

**多人协作：**首先，可以试图用**git push origin <branch-name>**推送自己的修改；

如果推送失败，则因为远程分支比你的本地更新，  
需要先用**git pull**试图**合并**；

如果合并有冲突，则解决冲突，并在本地提交；

没有冲突或者解决掉冲突后，再用**git push origin <branch-name>**推送就能成功！

如果git pull提示no tracking information，则说明  
本地分支和远程分支的链接关系没有创建，用命令  
**git branch --set-upstream-to <branch-name> origin/<branch-name>**。建立关系



**创建dev分支：** **git checkout -b dev** 或 **git switch -c dev**  
**切换回master：** **git checkout master** 或 **git switch master**  
**合并分支：** **git merge dev**，将dev合并到当前分支  
**删除分支：** **git branch -d dev**  
**查看当前分支：** **git branch**  
**流线形式查看分支修改记录：** **git log --graph --pretty=oneline --abbrev-commit**

**分支冲突：**冲突的时候git status会提示冲突内容，  
需要手动修改成需要的再提交

**合并分支：**合并分支时，加上--no-ff参数就可以用  
普通模式合并，合并后的历史有分支，能看出来曾  
经做过合并，而fast forward合并就看不出曾经做  
过合并。

**保护现场：**当dev分区工作未完成时，先git add，  
然后使用git stash保存现场，此时工作区将会没有  
保存的文件，想继续dev工作时，使用**git stash list**  
查看保存的现场。  
**git stash apply**可以恢复，但是现场仍保护，可以使  
用**git stash drop**删除被保护的现场  
**git stash pop**，恢复的同时把现场也删了

**不同分支修复相同bug：**在master分支上修复的  
bug，想要合并到当前dev分支，可以用**git cherry-  
pick <commit>**命令，把bug提交的修改“复  
制”到当前分支，避免重复劳动

**丢弃分支：**开发一个新feature，最好新建一个分  
支；  
如果要丢弃一个没有被合并过的分支，可以通过**git  
branch -D <name>**强行删除。

**与Github添加/删除远程库：**

- ①首次更新：**git push -u <远程库名> master**
- ②之后更新：**git push <远程库名> master**就可以将本地库  
更新到远程库（Github上）
- ③删除前查看信息：**git remote -v**
- ④删除指令：**git remote rm <name>**

**从远程仓库克隆：(ssh/http...)**  
**git clone git@github.com:lmMengK/GitMXK.git**

**rebase操作的特点：**把分叉的提交历史“整理”成  
一条直线，看上去更直观。缺点是本地的分叉提交  
已经被修改过了。**git rebase**

#### Part4-分支管理

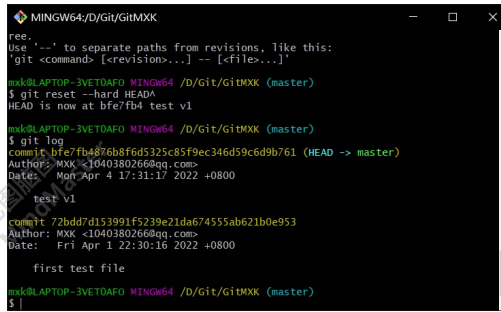
### Git学习

#### Part1-仓库建立

首先需要把文件放到仓库文件夹  
**git add readme.txt**将文件添加（从工作区添加，存放在暂缓区）  
**git commit -m "wrote a readme file"**来提交文件，其中-m后是提交  
说明，该指令可以一次提交add过的多个文件（git add file2.txt  
file3.txt）  
git commit提交的实际上是“修改”，是暂缓区里的文件内容，因此，git add->  
修改①-> commit-> 修改②，修改②是**不会被提交**的。  
正确思路：**第一次修改 -> git add -> 第二次修改 -> git add -> git commit**

**git status** 查看仓库状态  
**git diff**查看修改部分

**历史状态：** **git log**查看仓库文件历史状态



**版本回退：** **git reset --hard HEAD^**  
回退版本指令，其中HEAD标志前一次，HEAD~100代表上100次的版  
本  
回退后，git log查看，后面的版本都会消失。但运行框还在时，可以  
再使用指令根据黄色的版本号来回到新版本----->**git reset --hard 版  
本号开头**

要重返未来，用**git reflog**查看命令历史，以便确定要回到未来的哪个  
版本

**版本撤销：**命令**git checkout -- readme.txt**意思就是，把  
readme.txt文件在工作区的修改全部撤销。若已添加到暂存  
区，可以先回退一个版本（git reset --hard HEAD^）再  
check out

**文件删除：****rm test.txt**删除了文件，但**版本库里还有**，用**git rm  
test.txt**可以**彻底删除**。如果删除未commit，可以用**git checkout --  
test.txt**从**版本库恢复**，提交了就没法恢复了

情况①：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；

情况②：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。

#### Part3-远程仓库

#### Part2-文件控制