# **Git**

**Git**一、**Git是一个版本管理系统**二、Git的基本工作原理三、Git的使用四、提交步骤五、Git进阶六、分支命令七、暂时保存更改八、Github九、多人协作开发流程十、将本地仓库推送到远程仓库中十一、解决冲突十二、跨团队协作十二、SSh免登陆十三、Git忽略清单十四、为仓库添加详细的说明

###### **一、**Git是一个版本管理系统

Git 是一个程序源代码的分布式版本管理系统。

1.Git查看版本号：右击打开Git命令窗户，然后输入git -version

###### **二、Git的基本工作原理**

1.Git 仓库：用于存放提交记录

2.Git暂存区：临时存放被修改文件

3.Git工作目录：被Git管理的项目

###### **三、Git的使用**

在使用git前需要告诉git 你是谁，在向git仓库中提交时需要用到（如果要进行修改的话直接整个过程再来一遍）在需要使用Git的工作文件夹中直接进行配置,只需要在第一次使用的时候进行配置

1.配置提交人姓名：

git config --globle user.name 提交人姓名

2.配置提交人姓名：

git config --globle user.eamil 提交人邮箱地址

3.查看git 的配置信息：

git config --list

###### **四、提交步骤**

在项目文件夹下打开Git命令行程序

1.git init -------->在项目文件夹下初始化一个Git仓库（会自动生成一个git文件夹）

2.git status ------>查看Git管理下项目文件夹的文件状态（被管理、没被管理）

3.git add 文件列表 ----->追踪（红色：未追踪）想要管理为文件（将Git管理下的文件添加冬奥暂存区中）

4.git commit -m 提交说明 -----> 将暂存区中的文件提交到Git仓库中（每次提交值提交一个功能：减小后期维护难度）

5.git log ----->查看历史提交记录，可以看到提交ID

五、恢复记录和撤销：

1.将Git仓库中指定的更新记录恢复出来

git rest --hard commitID

2.撤销：

2.1、用暂存区中的文件覆盖工作目录中的文件：

git checkout 文件名字

2.2、将文件从暂存区中删除：

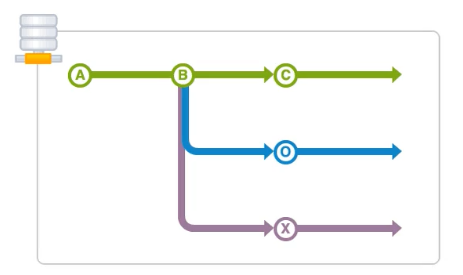
git rm --cached 文件名称

2.3、将git 仓库中指定的更新记录恢复出来，并且覆盖暂存区和工作目录：

git reset --hard commitID

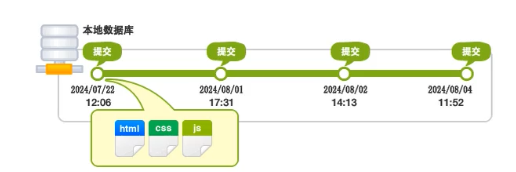
###### **五、Git进阶**

分支：大家暂时可以认为分支就是当前工作目录中代码的一份副本，使用分支，可以让我们从开发主线上分离出来，以免影响主线的开发。



1.1分支细分：功能分支--->开发分支--->主分支

1)主分支（master）：第一次向git仓库中提交更新记录时自动产生的一个分支(根据时间来的)



2）开发分支(develop)：作为开发的分支，基于master分支创建，在开发分支中实现所需功能之后合并到主分支上，分支和分支之间时独立的。

3）功能分支(feature)：作为开发具体功能的分支，基于开发分支创建，专门用于功能开发，开发完成之后合并到开发分支上。

###### **六、分支命令**

1、git branch -------->查看分支（前面带\*说明正处于该分支）

IMG_258

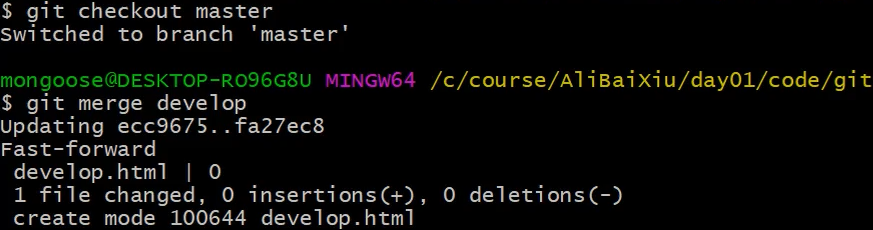
2、git branche 分支名称 -------->基于前面查看的正处于的分支创建分支

IMG_259

3、git checkout 分支名称 -------->切换分支（一定要保证当前工作目录已经提交到Git仓库中）

IMG_260

4、git merge 来源分支 -------->合并分支（先切换到主分支上，再合并，虽然合并了但是开发分支还是存在）



5、git branch -d 分支名称 -------->删除分支（分支被合并后才允许删除，-D是强制删除 ）如果分支没有合并时不允许被删除的

IMG_262

###### **七、暂时保存更改**

在Git 中，可以暂时提取分支上所有的改变并存储，让开发人员得到一个干净的工作副本，临时转向其他工作

1、存储临时改动：

git stash

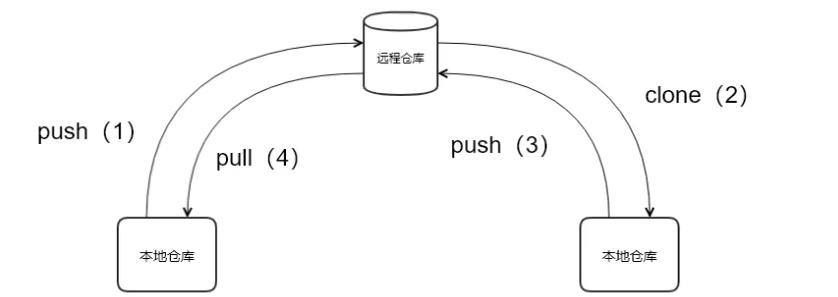
2、恢复改动（一定要切换会刚才需要临时存储的工作分支上）：

git stash pop

###### **八、Github**

这是一个远程仓库（公共的），用于和别人一起进行开发同一个项目的时候

###### **九、多人协作开发流程**

* 

1、A在自己的计算机中创建本地仓库

2、A 在github中创建远程仓库

3、A 将本地仓库推送到远程仓库中

4、B 克隆远程仓库到本地仓库进行开发

5、B将本地仓库中开发大的内容推送到远程仓库

6、A将远程仓库中的最新内容拉到本地

###### **十、将本地仓库推送到远程仓库中**

1、git push 远程仓库地址 分支名称

2、git push 远程仓库地址别名 分支名称

3、git push -u 远程仓库地址别名 分支名称

-u 记住推送地址及分支，下次推送只需要输入git push 即可

4、git remote add 远程仓库地址别名 远程仓库地址 （给远程仓库地址添加一个别名）

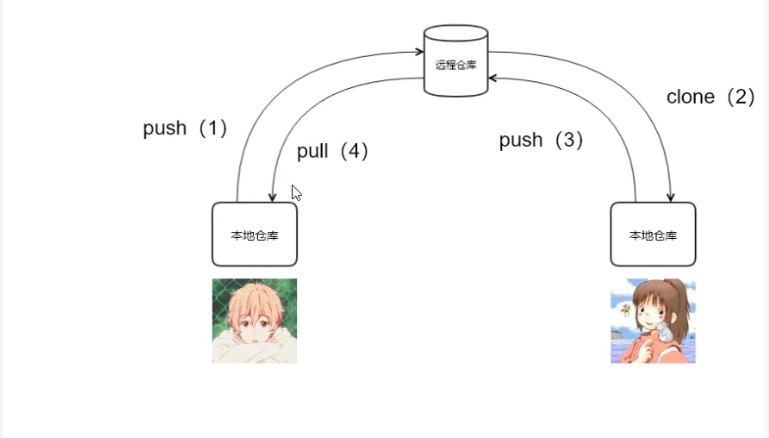
5、程序员B克隆远程数据仓库到本地

git clone 远程仓库地址

6、程序员A需要将程序员B推送的内容拉去到本地

6.1 拉取远程仓库中最新的版本：

git pull 远程仓库地址 分支名称



7.git pull 和git clone 的区别：git pull 只是拉去远程仓库中的最新的版本,在已经有本地仓库的基础上,git clone是完全克隆远程仓库，是在没有本地仓库进行的，只是在第一次开发的时候用到

8.如果远程仓库的版本高于本地仓库版本，此时本地仓库是不能向远程仓库进行提交，本地仓库必须先拉去远程仓库的内容到本地仓库，然后再向远程仓库进行提交

###### **十一、解决冲突**

1、在多人同时开发一个项目时，如果两个人修改了同一个文件的同一个地方（后推送的人是无法推送的），就会发生冲突，冲突时需要人为取解决。

###### **十二、跨团队协作**

登陆自己的账户之后访问别人创建的仓库

1、程序员C fork仓库（将别人的仓库复制下来 ，并且放在程序员C自己的账户中）

2、程序员 C将仓库克隆在本地然后进行修改

3、程序员C将仓仓库推送到程序员C远程仓库

4、程序员C发起 pull resqest

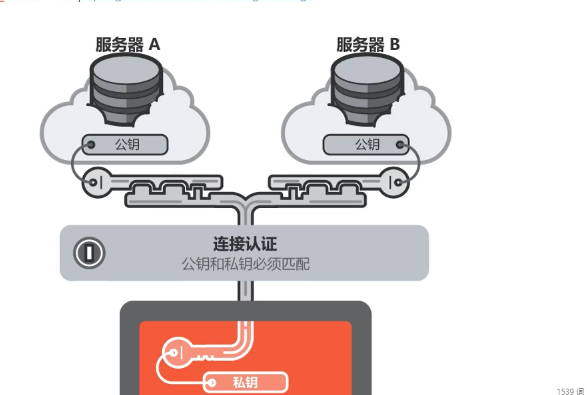
5、原仓库作者审核

6、原仓库作者合并代码

###### **十二、SSh免登陆**

1、https协议仓库地址：

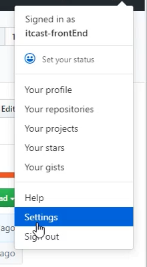
https://github.com/young-sudo

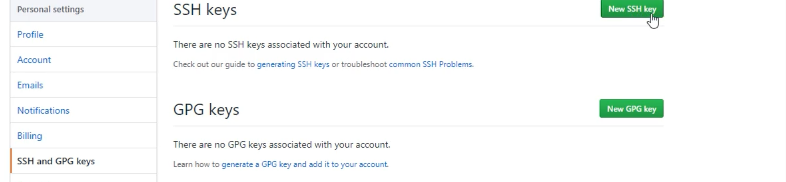
2、

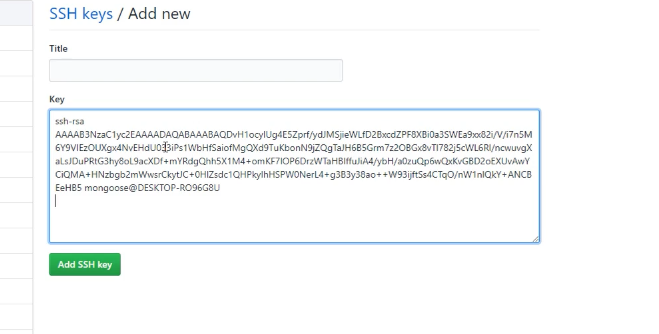
3、公钥：放在Github 账户中 id\_ras.pub 私钥：保存在开发者自己的电脑中id\_ras

4、生成密钥：ssh-keygen (一直回车)--->C\用户\当前所使用的系统用户

（有一个.ssh的文件） 5.id\_rsa.push放在GitHub账户中







6、给ssh 地址添加一个别名：

IMG_269

git remote add 仓库地址别名 ssh地址

###### **十三、Git忽略清单**

将不许要被Git管理的文件名字添加到此文件中，在执行git命令的时候，git就会忽略这些文件。git忽略清单文件名称：.gitignore，将不想被管理的文件夹的名字写在里面。

将工作目录中的文件全部添加到暂存区（实现同时添加多个）：git add . 。

###### **十四、为仓库添加详细的说明**

1、建一个.md文件，在里面进行说明，然后推送到远程仓库