**一、**下载 Watt Toolkit 软件 加速github----网址（https://steampp.net/）

**二、**git config --global user.name "名字"

git config --global user.email "邮箱地址"

git config --global credential.helper store (保存用户名和密码)

git config --global --list (查看git配置信息)

//如果用户名有空格则需要双引号括起来

若没有的话可以省略

省略(Local)：本地配置，只对本地仓库有效

--global：全局仓库，所有仓库生效

--system：系统配置，对所有用户生效

**三、**使用Git Bash软件clone代码

先进入要克隆到的文件夹

cd /D

cd 文件夹名称

git clone (git地址链接)

四、提交代码到git仓库

三个步骤：add(预添加代码到本地仓库) commit(提交代码到本地仓库) push(将本地代码推送到git仓库)

先输入（ls）进入仓库里 (.git文件是隐藏的 如果需要通过终端查询 命令 ls -a)

然后进入文件夹 cd

输入命令 git add (文件名)

git commit -m"注释信息"

git commit -a -m”注释信息” 可以一个命令完成添加暂存和提交两个动作

**若没有输入注释信息则会进入vim编辑器 补充注释信息**

**需按i键进入编辑模式**

**在第一行输入信息**

**按Esc按键返回**

**输入：wq退出**

git push ----推送到git仓库

.git所在的目录 工作区

git add 添加到暂存区 git commit 添加到本地仓库

git push 推送到git仓库

git 不会将空的文件夹添加到仓库里

五、创建git仓库

1、在电脑本地创建(git init) 2、git clone 一个网络的git仓库

(创建一个目录 mkdir 目录名字 (make directories))

查看文件夹里的全部文件 ls -altr

查看文件夹文件 ls -ltr

进入上一层文件夹 cd ..

删除文件夹 \rm -rf 文件夹名称

若在git init 后加一个文件夹名称则会初始化加的文件夹 若没有则创建一个

六、git常用命令

git init 创建仓库

git status 查看仓库状态(查看当前仓库处于那个分支 有哪些文件以及处在那个状态)

git add 添加到暂存区

git commit 提交

git restore --staged 文件名 将暂存区的文件卸载

echo “文件内容” > 文件名 创建一个文件并添加内容

echo “ 内容 “ >> 文件名 表示在该文件后追加内容

git add \*.txt 将所有.txt文件添加到暂存区、

git add . 将当前文件夹下所有文件添加到暂存区里

（ . 表示当前目录）

cat 文件名 查看该文件内容

git ls-files 查看暂存区的包含的文件

git log 查看git add提交记录

git log --oneline 只显示每次提交的ID和提交信息

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*git reset 撤销之前提交的版本

git reset --soft 表示回退到某一个版本 保留工作区和暂存区的所有修改内容

：：

git reset --hard 表示回退到某一个版本 丢弃工作区和暂存区所有修改内容（会删除文件）

git reset --hard HEAD^表示回退到上一个版本

会删除工作区和暂存区文件 （回退的版本）

git reset --mixed 回退到某一个版本保存工作区修改内容 丢弃暂存区修改内容 \*\*\*\*\*（**git默认配置**）可以省略--mixed

git reflog 查看之前操作历史记录

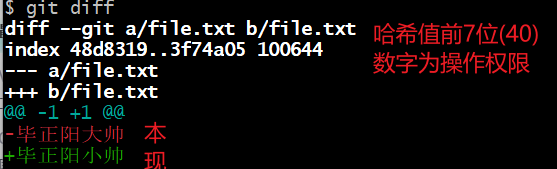
先找到删除之前的版本号 然后执行 git reset --hard +版本号

就可以回溯到之前的版本了

git diff 可以查看文件在工作区 暂存区以及版本库之间的差异

文件在不同版本之间的差异 或 文件在两个分支之间的差异

若git diff后什么都不加 则默认比较工作区和暂存区之间的差异内容 它会显示发生更改的文件以及更改的详细信息

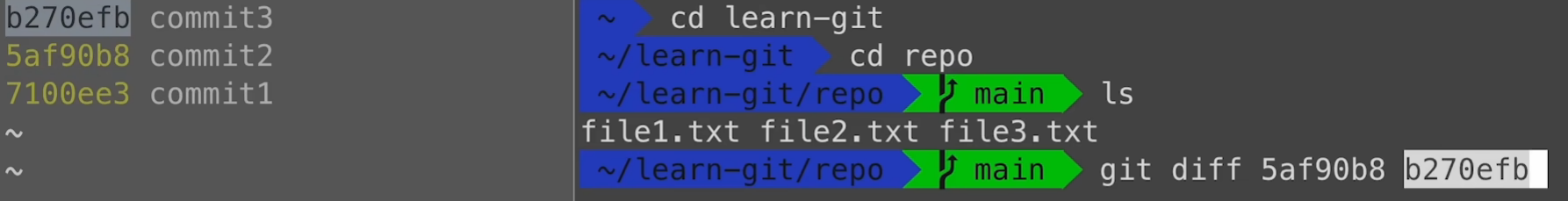


重新add后则不会出现

git diff HEAD 比较工作区和版本库之间差异

git diff --cached 比较暂存区和版本库之间差异

git diff (+) 两个版本的提交id可以查看两个版本的内容差异



也可以使用HEAD表示当前分支的最新提交

git diff HEAD~ HEAD表示当前版本与上一个版本比较

HEAD~2表示HEAD之前两个版本

G、it diff HEAD~ HEAD file.txt 表示只查看这个文件的差异

rm 文件名 表示在本地工作区删除文件

git restore 文件名 可以将刚删除的文件恢复（也可以使用git status获得提示）**注：：：该操作只在本地删除 需要将该操作上传到缓存区**

**使用git add git commit**

使用git rm 文件名 命令则会把工作区和暂存区的文件都删除：：：仍需提交

git rm --cached 文件名 只会在暂存区删除该文件：：仍需提交

.gitignore文件 可以用该文件忽略某些不想上传的文件

忽略日志文件和文件夹，忽略所有。Class文件，忽略所有。O文件，忽略所有.env文件，忽略所有.zip和tar文件，忽略所有.pem文件

\*\*\*系统或软件自动生成的文件

\*\*\*编译产生的中间文件和结果文件

\*\*\*运行时生成的日志文件、缓存文件、临时文件

\*\*\*设计身份、密码、口令、密钥等私密文件

生效前提：在将文件添加进.gitignore之前不能添加到版本库的文件，可以搭配git rm --cached 文件名 指令享用::::::::注：文件夹写法应在文件夹后加 “ / “

使用方法：

1. 创建一个.gitignore文件

------2、将需要隐藏的文件添加到.gitignore文件

echo other.log > .gitignore

------3、修改.gitignore文件：：：vi .gitignore 进入vim编辑器

加入一行 \*.log 则将全部.log文件添加进该文件

4、add 全部文件也不会显示添加进.gitignore文件的文件

忽略规则：

1、空行或以“ # “ 开头的行会被git忽略 ，#一般表示注释

2、使用标准Blob模式匹配：

星号 \* 匹配任意个字符

问号 ？匹配单个字符

中括号” [ ] “ 表示匹配列表中的单个字符，

比如：[ a b c ] 表示a / b / c

3、使用两个星号 \* \* 表示匹配任意的中间目录

4、中括号可以使用短中线“ - “ 连接，比如：

[ 0 – 9 ]表示任意一位字 [ a - z ] 表示任意一位小写字母

5、 “ ! “ 表示取反 表示要忽略指令模式以外的文件或者目录



Git有忽略常用语言gitignore文件模板 github.com/github/gitignore

需要注意的是git下载的.gitgnore文件需要删除前缀才能生效

Github创建仓库----仓库名称----仓库描述-----仓库可见性----是否初始化一README文件（后期可手动添加）----是否创建一个.gitignore文件-----开源许可证文件

https远程仓库方式把本地代码push到远程仓库时需要验证仓库名和密码

SSH协议则不需要 但需要在github上添加SSH公钥的配置

**步骤**：1、输入cd 进入根目录

1. 输入cd .ssh 进入.ssh目录
2. 使用ssh-key generate生成自己的ssh密钥

后面加上-t rsa表示指定协议为rsa -b 4096 表示生成大小为4096

完整为 ssh-keygen -t rsa -b 4096

1. 若第一次使用这个命令 需要输入的时候直接回车就行

若之前配置过了的话，不能直接回车，否则会覆盖掉之前生成的ssh密钥文件 且不可逆 需要输入一个新的文件名 例如tset 后回车

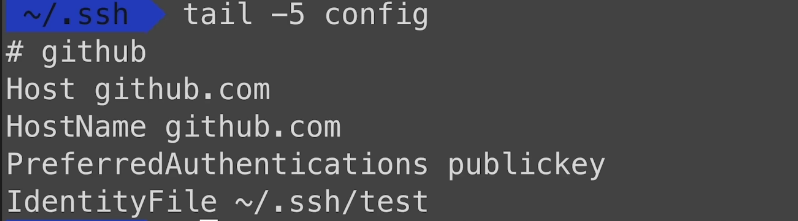
Ls -ltr 查看



第一条为私钥文件，不能发出来 第二条为公钥文

没有任何拓展名是私钥文件 以.pub结尾的是公钥文件

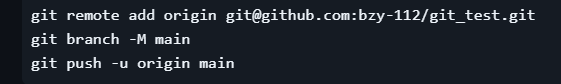
1. 打开公钥文件vi test.pub'
2. 复制内容 后输入：q退出
3. 添加到github上
4. 若不是第一次创建ssh密钥的话还需要创建一个config文件并添加五行内容在文件里 表示访问github.com时 指定使用.ssh文件下的test这个密钥(创建一个touch config / vi config)



若是第一次使用ssh协议需要确认一下 输入yes即可

注意：若使用git rm命令删除文件 仍需要git commit 命令上传到本地仓库 之后git push

将git仓库与本地仓库连接起来



命令：git remote add origin 仓库地址

git remote -v 可以查看当前仓库所对应远程仓库的别名和地址



本地仓库所对应远程仓库别名为origin 后面为所对应远程仓库地址

git branch -M main意思为指定分支的名称为main 因为默认分支名称就是main 所有可以省略

上传之前需要将该文件夹里的文件git commit 若还出现错误则需要将master换成main

git push -u origin main 把本地的main分支和远程origin仓库的main分支关联起来（全称应该为 git push -u origin main:main (-u是upstream的缩写)）

若在远程仓库修改了内容则需要使用pull命令将修改的内容拉取到本地

Git pull <远程仓库名称><远程分支名>:<本地分支名>

仓库名称和分支名称可以省略 为 git pull origin main

只写 git pull 会默认拉取仓库名为origin的main分支到本地进行合并

拉取git会自动执行一次合并操作 若远程仓库的修改内容和本地修改内容没有冲突的话就成功 若有冲突会报错

还可以使用git fetch命令 只会获得远程仓库的修改内容 并不会自动合并到本地仓库

VsCode文件后缀显示含义



git branch 查看当前仓库所有分支

命令行带有星号（ \* ）就是当前所处在的分支

--------在git早期默认分支称为master (带有种族歧视)现 为main

git branch 分支名 创建一个新的分支 （并没有切换到该分支上）

git checkout 分支名 切换到不同的分支

问题：git checkout 还可以用来恢复文件或者目录到之前某一个状态，若意外修改了某一个文件 则可以使用git checkout恢复之前问状态。若分支名称和文件名称相同，则该命令会默认切换分支，而不是恢复文件，可以使用专门切换分支的git switch命令

在分支创建的文件没合并时 ，切换回main分支则不会显示分支创建文件。

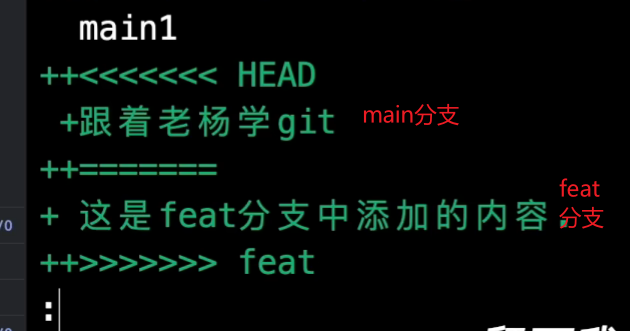
git merge 分支名 将分支合并到当前所在分支 但是分支还是存在的

git log --graph --oneline --decorate --all 查看分支图

git branch -d 分支名 删除分支（-d）表示删除已经合并的分支

git branch -D 分支名 -D表示强制删除

若两个分支修改了同一个文件的同一行代码

合并时会出现错误 使用git diff查看

使用vi 文件名 进入编辑文件 解决分歧



git rebase个分支移动到目标分支的最新提交记录后合并为一个分支

git checkout -b 分支名 删除分支的提交ID 可以恢复到这个分支的这一时间点的状态 **注意：该操作只会恢复该分支的状态 不会恢复被合并分支的状态**

查看ID:git log --oneline --graph --decorate --all

或者找到之前删除分支终端返回的提示

可以使用alias 命令将命令重命名

例如：alias graph=”git log --oneline --graph --decorate --all”

后输入grath 等同于git log --oneline --graph --decorate --all

优缺点：

git merge 优点不会破坏缘分支的提交历史，方便回溯和查看

缺点产生额外的提交节点，分支图比较复杂

Git rebase优点不会增加额外的提交记录，形成线性历史，比较直观干净

缺点会改变提交历史，改变了当前分支branch out节点避免在共享分支使用

Git 错误问题

1、如果出现错误：$ git clone https://github.com/vant-ui/vant-demo.git

Cloning into 'vant-demo'...

fatal: unable to access 'https://github.com/vant-ui/vant-demo.git/': SSL certificate problem:unable to get local issuer certificate

因为通过HTTPS访问Git远程仓库时候，本地没有配置https信任的服务器，导致验证不通过，

解决方法：需要将git中的sslverify关闭

git config --global http.sslverify false

2、Window平台下使用“git add .”出现“warning: LF will be replaced by CRLF”的提示:

解决方法：

1 保留仓库的LF行尾结束符，在git检出时自动转换CRLF到LF

2 git config --global core.autocrlf input (采用)

保留window系统的crlf 在git检出不自动转换

git config --global core.autocrlf true

3 若想避免这个警告，报纸原有的EOL

git config --global core.autocrlf false

3、若出现端口号22错误的话 需要设置端口号

[【git clone时git报错:port 22】解决方案 ssh:connect to host github.com port 22: Connection timed out\_git clone ssh-CSDN博客](https://blog.csdn.net/weixin_44955407/article/details/135647227?ops_request_misc=&request_id=&biz_id=102&utm_term=git%20clone%E9%94%99%E8%AF%AFssh:%20connect%20to%20hos&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-0-135647227.142%5ev100%5epc_search_result_base6&spm=1018.2226.3001.4187)