## 初始化仓库：

### 配置开发者信息：

git config --global user.name "your\_name"：配置全局的开发者名称。

git config --global user.email "your\_email@example.com"：配置全局的开发者邮箱。

git config –global credential.helper store：保存用户名和密码

git cofig –global –list：查看配置信息

## 初始化一个新的 Git 仓库

git init：在当前目录下初始化一个新的 Git 仓库。

### 查看文件变动状态：

**查看当前文件的状态，包括 哪些文件被修改、哪些文件已添加到暂存区等**

git status：/ˈsteɪtəs/ -s 查看缩略信息

**查看git的操作记录**

git reflog

**查看提交历史：**

git log：查看提交历史。

git log --oneline：查看简洁版本的提交历史。/ˌwʌnˈlaɪn/

git log --graph --oneline --all：图形化显示分支和提交历史。

git log --pretty=oneline

git log --pretty=oneline –graph：图于化显示各分支的提交

### .git所在目录

git文件的状态：未跟踪untrack,未修改unmodified，已修改modifide,已暂存staged

### 工作区：.git所在的目录

working directory

### 撤销某个修改

工作区删除某个文件但是已经存过暂存区输入git checkout 文件名,就可以丢弃工作区的改动，还原了被删除的文件

git checkout --文件名

### 添加进暂存区：git add

目录所在.git/index

staging area/index

**添加指定文件到暂存区。**

git add <file>

如：git add \*.txt

**添加当前目录下所有更改过的文件到暂存区。**

git add.

**如果工作区已删除文件的操作已进行暂存区想找回，先撤销暂存区的操作，再执行上面checkout的工作**

如：git reset head file 放弃暂存区的修改

git checkout –flie 放弃工作区的修改

**将已删除的文件也添加到暂存区，确保所有文件的变动都被追踪。**

git add -u：

**将当前文件夹里的所有文件增加到暂存区**

git add .

### 提交到版本库： git commit -m

目录所在.git/ob jects

local repository

**提交暂存区的修改到本地版本库，并添加提交说明。**

git commit -m "提交说明"：

git commit –am "提交说明" 这上命令直接把git add 这一步省略了

### 撤销已经提交的代码：

git revert <commit-hash>：撤销某个提交。

git revert <commit-hash1>^..<commit-hash2>：撤销一系列提交（commit-hash1和commit-hash2是要撤销提交的哈希值）。

### 回退版本：git reset

git reset --soft(表示回退到哪一个版本，保留工作区和暂存区的所有内容)

git reset --hard:(表示回退到哪一个版本，丢弃工作区和暂存区的所有内容)

git reset --mixed：默认参数(表示回退到哪一个版本，只保留工作区，丢弃暂存区内容)

如：git reset –s oft 4291559(4291559是回退的版本id)

hard~代表上一个版本，hard~10代表前10个版本

回到上一个版本： git reset --hard hard~

根据版本编号回到哪个版本：git reset --hard 4291559

### 比较文件差异git diff

git diff 比较工作区与暂存区的差异

git diff head 比较暂存区与本地仓库的差异

git diff --cached 暂存区与版本库之间的差异

git diff 版本号1 版本号2 两个版本之间的差异

git diff head 最先的版本，最近一次提交版本之间的差异

git diff head~ head, 最近两次版本之间的差异（还可以在波浪线后面加数字~2代表之前的两个版本）

git diff head~ head 4.txt 指定文件的差异

### 删除版文件

**工作区的删除 rm file1**

**暂存区的删除多个文件**

git rm file1 file2 file3

git commit -m "Remove multiple files"

**删除单个文件**

git commit -m "Remove filename"

**删除目录及其内容**

git rm -r directory\_name

git commit -m "Remove directory and its contents"

**仅从版本库删除文件，保留在工作目录**

git rm --cached filename

git commit -m "Remove filename from version control"

### 忽略文件 .gitgnore

echo \*.log > .gitignore

在.gitegnore文件里添加过滤文件：\*.lgo所有文件 temp/忽略文件夹

\*任意字符，？匹配单个字符，[]表示匹配列表中的任意字符，\*\*表示匹配任意的中间目录

如果在[0-9],[a-z]表示任意数字、任意字母。！表示取反

### 添加远程仓库

git remote add <远程仓库别名><远程仓库地址>

git push –u <远程仓库别名><分支名>

ssh方法：git remote add origin git@github.com:gentelhu/learn.git

http方法：git remote add origin <http://github.com/gentelhu/learn.git>

#### 添加新的远程仓库名字

git remote add gitee https://gitee.com/hjtguyue/learn.git

#### 本地ssh密钥生成

在.ssh目录下：ssh-keygen –t rsa –b 4096

#### 从远程仓库克隆版本库：

git clone <远程仓库地址>：从远程仓库复制一个完整的版本库到本地。

git clone --branch <分支名> <远程仓库地址>：克隆特定分支的代码。

### git branch查看本地仓库的分支列表

#### 创建并切换到 main 分支

git branch -m master main

#### 查看本地仓库和远程仓库所对应的名字

git remote –v查看本地仓库和远程仓库所对应的名字

#### 将本地的根目录和远程仓库的根目录关联起来

git push –u origin main:main

如果本地分支是master而不是main则修改：git push –u origin master:main

**也可以将本地的master分会重命名为main**

##### 本地分支重命名

git branch –m master main

git push –u origin main

**如果本地不存在main分支，可以创建一个新的main分支**

#创建一个新的main分支

git checkou t –b main

#将当前分支的内容合并到main分支

git merge<当前分支名称>

#将本地的main分支推送到远程仓库

git push –u origin main

### 本地与服务器之间文件推送

**从远程获取文件并合并**

git pull

**只获取远程文件的修改，但并不会自动合并到本地仓库中，需要自己手动合并**

git fethc

#### 将本地推送到服务器

git push origin <分支名>：

如：git push -u origin master:main

**强制本地覆盖远程**

git push origin --force：

如：git push -f -u origin master:main

如2：git push -f -u gitee main:main

**打标签，标记版本：**

git tag <标签名称>：创建一个标签。

git push origin <标签名称>：推送标签到远程仓库。

git push origin --tags：推送所有标签到远程仓库。

git tag -d <标签名称>：删除本地标签。

git push origin :refs/tags/<标签名称>：删除远程标签。

忽略文件：

创建一个.gitignore文件，在其中列出需要忽略的文件或目录。例如：

.idea/：忽略整个.idea目录。

\*.iml：忽略所有.iml文件。

file.txt：忽略指定的file.txt文件。

## 分支branch

可以独立存在，并且有自己的提交记录，可以看成代码库中的不同版本

### 查看当前分支并且看到在哪个分支下工作

git branch

### 分支图展示

git log --graph --oneline --decorate –all

### 创建分支git branch 分支名

创建一个分支dev并切换到其上进行工作

git checkout –b dev

### 切换分支git switch 分支名

### 合并分支：git merge <branch-name>

git merge <branch-name>：将指定分支的更改合并到当前分支（在执行合并操作之前，确保切换到目标分支。合并后，

#### **冲突**

**需要手动解决冲突文件合并后**，再主分支再重新提交一次文件，然后再合并分支。

#### 分支管理

如果某文件没有冲突但是在各个分支中各自有文件修改，在合并的时候git会自动再做一次提交，需要重新备注提交内容

当某个分支没有提交记录，只有一个分支有提交记录，就可以快速合并

#### 禁止快速合并

git merge –no-ff –m “禁用fast-forward” dev

### 删除分支git branch –d 分支名

git branch -d <branch-name>：删除本地分支（如果分支未合并到当前分支，Git 会阻止删除操作）。

git branch -D <branch-name>：强制删除本地分支。

删除远程分支：

git push origin --delete <branch-name>：删除远程分支。

## 其他常用命令：

git diff：查看文件的差异。

git stash：暂存未完成的更改。

git stash pop：恢复暂存的更改。

git clean：删除未跟踪的文件。

git remote：管理远程仓库。

git remote -v：查看远程仓库的地址。

git fetch：从远程仓库获取最新数据，但不合并。

git cherry-pick：应用指定提交的更改。

git rebase：将当前分支的提交重新应用到另一个分支的顶部。

这只是 Git 命令的一部分，Git 还有许多其他功能和命令，可以根据具体需求进一步学习和使用

## 其他GIT托管平台

gitee.com

gitlab.com

## git管理工具

github-desktop只能管理github上的仓库功能单一,只能简单的图形工具

sourcetree免费的，支持windows mac

https://sourcetreeapp.com/

gitkraken分为免费的和收费的，免费的限制较多

## 分支命名规范

版本发布分支/tag示例：v1.0.0

功能分支示例：feature-login-page

修复分支示例：hotfix-#issueid-desc

分支管理

定期合并已经成功验证的分支，及时删除已经合并的分支保持合适的分支数量，

为分支设置合适的管理权限

## widndow cmd

type 在cmd查看文件

echo 重复单行

（）多行重复

( 这里回车

More? echo 2 line shift+enter

More? echo 3 line

More? echo 4 line) >> 4.txt