GIT下载安装

下载链接：Git for Windows

配置个人用户名和邮箱
git config --global user.name "runoob"
git config --global user.email test@runoob.com

Github配置SSH密钥连接（附相关问题解决） - 知乎

创建新仓库

创建新文件夹，打开，然后执行

```
git init
```

以创建新的 git 仓库。

工作流

你的本地仓库由 git 维护的三棵“树”组成。第一个是你的 工作目录，它持有实际文件；第二个是 暂存区（Index），它像个缓存区域，临时保存你的改动；最后是 HEAD，它指向你最后一次提交的结果。

添加和提交

你可以提出更改（把它们添加到暂存区），使用如下命令：

```
git add <filename>
```

```
git add *
```

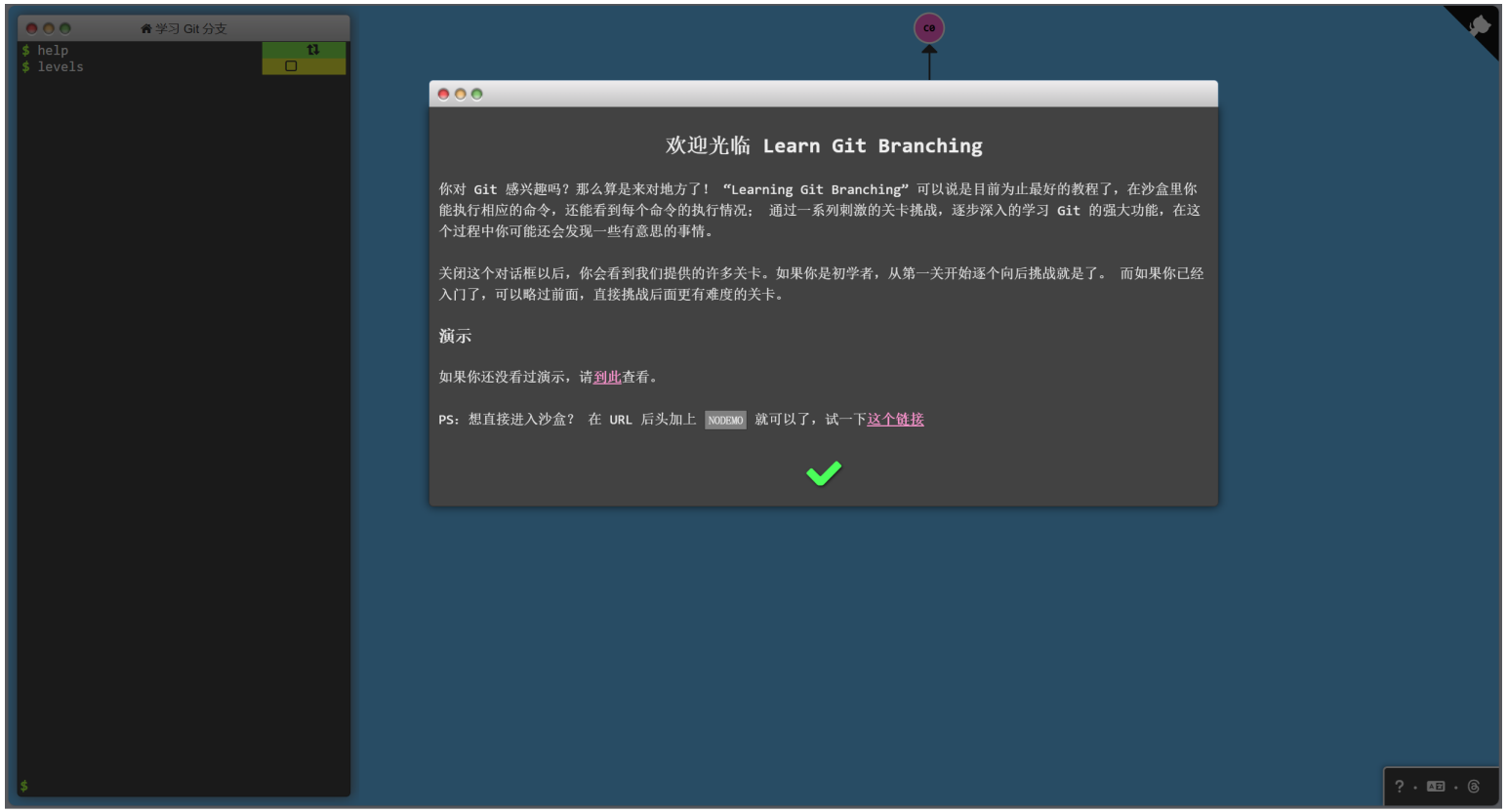
这是 git 基本工作流程的第一步；使用如下命令以实际提交改动：

```
git commit -m "代码提交信息"
```

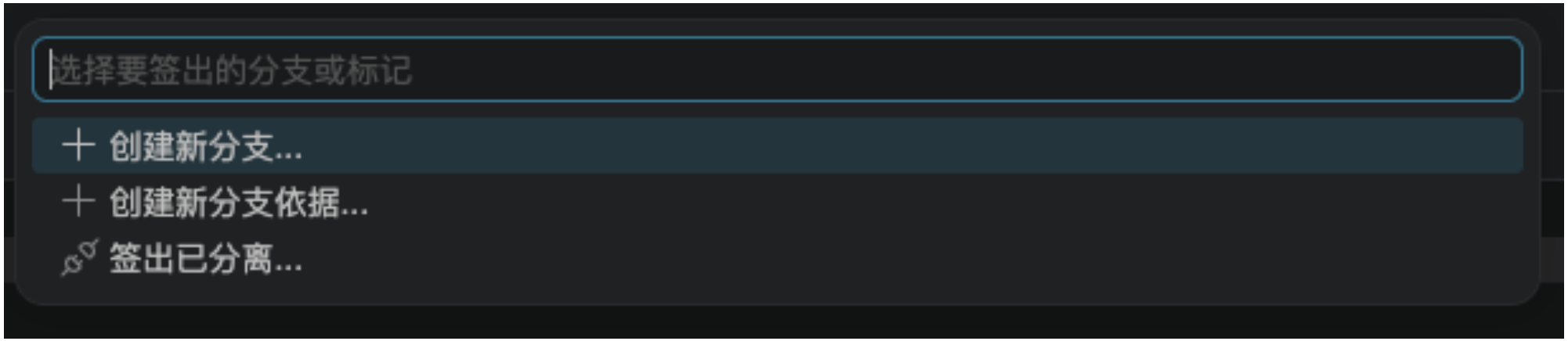
现在，你的改动已经提交到了 **HEAD**，但是还没到你的远端仓库。

学习git基本命令

- Learn Git Branching



以列表形式查看	把源代码管理面板里的更改文件按列表显示，而不是树状目录显示	只是 VS Code 显示方式，不是 Git 命令
查看和排序	调整更改文件的显示方式和排序方式，例如按文件名、路径、状态排序	只是 VS Code 视图设置
拉取	从远程仓库拉取最新提交，并合并到当前分支	`git pull`
推送	把本地提交上传到远程仓库	`git push`
克隆	从远程地址复制一个完整仓库到本地	`git clone <url>`
签出到…	切换到某个分支、标签或提交	`git checkout <name>` / `git switch <branch>`
抓取	获取远程仓库最新信息，但不自动合并到当前分支	`git fetch`
提交	把已暂存的更改保存成一次 Git 历史记录	`git commit`
更改	当前工作区里被修改、新增、删除但还没提交的文件	`git status` 可查看
拉取，推送	和远程仓库同步相关的操作菜单，通常包含拉取、推送、同步等	`git pull` / `git push`
分支	分支相关操作，如新建分支、切换分支、合并分支、删除分支	`git branch` / `git switch` / `git merge`
远程	远程仓库相关操作，如添加远程、查看远程、发布分支	`git remote`
存储	通常指 Git Stash，把当前未提交的修改临时保存起来	`git stash`
标记	Git Tag，给某个提交打标签，常用于版本号，如 `v1.0.0`	`git tag`
显示 GIT 输出	打开 VS Code 的 Git 日志输出面板，查看 VS Code 实际执行了哪些 Git 命令以及报错信息	不是 Git 命令，是调试窗口

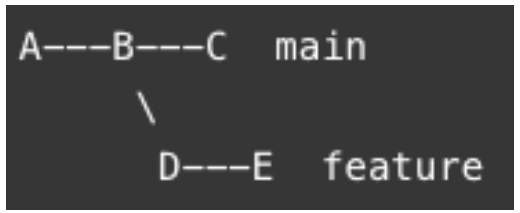


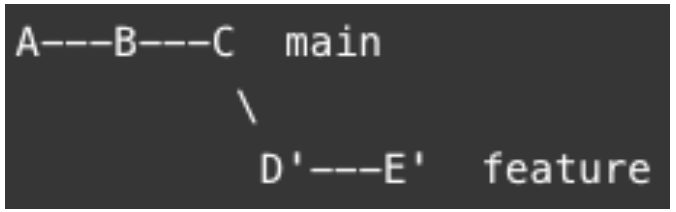
创建新分支	从**当前所在位置**新建一个分支，并切换过去	`git switch -c` 新分支名	你现在在 `main`，想从当前代码开始开发新功能
创建新分支依据…	选择一个**指定来源**作为起点来新建分支	`git switch -c` 新分支名 起点	想基于 `origin/main`、某个旧分支、某个 tag 或某个提交创建分支
签出已分离	切到某个提交/tag，但**不在任何分支上**	`git checkout <commit>`	临时查看旧版本、测试某个历史提交，不打算直接开发

创建新分支 = 从**当前位置**开分支
创建新分支依据**…** = 从**你指定的位置**开分支

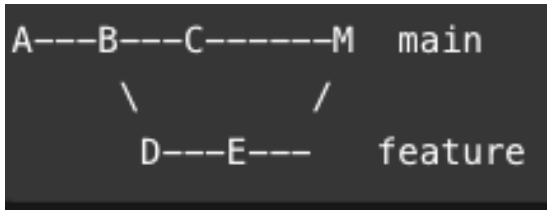
提交	把**已经暂存**的内容提交一次	`git commit`
提交已暂存文件	和上面基本一样，强调“只提交暂存区里的文件”	`git commit`
全部提交	把**所有已跟踪文件**里改动过的内容直接提交，不用先一个个 `add`	`git commit -a`
撤消上次提交	撤回最近一次提交，通常保留改动回到工作区或暂存区	常见是 `git reset --soft HEAD~1` 或 `git reset HEAD~1`
中止变基	取消当前正在进行的 `rebase`，回到变基前状态	`git rebase --abort`
提交（修改）	修改**上一次提交**，把当前暂存内容补进上一条提交里	`git commit --amend`
提交已暂存文件（修改）	和上面几乎一样，强调只用暂存区内容去改上一条提交	`git commit --amend`
全部提交（修改）	把所有已跟踪改动一起补进上一条提交	`git commit -a --amend`
提交（已签收）	提交时附带“Signed-off-by”签收信息	`git commit --signoff`
提交已暂存文件（已署名）	提交暂存区内容，并附带签收/署名信息	`git commit --signoff`
全部提交（已署名）	提交所有已跟踪改动，并附带签收/署名信息	`git commit -a --signoff`

变基 就是 `rebase`。你可以把它理解成：
把你这条分支上的提交，重新接到另一条分支的最新位置上。

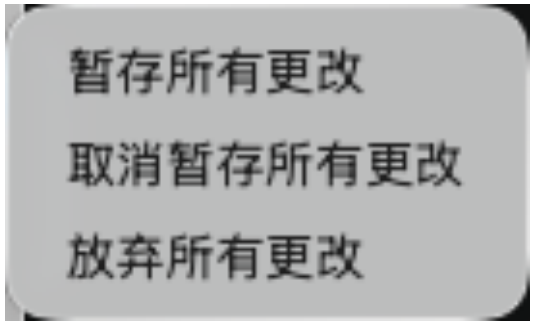




git switch feature
git rebase main



git switch main
git merge feature



暂存所有更改	把所有修改加入暂存区，准备提交	`git add .` 或 `git add -A`	不会
取消暂存所有更改	把已经暂存的内容撤回到工作区	`git restore --staged .`	不会
放弃所有更改	把工作区改动直接丢掉，恢复到最近一次提交状态	`git restore .`，有时也会包含删除未跟踪文件	会**丢代码**

同步	先拉取再推送，让本地和远程尽量保持一致	通常是 `git pull` + `git push`	一键同步，省事
拉取	从远程拿最新提交，并直接合并到当前分支	`git pull`	默认是“拉下来再 merge”
拉取（变基）	从远程拿最新提交，但不用 merge，而是 rebase 到最新远程后面	`git pull --rebase`	历史更直，不多 merge 提交
拉取自…	从你指定的远程或分支拉取	`git pull <remote> <branch>`	不是默认 origin/当前跟踪分支
推送	把本地提交上传到当前远程分支	`git push`	最常见
推送到	推送到你指定的远程或分支	`git push <remote> <branch>`	手动指定目标
抓取	只获取远程最新信息，不自动合并到本地	`git fetch`	安全，只更新“情报”
抓取（删除）	获取远程信息时，顺便清理本地已经失效的远程分支引用	`git fetch --prune`	会把已删除远程分支的本地引用清掉
从所有远程存储库中抓取	把所有远程仓库都 fetch 一遍	`git fetch --all`	不只抓 `origin`

git commit 规范

类型	说明
feat	✨ 新增功能
fix	🐛 修复 bug
docs	📝 文档变更（仅文档，不改代码）
style	💅 格式修改（空格、分号等）
refactor	♻️ 代码重构（非功能/修复）
perf	⚡ 性能优化
test	✅ 添加或修改测试代码
chore	🔧 构建过程、依赖管理等
revert	⏪ 回滚 commit

回退版本

临时查看旧版本（不改变分支）

git checkout <commit-id>

进入“游离 HEAD（detached HEAD）”状态。

想回到原分支再输入：git checkout main

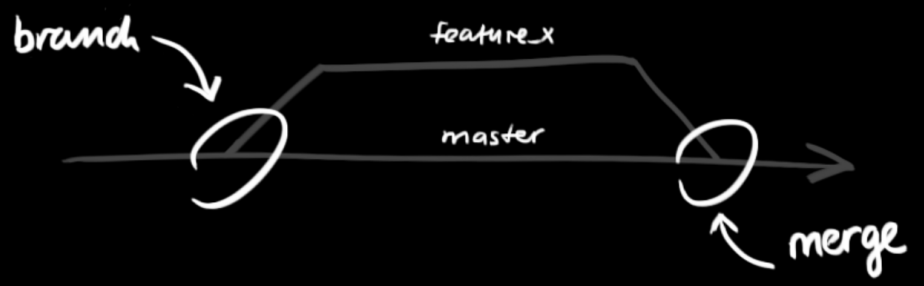
重置当前分支到旧版本（永久回退）

git reset --hard <commit-id>

目的	命令
查看旧版本	git checkout <commit-id>
永久回退	git reset --hard <commit-id>
安全回滚	git revert <commit-id>
只回退提交记录	git reset --soft <commit-id>

分支

分支是用来将特性开发绝缘开来的。在你创建仓库的时候，*master* 是“默认的”分支。在其他分支上进行开发，完成后再将它们合并到主分支上。



创建一个叫做“feature_x”的分支，并切换过去：

```
git checkout -b feature_x
```

切换回主分支：

```
git checkout master
```

再把新建的分支删掉：

```
git branch -d feature_x
```

除非你将分支推送到远端仓库，不然该分支就是 *不为他人所见的*：

```
git push origin <branch>
```

提交PR -Pull Request

有权限（协作者 / 组织成员）时

git clone 仓库地址

git checkout -b stu-xxx

修改、提交

git push origin stu-xxx

无权限（协作者 / 组织成员）时

第一步：fork 仓库（GitHub 页面右上角点 Fork）

第二步：克隆你自己的 fork 仓库

git clone https://github.com/你自己的用户名/项目名.git

第三步：创建分支，提交修改

git checkout -b my-fix

修改、提交、推送

git add .

git commit -m "修复拼写错误"

git push origin my-fix

第四步：在 GitHub 页面点“Pull Request”

base 仓库选上游原项目，compare 是你的分支