

[toc]

1:Git使用

1.1 Git使用前的配置

在使用Git前，告诉Git你是谁，提交项目时会用到

1: 配置提交人姓名: `git config --global user.name`

2: 配置提交人邮箱: `git config --global user.email`

3:查看配置信息: `git config --list`

```
浪子@DESKTOP-M17CR5F MINGW64 /d/gitexercise/wswdemo
$ git config --global user.name 浪子

浪子@DESKTOP-M17CR5F MINGW64 /d/gitexercise/wswdemo
$ git config --global user.email 1278833457@qq.com

浪子@DESKTOP-M17CR5F MINGW64 /d/gitexercise/wswdemo
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=G:/GIT/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
credential.helper=manager
user.name=浪子
user.email=1278833457@qq.com
```

1.2:提交步骤

1: `git init` 初始化git仓库

2: `git status` 查看文件状态

3: `git add` 文件列表纸 追踪文件状态

4: `git commit -m` 提交信息 向仓库中提交代码

5: `git log` 查看提交记录

```
浪子@DESKTOP-M17CR5F MINGW64 /d/gitexercise/wswdemo
$ git init
Initialized empty Git repository in D:/gitexercise/wswdemo/.git/

浪子@DESKTOP-M17CR5F MINGW64 /d/gitexercise/wswdemo (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html

nothing added to commit but untracked files present (use "git add" to track)

浪子@DESKTOP-M17CR5F MINGW64 /d/gitexercise/wswdemo (master)
$ git add index.html

浪子@DESKTOP-M17CR5F MINGW64 /d/gitexercise/wswdemo (master)
$ git commit -m 第一次提交
[master (root-commit) 3b3372b] 第一次提交
 1 file changed, 1 insertion(+)
 create mode 100644 index.html

浪子@DESKTOP-M17CR5F MINGW64 /d/gitexercise/wswdemo (master)
$ git log
commit 3b3372b4d43bec1605411bbf90086913ac7daf2e (HEAD -> master)
Author: 浪子 <1278833457@qq.com>
Date:   Mon Mar 9 20:39:13 2020 +0800

    第一次提交

浪子@DESKTOP-M17CR5F MINGW64 /d/gitexercise/wswdemo (master)
```

1.3 撤销操作

*用暂存区的文件覆盖工作目录的文件（例如文件修改错误恢复到暂存区的状态（此时还没更新此文件））：git checkout 文件名

*将文件从暂存区中删除（例如一些测试文件，不小心加入了暂存区）：git rm --cached 文件名

*将git仓库中指定的提交记录恢复出来，并且覆盖暂存区和工作目录最终结果就是恢复提交记录的状态（包括提交记录和暂存区以及工作目录，在此提交记录之后的文件全部删除掉了）：git rest --hard commit ID

1.4 分支操作

分支与主分支之间没有任何联系，主要作用就是对项目的各种操作并行开发（例如；查找修改bug，新功能的开发两者之间同时进行）

1.4.1 分支命令

*git branch 查看分支

*git branch 分支名称 -->创建分支

*git checkout 分支名称 -->切换分支（切换分支前，一定要将创建的分支进行提交，否则会出现问题（主分支与新建的分支之间会有联系，而分支是不能与主分支之间存在联系的）

*git merge 来源分支 -->合并分支（当前开发分支已经完成 将其合并到主分支，要在主分支上操作，合并后 开发分支依然存在）

*git branch -d 分支名称 -->删除分支（分支被合并后才允许删除）（-D强制删除）

1.4.2 暂时保存分支

当开发分支没有开发完成，有必须改动以前的分支内容（比如已经开发好的分支遇到bug），当前未开发好的开发分支不想提交，就可以暂时提取改分支上的所有改动并储存（不提交此分支）

*储存临时改动：git stash（由于git提供的储存临时改动是独立的，也就是说，你在其他开发分支使用此命令的时候，就会把改动添加到其他分支，所以在使用前，需要切换到你保存为提交的分支）

*恢复改动：git stash pop

2:Github

远程仓库，实现多人协作开发，本地仓库知识本人使用，远程仓库别人可以下载你写的代码，并进行操作。

2.1 注册

网址：<https://github.com/>

2.2 多人协作开发流程

*A在自己的计算机中创建本地仓库

*A在github中创建远程仓库

*A将本地仓库推送到远程仓库

*B克隆远程仓库到本地进行开发

*B将本地仓库开发的内容推送到远程仓库

*A将远程仓库中的最新版本拉到本地仓库

2.3 本地仓库提交到远程仓库

*初始化本地仓库

1: git init

2: git add .(添加到暂存区 .可以选择所有)


3:git commit -m 添加到本地仓库

*创建远程仓库

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner

 BCLZ-WSW ▾

 /

Repository name *

Description (optional)

☒  **Public**
Anyone can see this repository. You choose who can commit.☐  **Private**
You choose who can see and commit to this repository.☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer.Add .gitignore: **None ▾**Add a license: **None ▾** ⓘ

Create repository

*将本地仓库中的文件传到远程仓库中

1: git push 远程仓库地址 分支名称

ex: git push <https://github.com/>... master

2: git remote add 远程仓库地址的别名 远程仓库地址

3: git push 远程仓库别名 分支名称 (只要作用就是远程仓库地址太难记, 用一个容易记住的名字代替)

4: git push -u 远程仓库地址别名 分支名称 (主要作用就是简化指令, 在下次操作的时候直接git push 即可)