git总结流程演示

1. 安装git后需要设置 --global 参数 表示这台机器上所有的仓库都会使用这个配置
   1. git config --global user.name “your name”
   2. Git config --global user.email “your email”
2. 将预先创建好的文件夹设置为仓库
   1. cd F:\rest\ data (以我的文件路径为基准)
3. 在仓库中创建自己的文件后（如：readme.txt）
   1. git add readme .txt (将文件提交到暂存区)
   2. git status (查看仓库的当前状态，确定readme.txt是否在暂存区内准备提交)
   3. git commit -m “first tijiao” (将文件从暂存区提交到分支中，-m是本次提交的说明)
4. 如果工作区的文件被修改可以通过git status 来查看哪个文件被修改了，再用 git diff 来将工作区中被修改的文件与分支中的文件进行对比来确认修改了哪些东西，然后就可以进行提交了（提交步骤请参考3）
5. 工作中如果想要提交后突然需要以前版本的文件时，我们就需要进行版本回退的操作了
   1. git log (查看从最近到最远的提交日志，来查看要回到那个版本)
   2. git reset --hard HEAD^ (使用这个语句就可以将版本回退到上一个版本 HEAD^是上一个版本^^是上上个版本^可以递增，这样我们就回到了过去)
   3. 如果需要再回到未来的版本的话这个时候我们就需要有一个坐标， 因为如果已经回退后git log是看不到未来的commit id的所以要使用git reflo来查看命令历史，以便确定要回到未来的哪个版本
   4. git reset --hard commit id (利用commit id 就可以回到未来的版本了，commit id不用全部输入，写入前6位就可以了)
6. 如果工作中提交到版本库之后又在工作区做了一些你不想要的操作，你想要将它抛弃，变回和版本库中一样的版本的时候这个时候就可以用git checkout --readme.txt (把readme.txt在工作区的修改全部撤销，相当于用版本库中的文件替换掉工作区中的文件)
7. 如果你在工作区中做了一些不想要的操作并且不小心的存储到了暂存区中（git add），所幸在提交到分支（git commit -m “点点滴滴”）之前你发现了这些问题，那么就可以使用git reset HEAD readme.txt（将暂存区的修改撤销掉重新放回工作区，然后再用git checkout --readme.txt就可以将这些讨厌的操作清理干净不管是暂存区内还是工作区内）
8. 如果你在工作区中做了一些不想要的操作并且不小心的存储到了暂存区中（git add），并且还存储到了分支当中（git commit）还记得版本回退吗，当然如果你已经推送到了远程库中那么就没有办法了。
9. 如果在工作中，你提交了一个文件到分支当中，但是过后又发现这个文件是不需要的，你在工作区把这个文件删掉了，但是分支中还存在怎么办，这个时候应该git rm text.txt，并且 git commit -m ”remove text.txt”.现在文件就从分支当中删除了。当然还有另外一种情况，就是发现自己删错了，而版本库中还有这个文件的话，那么就用git checkout --text.txt 来用版本库中的文件来替换工作区中的文件。
10. 远程仓库
    1. git remote add origin [git@github.com:CherrySUKURA/Data.git](mailto:git@github.com:CherrySUKURA/Data.git) (关联远程仓库，这里关联的是github中的仓库)
    2. git push -u origin master (将master分支推送到远程库中的master分支上)
    3. git clone [git@github.com:CherrySUKURA/gitskills.git](mailto:git@github.com:CherrySUKURA/gitskills.git) (从github上将gitskills库克隆到本地中)
11. 分支管理
    1. git checkout -b dev 创建分支并切换到dev分支上，在创建并切换到dev分支上后，你在做的所有提交和操作都是在dev分支上的，master上不会有任何显示
    2. git checkout -b dev 相当于 git branch dev(创建dev分支)&&git checkout dev(切换dev分支)
    3. git branch （查看所有分支，当前分支前面会有一个\*号）
    4. git merge dev （用来合并dev分支到当前分支上，这是快速合并是按不到分支的历史的）
    5. git merge --no-ff -m ”merge with on-ff” (普通合并可以看到分支的历史)
    6. git branch -d dev (删除dev分支)