第一题puls

5

.git/ 目录是什么，.git目录里放了什么？

我理解的git就是一本“历史书”，记录程序的演变过程，而git目录就是git项目的核心，储存了这本书的内容，包括程序的前后代，修改时间,注释等等

> .git目录包含详细内容（以下来自网络）
> hooks目录 ：该目录包含了一些脚本文件，这些脚本可以在Git操作中被调用，比如在提交代码时运行pre-commit钩子
> refs目录 ：该目录包含了Git引用的信息，如分支、标签等。这些引用可以帮助Git跟踪代码提交的历史记录
> objects目录 ：该目录包含了Git存储代码提交的实际数据，其中分为三个子目录，分别是blob、tree和commit
> index ：文件保存暂存区的信息
> HEAD文件 ：该文件指向当前工作分支的最新提交
> config文件 ：该文件包含了Git项目的配置信息，如用户信息、远程仓库信息等

在IDEA的git工具

添加 : 将文件或改动添加到暂存区

提交 : 将暂存区中的改动保存为一个新的版本

回滚 : 返回到之前的某个提交点

签出 : 恢复指定分支的内容到工作目录， git checkout branch_name.

删除 : 删除本地分支（不是从远程仓库删除）用 git branch -d branch_name，删除文件或目录使用 git rm [file]

合并: 合并两个或多个分支到一起，比如 git merge branch_name。

变基: 使用 git rebase 进行基于另一个分支上的修改，这会影响到历史提交顺序。

克隆 : 创建一个全新的Git仓库副本，git clone repository_url.

提取 : 这个术语可能不直接对应Git命令，但如果是从某次提交中提取特定变动，可以考虑使用 git diff 和 git cherry-pick。

更新: 更新当前分支到与远程分支同步，通常用 git pull 或 git fetch && git merge origin/branch_name.

将传入更改合并到当前分支 : 类似于 update，但同时合并远程分支的更改，git pull origin branch_name.

在传入更改上变基当前分支 : 就是在本地分支基础上应用远程分支的更改，然后调整提交记录。

推送 : 将本地分支的最新变化推送到远程仓库，git push origin branch_name.

fork和clone有什么区别，Pull Request和push有什么区别

Fork：通常在GitHub等托管平台上，fork是指用户复制一份其他用户的仓库到自己的账户下。这相当于创建了一个副本，你可以对这个副本进行修改，并独立开发。fork的主要目的是共享和协作，比如创建项目分支。

Clone：则是将远程仓库克隆到本地机器上，这样你就有了该代码库的所有内容，可以对其进行编辑、提交更改后再推送回原仓库，或者创建新的分支。克隆是个人开发者获取项目副本以便进行工作的一部分。

（我的理解就是要不要把代码返回，如果是复制别人的，就是fork，如果在修改过后还要把代码传回远程仓库，就是clone）

Pull Request和Push的区别：

Push：当你完成了代码修改并认为它们值得添加到原始仓库时，会执行push操作。Push将你的本地更改推送到远程仓库，让其他人能够在其本地克隆并看到你的更新。

Pull Request：这是一种请求合并的方式，当你在一个Forked的仓库上做了改动，你想把这些改动合并到原始仓库（通常是开源项目）。通过创建Pull Request，你邀请项目的维护者审查你的更改，并允许他们将其合并到主分支。这是为了保持代码质量控制和避免破坏现有功能的一种流程。

（我的理解就是push是你主动将更改发布出去，而Pull Request是在多人合作环境中请求管理员接受你的更改）

工作区、暂存区、本地仓库、远程仓库

工作区：编写代码的地方
暂存区：就是代码写完了，但是还没有正式提交到远程仓库时，文件存放的地方。
本地仓库：文件在本地储存的地方
远程仓库：文件在服务器上存储的地方
现在工作区编写文件，然后放在暂存区（准备提交，但是还没有正式提交），当执行commit后，暂存区的文件会被储存在本地仓库，还可以通过push，把文件传输到远程仓库。

什么是git冲突？冲突发生的条件是什么，有哪些操作会引发冲突？应该如何处理？

当一份文件被不同的人更改，再试图合并时（产生不兼容冲突)就可能会发生。
我搜到冲突原因都有双向合并 分支切换 外部合并，但其实好像就是提交冲突，比如远程本地仓库版本信息不同等。
解决办法：
查看冲突办法：git status或git diff
找到冲突，解决（选择哪保留哪更改）
最后提交变更

[拓展]

感觉应该没有什么变化吧，文件状态会保持原状。

6

之前在配置githob交作业的时候，完成了前三步。 git@github.com:cyh6213/diyiti.git

相当于develop分支已经有了新的提交，如果这些提交与我在本地的更改产生了冲突，git就会拒绝推送。

6

git-flow 通过 5 种分支来管理整个工程

分支	周期	说明
master	长期	主分支，用于存放对外发布的版本，任何时候在这个分支拿到的，都是稳定的分布版
develop	长期	开发分支，用于日常开发，存放最新的开发版
feature	短期	功能分支，它是为了开发某种特定功能，从 develop 分支上面分出来的。开发完成后，要再并入 develop
release	短期	预发分支，它是指发布正式版本之前（即合并到 master 分支之前），我们可能需要有一个预发布的版本进行测试。预发布分支是从 develop 分支上面分出来的，预发布结束以后，必须合并进 develop 和 master 分支
hotfix	短期	bug 修补分支，从 master 分支上面分出来的。修补结束以后，再合并进 master 和 develop 分支

github-flow 可以认为是 git-flow 的一个简化版，它适用于持续部署的工程，直接将最新的功能部署到 master 分支上，不再操作 develop 分支。同时通过 CI&CD 的使用，不再需要额外的 release 和 hotfix 分支。github 还结合了推送请求（pull request）功能，在合并 feature 分支之前通过PR请求其他成员对代码进行检查。

> 第一步：根据需求，从 `master` 拉出新分支，不区分功能分支或补丁分支。
>
> 第二步：新分支开发完成后，或者需要讨论的时候，就向 `master` 发起一个 pull request（简称PR）。
>
> 第三步：Pull Request既是一个通知，让别人注意到你的请求，又是一种对话机制，大家一起评审和讨论你的代码。对话过程中，你还可以不断提交代码。
>
> 第四步：你的Pull Request被接受，合并进 `master`，重新部署后，原来你拉出来的那个分支就被删除。（先部署再合并也可。）

如果是我从头开发一个项目，那还需要对项目进行初始化

可能遇到的问题
冲突，分支管理混乱，代码规范不一致等等