**\*git跟踪并管理的是修改**

git-bash进入

设置姓名和邮箱

$ git config --global user.name "hank"

$ git config --global user.email "li583708195@gmail.com"

mkdir xxx 在C盘用户下生成一个xxx的文件夹

cd xxx 进入该文件夹//注意正斜线反斜线

pwd 查看该文件夹路径（当前目录）

git init 将当前文件夹作为仓库（repository）

编写一个xx.txt文件放到xxx（子）目录下，把该文件添加到仓库（共两步）：

git add xx.txt 添加文件（可多次添加后一次commit）

git commit –m ”something” 提交文件

git status 查看仓库当前状态（比如xx.txt被修改啦，还没提交啦，谁被删除啦）

git diff xx.txt 查看具体修改的内容

（补充：gif diff HEAD - - xx.txt 查看工作区与版本库最新版本区别）

git log 查看历史版本

git log --pretty=oneline 查看最近三次的历史版本（此时显示的一连串16进制数字是commit id）

git reset - -hard HEAD^ 回退到上一个版本。head表示当前版本，加^表示上一个，可以连续加，缩写为HEAD~100（回退实际是改变HEAD指针指向的位置）

git reset - -hard 585568 向后到某个版本（去未来），后面数字为版本的commit id的前几（7）位

cat xx.txt 查看xx.txt的内容

git reflog 查看历史命令，会显示当前的commit id

**概念：**

工作区：xxx文件夹；版本库：.git文件夹；暂存区（stage）；master唯一，不断加分支



场景1：当你改了工作区某个文件的内容，想直接丢弃工作区的修改时，使用git checkout - - xx.txt

场景2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区（已经add了）时，想丢弃修改，分两步，第一步用命令git reset HEAD xx.txt，就回到了场景1，第二步按场景1操作。

场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考[版本回退](https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/0013744142037508cf42e51debf49668810645e02887691000)一节，不过前提是没有推送到远程库。

rm删除工作区文件之后：

一：git rm从版本库中删除该文件，并git commit；

二：如果rm删错了，git checkout --readme.txt恢复

**远程仓库（origin（git remote –v可查看远程库的详细信息））**：关于github

创建SSH Key

ssh-keygen –t rsa –C [li583708195@gmail.com](mailto:li583708195@gmail.com)

用户名文件夹下有.ssh目录，内有私有密钥id\_rsa和公有密钥id\_rsa.pub（可以任意文本编辑器打开）

如果公司和家里都需要向github传递，那么在github上绑定两个key即可。

git remote add origin https://github.com/losthank/learngit.git关联github

git push -u origin master 推送本地分支master到github

**修改后提交顺序：**

add——commit——git pull --rebase origin master（拉入合并）——git push origin master （推入master分支）

clone远程库git到本地库：git clone [git@github.com:losthank/git.git](mailto:git@github.com:losthank/git.git)

分支管理：要提交没写完的代码使自己下次继续，需要创建一个自己的分支，别人看不到，在自己的分支上工作，开发完毕后再合并到原来的分支上。

**使用分支：**

1. 创建+切换分支：git checkout –b dev （dev为分支名），此处包括两个步骤：git branch dev和git checkout dev
2. 查看分支：git branch，此处会显示所有分支（此时是master和dev），当前分支标有\*
3. 合并dev到当前分支master：当前分支为master（分支切换过来），执行git merge dev（git merge - -no-ff –m”xxx” dev）

[- -no-ff表示禁用Fast forward，即合并后生成一个新的commit，可以看分支历史]

[-m参数添加描述，可以省略]

1. 删除分支：git branch –d dev

Q:如果新分支devnew修改了xx.txt并commit，切换回master上（Git提示当前master分支比远程master分支超前一个提交），继续修改xx.txt并commit，造成的结果是下图：

此时无法快速合并（git merge devnew失效），手动合并。

git log - -graph查看分支合并图



上图：在dev分支上干活，在master上发布版本

**临时任务修复bug：**

在dev上工作，代码还没写完，领导下发修改其他代码的一个bug。

第一步，挂起dev：git stash，此时工作区是干净的；

第二步，确定在哪个分支修复bug（假设在master分支上修复），创建临时分支：git checkout master、git checkout –b bug（先转到master分支，再创建bug分支）

第三步，修复并add、commit，切换到master上合并，再删除bug分支，准备回到dev继续干活

第四步，查看stash内容：git stash list；恢复dev工作区： git stash apply/git stash pop（前者不删除stash，后者删除）；

ps：可以多次stash，恢复时git stash list后执行git stash apply stash@{0}恢复相应stash工作区。

**开发一个新功能（feature）：**创建分支feature，完成，切回主分支，准备合并时临时取消，需要删除，执行git branch –D feature（-d无法删除未合并分支，需-D强行删除）

**协同开发：**某人clone远程库之后，只能看到本地master，要在dev分支开发，就必须创建远程origin的dev分支到本地（git checkout –b dev origin/dev），边修改边push到远程（git commit –m “xxx”）

此时如果我也要push同一个修改过的文件，会commit失败，解决办法：先git pull，如果pull失败（没有指定本地dev分支与origin/dev分支链接），执行git branch - -set-upstream dev origin/dev，再git pull，再合并再commit再push。

ps：通常本地和远程的分支名一致；出现错误git一般会提示执行操作的代码。

**（在已发布版本上）打标签：**git tag 标签名（如v1.0）；查看标签名：git tag

默认标签在最新的commit上，如果之前忘记打标签，查找历史提交的commit id打上即可。（git tag 标签名 commit’id）

标签是按字母顺序排列的，git show v1.0查看标签信息。

git tag –a v1.0 –m “xxxx” -a指定标签名 –m加标签说明。-s替换-a的话，最后可以用私钥签名（PGP签名）一个标签。（这里不懂）

git tag –d v1.0删除本地标签

git push origin v1.0 推送标签到远程

git push origin - -tags一次性推送全部标签到远程

删除远程标签：先删除本地标签，再远程删除（git push origin :refs/tags/v1.0）

忽略特殊文件（创建.gitignore文件夹）：留待后续实践

**配置别名**（输入st即表示status）：git config - -global alias.st status

其他简写：co/checkout ci/commit br/branch unstage/’reset HEAD’（撤销暂存区内容） last/’log -1’（git last显示最后一次提交信息） 输入git lg可以直接看颜色等配置git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"

所以以后可以写成 git ci –m”xxx”

**配置文件：** - -global针对当前用户，不加是针对当前仓库；

.git/config存放仓库的Git配置文件（cat .git/config查看），当前用户的git配置文件在用户主目录下的.gitconfig文件中。

git服务器搭建、权限管理后续实践

*整理来源：廖雪峰的官方网站*