git入门教程

git安装

- 登录官网：`https://git-scm.com/`
- 安装过程：一直点下一步即可。

    1. 可以自行选择安装盘符位置
    2. git的环境路径选择“use git from git bash only”即可
    3. 其他设置默认即可
    4. 安装成功后，即可鼠标右键，选择git bash here打开终端

git结构

- git的本地结构分为工作区（Working Directory），暂存区（Index / Staging Area），和本地仓库（Local Repository）

    1. 工作区：项目的实际目录，包含编辑的文件和子目录，以及实际进行修改和编辑文件的地方
    2. 暂存区：一个暂时存放改动的地方，在做出一些修改后，可以使用 git add 命令将这些修改添加到暂存区，这个阶段允许选择性地暂存某些文件，而不是一次性提交所有的更改
    3. 本地仓库：是包含项目完整历史记录的地方，保存在本地计算机上，运行 git commit 命令时，暂存区中的改动会被提交到本地仓库，本地仓库包含完整的项目历史记录，每个提交都有一个唯一的 SHA-1 标识符

初始化本地库

- 创建一个文件夹
- 打开git bash here终端，在终端内部右击选择options-test-select（可以设置字体大小），以及设置编码形式，options-test-locae(c)-character set(UTF-8)
- git命令和linux命令一样
- 设置用户名：`git config --global user.name "username"`
- 设置邮箱：`git config --global user.email "123@gmail.com"`
- 进入初始化目标的文件夹`cd path`
- 然后使用初始化命令`git init`，初始化会得到.git的文件夹（是隐藏文件夹），可以通过`ll -la`查看
- 注意.git目录下的文件不要修改或是删除

git常用命令

- git add，git commit命令

    1. 首先在上述创建的文件夹中，新建一个文件
    2. 将文件提交到暂存区，在终端输入`git add Demo.txt`
    3. 将暂存区的文件提交到本地库`git commit -m "注释文件信息（message）" Demo.txt`
    4. 注意：不放在本地仓库中的文件，git是不进行管理的，即使放在本地仓库文件，git也不管理，必须通过add、commit命令操作才可以将内容提交到本地库。

- git status命令

    1. git status查看 Git 仓库当前状态的命令。执行这个命令将显示有关工作区、暂存区和本地仓库的信息，包括未提交的更改、已暂存的更改以及分支信息。

- git log命令

    1. 是一个用于显示 Git 仓库提交历史的命令。执行这个命令将显示从最新到最旧的提交记录，包括每个提交的作者、日期、以及提交信息。
    2. 当历史记录过多，查看时会有分页情况，到下一页用空格，到上一页用b，到尾页显示end，退出点击q。
    3. `git log --pretty=oneline`或是`git log --oneline`
    4. `git reflog`一个用于显示引用日志的 Git 命令。它记录了 HEAD 和分支引用的历史，包括每次提交和分支切换的信息。git reflog 主要用于查看仓库中的引用变更历史，特别是当需要找回丢失的提交或分支时。

- git reset命令

    1. 用于撤销提交、移动分支的 HEAD 指针，以及更改暂存区和工作区的状态。这个命令有不同的模式，可以根据需要选择
    2. `git reset <commit-hash>`带任何参数时，默认是软重置,保留暂存区和工作区的更改，只移动 HEAD 指针到指定的提交。
    3. `git reset --hard <commit-hash>`,会重置 HEAD 指针，清空暂存区，并且丢弃工作区的所有更改
    4. `git reset --mixed <commit-hash>`,会重置 HEAD 指针，清空暂存区，但保留工作区的更改
    5. `git reset <file>` 用于撤销暂存的特定文件,取消对指定文件的暂存，但保留工作区中的修改

- git删除命令

    1. 删除工作区中的文件`rm doc_name`
    2. 将删除操作同步到本地库`git add doc_name`,`git status`,`git commit -m "notes" doc_name`
    3. 找回本地库中删除的文件，实际上就是将历史版本切换到刚才添加文件的那个版本即可`git reset --hard hash`

- git中的diff命令

    1. 当工作区和暂存区不一致时，要对比差异。`git diff doc_name`， git是按照行为单位管理数据。
    2. 多个文件的比对使用`git diff`，比较工作区和暂存区中的所有文件
    3. `git diff [历史版本] [文件名]`，这是比较暂存区和工作区的内容

git分支

- 分支查看 1.`git branch -v`查看分支
- 创建分支
    1. `git branch [分支名称]`在哪个分支上，分支前有*显示
- 切换分支
    1. `git checkout [分支名称]`

git本地库

- 查看别名
    1. `git remote -v`

- 添加别名
    1. `git remote add origin https://github.com/zhangsan/lisi.git`git remote add: 这是 Git 命令，用于添加一个远程仓库。origin: 这是给远程仓库起的一个别名，通常约定使用 "origin" 表示主要的远程仓库。最后是远程仓库的 URL 地址，指向 GitHub 上的 zhangsan/lisi.git 仓库。可以使用 "origin" 作为这个远程仓库的快捷名称，进行拉取（fetch）、推送（push）等操作
- 推送操作
    1. `git push origin master`，origin是远程库的别名，master是你要推送的分支
- 克隆操作
    1. `git clone [远程仓库的 URL 地址]`，git克隆后可以完成初始化本地库，将远程库内容完整的克隆到本地，以及创建远程库的别名。
- 远程库修改的拉取操作
    1. 拉取操作pull操作，相当于fetch+merge。
    2. `git fetch origin master`，在抓取操作执行后，只是将远程库的内容下载到本地，但是工作区中的文件并没有更新，工作区还是原来的内容。可以查看远程库的内容`git checkout origin/master`,`ll`,当内容没有问题，可以进行合并操作。
    3. 进行合并前要将分支切换回来`git checkout master`,`git merge origin/master`
    4. 远程库的拉取可以直接使用pull命令，`git pull origin master`

免密操作

- 进入用户主目录 `cd ~`
- 执行命令，生成一个.ssh的密匙`ssh-keygen -t rsa -C [github账号注册的邮箱]`，三次回车默认
- 进入C盘用户下，会有.ssh目录，其下面有两个文件夹，打开id_rsa.pub文件，将内容复制。将其粘贴在github中的setting-ssh and GPG keys中，title随便写，key处即粘贴刚才的内容
- 生成密钥以后，就可以正常进行push操作了。
- 对ssh地址起别名`git remote add origin_ssh[别名] [ssh地址]`

相关问题

1. Line Endings
    - LF (Line Feed): This is the line ending used in Unix-based systems (e.g., Linux, macOS).
    - CRLF (Carriage Return + Line Feed): This is the line ending used in Windows systems.
    - LF will be replaced by CRLF the next time， Git is warning you that line endings in your files, which are currently using LF, will be converted to CRLF when Git processes these files. This usually happens because your Git configuration is set to handle line endings differently depending on your operating system.
2. Git 提供了以下设置来处理行结束符：
    - core.autocrlf： true: Git 在检出文件时将 LF 转换为 CRLF（适用于 Windows 用户），在提交文件时将 CRLF 转换为 LF。input: Git 在提交文件时将 CRLF 转换为 LF，但在检出文件时不会将 LF 转换为 CRLF（适用于 Unix 系统）。 false: Git 不会自动转换行结束符。
3. 检查 Git 配置：
    - 可以通过以下命令检查当前的 Git 配置：`git config --global core.autocrlf`
    - 对于 Windows 用户（以避免行结束符问题）:`git config --global core.autocrlf true`
    - 对于 Unix 系统用户（保留 LF 行结束符）:`git config --global core.autocrlf input`

4. 标准化行结束符

    - 在仓库根目录中添加 .gitattributes 文件,内容如下"* text=auto"
    - 提交 .gitattributes 文件,"git add .gitattributes, git commit -m "Add .gitattributes to control line endings"
    - 重新标准化文件,"git add --renormalize . , git commit -m "标准化行结束符""