Git 命令合集

## 常用命令

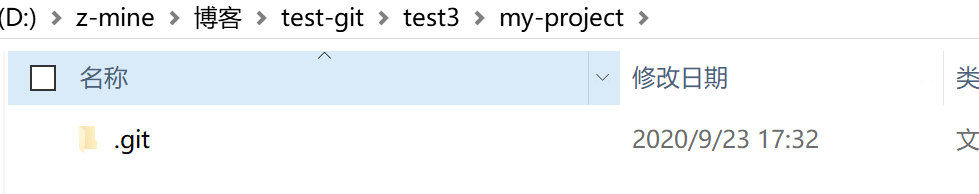
|  |
| --- |
| 1）克隆代码 [git clone [url]](#_git_clone_[url])  2）阿萨德  新安装的git与github创建ssh关联:  由于本地Git仓库和Github仓库之间的传输是通过SSH加密的，所以连接时需要设置一下：  　　 $ ssh-keygen -t rsa -C "youremail@example.com"  然后一路回车。这时你就会在用户下的.ssh目录里找到id\_rsa和id\_rsa.pub这两个文件     新建一个目录[project-name]，并在该目录下初始化Git代码仓库。  示列：  1）新建test3。    2）输入git init my-project  3）此时test3文件夹下就出现了my-project文件夹，my-project下有 .git 。    第四步：登录Github,找到右上角的图标，打开点进里面的Settings，再选中里面的SSH and GPG KEYS，点击右上角的New SSH key，然后Title里面随便填，再把刚才id\_rsa.pub里面的内容复制到Title下面的Key内容框里面，最后点击Add SSH key，这样就完成了SSH Key的加密。具体步骤也可看下面：        上面key默认在C:\Users\Administrator\.ssh下的id\_rsa.pub文件中，把这个文件中的内容全部复制到上面的key里面就可以了。如下图： |

## 1.新建代码库

### git init

|  |
| --- |
| 在当前目录新建一个Git代码仓库。  示列：  1）创建test文件夹。    2）右键Git Bash Here，输入git init。     1. 此时test文件夹就出现一个隐藏文件夹 .git ，说明我们已经创建了一个本地代码仓库。 |

### git init [project-name]

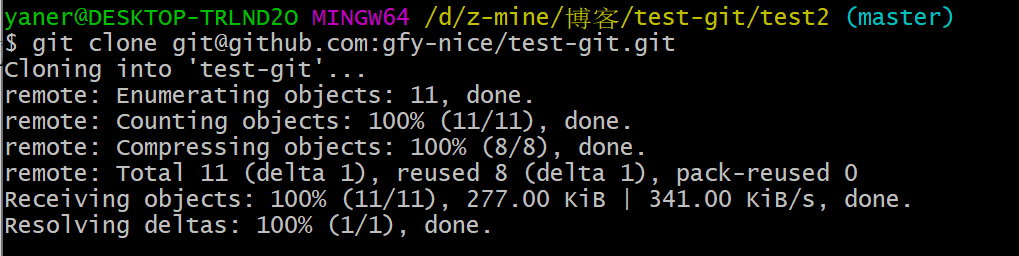


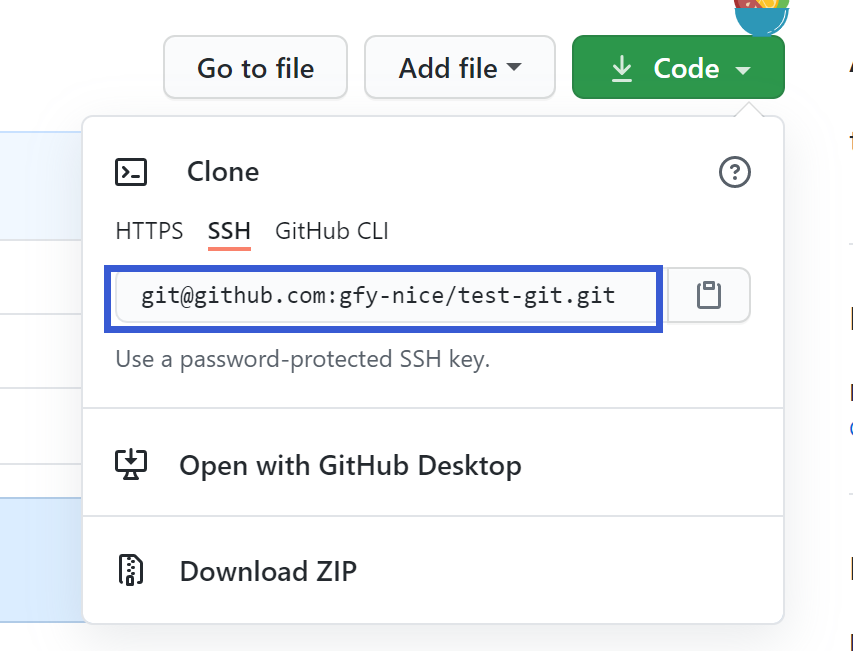
### git clone [url]

克隆git url上的项目，和它的整个历史版本。

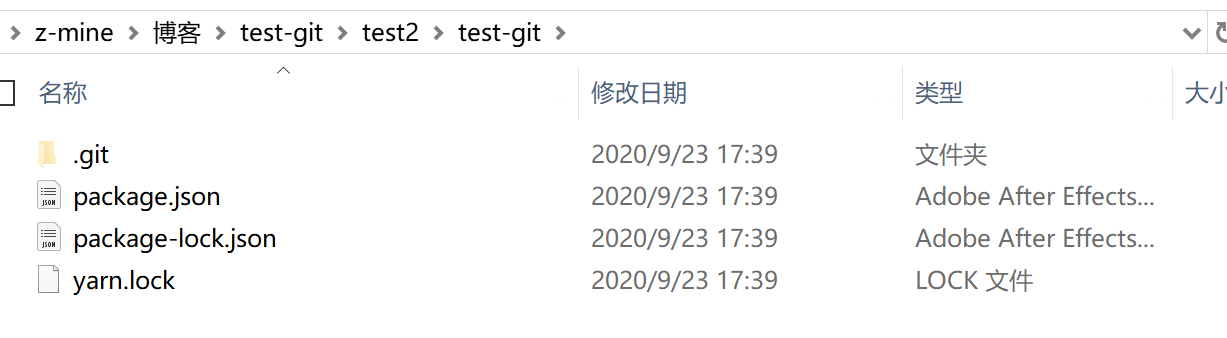
示列：

1）输入git clone [git@github.com:gfy-nice/test-git.git](mailto:git@github.com:gfy-nice/test-git.git) ,url是GitHub的地址。

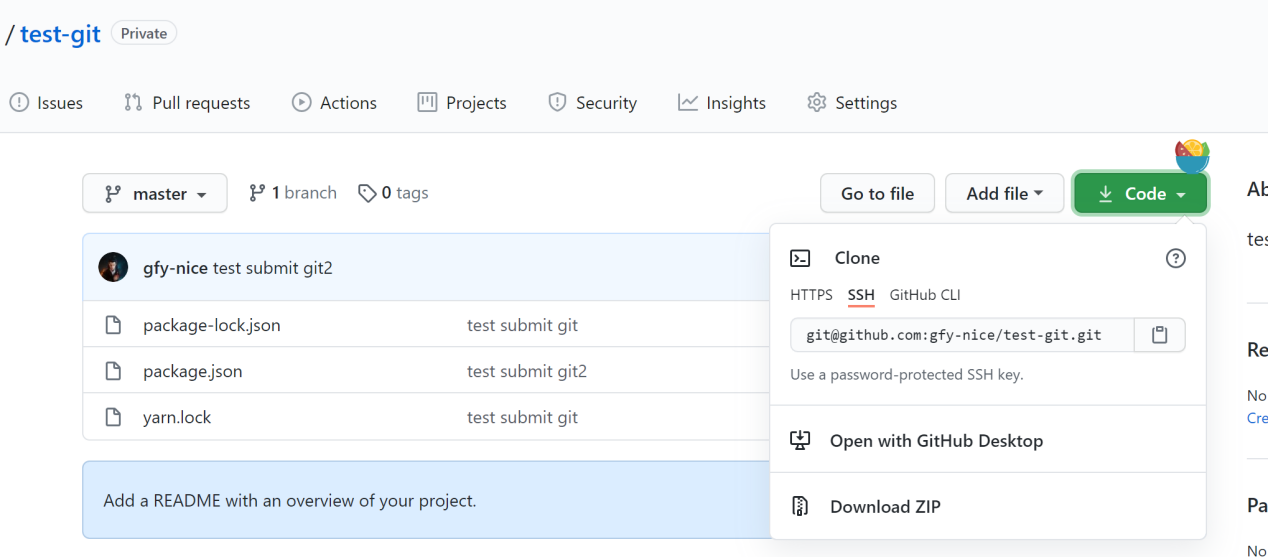




1. 下载后如下。



原来github上的文件如下：



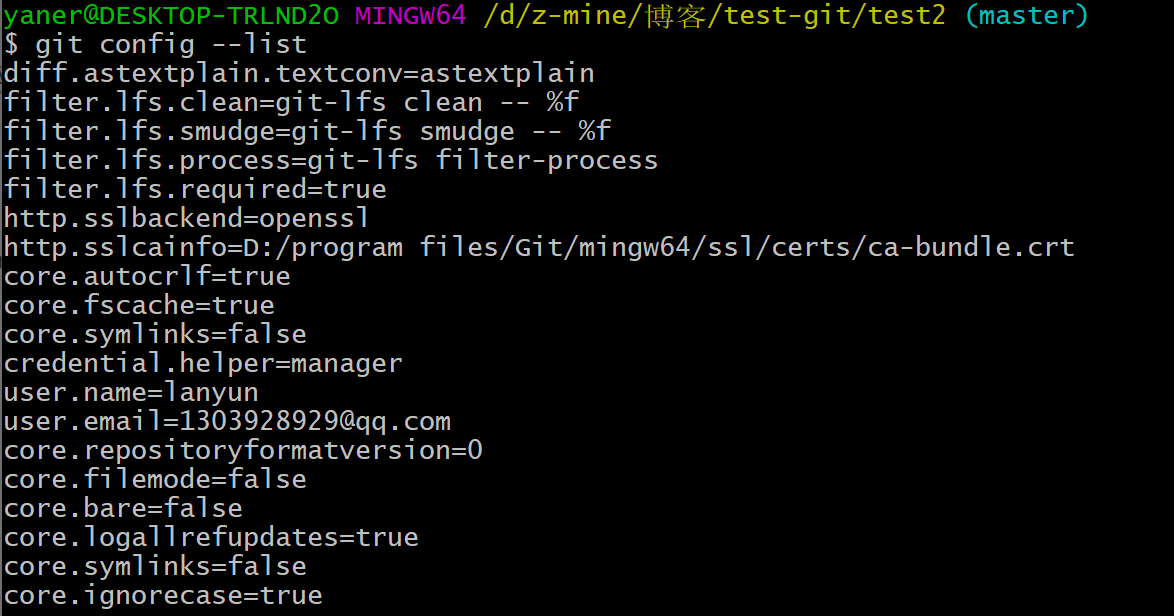
## 2.配置信息

### git config --list

显示当前的git配置信息，也就是传说中的.gitconfig文件，一般这个文件会在项目中（项目配置）或者用户主目录下（C:\Users\yaner）（全局配置）。

示列：

1）输入git config --list



1. 如果是一个初始化项目时，如何**生成.gitignore文件，可输入命令 vim .gitignore**

可以**输入node\_modules/ ，意思就是Git提交代码时会忽略掉这个文件。**

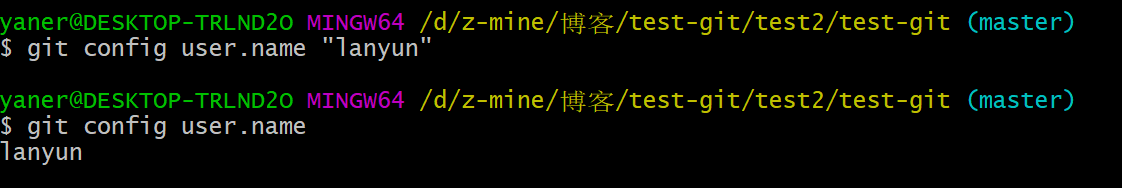
1. 在编辑时，**退出编辑，可长按Esc，在按两次大写的ZZ。**

### git config -e [--global]

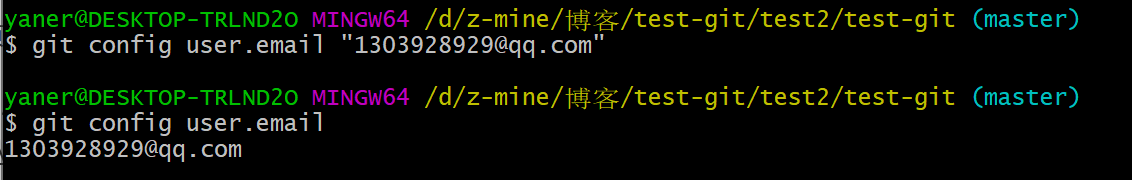
编辑Git配置文件

### git config [--global] user.name “[name]”

设置提交代码时的用户信息



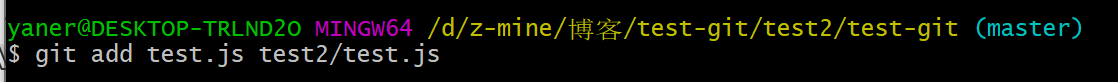
### git config [--global] user.email “[email]”



## 3.增加文件

### git add [file1] [file2] ...

添加指定文件到暂存区



### git add [dir]

添加指定目录到暂存区，包括子目录

### git add .

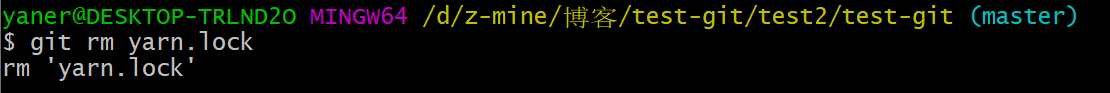
添加当前目录的所有文件到暂存区

git add -p对于同一个文件的多处变化，可以实现分次提交

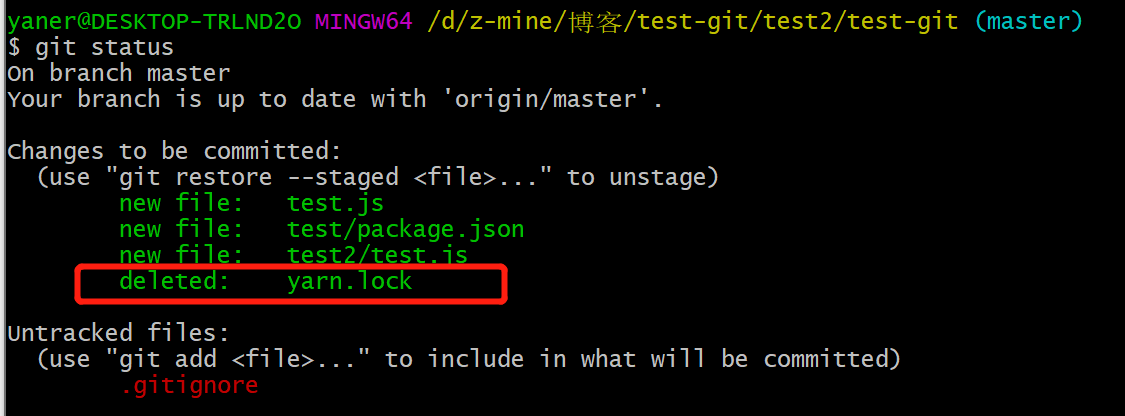
## 4.删除文件

### git rm [file1] [file2] ...

删除工作区文件，并且将这次删除放入暂存区



删除后查看有变更的文件

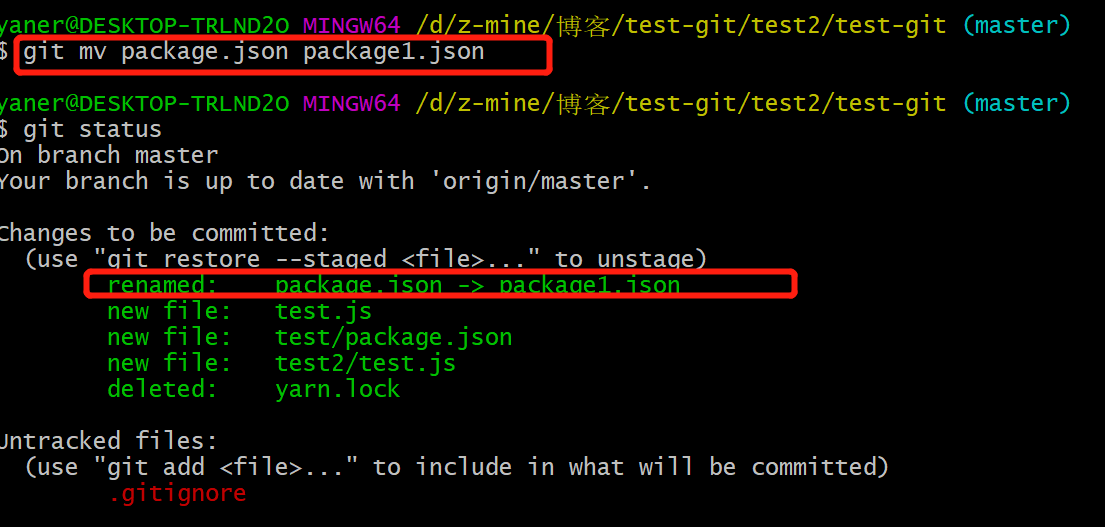


### git rm --cached [file]

停止追踪指定文件，但该文件会保留在工作区

### git mv [file-original] [file-renamed]

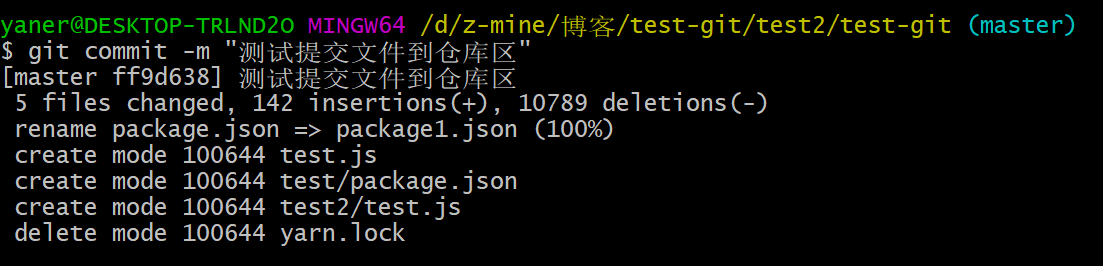
改名文件，并且将这个改名放入暂存区



## 5.提交文件

### git commit -m [message]

提交暂存区到仓库区



### git commit [file1] [file2] ... -m [message]

提交暂存区的指定文件到仓库区

### git commit -a

提交工作区自上次commit之后的变化，直接到仓库区

### git commit -v

提交时显示所有diff信息

### git commit --amend -m [message]

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

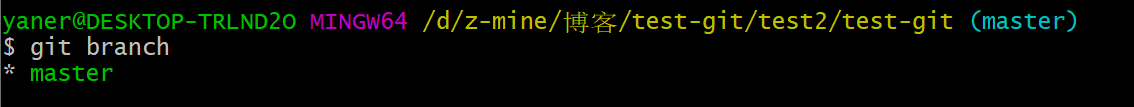
### git commit --amend [file1] [file2] ...

重做上一次commit，并包括指定文件的新变化

## 6.分支

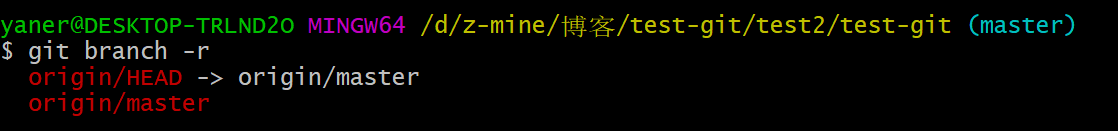
### git branch

列出所有本地分支



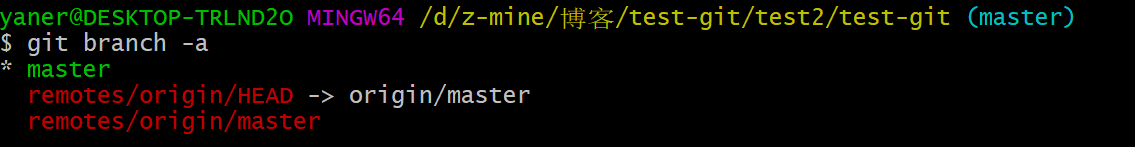
### git branch -r

列出所有远程分支



### git branch -a

列出所有本地分支和远程分支



git branch [branch-name]

新建一个分支，但依然停留在当前分支

### git checkout -b [branch]

新建一个分支，并切换到该分支

### git branch [branch] [commit]

新建一个分支，指向指定commit

### git branch --track [branch] [remote-branch]

新建一个分支，与指定的远程分支建立追踪关系

### git checkout [branch-name]

切换到指定分支，并更新工作区

### git checkout -

切换到上一个分支

### git branch --set-upstream [branch] [remote-branch]

建立追踪关系，在现有分支与指定的远程分支之间

### git merge [branch]

合并指定分支到当前分支

### git cherry-pick [commit]

选择一个commit，合并进当前分支

### git branch -d [branch-name]

删除分支

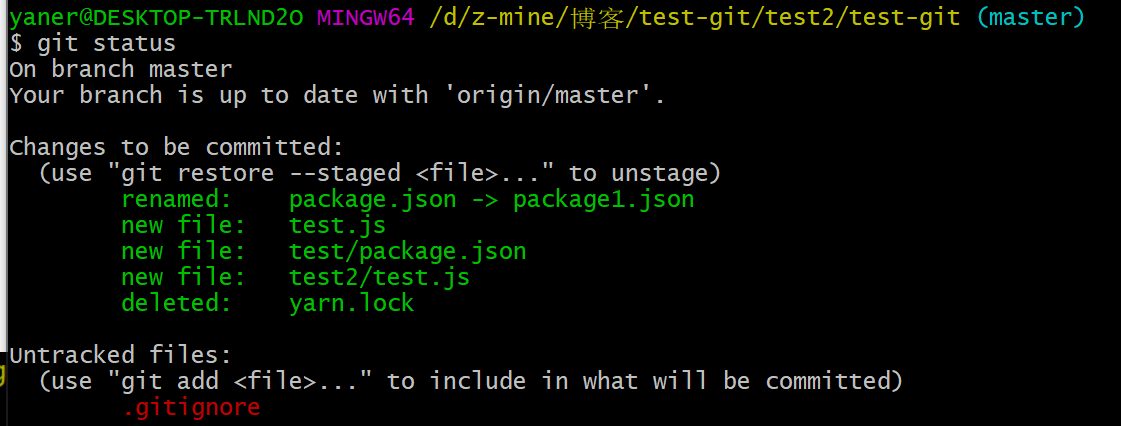
### git push origin --delete [branch-name]

删除远程分支

### git branch -dr [remote/branch]

## 7.查询

### git status

显示有变更的文件

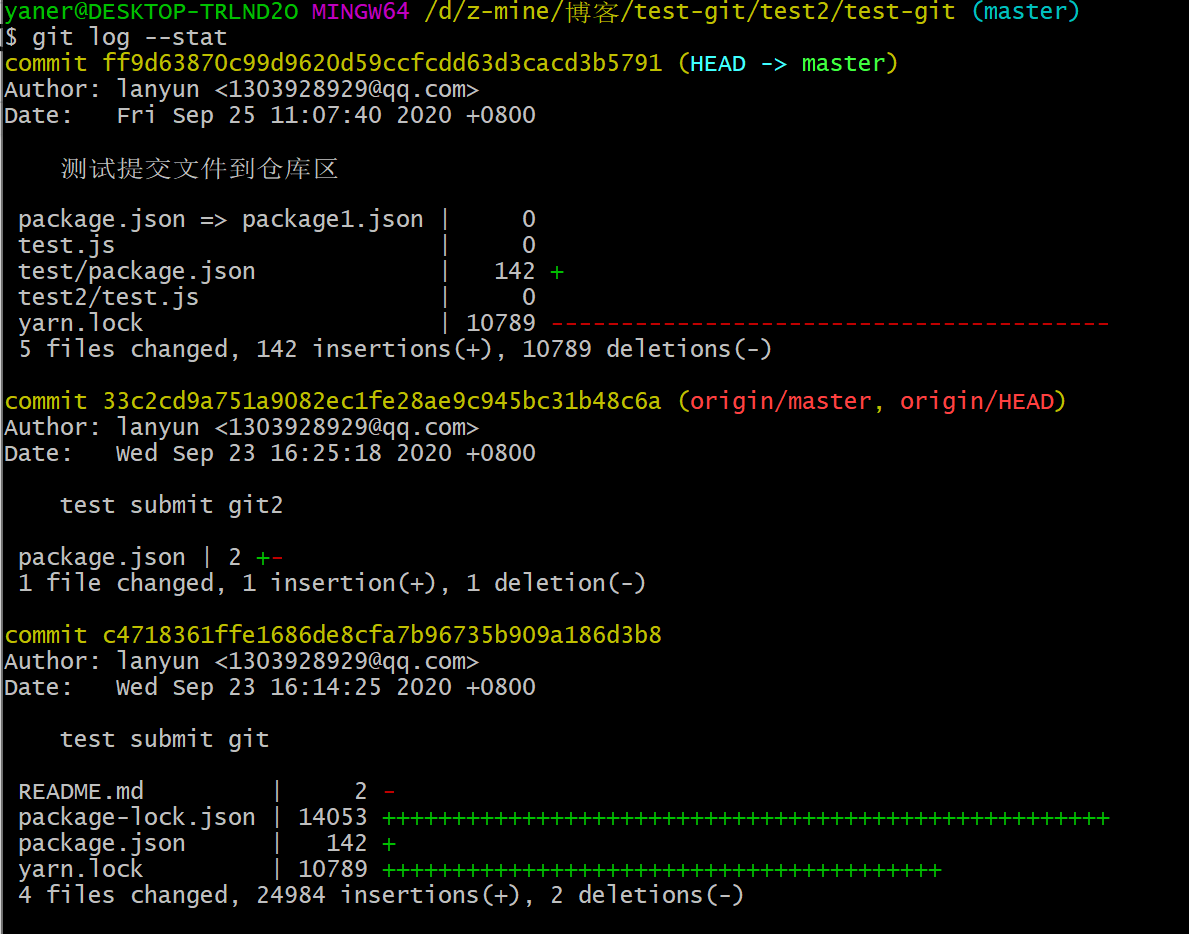
### git log

显示当前分支的版本历史



### git log --stat

显示commit历史，以及每次commit发生变更的文件



### git log -S [keyword]

搜索提交历史，根据关键词

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

显示某个commit之后的所有变动，每个commit占据一行

### git log [tag] HEAD --grep feature

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 --follow [file]

显示某个文件的版本历史，包括文件改名

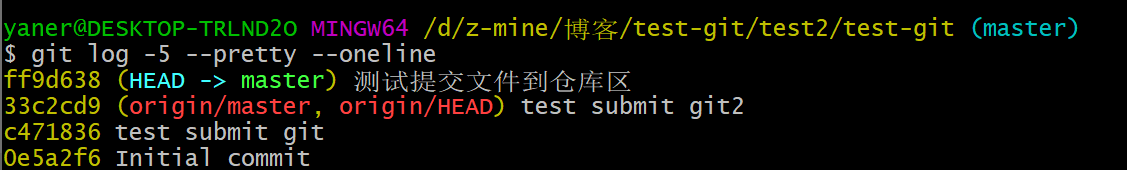
### git whatchanged [file]

### git log -p [file]

显示指定文件相关的每一次diff

### git log -5 --pretty --oneline

显示过去5次提交



### git shortlog -sn

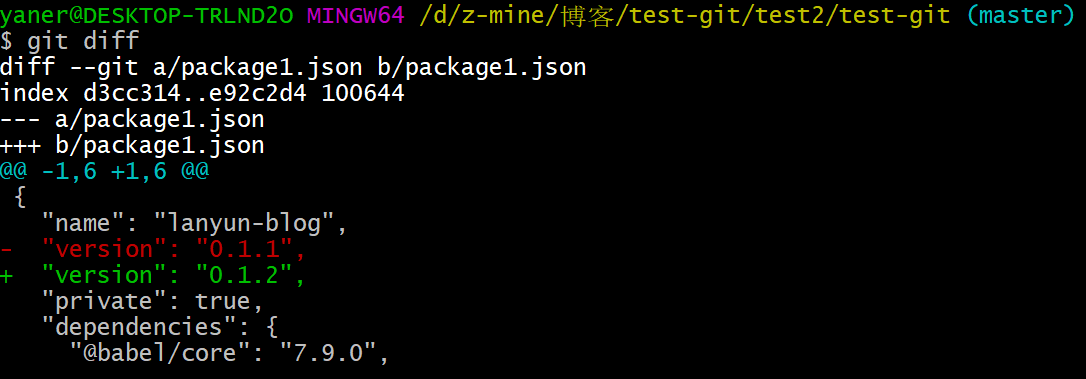
显示所有提交过的用户，按提交次数排序

### git blame [file]

显示指定文件是什么人在什么时间修改过

### git diff

显示暂存区和工作区的差异



### git diff --cached [file]

显示暂存区和上一个commit的差异

### git diff HEAD

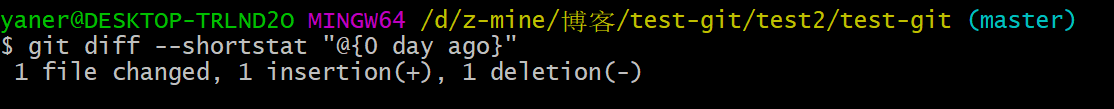
显示工作区与当前分支最新commit之间的差异

### git diff [first-branch]...[second-branch]

显示两次提交之间的差异

### git diff --shortstat "@{0 day ago}"

显示今天你写了多少行代码



### git show [commit]

显示某次提交的元数据和内容变化

### git show --name-only [commit]

显示某次提交发生变化的文件

### git show [commit]:[filename]

显示某次提交时，某个文件的内容

### git reflog

显示当前分支的最近几次提交

## 8.远程仓库

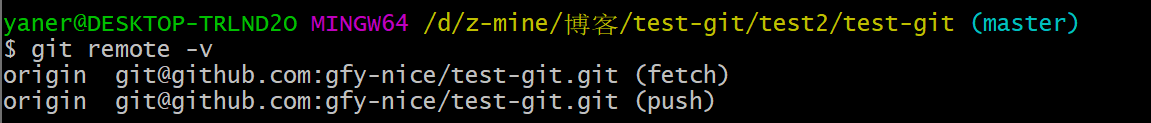
### git fetch [remote]

下载远程仓库的所有变动， [remote]是仓库地址。



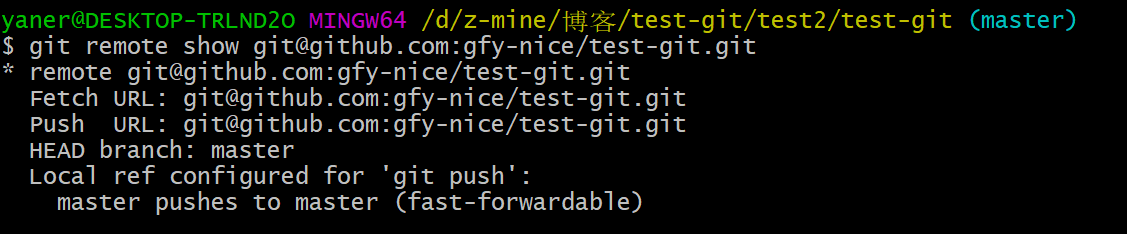
### git remote -v

显示所有远程仓库



### git remote show [remote]

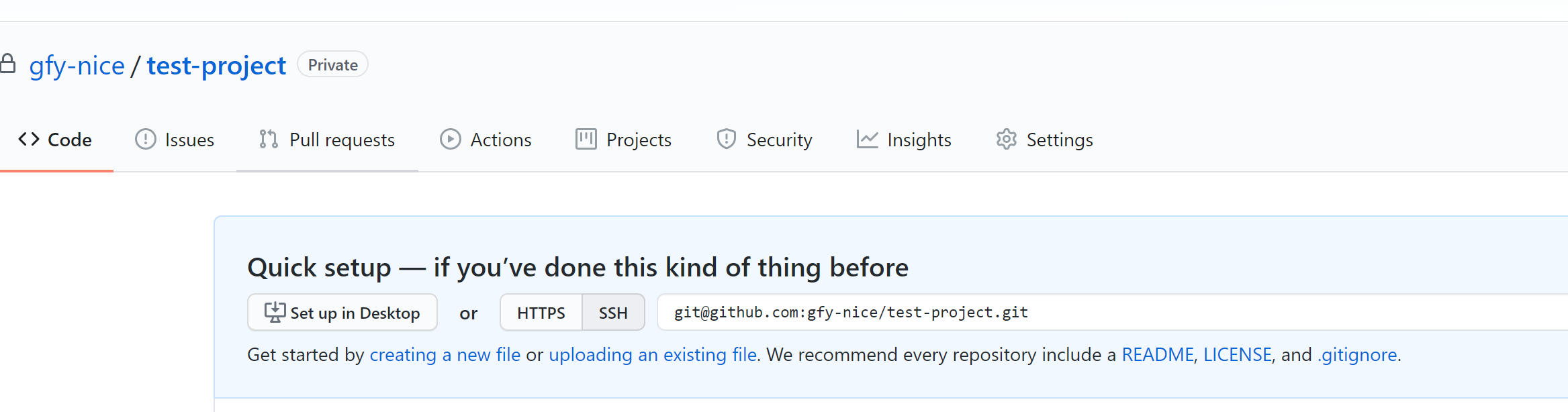
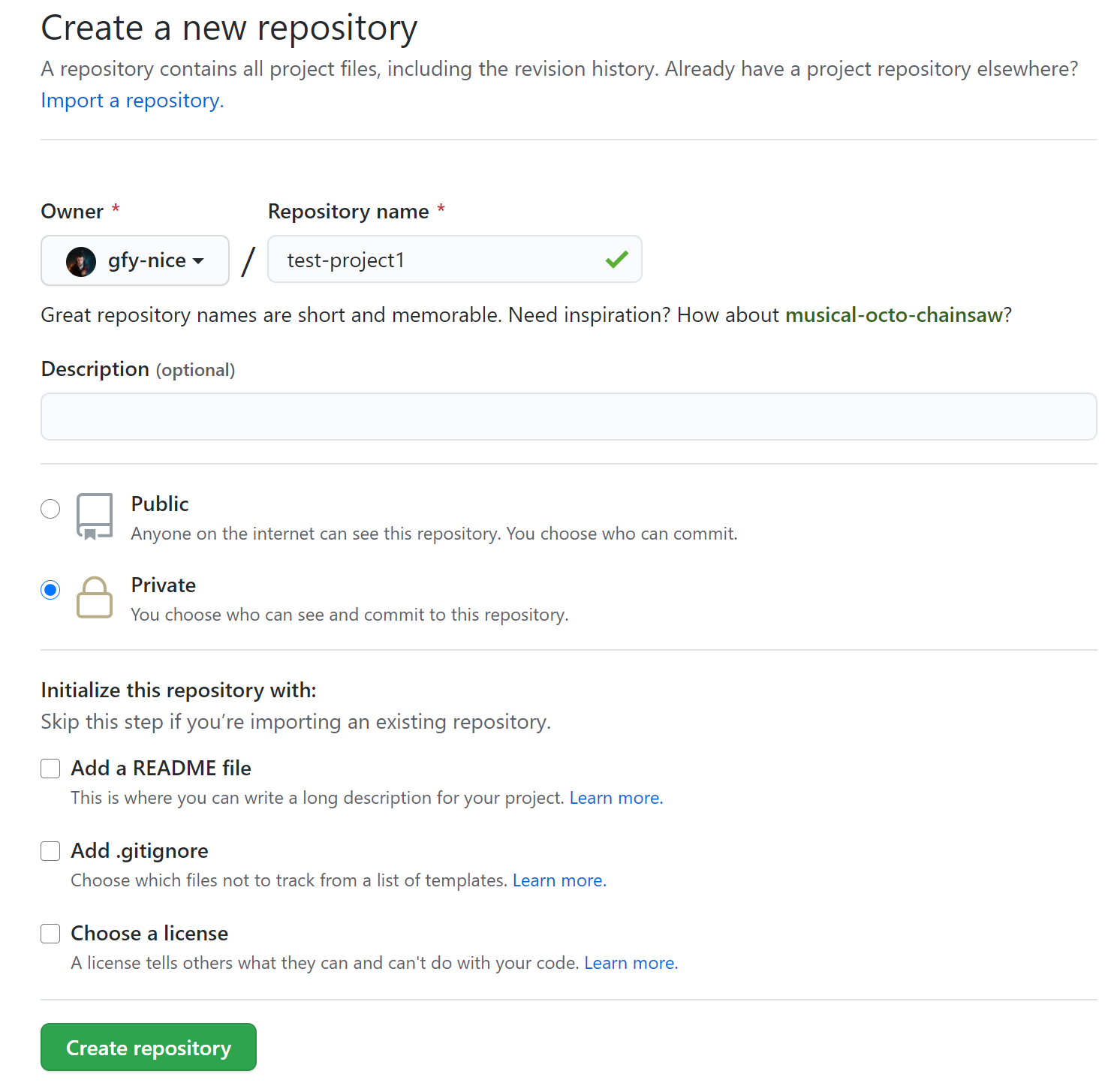
显示某个远程仓库的信息

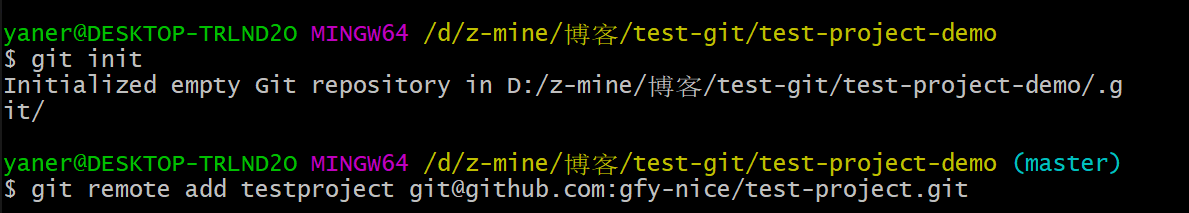


### git remote add [shortname] [url]

增加一个新的远程仓库，并命名

如果没有git仓库，先到GitHub上创建，





### git pull [remote] [branch]

取回远程仓库的变化，并与本地分支合并

### git push [remote] [branch]

上传本地指定分支到远程仓库

### git push [remote] --force

强行推送当前分支到远程仓库，即使有冲突

### git push [remote] --all

推送所有分支到远程仓库

### git checkout [file]

恢复暂存区的指定文件到工作区

### git checkout [commit] [file]

恢复某个commit的指定文件到暂存区和工作区

### git checkout .

恢复暂存区的所有文件到工作区

### git reset [file]

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 --hard

重置暂存区与工作区，与上一次commit保持一致

### git reset [commit]

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 --hard [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 --keep [commit]

重置当前HEAD为指定commit，但保持暂存区和工作区不变

### git revert [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 stash

暂时将未提交的变化移除，稍后再移入

### git stash pop