Git常用命令集锦

1. 新建文件夹 mkdir 文件夹名
2. 查看目录机构： pwd
3. 将文件添加至Git管理范围：git init
4. 文件添加到Git本地仓库暂存区（stage）
   1. git add 文件夹名
5. 文件提交到Git本地仓库分支（master）
   1. Git commit -m “提交备注描述”
6. 生成 .ssh后缀文件：
   1. ssh-keygen -t rsa -C”自己的Git邮箱号”
7. 查看文件夹下文件提交情况
   1. git status (查看该文件夹下还有多少文件没有进行提交至Git仓库)
   2. git diff 文件名 查看该文件修改了什么内容（如：git diff test.txt）
   3. git diff HEAD -- test.txt 第二种查看文件修改内容写法
   4. cat 文件名 查看文件内容 如：cat test.txt | iconv -f GBK -t UTF-8
8. 版本回退
   1. git log 可以看见最近3次的提交记录
   2. git reset --hard HEAD^ 回退到上一个版本
   3. git reset --hard HEAD~5 回退到往上第五个版本
   4. git reflog 查看我之前的每一次命令（提交，修改，版本回退等）
   5. Git reset --hard 1002b (1002b指的是commit id的前几位数）
   6. git log --graph:查看分支合并图
9. 撤销修改
   1. git checkout -- 文件名 如：git checkout test.txt (**一定要有--**)

说明:命令git checkout -- test.txt意思就是，把test.txt文件在工作区的修改全部撤销，这里有两种情况：

一种是test.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test.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。

* 1. 将暂存区的修改文件撤销回放到工作区

git reset HEAD test.txt

1. 删除文件
   1. rm 文件名 如：rm test.txt
      1. git rm test.txt 将文件从版本库中删除
      2. git commit -m “删除test文件”
   2. git checkout -- test.txt
      1. 如果版本库中存在test.txt文件，可以将版本库中的文件替换成已删除的该文件
2. 连接远程库
   1. git remote add origin [git@github.com:](mailto:git@github.com:chenyuanbo888/first_repository.git)Git远程库地址
   2. 如：git remote add origin [git@github.com:chenyuanbo888/first\_repository.git](mailto:git@github.com:chenyuanbo888/first_repository.git)
3. 推送master分支内容：
   1. 第一次推送：git push -u origin master
   2. 后面推送最新更改：git push origin master
4. 克隆远程仓库
   1. 进入本地要存放的文件夹下，输入以下命令
   2. git clone [git@github.com:chenyuanbo888/test.git](mailto:git@github.com:chenyuanbo888/test.git)
5. 创建与合并分支
   1. git checkout -b dev 创建dev分支并切换至dev：相当于执行以下两条命名
      1. git branch dev
      2. git checkout dev
   2. git branch 查看所有分支
   3. git checkout master 切换至master分支
   4. 合并某分支到当前分支：
      1. git merge dev 将分支dev合并到master分支，前提是现在在master分支下
   5. 删除分支：git branch -d dev
6. 分支管理
   1. 禁用 Fast -forward合并分支，如：
      1. git merge --no--ff -m ”merge with no-ff” dev 禁用fast-forward,将dev分支合并到master分支，并创建了一个新的commit
   2. Bug分支
      1. git stash 将该工作区内容都储藏起来
      2. git stash list 查看工作区的内容
      3. git stash apply 恢复工作区内容，但不删除stash中的内容
      4. git stash pop 恢复工作区内容，并将stash中的内容进行删除
   3. Feature分支
      1. 新增、合并和bug分支操作都一致
      2. git branch -D 分支名称 注意D为大写（强行删除）
   4. 多人协作
      1. 查看远程库信息
         1. git remote -v
      2. 将分本地支推送到远程仓库
         1. git push origin 分支名称、
      3. 推送失败时用git pull抓取远程的新提交文件
      4. 在本地创建和远程分支对应的分支，使用git checkout -b branch-name origin/branch-name，本地和远程分支的名称最好一致；
      5. 建立本地分支和远程分支的关联，使用git branch --set-upstream branch-name origin/branch-name；
      6. 从远程抓取分支，使用git pull，如果有冲突，要先处理冲突。
      7. 同时推送相同修改文件版本冲突问题
         1. 先指定本地dev分支与远程origin/dev分支的链接，根据提示，设置dev和origin/dev的链接：
            1. git branch --set-upstream-to=origin/dev dev
         2. git pull 将远程的文件拉取到本地进行合并
         3. 手动将文件进行更改后在进行push提交至远程库
      8. Rebase
         1. git rebase 将提交历史整理成一条线
7. 标签管理
   1. 创建标签
      1. git tag 版本名： git tag v1.0
   2. 查看所有标签
      1. git tag
   3. 根据commit id进行创建标签
      1. git tag 版本名 commit id前缀 如：git tag v1.0 f54c633
   4. 查看标签信息
      1. git show 标签名 如：git show v1.0
   5. 创建带说明的标签
      1. git tag -a 标签名 -m “说明” f54c633
   6. 删除标签
      1. git tag -d 标签名
   7. 将指定标签推送到远程
      1. git push origin 标签名
   8. 将所有标签推送到远程
      1. git push origin --tags
   9. 删除远程标签
      1. 先删除本地对应的标签
      2. git push origin :refs/tags/v1.0