# git一般命令行

所有git仓库的用户名和邮箱

git config --global user.name “wudongyue”

git config --global user.email [“m18207144079@163.com”](mailto:\“m18207144079@163.com\”)

将一个目录变为git可以管理的仓库

git init 会生成一个 .git的目录，是用来跟踪管理版本库的（不要手动修改）

将文件添加git仓库

git add test.txt

将当前文件夹中所有文件提交

git add .

提交文件

git commit -m ‘this is remark’

提交说明

回退

git reset --hard HEAD^

上个版本 （上上个版本 HEAD^^, HEAD~10 往前10个版本）

这里的log也会被消除，无法返回后面的版本，但是通过commit id还是可以回到后面

的版本

--hard : 硬恢复 --soft: --mixed:

git reset --hard e2e8c40df8cf4451e818e2eab39ba8a75a73eaa8

不写全也可以

实际上不能回退新增的文件，因为还没有上传的文件，没有入仓库，无法回退。

查看提交记录(配合回退)

git log

查看每次操作后的commit id (配合git reset --hard commitId 任意切换版本)

git reflog

查看工作区和版本库中暂存区文件的状态

git status

Untracked ： 未被跟踪的文件

changes not staged ： 修改未被放入暂存区

changes to be committed : 修改已提交

new file：

modified：

deleted

对比本地文件与仓库中文件

git diff a -- b

去除工作区文件的修改

git checkout -- a

会将a文件恢复到add 或commit后的状态，其中工作区的未被add修改会被去掉

但是对于新增文件（没有被add），git仓库没有保存，无法查找，也就不能删除它。

删除也是修改，因此 此命令可以恢复被删除的文件 ！！！ 但是好像不能用，不能匹配。。

将add 回退（就是将暂存区的删除）

git reset HEAD test.txt

配合 git checkout -- test.txt可以对已add 的test.txt文件工作区回退修改以前状态

添加本地远程库关联

git remote add origin git@github.com:piLurk/gitTest.git

删除本地远程库关联

git remote rm origin

查看本地分支

git remote

git remote -v 远程库详细情况

C:\Program Files\TortoiseGit\bin\TortoiseGitPlink.exe

修改本地远程库管理

git remote origin set-url [url]

向远程库推送

git push -u origin master

-u:第一次推送使用，将本地mater分支内容推送大远程新的master分支，并将本地的mater分支与远程的master分支关联起来。

远程库克隆到本地

git clone git@github.com:piLurk/gitTest1.git

创建分支

git checkout -b dev

-b：创建并切换

相当于

创建分支 git branch dev

切换分支 git checkout dev

查看分支

git branch

合并分支

git merge dev

删除分支

git branch -d dev

将远程仓库拉到本地仓库

git pull （需要本地仓库与远程仓库有链接）

隐藏工作区

git stash

显示隐藏的列表

git stash list

恢复到隐藏时的文件状态

git stash apply + git stash drop (删除stash)

git stash pop 恢复同时删除

git stash apply stash@{0} 恢复到指定的stash

查看标签列表

git tag

展示的顺序是按照字母排序，而不是时间排序的。

git tag -a v1.0 -m “有个说明”

-a:标签名 说明

查看标签详情

git show v1.0

打标签

git tag 1.0

标签可以是汉字

删除标签

git tag -d v0.1

向远程推送标签(对应github上的releases)

git push origin v1.0

推送所有标签

git push origin --tags

删除远程库上的标签

git push origin :refs/tags/0.14.0

# 工作区和缓存区

## 工作区：

电脑中能看到的目录就是工作区

## 版本库

.git就是git的版本库



stage(index):暂存区

master：git自动创建的第一个分支

HEAD: 指向master的一个指针

git add 将文件添加到暂存区

git commit 实际上就是讲暂存区的内容提交到当前分支

将所有需要提交的文件修改统统放在暂存区，然后，一次性提交暂存区所有修改。

# 管理修改

git跟踪管理的是修改，而非文件。

修改字符，添加文字，创造文件都是修改

# 远程仓库

如 github

# 分支管理

分支的意义：

短时间无法完成的开发，不完整的代码上传会影响别人，等到开发完毕再提交又有丢失的风险。

先开一个自己的分支，别人不可见，等到开发完毕，在一次合并到原来的分支上。

实际上svn也是有分支，但是操作非常慢

主分支master



分支：新建指针dev ，将HEAD指向dev。



dev分支提交一次后，dev指针向前移动一步，而master指针不变



合并就是讲master指向dev的当前提交，就完成了合并。



## 删除分支后保留分支信息

默认情况下Git采用Fast forward模式（这种模式下，会丢失分支信息）

用--no-ff方式的git merge

git merge --no-ff -m “merge with no-ff” dev

此次合并要创建一个新的commit， 所有加上 -m ,这样会保留分支信息。在git reflog中可以找到。

这个参数可以看出曾经做过合并，而fast forward看不出来曾经做过合并

## bug分支

当当前开发一时完成不了，可以先将当前文件状态保存隐藏（stash）起来，配合新开一个bug分支，修复后合并主干，并删除bug分支。在提取隐藏的文件状态。

## Feature分支

与bug分支类似，但是不确定性很高。

## 多人协作

master主干，时刻与远程同步。

dev开发分支，也需要与远程同步。

bug分支，没有必要退到远程。

feature分支是否需要推送到远程，取决于你是否与同事在上面合作开发。

git pulll

# 标签

历史版本标签，比commit id好管理。

# .gitignore

在根目录配置忽略文件

*# Windows:*Thumbs.db

ehthumbs.dbDesktop.ini

*# Python:*

\*.py[cod]

\*.so

\*.egg

\*.egg-info

dist

build

*# My configurations:*

db.ini

deploy\_key\_rsa

将这个文件提交到git上， ok

强制提交 git add -f a.class

# 其中碰到的问题和错误

1. GitHub远程创建仓库时，如果用README初始化仓库，只能在本地克隆仓库，在本地创建与远程仓库origin分支的链接，不会成功上传本地库，因为已经初始化了。