

1. 创建版本库：

mkdir learngit # 创建空目录。

cd learngit

pwd # 显示当前目录。

git init # 将这个目录变成 git 可以管理的仓库

(要将文件放在 learngit 文件夹下 (子目录也可以))

2. 将文件增加到仓库。

① git ^{add} file.name # 将文件增加到仓库。

② git commit -m "备注 内容" # 将文件提交到仓库。

3. git status: 输出工作区状态。

4. git diff: 查看两次修改内容。

5. git log 查看历史记录。

git log --pretty=oneline, 每个历史输出为一行

Git 里序号不是用 1, 2, 3... 数字表示。

6. 版本回退。HEAD 表示版本, HEAD^ 表示上个版本

HEAD^^ 表示上上个, HEAD~100, 100 个版本。

git reset --hard HEAD^, 回退到上一个版本。

git reset --hard 版本号, 可以回到特定版本, 限制少

git reflog 查看命令历史

cat <filename> 查看文件内容。

7. 工作区

电脑中能看到的目录，比如 learngit 就是一个工作区

8. 版本库：

工作区中有个隐藏目录 .git，是 git 的版本库，不是工作区。
版本库最重要的是 stage(或者 index) 的暂存区。还有 Git 为我们创建的第一个分支 master，及指向 master 的一个指针叫 HEAD。
需要提交的文件修改都会先在暂存区然后，一次性提交所有修改。
但暂存区提交后，并没有对工作区做任何修改，那么工作区就“干净”

9. 修改：Git 管理的是修改，而不是文件。

第一次修改 → git add → 第二次修改 → git commit
comment

只会提交第一次修改，不会再提交第二次，工作区不干净

10. 撤销修改

git checkout -- <filename> #丢弃工作区的修改

包含两种情况：①. file 自修改后没有被放到暂存区，现在撤销修改回到和版本库一模一样的状态。

② file 已经被添加到暂存区，又作了修改，现在撤销修改回到添加到暂存区后的状态

总之就是让这个文件回到最近一次 git commit 或 git add 之后。

git reset HEAD <file> #将暂存区的修改撤销并放回工作区
已经提交了不适合的修改到版本库，要撤销，可版本回退

1. 删除文件

rm <filename> # 直接删除工作区文件

git rm <filename> # 从版本库中删除

git commit -m "说明"

2. 链接远程仓库

git remote add origin

git push -u origin master # 将本地所有内容推送到远程中

git push origin master. # 将本地 master 最新修改推送到远程仓库

git remote -v # 查看远程仓库信息

git remote rm origin # 删除“解除本地和远程绑定”

3. 从远程克隆

git clone git@github.com: name/gitskills.git

4. 创建分支

git checkout -b dev # 创建 dev 分支并切换到 dev

git branch # 查看所有分支

git merge dev # 合并 dev 分支到当前分支

git branch -d dev # 删除 dev 分支

5. switch

git switch -c dev # 创建并切换到新 dev 分支

git switch master # 切换到已有 master 分支

16. Bug 分支

git stash # 将当前工作现场“储藏”起来

git stash list # 查看储藏列表

恢复方法：

① git stash apply # 恢复后，stash 内容并不删除

② git stash pop # 恢复的同时将 stash 内容也删除了

17. 处理提交到上游分支

git cherry-pick <commit>

18. 开发一个新 feature，最好新是一个分支；

进行删除

如果需要并一个没有合并过该分支，可以使用 git branch -D <name>

19. 创建远程 origin 上的 dev 分支到本地，用 硬链接 创建 dev 分支

git checkout -b dev origin/dev

20. git pull # 将最新的提交从 origin/dev 抓下来

21. git rebase # 将分支的提交变成一条直线 但是本地的对提交有修改

git log --graph --pretty=oneline --abbrev-commit

查看 Commit 历史记录 查看合并分支图

22. 创建标签：(分支)

git tag v1.0 # 首先切换到需要打标签的分支上 commit

git tag # 可以查看所有标签

git tag v1.0 <文件名> # 对历史记录打标签

git show v1.0 # 查看标签信息

git tag -a <tagname> -m " " # 指定标签信息

标签总是和某个 commit 挂钩，如果这个 commit 既出现在 master 分支，又出现在 dev 分支，那么这两个分支上都能看到这个标签。

git tag -d v0.1 # 删除标签

git push origin <tagname> # 推送某个标签到远程

git push origin --tags # -v 强制推送尚未推送到远程的本地标签

git push origin :refs/tags/<tagname> # 删掉一个远程标签