HenCoder Plus 第 23 课 讲义

Feature Branching 和实用常用指令

branch 的作用

- 让项目可以同时做多件事
- 未做完的事不会被项目真正收录

Feature Branching

做法：每开发一个新的功能做一个修复，都使用单独的分支，在做完之后 merge 到 master 去

- 本地 merge：由于别人可能在你之前 push 过，所以你的 push 可能失败。所以通常会需要先 pull 一下，然后再 push
- 使用 github：先创建 pull request，在同事审阅完成之后，通过按钮实现在线 merge
  - pull request 是什么：是开发者对远端仓库的提出的「拉取某个 branch」的请求

merge 冲突

当 Git 不知道怎么合并某两处冲突的修改时，会中断自动合并，并对冲突文件进行标记。解决方法：

1. 把文件手动修改好
2. 把修改的内容使用 add 来添加进暂存区
3. 用 `git merge --continue` 来继续自动合并流程

rebase

把当前 commit（以及它之前的 commits）放应用到指定的需要 rebase 的 commit 上。

> Git 中的每一个 commit 都是不会改变的，所以 rebase 之后的每个 commit 都是新产生的，而不是对原先的 commit 进行「修改」

rebase 冲突

rebase 的冲突解决方法和 merge 冲突一样，只是把 `git merge --continue` 改成 `git rebase --continue` 就行了

提交过的东西写错了

1. 最新的一条内容需要修改：

   `git commit --amend`

   > 「修改」只是概念行为，实质上会产生一个新的 commit

2. 旧的内容需要修改：

   `git rebase -i 指定的commit` 交互式 rebase

   交互式 rebase 不止可以修改和删除 commit，还能增加

3. 已经 push 到 master 的内容需要删除：

   不能用 rebase，因为 master 上的东西是不能强行修改的

   可以用 `git revert 指定commit` 来撤销。它的原理是创建一个新的 commit，内容是指定 commit 的「相反内容」

reset

把当前 branch 指向指定的 commit。

- `git reset 指定commit`

  移动到指定 commit，并把当前位置和指定位置的文件差异加入 working tree

- `git reset --hard 指定commit`

  移动到指定 commit，并重置 working tree

reset 和 checkout 的区别

它们都是移动 HEAD，但 chekcout 移动的时候是自己移动，不带着 branch 一起；而 reset 会带着 branch 一起移动。