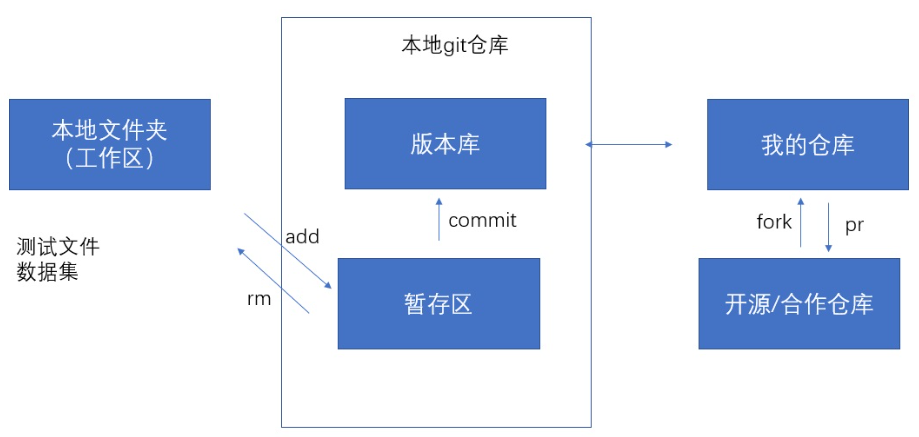
参考北京理工大学周弈帆个人博客[git使用方法、git与github的关系 | 周弈帆的博客 (zhouyifan.net)](https://zhouyifan.net/2021/09/21/20210916-git-note/)

# git总体流程概览



# git命令

**常用命令：**

git init ：新建文件夹，进行git仓库初始化

git add XXX ：XXX为文件名，将本地文件添加到暂存区

git commit -m “XXX”：XXX为自定义改动信息，将暂存区文件提交到本地版本库

git pull <远程主机名> <远程分支名>:<本地分支名>**将本地仓库更新为远程仓库内容** 如果本地远程一样冒号可以省略

git push <远程主机名> <本地分支名>:<远程分支名>**将远程仓库更新为本地仓库内容** 如果本地远程一样冒号可以省略

git status ：查看当前暂存区改动状况

git branch：查看分支情况

git diff ：查看当前暂存区具体修改细节

## 1.1版本回退（较严重已经commit过了）：

1. 错误改动，但已经commit进入版本库。
2. git log：查看所有改动日志
3. git reset -hard HEAD^：回退版本

HEAD为当前版本，每多一个^多回退一个版本。 且HEAD可以用具体某一版本的ID来替代。

-hard表示不仅修改暂存区文件还同时修改本地文件。

## 1.2版本回退（不严重，还未add或add后但未commit）：

1. 错误改动1

从github中pull下来后不小心改动了但是还未add进入暂存区，

使用git checkout -- <filename>进行恢复

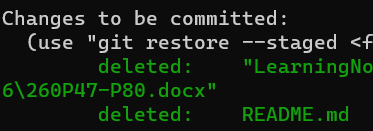
1. 错误改动2

或add后但未commit至版本库，此时想对add进暂存区的文件进行修改，

首先git reset HEAD <filename>把git暂存区中的文件恢复到没有add之前的状态（最后一次git commit的状态）。等于回到了上个问题，文件没有add该怎么恢复。再一次执行git checkout -- <filename>就可以把文件的内容也彻底恢复回去了。

## 1.3版本回退（不小心add了不需要的文件）：

使用git rm --cache <filename>：将暂存区不需要的文件清除，会显示状态为deleted：



git rm -f <filename>：会同时将本地文件也删除

## 1.4使用gitignore：

有一个文件，我想在文件系统保留它，却不想让它进入git仓库。每次git status时都会有显眼的 untracked files。

可以在git仓库的根目录下创建一个.gitignore文件，用于描述不被git跟踪的文件。描述的形式可以是直接描述文件的全名，也可以用\*来模糊匹配，还可以直接忽略一整个文件夹。比如在.gitignore中加入test.py，再输入git status就不会提示有文件没有加入git仓库了。

## 1.5关联远程仓库

git remote add origin [git文件的https地址]：将本地库与github仓库关联起来。效果等同于git clone [git文件的https地址]。

git pull origin：在绑定远程仓库后，将远程仓库的master分支与本地仓库合并。

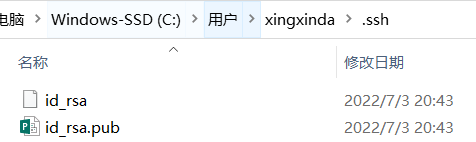
git remote remove origin：撤销远程库的关联。

git push origin master（克隆的分支名）：将本地版本库中内容推送至github。

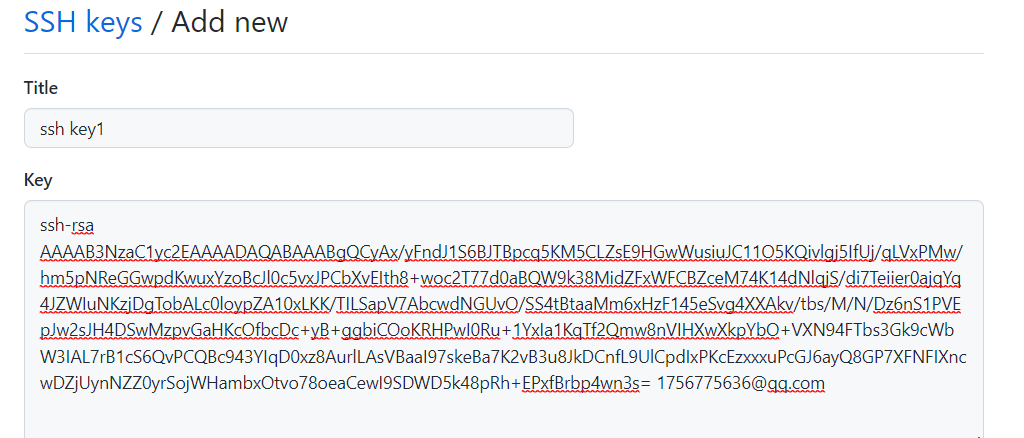
使用https的方法通信，很容易出现超时的问题。如果之前不小心对https格式的地址进行了绑定，可以先remove掉。

### 1.5.1 添加ssh

1. 在个人文件夹中 输入 ssh-keygen -t rsa -C [1756775636@qq.com](mailto:1756775636@qq.com)



1. 在github中新建ssh key将id\_rsa.pub公钥内容复制进去，



1. 最后使用 git remote add origin [自己的远程仓库https地址]把本地仓库和远程仓库关联。或者使用git clone [自己的远程仓库https地址]将远程仓库克隆下来。之后不用梯子也可以push pull。

### 1.5.2 克隆别人的仓库

克隆别人的仓库无法修改。如果只是用别人的代码，clone仓库本身就行了。如果想要对别人的仓库进行贡献，则最好先去github上面点一下fork，分支出一个完全一样的，属于你自己的仓库，再clone这个属于自己的仓库。因为对别人的仓库是无法进行修改等操作的。

## 1.6使用分支

* 创建分支：git branch branch-name
* 删除分支：git branch -d branch-name
* 切换到新分支：git checkout branch-name或者 git switch branch-name

（以上两步可以用git checkout -b branch-name 一步完成）

* 查看所有分支：git branch
* 合并分支：git merge B

Ps：假设当前所在分支为A，merge后分支是B，则是将B合并到A分支。

## 1.7 Pull Request

不管是开发自己还是别人的开源项目，都建议创建一个新的分支开发。开发完一项功能后，把新分支推送到github上，再利用github的pull request把分支合并到主分支上。这种开发方式使得在添加新功能时，所有人都能看到哪些代码进行过修改，有利于多人协作。

更新github上的分支的几分钟内，github会主动提示你要不要进行pull request。或者是点击pull request一栏主动进行”New pull request“，都可以开启一个新的PR。

PR的标题和内容一般都有格式要求。如果是向别人贡献代码，一定要向别人的要求对齐。输入完对PR的描述后，就可以正式创建一个PR了。

（这里没截图）PR的审阅者可以在PR的file changes里面查看有哪些代码被修改，确认无误后可以点击approve，并在整个pr的最后面点击merge。这样，一个带有新功能的分支就正式被并入主分支了。