所有的版本控制系统, 其实只能跟踪文本文件的改动, 比如TXT文件, 网页, 所有的程序代码等等, Git也不例外.

初始化一个Git仓库：git init

添加文件到Git仓库，分两步：

git add <file>，注意，可反复多次使用，添加多个文件；

git add –-all命令全部加入缓存区

git commit -m <message> 提交文件到版本库

要随时掌握工作区的状态：git status

如果git status告诉你有文件被修改过，用git diff <file>可以查看修改内容。工作区与缓冲区比较。

命令显示从最近到最远的提交日志

git log

简化log的输出信息

git log --pretty=oneline

回退到上一个版本

git reset --hard HEAD^

回退到指定版本(id可以只写前几位)

git reset --hard <commit\_id>

查看版本变更历史

git reflog

工作区: 在电脑里能看到的目录.

版本库(repository): 就是.git文件夹. Git的版本库里存了很多东西, 其中最重要的就是称为stage(或者叫index)的暂存区, 还有Git为我们自动创建的第一个分支master, 以及指向master的一个指针叫HEAD.



工作区, 版本库和暂存区的关系.

untracked状态是指该文件还没有被添加过.

查看工作区和版本库里面最新版本的区别：

git diff HEAD -- <file>

把<file>文件在工作区的修改全部撤销

git checkout -- <file>

一种是readme.txt自修改后还没有被放到暂存区, 现在, 撤销修改就回到和版本库一模一样的状态.

一种是readme.txt已经添加到暂存区后, 又作了修改, 现在, 撤销修改就回到添加到暂存区后的状态.

总之, 就是让这个文件回到最近一次git commit或git add时的状态.

已经add, 还没有commit, 取消缓存区的修改:

(1) git reset HEAD <file>

(2) git checkout -- <file>

如果已经commit了, 那就回退版本(看前面的内容).

git rm –cached <file>，可以从缓存区移除文件，使该文件变为未跟踪的状态，同时下次提交时从本地库中删除。

从版本库中删除文件:

(1) rm <file> //删除工作区的文件

(2) git rm <file> //删除版本库中的文件

(3) git commit -m "remove <file>" //提交删除

如果误删了, 要从版本库中回复最新的版本:

git checkout -- test.txt

本地git仓库关联远程的github服务器

先看看/c/Users/<user\_name>/.ssh/文件夹受否存在, 如果存在是否有id\_rsa和id\_rsa.pub这两个文件.

如果已经有了, 可以直接跳到下一步.

第一步:

创建SSH Key, 一路回车

ssh-keygen -t rsa -C "youremail@example.com"

id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥，可以放心地告诉任何人。

第二步:

登录github, 打开Account settings, SSH Keys, 然后, 点Add SSH Key,填上任意Title, 在Key文本框里粘贴id\_rsa.pub文件的内容.

git本地已经有一个仓库了, 可以在github上创建一个空的仓库<repository\_name>, 然后将本地仓库与远程仓库进行关联.

关联远程仓库：

git remote add origin [git@github.com:<github\_user\_name>/<repository\_name>.git](mailto:git@github.com:%3cgithub_user_name%3e/%3crepository_name%3e.git)

如git remote add origin [git@github.com:cMH2017/learngit.git](mailto:git@github.com:cMH2017/learngit.git)

远程仓库的名字就是origin.

把本地库的所有内容推送到远程库上：

git push -u origin master

-u参数, 把本地的master分支和远程的master分支关联起来, 在以后的推送或者拉取时就可以简化命令.

关联以后，推送最新修改:

git push origin master

查看远程仓库的信息:

git remote -v

解除本地仓库与远程仓库的关联关系:

git remote rm origin

克隆一个本地库：（一定要用ssh地址， 如果用http地址， 每次push都要弹窗登录）

git clone git@github.com:<github\_user\_name>/<repository\_name>.git

Git自动把本地的master分支和远程的master分支对应起来了，并且，远程仓库的默认名称是origin。

查看分支：git branch

创建分支：git branch <name>

切换分支：git checkout <name>

创建+切换分支：git checkout -b <name>

合并某分支到当前分支：git merge <name>

删除分支：git branch -d <name>

强行删除分支：git branch -D <name>

git log --graph --pretty=oneline --abbrev-commit 看到分支的合并情况

git merge --no-ff -m "merge with no-ff" dev --no-ff参数，表示禁用Fast forward

git stash 把当前工作现场“储藏”起来，等以后恢复现场后继续工作

git stash list 查看被stash的工作现场

git stash apply 恢复后，stash内容并不删除

git stash apply stash@{0} 恢复指定的stash

git stash drop 删除stash中的内容

git stash pop，恢复的同时把stash内容也删了

git remote [-v] 查看远程库的信息

git push origin <name> 推送分支到远程仓库

基于远程分支origin/dev创建本地分支dev:

git checkout -b dev origin/dev

指定本地dev分支与远程origin/dev分支的链接:

git branch --set-upstream-to=origin/dev dev

git pull 把最新的提交从origin/dev抓下来

命令git tag <tagname>用于新建一个标签，默认为HEAD，也可以指定一个commit id；

命令git tag -a <tagname> -m "blablabla..."可以指定标签信息；

命令git tag可以查看所有标签

命令git push origin <tagname>可以推送一个本地标签；

命令git push origin --tags可以推送全部未推送过的本地标签；

命令git tag -d <tagname>可以删除一个本地标签；

命令git push origin :refs/tags/<tagname>可以删除一个远程标签。

让Git显示颜色，会让命令输出看起来更醒目：

git config --global color.ui true

.gitignore文件 把要忽略的文件名填进去，Git就会自动忽略这些文件。

所有配置文件可以直接在线浏览：https://github.com/github/gitignore

git add -f App.class 用-f强制添加到Git

git check-ignore -v App.class 找出来到底哪个规则写错了

git config --global alias.st status 以后st就表示status

--global参数是全局参数，也就是这些命令在这台电脑的所有Git仓库下都有用