实验目的：掌握 Git 核心命令的使用方法，理解版本控制流程，提升团队协作效率  
实验环境：Git 2.30+，命令行终端，远程仓库

一、基础命令操作​​

1. Git Clone​​

功能：克隆远程仓库到本地。  
语法：git clone <仓库URL> [目录名]  
操作流程：

说明：

默认克隆所有分支和历史记录

支持指定分支：git clone -b dev <https://github.com/user/repo.git>

2. Git Add​​

功能：将工作区修改添加到暂存区。  
常用语法：

说明：

git add . 包含新文件，git add -u 仅包含已跟踪文件

3. Git Commit​​

功能：将暂存区内容提交到本地仓库。  
语法：

关键选项：

--amend：修改上一次提交信息

提交前需配置用户信息

4. Git Pull​​

功能：拉取远程仓库更新并合并到本地分支。  
语法：git pull origin <分支名>  
操作流程：

注意：

若存在冲突需手动解决后再次提交

等效于 git fetch + git merge

5. Git Push​​

功能：将本地提交推送到远程仓库。

强制推送：

git push -f origin main（覆盖远程提交，慎用）

6. Git Reset --hard​​

功能：强制回退到指定提交，丢弃工作区和暂存区所有修改。  
语法：git reset --hard <commit-id>

警告：

回退后未提交的修改将永久丢失

恢复误操作：通过 git reflog 找回历史提交ID

二、进阶命令操作​​

7. Git Branch​​

功能：管理分支（创建、查看、删除）。  
常用命令：

8. Git Checkout​​

功能：切换分支或恢复文件。  
常用场景：

分离头指针模式：

git checkout <commit-id>：临时查看历史代码（不关联分支）

三、实验总结​​

核心流程：  
克隆 → 修改 → 暂存 → 提交 → 拉取 → 推送​​，形成完整协作闭环

安全建议：

推送前先拉取，避免冲突

谨慎使用 reset --hard 和 push -f。

进阶技巧：

分支管理是团队协作的核心，推荐基于功能或修复创建独立分支

checkout 的恢复功能可挽救误删文件