

git 使用方法

2022 年 6 月 18 日

17:59

准备

- 1、安装 git: [Git \(git-scm.com\)](https://git-scm.com)
- 2、设置一下用户名和邮箱:

```
git config --global user.name "xxx"  
git config --global user.email xxx@xx.com
```

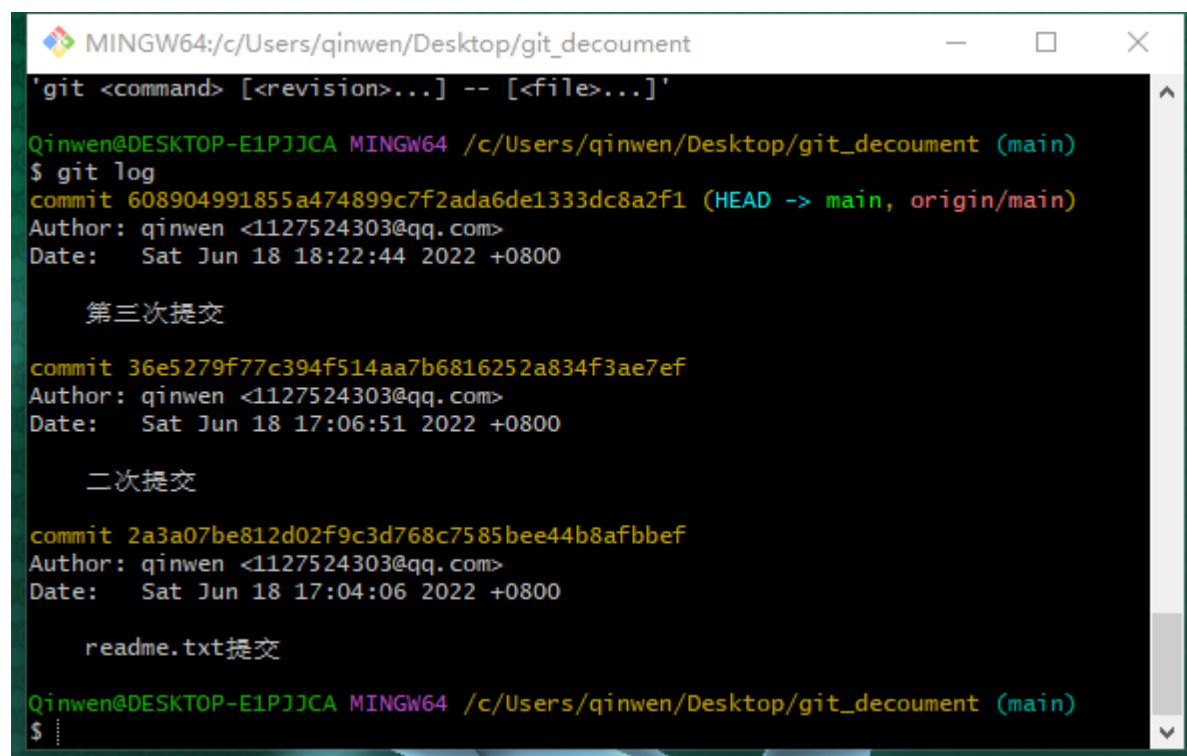
使用:

一、提交到本地仓库

- 1.创建仓库:
 - (1)使用 Linux 基础命令创建一个文件
 - (2) git init
- 2.提交
 - (3) git add 文件\目录 //将文件添加到暂存区
 - (4) git commit -m "备注" //把文件提交到分支
- 3.查看结果
 - (5) git status //来查看下结果
 - (6) git diff //查看到底修改了什么

二、版本退回

- 1.git log 查看提交的历史纪录



```
MINGW64:/c:/Users/qinwen/Desktop/git_decoument
'git <command> [<revision>...] -- [<file>...]'

Qinwen@DESKTOP-E1PJJCA MINGW64 /c:/Users/qinwen/Desktop/git_decoument (main)
$ git log
commit 608904991855a474899c7f2ada6de1333dc8a2f1 (HEAD -> main, origin/main)
Author: qinwen <1127524303@qq.com>
Date: Sat Jun 18 18:22:44 2022 +0800

    第三次提交

commit 36e5279f77c394f514aa7b6816252a834f3ae7ef
Author: qinwen <1127524303@qq.com>
Date: Sat Jun 18 17:06:51 2022 +0800

    二次提交

commit 2a3a07be812d02f9c3d768c7585bee44b8afbbef
Author: qinwen <1127524303@qq.com>
Date: Sat Jun 18 17:04:06 2022 +0800

    readme.txt提交

Qinwen@DESKTOP-E1PJJCA MINGW64 /c:/Users/qinwen/Desktop/git_decoument (main)
$
```

(commit :每次提交的版本号)

2.退回上个版本

`git reset --hard HEAD^` //退回上一个版本

3.退回上上个版本 (以此类推)

`git reset --hard HEAD^^`

4.退回上 N 个版本

`git reset --hard HEAD~100`

5.按指定版本号退回

(1) 查看版本: `git reflog`

(2)退回: `git reset --hard [版本号]`

三、撤销或者修改文件

一、撤销和修改

1.git checkout --文件\目录 //撤销对文件的所有修改(还没有添加到缓存区)

(如果文件已经提交暂存区, 而且文件还做了修改, 那么使用此命令只会还原到上传暂存区的版本)

二、删除文件

1.在目录中 rm 命令即可

四、连接远程仓库

1、推送

(1)创建 ssh KEY 运行:

`ssh-keygen -t rsa -C "xxx@xxx.com"`

id_rsa 是私钥 id_rsa.pub 是公钥

(2) 登录 github 添加 SSH KEY

(3) 关联仓库

`git remote add origin " (ssh 版) 仓库地址"`

(4) 将本地仓库推送至远程

`git push -u origin master` // (第一次提交) 把本地仓库分支 master 内容推送到远程仓库

注:

{Git 不但会把本地的 master 分支内容推送的远程新的 master 分支，还会把本地的 master 分支和远程的 master 分支关联起来，在以后的推送或者拉取时就可以简化命令}

git push origin master //之后版本提交只需此命令

2、克隆远程仓库

git clone "远程仓库地址" //别人的代码真香

五、创建与合并分支

1、创建分支并切换：

git checkout -b [分支名]

相当于下面两条命令：

git branch [分支名] //创建分支

git checkout [分支名] //切换分支

2.查看分支

git branch //会列出当前仓库所有分支

3.将分支合并

(1) 将指定分支合并到当前分支中

git merge [指定分支名]

4.删除分支

git branch -d [分支名]

命令词典

一、新建代码库

在当前目录新建一个 Git 代码库

\$ git init

新建一个目录，将其初始化为 Git 代码库

\$ git init [project-name] # 下载一个项目和它的整个代码历史

\$ git clone [url]

二、配置

显示当前的 Git 配置

\$ git config --list

编辑 Git 配置文件

\$ git config -e [--global] # 设置提交代码时的用户信息

```
$ git config [--global] user.name "[name]"  
$ git config [--global] user.email "[email address]"
```

三、增加/删除文件

添加指定文件到暂存区

```
$ git add [file1] [file2] ... # 添加指定目录到暂存区，包括子目录
```

```
$ git add [dir] # 添加当前目录的所有文件到暂存区
```

\$ git add . # 添加每个变化前，都会要求确认 # 对于同一个文件的多处变化，可以实现分次提交

```
$ git add -p
```

删除工作区文件，并且将这次删除放入暂存区

```
$ git rm [file1] [file2] ... # 停止追踪指定文件，但该文件会保留在工作区
```

```
$ git rm --cached [file] # 改名文件，并且将这个改名放入暂存区
```

```
$ git mv [file-original] [file-renamed]
```

四、代码提交

提交暂存区到仓库区

```
$ git commit -m [message] # 提交暂存区的指定文件到仓库区
```

```
$ git commit [file1] [file2] ... -m [message] # 提交工作区自上次 commit 之后的变化，直接到仓库区
```

```
$ git commit -a
```

提交时显示所有 diff 信息

```
$ git commit -v
```

使用一次新的 commit，替代上一次提交 # 如果代码没有任何新变化，则用来改写上一次 commit 的提交信息

```
$ git commit --amend -m [message] # 重做上一次 commit，并包括指定文件的新变化
```

```
$ git commit --amend [file1] [file2] ...
```

五、分支

列出所有本地分支

```
$ git branch
```

列出所有远程分支

```
$ git branch -r
```

列出所有本地分支和远程分支

```
$ git branch -a
```

新建一个分支，但依然停留在当前分支

```
$ git branch [branch-name] # 新建一个分支，并切换到该分支
```

```
$ git checkout -b [branch] # 新建一个分支，指向指定 commit
```

```
$ git branch [branch] [commit] # 新建一个分支，与指定的远程分支建立追踪关系
```

```
$ git branch --track [branch] [remote-branch] # 切换到指定分支，并更新工作区
```

```
$ git checkout [branch-name] # 切换到上一个分支
```

```
$ git checkout - # 建立追踪关系，在现有分支与指定的远程分支之间
```

```
$ git branch --set-upstream [branch] [remote-branch] # 合并指定分支到当前分支
```

```
$ git merge [branch] # 选择一个 commit，合并进当前分支
```

```
$ git cherry-pick [commit] # 删除分支
```

\$ git branch -d [branch-name] # 删除远程分支

\$ git push origin --delete [branch-name]

\$ git branch -dr [remote/branch]

六、标签

列出所有 tag

\$ git tag

新建一个 tag 在当前 commit

\$ git tag [tag] # 新建一个 tag 在指定 commit

\$ git tag [tag] [commit] # 删除本地 tag

\$ git tag -d [tag] # 删除远程 tag

\$ git push origin :refs/tags/[tagName] # 查看 tag 信息

\$ git show [tag] # 提交指定 tag

\$ git push [remote] [tag] # 提交所有 tag

\$ git push [remote] --tags

新建一个分支，指向某个 tag

\$ git checkout -b [branch] [tag]

七、查看信息

显示有变更的文件

\$ git status

显示当前分支的版本历史

\$ git log

显示 commit 历史，以及每次 commit 发生变更的文件

\$ git log --stat

搜索提交历史，根据关键词

\$ git log -S [keyword] # 显示某个 commit 之后的所有变动，每个 commit 占据一行

\$ git log [tag] HEAD --pretty=format:%s

显示某个 commit 之后的所有变动，其"提交说明"必须符合搜索条件

\$ git log [tag] HEAD --grep feature

显示某个文件的版本历史，包括文件改名

\$ git log --follow [file]

\$ git whatchanged [file] # 显示指定文件相关的每一次 diff

\$ git log -p [file] # 显示过去 5 次提交

\$ git log -5 --pretty --oneline

显示所有提交过的用户，按提交次数排序

\$ git shortlog -sn

显示指定文件是什么人在什么时间修改过

\$ git blame [file] # 显示暂存区和工作区的差异

\$ git diff

显示暂存区和上一个 commit 的差异
\$ git diff --cached [file] # 显示工作区与当前分支最新 commit 之间的差异
\$ git diff HEAD

显示两次提交之间的差异
\$ git diff [first-branch]...[second-branch] # 显示今天你写了多少行代码
\$ git diff --shortstat "@{0 day ago}" # 显示某次提交的元数据和内容变化
\$ git show [commit] # 显示某次提交发生变化的文件
\$ git show --name-only [commit] # 显示某次提交时，某个文件的内容
\$ git show [commit]:[filename] # 显示当前分支的最近几次提交
\$ git reflog

八、远程同步

下载远程仓库的所有变动
\$ git fetch [remote] # 显示所有远程仓库
\$ git remote -v

显示某个远程仓库的信息
\$ git remote show [remote] # 增加一个新的远程仓库，并命名
\$ git remote add [shortname] [url] # 取回远程仓库的变化，并与本地分支合并
\$ git pull [remote] [branch] # 上传本地指定分支到远程仓库
\$ git push [remote] [branch] # 强行推送当前分支到远程仓库，即使有冲突
\$ git push [remote] --force

推送所有分支到远程仓库

\$ git push [remote] --all

九、撤销

恢复暂存区的指定文件到工作区

\$ git checkout [file] # 恢复某个 commit 的指定文件到暂存区和工作区

\$ git checkout [commit] [file] # 恢复暂存区的所有文件到工作区

\$ git checkout . # 重置暂存区的指定文件，与上一次 commit 保持一致，但工作区不变

\$ git reset [file] # 重置暂存区与工作区，与上一次 commit 保持一致

\$ git reset --hard

重置当前分支的指针为指定 commit，同时重置暂存区，但工作区不变

\$ git reset [commit] # 重置当前分支的 HEAD 为指定 commit，同时重置暂存区和工作区，与指定 commit 一致

\$ git reset --hard [commit] # 重置当前 HEAD 为指定 commit，但保持暂存区和工作区不变

\$ git reset --keep [commit] # 新建一个 commit，用来撤销指定 commit # 后者的所有变化都将被前者抵消，并且应用到当前分支

\$ git revert [commit] # 暂时将未提交的变化移除，稍后再移入

\$ git stash

\$ git stash pop