**Git实践任务学习笔记**

**一、学习资料来源**

Git官方文档：https://git-scm.com/doc

GitHub Guides：https://guides.github.com/

CSDN:https://blog.csdn.net/qq\_42808052/article/details/137476129?fromshare=blogdetail&sharetype=blogdetail&sharerId=137476129&sharerefer=PC&sharesource=ll0902&sharefrom=from\_link

CSDN:https://blog.csdn.net/sereasuesue/article/details/117080435?fromshare=blogdetail&sharetype=blogdetail&sharerId=117080435&sharerefer=PC&sharesource=ll0902&sharefrom=from\_link

**二、实践流程**

1、安装Git并配置用户名、邮箱。

2.创建本地仓库，提交三次代码以及笔记并推送至GitHub。

3.解决SSH密钥认证问题。

注：在git bash中的主要命令--

-git config --global user.name "你的名字" #全局配置

-git config --global user.email "你的邮箱" #全局配置

-cd /d # 进入项目目录

-mkdir git-project #新建文件夹

-cd /d/git\_project #已有文件夹，直接转

-git init # 初始化本地仓库

-echo "# My Git Project" > README.md # 创建 README 文件

-touch main.py # 创建示例代码文件

-git remote add origin https://github.com/LLinmagic/learning-notes.git

-git add . # 添加所有文件到暂存区

-git add <filename> ... # 添加某几个文件到暂存区

-git commit -m "初次提交" # 提交到本地仓库

-git branch -m master main # 修复分支名称问题（假设本地分支为 master）

-git push -u learning-nates main # 推送到远程仓库learning-notes的 main 分支

**三、遇到的困难及解决**

1.问题：推送时报错 Permission denied (publickey)。

解决：重新生成SSH密钥并添加到GitHub账户。

1. 问题：fatal: refusing to merge unrelated histories ：

这个错误表明你的本地仓库和远程仓库的历史记录（commit history）是完全独立的，Git 不知道如何将它们合并在一起。 这通常发生在以下情况： a.从一个全新的本地仓库开始，然后尝试将它推送到一个已经存在的远程仓库。 b.或者，远程仓库和本地仓库的初始提交（initial commit）不同。

解决：git pull learning-notes main --allow-unrelated-histories 允许 Git 合并不相关的提交历史

1. 问题：解决 non-fast-forward ：

这个错误表明你的本地分支落后于远程分支，Git 不允许你在本地分支落后的情况下直接推送更改。

解决： git pull learning-notes main git push -u learning-notes main

四、推荐书籍

《Pro Git》（免费在线阅读：https://git-scm.com/book/zh/v2）

**五、Git﻿知识点**

**1.概述**

Git 的官方网站为 https://git-scm.com。

（1）版本控制介绍

版本控制系统（VCS）可以记录文件随时间的变化历史，支持以下核心功能：

1. 版本回溯：可恢复任意历史版本，例如回退错误提交：

git reset --hard [commit-hash]

1. 协作管理：多人并行开发时追踪代码变更来源，如查看某行代码修改者：

git blame [filename]

1. 分布式版本控制 VS 集中式版本控制

| **类型** | **特点** |
| --- | --- |
| **集中式（如 SVN）** | **依赖中央服务器，需联网提交代码，单点故障风险高** |
| **分布式（如 Git）** | **每个开发者拥有完整仓库，支持离线工作，例如本地提交：git commit -m "msg"** |

（3）Git 工作机制与代码托管平台

Git 使用快照记录文件，支持快速创建和管理分支。代码托管平台（如 GitHub、Gitee、GitLab）提供便捷的仓库托管、协作功能。

**2.Git 安装**

（1）下载安装

Windows：官网https://git-scm.com/downloads下载安装包后勾选 Add to PATH 以支持命令行操作

Linux：使用包管理器快速安装，例如 Ubuntu：sudo apt-get install git

macOS：通过 Homebrew 安装：brew install git

（2）使用命令行验证安装：

git --version

（3）配置用户信息：  
# 全局配置（适用于所有仓库）

git config --global user.name "Your Name"

git config --global user.email "your.email@example.com"

# 为特定仓库单独配置（在仓库目录内执行）

git config --local user.email "work@company.com"

**3.Git 基本命令**

（1）配置用户信息

git config --global user.name "Your Name"

git config --global user.email "your.email@example.com"

可通过 --global 参数设置全局用户名和邮箱。

（2）初始化仓库

在项目根目录下初始化 Git 仓库：

mkdir git-project && cd git-project # 创建项目目录

git init # 初始化仓库（生成隐藏的 .git 文件夹）

已有文件夹，则：

cd /d/git\_project

git init

（3）查看仓库状态

查看当前工作区与暂存区状态，包括文件的改动、添加、删除等：

git status

（4）暂存、提交与版本管理

添加文件到暂存区：  
git add <filename>   
git add . # 添加所有改动

提交到本地库：  
git commit -m "提交描述信息"

查看提交日志：

git log # 显示详细的提交日志

git log --oneline # 简略显示

查看提交的差异：

git diff # 显示工作区与暂存区的差异

git diff HEAD~2 HEAD # 比较最近两次提交的差异

git diff --staged # 显示暂存区与本地库的差异/查看暂存区与上次提交的差异

例子:  
git add . # 添加所有修改到暂存区

git commit -m "fix: 修复登录按钮样式问题" # 语义化提交信息

git log --graph --oneline # 图形化显示提交历史

（5）分支管理

A.分支的创建与切换：分支使得开发多个功能独立进行，避免主分支的代码冲突和不稳定。

查看当前分支：

git branch

创建分支：

git branch <branch-name>

切换分支：

git checkout <branch-name>

创建并切换新分支：

git checkout -b <branch-name>

B.合并分支与解决冲突

合并分支（切换到主分支并合并）：

git checkout main

git merge <branch-name>

解决冲突：

Git 会标记冲突位置，手动编辑并解决冲突后：

git add <filename>

git commit -m "解决冲突描述"

例子：  
git branch -a # 查看所有分支（含远程分支）

git checkout -b dev # 创建并切换到 dev 分支

git merge dev --no-ff # 合并 dev 分支（保留合并记录）

git push origin --delete dev # 删除远程 dev 分支

（6）团队协作

Git 提供分布式版本控制，适用于不同场景的协作。

克隆仓库：git clone https://github.com/<user\_name>/<project\_name>.git

A.团队内协作

创建远程仓库并推送：

git remote add origin <repository-url>

git push -u origin main

拉取远程仓库更新：

git pull origin main

B.跨团队协作

a.Fork 项目：在 GitHub 上 fork 他人的项目，建立自己版本的副本。

b.提交 Pull Request：在自己的项目仓库修改后，发起 Pull Request 请求，将代码合并回原始项目。

**4.GitHub 集成**

创建远程仓库并设置别名：

git remote add origin <repository-url>

推送代码：

git push -u origin main

克隆远程仓库：

git clone <repository-url>

Eg.git clone https://github.com/<user\_name>/<project\_name>.git

SSH 免密登录：

# 生成密钥对（默认保存到 ~/.ssh/）

ssh-keygen -t ed25519 -C "your.email@example.com"

# 将公钥内容粘贴到 GitHub > Settings > SSH Keys

**5.IDEA 集成**

设置 Git：在 IDEA 设置中指定 Git 安装路径。

创建本地仓库：在 IDEA 中点击 Git > Enable Version Control Integration 以初始化本地仓库。

添加和提交：右键文件或项目，选择 Git > Commit 提交变更。

分支管理：在 IDEA 的 Git 工具窗口中可以创建、切换、合并分支。

推送与拉取：在 IDEA 中可以方便地将项目推送到远程仓库或拉取更新。