 运行图

图 23: 被更新的本地文件

19 git blame

git blame 可以显示文件的修改信息。

```
Administrator@vice-Gabby MINGW64 ~/Desktop/Latex_Used/Latex_Used (main)
$ git blame
usage: git blame [<options>] [<rev-opts>] [<rev>] [--] <file>

    <rev-opts> are documented in git-rev-list(1)

    --[no-]incremental   show blame entries as we find them, incrementally
    -b                   do not show object names of boundary commits (Default:
 off)
    --[no-]root          do not treat root commits as boundaries (Default: off)
    --[no-]show-stats    show work cost statistics
    --[no-]progress      force progress reporting
    --[no-]score-debug   show output score for blame entries
    -f, --[no-]show-name show original filename (Default: auto)
    -n, --[no-]show-number
                         show original linenumber (Default: off)
    -p, --[no-]porcelain show in a format designed for machine consumption
    --[no-]line-porcelain show porcelain format with per-line commit information
    -c                   use the same output mode as git-annotate (Default: off
)
    -t                   show raw timestamp (Default: off)
    -l                   show long commit SHA1 (Default: off)
    -s                   suppress author name and timestamp (Default: off)
    -e, --[no-]show-email show author email instead of name (Default: off)
    -w                   ignore whitespace differences
    --[no-]ignore-rev <rev>
                         ignore <rev> when blaming
    --[no-]ignore-revs-file <file>
                         ignore revisions from <file>
    --[no-]color-lines   color redundant metadata from previous line differentl
y
    --[no-]color-by-age  color lines by age
    --[no-]minimal       spend extra cycles to find better match
    -S <file>            use revisions from <file> instead of calling git-rev-l
ist
    --[no-]contents <file>
                         use <file>'s contents as the final image
    -C[<score>]          find line copies within and across files
    -M[<score>]          find line movements within and across files
    -L <range>           process only line range <start>,<end> or function :<fu
ncname>
    --[no-]abbrev[=<n>]  use <n> digits to display object names
```

图 24: git blame 运行图

20   git bisect

git bisect 可以用二分法快速查找引入 bug 的提交。

```
Administrator@vice-Gabby MINGW64 ~/Desktop/Latex_Used/Latex_Used (main)
$ git bisect
fatal: need a command

usage: git bisect start [--term-(new|bad)=<term> --term-(old|good)=<term>]    [--no-checkout] [--first-parent] [<bad> [<good>...]] [--]    [<pathspec>...]
   or: git bisect (good|bad) [<rev>...]
   or: git bisect terms [--term-good | --term-bad]
   or: git bisect skip [(<rev>|<range>)...]
   or: git bisect next
   or: git bisect reset [<commit>]
   or: git bisect visualize
   or: git bisect replay <logfile>
   or: git bisect log
   or: git bisect run <cmd> [<arg>...]
```

图 25: git bisect 运行图