Git Case 整理

1、 某些本应该忽略的文件（例：“1.java”），结果忘记加到“.gitignore”文件中，导致被提交到了远程仓库：

解：把本应该忽略的文件提交到远程仓库，意味着该文件需要从仓库中删除，不要再继续追踪，但是在本地还需要继续使用。

（1）、备份“1.java”文件。

（2）、执行“git rm 1.java”命令。

（3）、将“1.java”添加到忽略文件中。

（4）、提交所有改动并 push 到远程仓库。

（5）、恢复“1.java”文件。

2、 在做了很多修改之后，发现改错分支了，这些修改本该在“dev”分支做，结果在“master”分支做了，分别对应了四种情况：已修改，已暂存，已提交，已推送（已 push）：

解：在做了很多修改之后，意味着不能简单地回退，不可能在 master 分支上全部回退，再切到 dev 分支再改一次（累死了），所以最好采取储藏方式来解决。

已修改：

（1）、执行“git stash”命令。

（2）、切换到“dev”分支。

（3）、执行“git stash pop”命令或“git stash apply”命令，前者为应用储藏并清除，后者为只应用储藏。

已暂存：

（1）、执行“git stash”命令。

（2）、切换到“dev”分支。

（3）、执行“git stash pop --index”命令或“git stash apply --index”命令，前者为应用储藏并清除，后者为只应用储藏。加上“--index”参数表明应用储藏后还要是已暂存状态。

已提交：

（1）、执行“git reset --soft HEAD~”命令或者“git reset --mixed HEAD~”来选择回退到什么程度。

（2）、执行“git stash”命令。

（3）、切换到 “dev” 分支。

（4）、根据回退的程度来选择是否带 “--index” 参数执行 “git stash pop” 命令或 “git stash apply” 命令。

已推送（已 push）：

（1）、首先保证其他人不要 push 代码。

（2）、执行 “git reset --soft HEAD~” 命令或者 “git reset --mixed HEAD~” 来选择回退到什么程度。

（3）、执行 “git stash” 命令。

（4）、执行 “git push -f origin master” 命令强制回退远程分支版本。

（5）、切换到 “dev” 分支。

（6）、根据回退的程度来选择是否带 “--index” 参数执行 “git stash pop” 命令或 “git stash apply” 命令。

3、 从“dev”分支 merge 到“master”分支，发现有冲突，解决冲突中发现有严重错误，不想 merge 了，想恢复到 merge 之前，分别对应了四种情况：merge 后，已暂存，已提交，已推送（已 push）：

解：发生冲突之后，是可以通过命令来终止此次 merge 的，但是，如果已经提交或者推送到远程仓库，那么就需要用 reset 命令回退。

Merge 后：

（1）、执行“git merge --abort”命令。

已暂存：

（1）、执行“git merge --abort”命令。

已提交：

（1）、执行“git log”命令查看 merge 之前最后一次提交的版本号【version-name】，非常长的那个 SHA-1 签名，例 :5ef0eb88e91b1677810197ab6201501d851dd34c。

（2）、执行“git reset --hard 5ef0eb88e91b1677810197ab6201501d851dd34c”命令。

已推送：

（1）、首先保证其他人不要 push 代码。

（2）、执行“git log”命令查看 merge 之前最后一次提交的版本号【version-name】，非常长的那个 SHA-1 签名，例 :5ef0eb88e91b1677810197ab6201501d851dd34c。

（3）、执行“git log”命令查看 merge 之前最后一次提交的版本号【version-name】，

非常长的那个 SHA-1 签名, 例 :5ef0eb88e91b1677810197ab6201501d851dd34c。

（4）、执行 “git push -f origin master” 命令强制回退远程分支版本。

4、　远程仓库出现了问题，需要以本地仓库为基本，恢复远程仓库。

解：这可能有两种情况，一种是远程仓库某几个分支出现了问题，还有可能是远程仓库整体遭到了删除，一个核心的思想就是把本地的代码 push 到远程即可。

某些分支出现问题（假设 dev 分支）：

（1）、执行“git checkout dev”命令。

（2）、执行“git push origin dev”命令，必要的时候加上“-f”参数，命令是“git push -f origin dev”。

远程仓库整体删除：

（1）、在远程环境执行“git init”命令创建一个远程仓库，假设地址为“git@github.com:test/test.git”。

（2）、在本地仓库执行“git remote add origin git@github.com:test/test.git”命令，将本地仓库关联到远程仓库。

（3）、执行“git push -u origin [branch-name]”命令将分支推送到远程仓库并关联上，假设为 master 分支，则命令为“git push -u origin master”。

5、 执行“git reset --hard HEAD~”命令回退到了上一个版本，此时发现被回退的提交还有用处，此时该怎样找回这次提交。

解：核心思想是找到丢失提交的 SHA-1 值，然后从这个值创建一个分支来继续操作。

Ref log 存在：

（1）、执行“git reflog”命令查看 HEAD 变化的情况，假设丢失的提交 SHA-1 值为 5060539。

（2）、执行“git branch recover-branch 5060539”命令创建一个恢复分支指向这次提交。

（3）、可以在恢复分支做完操作再 merge 回原来的开发分支。

Ref log 不存在：

（1）、执行“git fsck --full”命令查看哪些没有被引用的对象，假设丢失的提交 SHA-1 值为 5060539。

（2）、执行“git branch recover-branch 5060539”命令创建一个恢复分支指向这次提交。

（3）、可以在恢复分支做完操作再 merge 回原来的开发分支。

6、 在“master”分支的提交记录为 commit1 <-- commit2 <-- commit3，在“dev”分支的提交记录为 commi1<-- commit2 <-- commit3 <-- commit4 <-- commit5<--commit6，我只想把“dev”分支的 commit5 合并到“master”分支，或者把“dev”分支的 commit4 和 commit5 合并到“master”分支。

解：如果是单个分支采用“cherry-pick”命令即可，多个分支采用“rebase”命令。

单个分支：

（1）、执行“git checkout master”命令。

（2）、执行“git cherry-pick commit5”命令，其中的“commit5”为这次提交的 SHA-1 值。

多个分支：

（1）、执行“git checkout dev”命令。

（2）、执行“git checkout -b dev_backup commit5”，其中的“commit5”为这次提交的 SHA-1 值。

（3）、执行“git rebase master”命令。

7、 在“dev”分支修改了文件“1.java”和“2.java”，但是只想把“1.java”提交并 merge 到 “master” 分支。

解：核心思想是部分提交，只提交一部分，其他的储藏，merge 后再回来。

(1)、执行 “git add 1.java” 命令。

(2)、执行 “git stash --keep-index” 命令。

(3)、执行 “git commit -m “comment”” 命令。

(4)、执行 “git checkout master” 命令。

(5)、执行 “git merge dev” 命令。

(6)、执行 “git checkout dev” 命令。

(7)、执行 “git stash pop” 或者 “git stash apply” 命令。