【Git学习】

【简单指令】

$ git config --global user.name "username" 创建全局的用户名

$ git config --global user.email "email" 创建全局的用户邮箱

$ mkdir xxxx 创建文件夹

$ cd Y:/WS/Git/GitSkill 打开目录 （注意用 / ）

$ pwd 查看当前目录

$ ls 查勘当前目录下的文件清单

$ rm text.txt 删除文件

【简单Git指令】

$ git init 初始化文件夹，创建.git文件

$ git add readme.txt 添加文件到暂存区

$ git commit -m "wrote a readme.txt file" 将暂存区的文件提交到分支

$ git status 查看当前的状态（显示当前分支，暂存区是否有文件可以被提交，工作区是否有文件被修改）

$ git diff readme.txt 查看工作区对最新版本做了哪些修改

$ git log 查看历史提交详细日志（包括id，分支，用户名，时间，log）

$ git log --pretty=oneline 查看历史提交日志的简单信息（包括id，分支，log）

$ git reset --hard HEAD^ 回滚到上一个版本

$ git cat readme.txt 打印出文件的内容

$ git reset --hard 574b4f 回滚到某一个id的版本（id是16进制的提交记录代码）

$ git reflog 查看命令历史，以便确定要回到未来的哪个版本。

$ git diff HEAD -- readme.txt 查看工作区和版本库里面最新版本的区别

【撤销、恢复与删除指令】

$ git reset HEAD readme.txt 把暂存区中对文件的修改撤销掉（unstage），不管修改了几次，就像删除了暂存区的文件。（git reset命令既可以回退版本，也可以把暂存区的修改回退到工作区。当我们用HEAD时，表示版本库的最新版本。此时再使用checkout就会把工作区恢复到版本库最新版的状态。）

$ git checkout -- readme.txt 如果修改了工作区但没添加到暂存区，就会恢复到和版本库一样的状态；如果添加到了暂存区后又修改了工作区，就会恢复到和暂存区一样的状态。（--很重要，没有--，就变成了“切换到另一个分支”的命令）

【场景1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令git checkout -- file。

场景2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令git reset HEAD file，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，参考版本回退一节，不过前提是没有推送到远程库。】

$ git rm text.txt 删除工作区和暂存区的文件。删除之后，如果直接用$ git checkout text.txt无法恢复工作区的text.txt，因为最近的一次$ git rm操作后没有commit，所以工作区恢复的是暂存区的状态，而暂存区已经没有text.txt了。所以要用$ git reset HEAD text.txt撤销暂存区中删除text.txt的操作，再$ git checkout text.txt把工作区恢复成暂存区的状态。

【创建远程仓库】

$ git remote add origin https://github.com/cfjsmhwp/Git-Learning.git

关联本地的仓库和远程仓库

$ git push -u origin master

第一次推送远程仓库，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的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。

$ git push origin master

把本地master分支的最新修改推送至GitHub。

$ git clone git@github.com:cfjsmhwp/Git.git

克隆一个远程库到本地

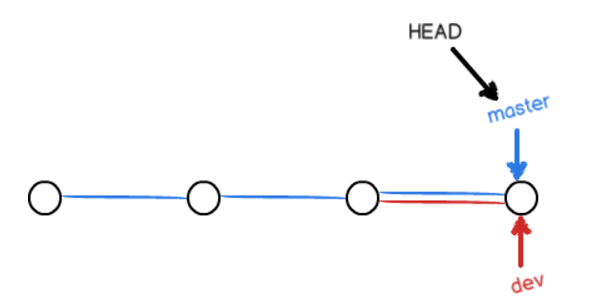
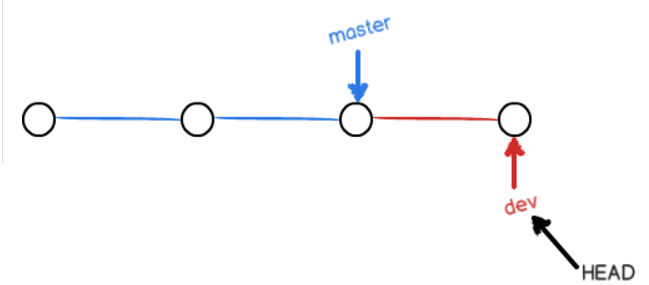
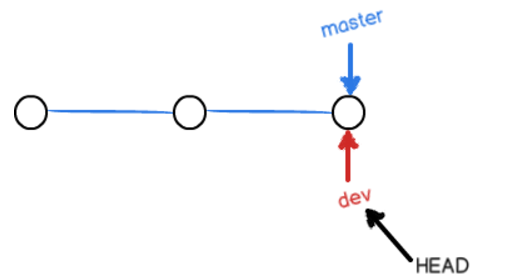
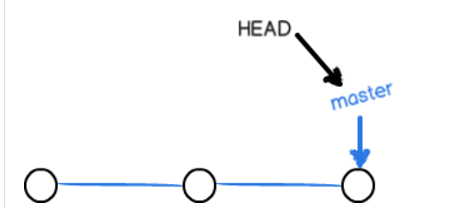
【https://github.com/cfjsmhwp/Git.git

git@github.com:cfjsmhwp/Git.git

Git支持多种协议，默认的git://使用ssh，但也可以使用https等其他协议。Git支持多种协议，包括https，但通过ssh支持的原生git协议速度最快。

使用https除了速度慢以外，还有个最大的麻烦是每次推送都必须输入口令，但是在某些只开放http端口的公司内部就无法使用ssh协议而只能用https。】

【分支】



每次提交，Git都会把它串成一条时间线。第一幅图只有一条分支，就是主分支。HEAD指向master，而master才是指向提交的。HEAD指向的就是当前的分支。因此就能确定当前分支以及分支的提交点。

创建一个新的分支dev的时候，Git创建了一个新的指针dev，指向master相同的提交点，再把HEAD指向dev，就表示当前分支在dev上。因此创建一个新的分支很快，只是增加了一个指针。

接着工作区修改和提交就会在dev分支上增加时间节点。

Dev工作完成之后，要把dev合并到master上，就直接把master指向dev当前提交的时间节点，完成合并。