# 新手上路

作为一个新人，仅需要了解

GitHub = “项目版本管理工具” = “能和其他项目成员提交自己所负责的代码到远程的一个项目文件存储的‘服务器’中，并针对相同代码区域的改动（“冲突”）会提供解决方式， GitHub会记录所有提交时的历史记录，并能够及时的回退到某个版本” 作用不止如此。此处仅说明新手最普遍见到的功能

三个概念： “未加入本地版本管理的文件” “已加入本地版本管理的文件（本地仓库）” “远程仓库”

上手命令仅4个

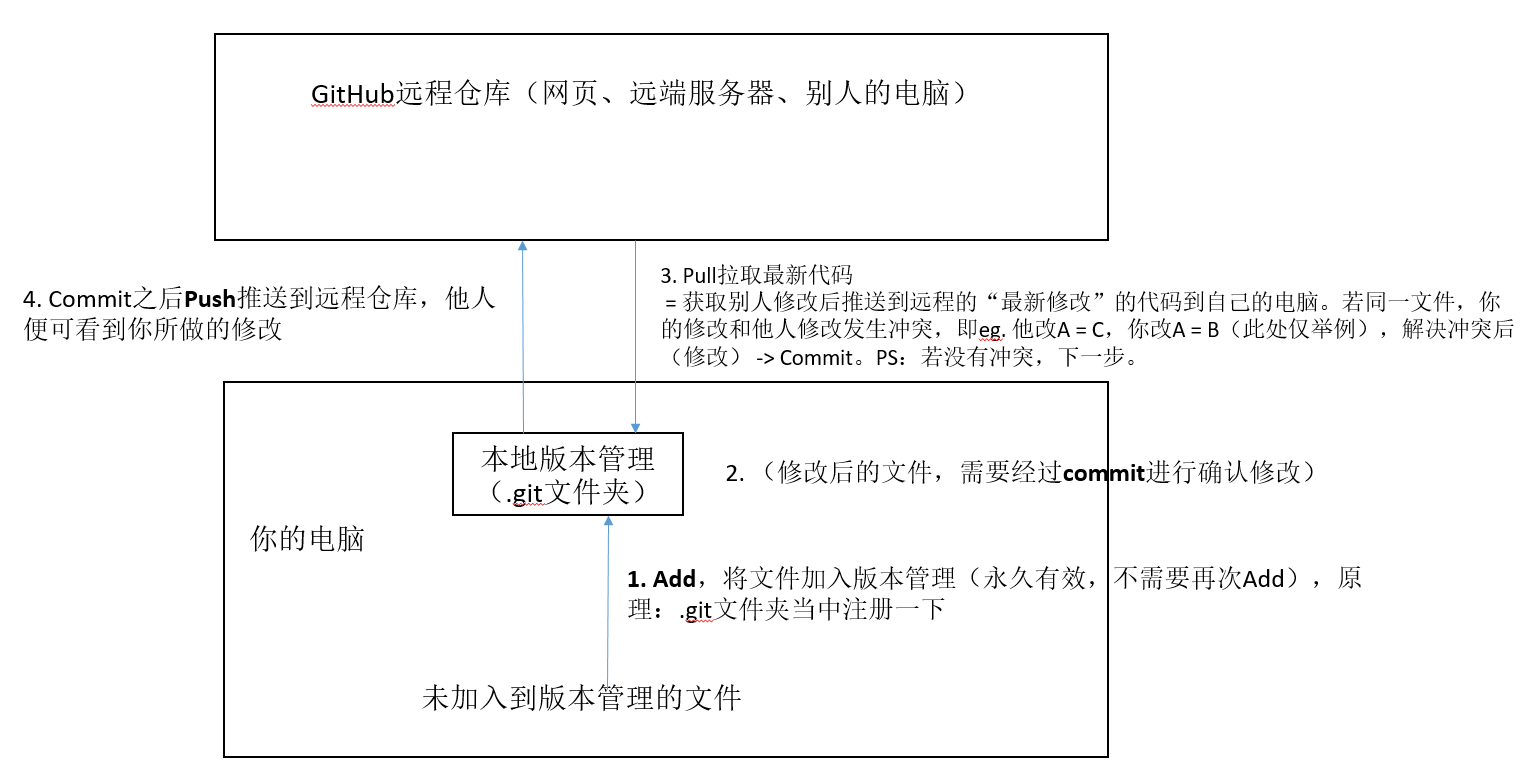
add

commit

push

pull

以下步骤为 提交自己的修改到远程仓库 以及 为 上述 3个概念和 4个命令 解释 大致作用和含义；



外加一个Tortoise Git客户端（可视化界面）。PS：建议日后多使用命令行进行操作，用到Linux时会更方便 PPS：命令行的作用……也可以自己写个脚本（多个命令行拼凑出来就叫脚本了）出来，用于某些特定的项目当中的功能。

.gitignore - 屏蔽某些不需要add进版本管理的文件类型 或 文件夹

PS：实践出真知，第一步：创个Github账号，自己建个远程仓库，将你的代码提交上去。

地址：[地址](#_创建远程仓库：)

官网：

<https://git-scm.com/book/zh/v1/Git-%E5%B7%A5%E5%85%B7-%E5%82%A8%E8%97%8F%EF%BC%88Stashing%EF%BC%89>

## 1. 设置远程主 仓库

git remote add gs http://172.24.132.26:10080/TH/TreasureHunter.git

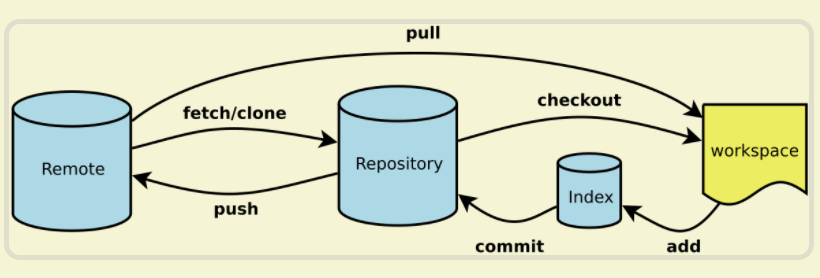
git remote set-url origin <http://172.24.132.26:10080/zhanziqi_971030/TreasureHunter>

PS: git remote -v 可查看对应别名 - 网址

# 应用：GitHub

## 仓库存储管理代码结构图（常用命令清单）：

http://www.ruanyifeng.com/blog/2015/12/git-cheat-sheet.html



## 官网文档

<https://git-scm.com/book/zh/v1/Git-%E5%9F%BA%E7%A1%80-%E5%8F%96%E5%BE%97%E9%A1%B9%E7%9B%AE%E7%9A%84-Git-%E4%BB%93%E5%BA%93>

## 书籍：

Pro Git 第二版

<https://www.kancloud.cn/kancloud/progit/70203>

## 参考实例网址（未涉及远程仓库）：

https://blog.csdn.net/hong10086/article/details/77188308

## 【】可视化界面的处理

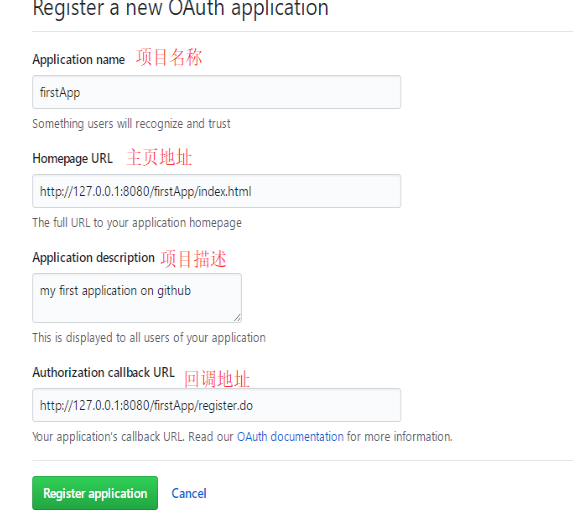
www.360doc.com/content/16/0920/17/1073512\_592302821.shtml

## 【】远程仓库

### 创建远程仓库：

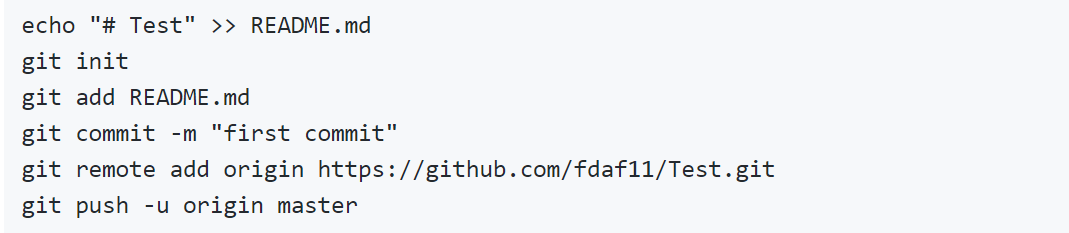
<https://blog.csdn.net/xingkaifan/article/details/81105352>

#### 创建项目（https://github.com/ ）



### 提交至远程仓库：

https://blog.csdn.net/liuweixiao520/article/details/78971221





git remote add origin http...... // 关联仓库地址

git push –u origin (分支名) //将本地仓库推送到远程仓库



### 修改账号密码

<https://blog.csdn.net/weixin_42328925/article/details/80482619>

## 获得历史版本的代码

https://blog.csdn.net/liangfeng093/article/details/79199548

不过没指名怎么用- - ，只有把版本放到了一个新的分支里，没有说明怎么用

## 一个较为系统的笔记

https://www.cnblogs.com/forget406/p/6045213.html

## 各种git基本命令

### git add (file) 加入缓存，并建个索引“commit\_a3.txt”

git add a3.txt

|  |  |
| --- | --- |
| add . | 添加当前所有文件到暂存区 |
| -p | 每个变化都会要求确认？ |
| add [file1] [file2] ... |  |

### git blame [file]：显示指定文件是什么人是在什么时间修改过

### git branch

#### 建立追踪关系的创建分支（本地仓库关联远程仓库）

git branch --track master origin/next

【】应该是这样吧？？？

next是什么？

关联目的是在执行git pull, git push操作时就不需要指定对应的远程分支。直接使用”git pull”就可从远程获取信息

### git brach：分支情况

|  |  |
| --- | --- |
| -a | 查看所有分支（包括远程仓库的，显示为红色） |

[Git](http://lib.csdn.net/base/git" \t "_blank" \o "Git知识库) branch -d XX 提示:  the branch  XXX is not fully merged

原因：XXX分支拥有 没有合并到当前分支的内容

解决方法：使用大写的D 强制删除  [git](http://lib.csdn.net/base/git" \t "_blank" \o "Git知识库) branch -D XXX

另外不能删除当前checkout 的分支

### git checkout 恢复文件/切换分支

仅限在“缓存区” 或 “已提交”的文件

|  |  |
| --- | --- |
| git checkout [file] | 恢复“暂存区”的指定文件到工作区 |
| git checkout [commit] [file] | 恢复某个commit的指定文件到“暂存区和工作区” |
| git checkout . | 恢复暂存区的所有文件到工作区 |

还有切换到某个分支的效果

### git commit： 提交（缓存区中所管理的文件的）一系列操作至本地仓库

|  |  |
| --- | --- |
| -v | 显示所有diff信息 |
| -m [message] | 提交缓存区所有操作至仓库区（本地管理） |
| commit [file1] [file2] ... -m [message] | 提交暂存区指定文件至仓库区 |
| -a | 提交自上次commit之后的变化【】TODO |
| commit --amend -m [message] | 重做上一次提交 |
| commit --amend [file1] [file2] ... | 重做上一次提交，指定文件 |

### git config：修改Git配置

都存在.gitconfig文件下

项目配置（项目目录文件） 和 全局配置（用户主目录？）C盘user目录下？

|  |  |
| --- | --- |
| --list | 显示当前配置 |
| -e [--global] | 编辑配置文件[可修改全局] |
| git config [--global] user.name "[name]" | 修改提交时的用户名 |
| XXXXXXXXXXXXXXXXXX.email "[email.address]" | XXXXXXXXXXX用户邮箱 |
|  |  |

### git diff：显示暂存区和工作区（上一次Commit到本地管理之后与当前暂存区的区别）

### git init：新建本地Git代码库

在当前目录（称为工作区）下创建仓库（会生成一个.git文件夹）

在master分支下（即d/GitStore）创建3个文件夹 abc1.txt abc2.txt abc3.txt

### git log：显示当前分支，版本历史

|  |  |
| --- | --- |
| --stat | 显示commit历史，以及每次Commit对应变更的文件 |
| -S [keyword] | 根据关键字，搜索提交历史 |
| git log [tag] HEAD --pretty=format:%s | 显示某个commit 之后的所有变动个，每个commit占一句 |

### git mv [file-orginal] [file-renamed]：改名，并将此操作加入

### git pull：拉取远程仓库至本地仓库（更新远仓代码至本地，若有冲突需要解决冲突）

git pull origin master:zzq\_store\_lottery

origin: 远程服务器名

将远程master分支 -> 本地的 zzq\_store\_lottery 分支下

https://www.yiibai.com/git/git\_pull.html

### git push：推送本地仓库 至 远程仓库

git push [remote] [branch]

|  |  |
| --- | --- |
| push [remote] -all | 推送所有分支至仓库 |
| --force | 强行推送（即使有冲突） |

### git relog：显示当前分支的最近几次提交

### git remote：对本地对应的远程仓库进行操作

#### 重新设置origin网址

git remote set-url origin http://172.24.132.26:10080/TH/TreasureHunter.git

#### 查看拥有的远程仓库（仅显示变量Origin之类的）

git remote

-v 输出：变量 + 网址

#### 添加一个远程仓库网址（并赋值给一个别名）

git remote add orign XXXX.git

### git reset：未知的撤销操作

|  |  |
| --- | --- |
| git reset [commit] | 重置当前分支的指针为指定commit，同时重置暂存区，工作区不变？ |
| --hard [commit] | 重置当前分支的HEAD为指定commit，重置暂存区和工作区，与指定commit一致 |
| --keep [commit] | 重置当前HEad指针为指定commit，暂存区和工作区不变 |
| --soft [commit] | 回退 至 某一次commit（错误提交时commit的SHA）之前的commit 情况，已提交、更改内容会返回给自己（已更改内容不会改变，仅仅让自己拥有“再一次拉取、提交”的权利），以供修改。【】适用范围：返回上一次自己提交的时候，进行再次拉取，重新解决冲突。 |

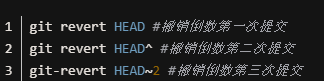
git revert 和 git reset 的区别 【】虽然还是看不懂

<https://blog.csdn.net/qq_33442844/article/details/80506041>

revert会保留此次撤销操作 作为 一次 commit操作，保存在历史记录中

### git revert：撤销操作

|  |  |
| --- | --- |
| git revert [commit] | 新建一个commit，用来撤销某次commit【】 |
| 普通的revert操作 | 回到上一次push后的状态？还是commit之后的状态？ |



撤销之后会出现Swap什么的， O只读查看

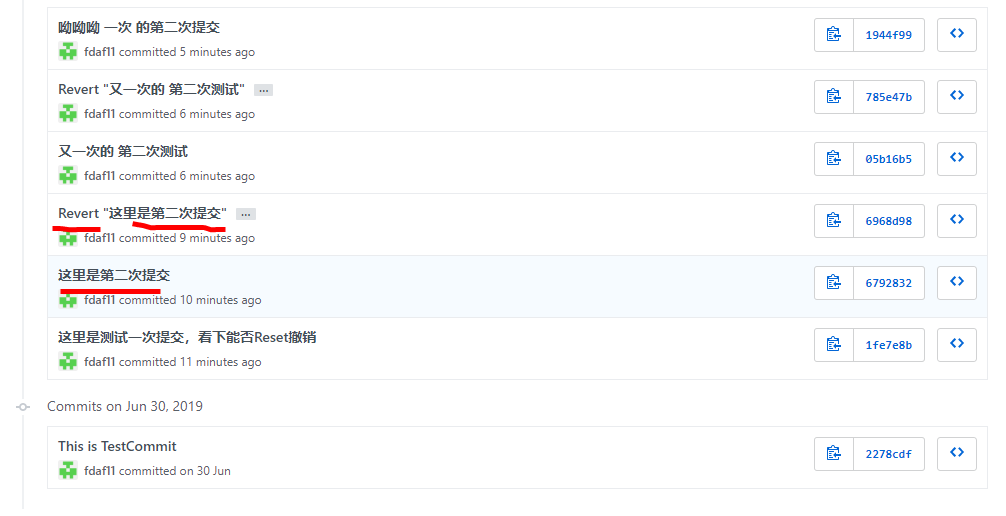
:q即可。

eg. git revert HEAD

此次撤销将返回返回至上上一次提交的状态（之后所做的更改将被删除（可恢复，勿忧））

PS：若想要恢复，可以通过ShowLog选择文件进行Rever TO Commit: XXXXX

最后提交给远端，会有一个Rever: XXXXX 的记录。（至于不保留历史的操作，可见Git命令认证考试里面的内容。）



总结：

这种撤销Commit的操作会保留操作历史，将撤销“作为一个commit”提交给远端. 撤销的文件和修改也都可恢复。

### git rm 删除工作区文件，并将此次删除操作放入 (缓/暂存区)

删除“缓存区”的文件

或 将“已提交”的文件加入删除队列等待commit之类的。

|  |  |
| --- | --- |
| --cache | 停止追踪此文件，会保留此文件（取消add的作用就是） |

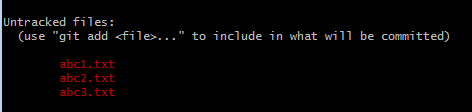
### git reset：重置暂存区指定文件，与上一次Commit保持一致，工作区文件不变（其实没差了）

### git status：查看距上次Commit - push之后更改过的文件（有变更的文件）

d:\Users\Public\Documents\imData\im\971030@nd\Image\9e1262b29a522dfc2272dc4a7f125c18.png

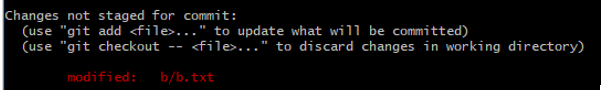
在分支下使用此命令可得到未“备份”的和已“备份的”

Untracked files - 未加入版本管理的文件（需要Add进行追踪）



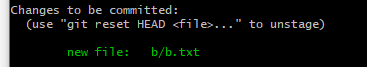
Changes not staged for commit - 已修改，尚未被Commit

红色：modified



Changes to be committed: - 已Add新加入版本管理的文件，尚未Commit

绿色 new File显示



|  |  |
| --- | --- |
| -s | 能够以一种更加紧凑的方式输出更改后的文件  D:\Users\Public\Documents\imData\im\971030@nd\Image\17652bece405157d464f70a8f51ae351.png  右边M - 已修改，未Commit  左边M - 修改后已Commit  两个MM - Commit之后又发生了修改，待Commit  A - 刚加入Add，未Commit  ?? - 未Add |

### git stash：本地缓存当前项目，之后发生修改失误可随时回退

你不想提交进行了一半的工作，否则以后你无法回到这个工作点。但是：现在需要远程仓库信息。解决这个问题的办法就是git stash命令。

官网：

<https://git-scm.com/book/zh/v1/Git-%E5%B7%A5%E5%85%B7-%E5%82%A8%E8%97%8F%EF%BC%88Stashing%EF%BC%89>

<https://www.cnblogs.com/yanghaizhou/p/5269899.html>

原文：

功能做一半，不想在commit的情况下pull代码。

将修改过的代码暂存到栈中，拉取远端代码后，再出栈接收，更改冲突。

<https://www.cnblogs.com/canda/p/7835663.html>

命令行操作：

git stash save "save log"

// 查看保存的List以及相应注释

git stash list

git pull url Develop\_Alpha

git stash apply

解决完冲突

git commit -m “commit message”

git push

1. 在没有提交前，有修改（红色感叹号）的情况，使用stash，会发生什么？老子项目丢了……别TM随便使用不熟的东西。

2. pop是否会将所有暂存列表和删除？那我应该怎么跳转查看对应的某条内容？apply跳转后是否会消失?是否需要重新存储？

### git tag：标签？

|  |  |
| --- | --- |
| git tag | 显示所有标签 |
| $ git tag -l 'v1.4.2.\*' | 列表显示，正则\*代表任意个任意字符 |
| git tag -a v.14 | 增加注解，用于在显示时显示 |
| -m [message] | 标签说明 |
| git show [annotated1] | 可用于查看标签注解对应的标签说明 |
| git tag [tag] | 新建一个Tag于本次Commit |
| git tag [tag] [commit] | 新建一个tag在指定commit |
| -d [tag] | 删除本地tag |
| git push [remote] :refs/tags/[tagName] | 删除远程Tag |
| git push [remote] [tag] | 推送指定Tag至远程 |
| git push [remote] --tags | 提交所有tag |
| git checkout -b [brach] [tag] | 新建一个分支，指向某个Tag |

### git XXX --help

可查看官方文档（离线）

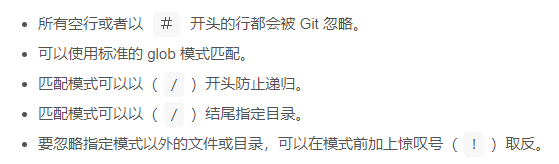
### .gitignore文件：忽略文件；避免将某些格式文件被提交；

不过好像需要提交到远程上面，.gitIgnore才有用

[https://github.com/github/gitignore](https://github.com/github/gitignore" \t "_blank)

eg. 日志文件、缓存文件、项目生成的临时文件

应该和正则有关？



# 后面跟着注释

空文件夹自动忽略。

glob模式？

匹配模式是啥子？正则？还是相对路径？

/a - 相对路径有效

|  |  |
| --- | --- |
| \*.[oa] | - 忽略所有以 .o 或 .a 为结尾的文件 |
| \*~ | 忽略所有以 ~ 为结尾的文件 |
| !/a | 忽略除了a文件夹下内容的其他所有文件 |
| # 今天是个好日子 | # - 注释 |
| /d | 忽略根目录下d文件夹中所有内容 |
| doc/\*\*/\*.pdf | 忽略doc目录下所有.pdf文件，包含子目录 |
| doc/\*.pdf | 忽略doc目录下（单层目录）中的.pdf文件，不包含子目录（子目录中的.pdf会被加入到版本） |
|  |  |
|  |  |
|  |  |

本地忽略：

D:\Users\Public\Documents\imData\im\971030@nd\Image\f8ae1554d3ece0bc7905797b834baae7.png

## 案例

### 撤销距上一次Commmit操作 之后的所有操作（增删改）

<https://stackoverflow.com/questions/4630312/reset-all-changes-after-last-commit-in-git>

git reset HEAD --hard

You can remove ignored and untracked files and directories // 移除 所有.ignored无视文件、未跟踪文件以及文件夹。但无法回退对那些“无视”文件的修改操作

git clean -fdx

You can also set clean.requireForce to false:

git config --global --add clean.requireForce false

to avoid using -f (--force) when you use git clean.

### git reset --hard的文件丢失（若有过git add 则可找回）

<https://blog.csdn.net/qq_31608451/article/details/78342365>

|  |
| --- |
| 如果你没有commit你的本地修改（甚至于你都没有通过git add追踪过这些文件，当他们被删除，git reset –hard对于这些没有被commit过也没有git add过的修改来说就是具有毁灭性的，destructive！！）  but，如果你幸运的是曾经通过git add命令追踪过这些文件，只是没有commit它们而已！那么试试git fsck –lost-found这个命令吧！然后你就可以在本地项目文件中路径为.git/lost-found/other（楼主亲自试验就是这个路径）中找到它们！！并且呢，这里面包含了所有的没有被commit（指定到某次commit）的文件，甚至可能还包括你每次git add的版本（version一词实在不知道在这里怎么翻译，姑且就认为是版本吧）！ |

## remote远程仓库处理

### 增加一个远程仓库地址

git remote add game-server XXX网址.git

## GitLab

### Fork项目流程

https://www.jianshu.com/p/4f4c04c6e98b

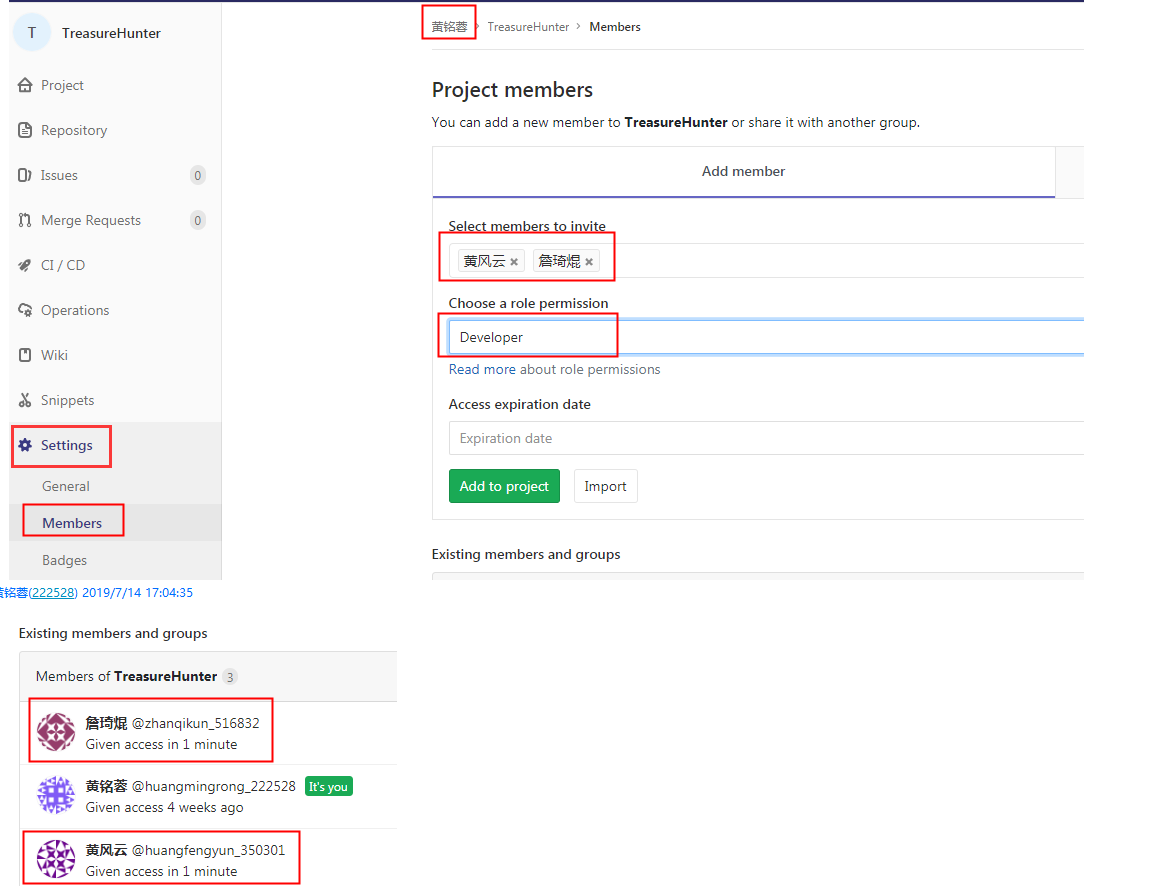
### 提供给合并者的修改权限

1. 打开自己的项目页面； Settings - Members

2. SelectMembersToInvite增加合并者（仅支持英文名）

3. 设置为Developer

4. Add To Project



### Fork工作项目拉取、解决冲突

<https://blog.csdn.net/jack22001/article/details/87946037>

|  |
| --- |
| 1. 增加远程仓库XXXX网址映射给变量gs   git remote add gs 网址.git  gs - 对远程仓库自定义命名：git remote可查看列表  2. 将远程gs仓库的master分支拉取到本地  git pull gs master --allow-unrelated-histories  gs - 刚刚定义的远程网址映射变量  master - 远程的分支  // 以下这两个用不上  //git branch --set-upstream-to=gs/master master  // git push |

Fork工作项目拉取、解决冲突

https://blog.csdn.net/jack22001/article/details/87946037

添加远程对应仓库网址，设置别名

git remote add gs gs网址.git

gs - 对远程仓库自定义命名：git remote可查看列表

从远程仓库gs的master分支上进行拉取

git pull gs master --allow-unrelated-histories

## 【知识点】

### SSH—key生成

https://blog.csdn.net/zdx1515888659/article/details/72954000

### 使用bat处理git命令，实现快速从公司Master pull内容

往环境变量：PATH中加入 git安装目录/bin

即可定义BAT

eg.

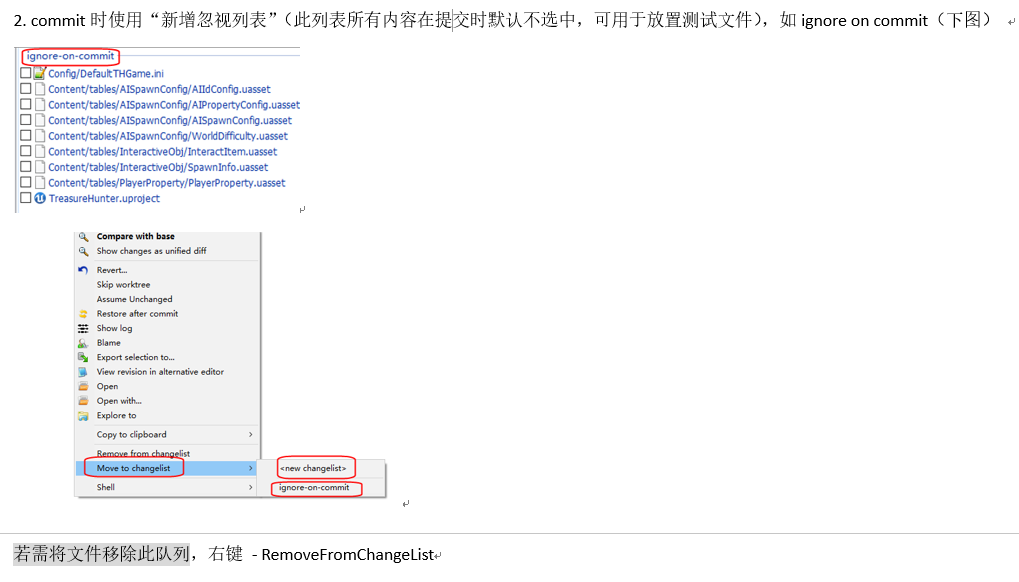
|  |
| --- |
| cmd /k git remote |

拉取公司gitlab master 至本分支

|  |
| --- |
| git remote add gs <http://172.23.8.138:10080/TreasureHunter/TreasureHunter/TreasureHunter_gameserver.git>  cmd /k git pull gs master --allow-unrelated-histories |

### 若在解决冲突前Commit，合并时提交不会显示文件

### 新增忽视列表（Commit时依旧可见，仅默认不选中，并加以分类归类）



## 分支

### 创建分支

### 查看远程分支

d:\Users\Public\Documents\imData\im\971030@nd\Image\d3d4ec95265806e3b8e3c4b4213d2e7a.png

### 查看尚未合并的分支

git branch --no-merged 查看尚未合并的工作：

### 删除分支

git branch –d XXXX

没有合并的分支被删除会报错

### 强行删除

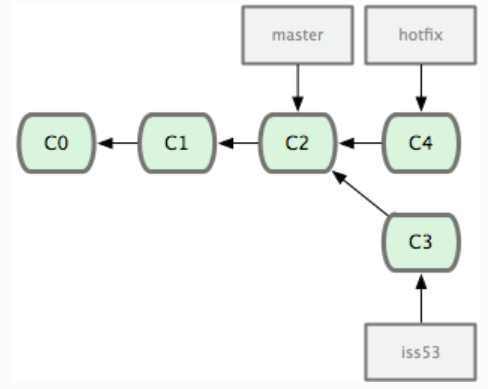
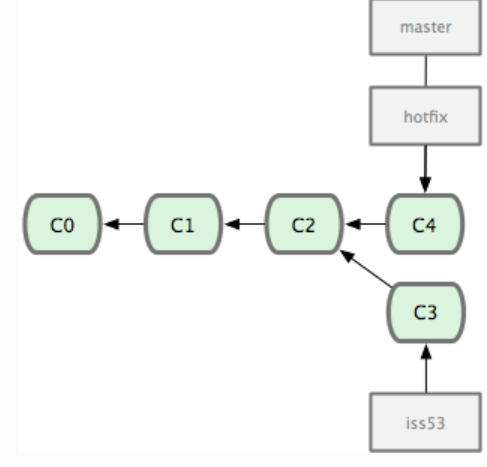
git branch –D XXXX

### git merge 合并分支（在测试过当前分支有效、合理、正确时，将master并入此）

git merge hotfix （将当前分支[master] 合并[移动] 到 目标分支[hotfix]）

d:\Users\Public\Documents\imData\im\971030@nd\Image\ed08fdb4fc8e6b68b4118d5d0b17108b.png

过程：

原： 合并后：

合并冲突常见格式：



修改之后add

commit提交

### 更改本地和远程分支

|  |
| --- |
| git branch -m old\_branch new\_branch # Rename branch locally  git push origin :old\_branch # Delete the old branch  git push --set-upstream origin new\_branch # Push the new branch, set local branch to track the new remote，新名字得和本地新名字对应。 |

### 【】复用，git提交内容至分支

创建分支

git branch zzq\_friend

git checkout zzq\_friend

## git mergetool 调用冲突解决图形化工具

## 冲突情况1：不出现冲突的本地修改（无commit）

git对比的是两个commit之后的版本，若你没有进行commit，那么即使本地master被你更改了，之后也不会出现pull最新的master分支到本地出现冲突的情况

l

## 【】转换分支

案例：在修改完之前的代码后，发现处于的代码是master，切换不到我的分支了。此时报错：Please commit your changes or stash them before you switch branches.

### 创建一个新的文件夹需要对远程仓库进行互相关联。

git pull进行拉取（关联分支）。

## 【】撤销或修改commit操作

### git reset --hard请慎用，尽量不动代码，多提交几次也少用着东西。（HEAD指针退回XX版本，并修改本地代码）

<https://www.cnblogs.com/hope-markup/p/6683522.html>

可用git fsck -–lost-found

来进行找回最近的一次提交

**案例：**

之前用提交commit时，不满意注释内容……想退回重改，退到了本次提交的上一版本。然后……上一版本是10天之前的代码……又没做备份……GG。

### git撤销commit，但未git push的命令

### 撤销最后一次暂存

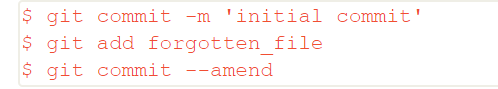
获取项目更新（暂时解决冲突的拉取方式）  


<https://blog.csdn.net/jtracydy/article/details/75665638>

这个网站关于Stash的内容虽然有点白，但是能看。

### 修改最后一次提交

漏掉文件没有提交，补充，再次提交（只产生一次提交记录）



### 撤销add操作

## SVN操作

update 是从SVN服务器上把最新版本下载到本地来；

commit 是将本地做过的改动（修改、新增、删除、改名、移动等）上传更新到SVN服务器；

add 是将本地新增文件或文件夹标注为受SVN控制，但还未上传到服务器，一般后面还会跟一个commit操作。

## 错误：

### git add : warning: LF will be replaced by CRLF

<https://www.jianshu.com/p/450cd21b36a4>

应该是换行符的问题。

git branch –vv 限时分支映射信息

git remote –vv 显示对应网址？push 和 fetch

### ! [rejected] master -> master (non-fast-forward)

### fatal: The current branch master has no upstream branch.

To push the current branch and set the remote as upstream, use

git push --set-upstream origin master

<https://blog.csdn.net/benben_2015/article/details/78803753>

// 查看所有分支（包括远程）

git brach -a

原因解析；

若绿色（本地） 和 红色（远程）之间名字不同可能出现此错误

解决：

git push -u origin master

### git clone error:RPC failed; curl 18 transfer closed with outstanding read data remaining

https://blog.csdn.net/PHY1161460191/article/details/88656003

--depth=1 只能拉取最近一次Commit内容。并不能把整个版本库拉取下来，所以，只需要改第一个就好，depth = 1什么的不要尝试

若使用了depth，则使用

https://segmentfault.com/q/1010000000409170

vim .git/config

按照如下示例修改：

改成这个即可：

fetch = +refs/heads/\*:refs/remotes/origin/\*

# 遇到的情况

## 发现自己解决错了冲突时，应该做的处理（eg. 文件格式不同，用了自己的）

更新至 公共库 最新

git reset --soft 回退 至showlog上一次commit（错误提交时commit的SHA）之前的commit版本号

将上一次commit的内容（提交的内容），暂存。

回退至上上一次 “自己提交”的commit

拉取

弹出，解决冲突。

提交、push即可。

删除远端commit

git reset --soft [commit]

git stash save ""

git push -f

但是只要标记了已解决，就会以“已解决的版本为主，不会再报冲突了，绝望。”

## 【1】Git revert对于撤销提交，能在别人提交之后也能起作用么？

# 例题

<http://www.360doc.com/content/12/0424/18/1016783_206221262.shtml>

答案（翻墙）：

<http://loveky2012.blogspot.com/2012/08/git-knowledge-test-answer.html>

## 题目

1. 如果提示提交内容为空、不能提交，则最为合适的处理方式是：\_\_\_A\_\_

a) 执行 git status 查看状态，再执行 git add 命令选择要提交的文件，然后提交。

b) 执行 git commit --allow-empty ，允许空提交。

c) 执行 git commit -a ，提交所有改动。

d) 执行 git commit --amend 进行修补提交。

1. 如果把项目中文件 hello.c 的内容破坏了，如何使其还原至原始版本？ \_\_\_B\_\_

a) git reset -- hello.c

b) git checkout HEAD -- hello.c

c) git revert hello.c

d) git update hello.c

1. 修改的文档 meeting.doc 尚未提交，因为错误地执行了 git reset --hard 导致数据丢失。丢失的数据能找回么？ \_\_\_C\_\_

a) 不能。执行硬重置使工作区文件被覆盖，导致数据丢失无法找回。

b) 能。可以通过 git checkout HEAD@{1} -- meeting.doc 找回。

c) 不确定。如果在重置前执行了 git add 命令将 meeting.doc 加入了暂存区，则可以在对象库中处于悬空状态的文件中找到。

d) 不能。因为未提交所以无法找回。

1. 仅将工作区中修改的文件添加到暂存区（新增文件不添加），以备提交，用什么命令标记最快？ \_\_D\_\_\_

a) git add -A

b) git add -p

c) git add -i

d) git add -u

1. 下面哪一个命令 不 会改变提交历史？ B\_\_\_\_\_

a) git reset --hard HEAD~1

b) git checkout HEAD^^ .

c) git rebase -i HEAD^^

d) git commit --amend

6 我使用和其他人不一样的IDE软件，总是在目录下生成以 .xx 为后缀的临时文件。如何避免由于自己的误操作导致此类文件被添加到版本库中呢？ \_\_C\_\_\_

a) 执行 git clean -f 删除临时性文件。

b) 向版本库中添加一个 .gitignore 文件，其中包含一条内容为 \*.xx 的记录。

c) 在文件 .git/info/exclude 中添加一条内容为 \*.xx 的记录。

d) 更换另外一款IDE软件。

7 项目跨平台导致文件中的换行符不一致。其中有 Linux 格式换行符（0A），也有DOS格式换行符（0D 0A）。要如何避免此类情况呢？ \_\_\_\_D\_

a) 修改 /etc/gitattributes 文件，在其中包含一条内容为 \* text=auto 的设置。

b) 执行命令 git config --global core.autocrlf true 。

c) 执行命令 git config --global core.autocrlf input 。

d) 向版本库中添加一个 .gitattributes 文件，在其中包含一条内容为 \* text=auto 的设置。

8 下列对于版本库授权说法正确的是：\_\_A\_\_\_

a) 可以为分支或路径设置不同的写入权限，但不能设置不同的读取权限。

b) 除管理员外，版本库的创建者都可以为自己创建的版本库授权。

c) 只要通过授权后，便不能限制所推送的提交的署名作者，可以是任何人。

d) 如果没有向版本库的写入权限，就一定没有读取权限。

9 取消服务器版本库中ID为 a2387 的提交，而且不能引起历史提交的变更，用什么操作？ \_\_C\_\_\_

a) git rebase -i a2387^

b) git checkout a2387^ -- .

c) git revert a2387

d) git reset --hard a2387^

10 从版本库中的历史提交中彻底移除ID为 a2387 的提交，用什么操作？ \_\_\_D\_\_

a) git reset --hard a2387^

b) git checkout a2387^ -- .

c) git revert a2387

d) git rebase --onto a2387^ a2387 HEAD

11 所有改动的文件都已加入暂存区，若希望将其中的 other.py 文件下次再提交，如何操作？ \_A\_\_\_\_

a) git reset -- other.py

b) git checkout -- other.py

c) git checkout HEAD other.py

d) git reset --hard -- other.py

12 若产品的版本号显示为 1.7.10.rc0-33-g9678d-dirty ，可以判断出此版本号是如何生成的么？ \_\_\_B\_\_

a) git tag

b) git describe --tags --always --dirty

c) git name-rev HEAD

d) git --version

13 关于 git clone 下面说法 错误 的是：\_\_\_D\_\_

a) 克隆时所有分支均被克隆，但只有HEAD指向的分支被检出。

b) 可以通过 git clone --single-branch 命令实现只克隆一个分支。

c) 克隆出的工作区中执行 git log 、 git status 、 git checkout 、 git commit 等操作不会去访问远程版本库。

d) 克隆时只有远程版本库HEAD指向的分支被克隆。

14 关于删除分支 XX ，下列说法正确的是： \_\_A\_\_\_

a) 执行 git push origin :XX 来删除远程版本库的 XX 分支。

b) 执行 git branch -D XX 删除分支，总是能成功。

c) 远程版本库删除的分支，在执行 git fetch 时本地分支自动删除。

d) 本地删除的分支，执行 git push 时，远程分支亦自动删除。

15 下面的操作中哪一个不能确认维护分支 maint 上所有的 bugfix 提交均已合并至当前分支 master 中。 \_\_\_B\_\_

a) git rev-list ..maint 的输出为空。

b) 在 maint 分支成功地执行 git merge master 。

c) git log ..maint 的输出为空。

d) 新版本发布，在 maint 分支执行 git merge --ff-only master 成功。

16 一个图片文件 logo.png 冲突了，如何取出他人的版本。 \_C\_\_\_\_

a) git show :1:./logo.png > logo.png-theirs

b) git show :2:./logo.png > logo.png-theirs

c) git show :3:./logo.png > logo.png-theirs

d) git show :0:./logo.png > logo.png-theirs

17 工作在特性分支，常常因为执行 git push 默认推送所有本地和远程共有分支，导致非当前分支报告 non-fast-forward 错误。如果设置只推送当前分支可避免此类问题。下面操作正确的是：\_\_\_\_A\_

a) git config --global push.default upstream

b) git config --global pull.rebase true

c) git config --global receive.denyDeletes true

d) git config --global pager.status true

18 关于对象库（.git/objects）说法 错误 的是：\_\_\_B\_\_

a) 两个内容相同文件名不同的文件，在对象库中仅有一个拷贝。

b) 对象库执行 git gc 操作后，reflog 会被清空导致其中记录的未跟踪提交及指向的文件被丢弃。

c) 删除文件后，再通过添加相同文件找回，不会造成版本库的冗余。

d) 对象库并非一直保持最优存储，而是通过周期性地执行 git gc 优化版本库。

19 关于子模组 错误 的说法是：\_\_\_C\_\_

a) 克隆父版本库，默认不会克隆子模组版本库。

b) 子模组可以嵌套。执行 git submodule update --recursive 可对嵌套子模组进行更新。

c) 子模组和父版本库的新提交，要先推送父版本库，后推送子模组。

d) 子模组检出处于分离头指针状态（gitlink的指向），在子模组中工作需要手动切换分支。

20 当一个提交说明显示为 souce code refactor (fix #529) ，下面哪个说法是正确的？ \_\_D\_\_\_

a) 这个提交只是代码重构，并未修复任何东西，因此没有改变版本库的提交历史。

b) 这个提交修正了第529号提交，没有改变版本库的提交历史。

c) 这个提交撤销了第529号提交，改变了版本库的提交历史。

d) 这个提交和项目的缺陷跟踪平台（如Redmine）关联，并会更新相关问题的状态。

## 解析

解答如下：   
  
题目1  
这种情况往往是修改了工作区的文件，但是没有运行git add来将修改的文件添加到index中。正确方法为git add需要commit的文件，再执行git commit。  
git commit --allow-empty只会创建一个空提交，工作区的内容仍旧未提交至git repo里。  
git commit -a 会直接提交所有修改/删除的文件，通常不推荐此操作，因为很可能会将尚未ready的改变一并提交至git仓库。  
git commit --amend 会进行一次修补提交，即不会产生一个新的commit，而是将此次commit的change直接apply到最近一次提交，提交相关信息不变。  
  
题目2  
git reset -- hello.c 只能恢复index中的文件，工作区的文件并不会被重置。  
git checkout HEAD -- hello.c 同时恢复index与工作区中的文件。  
git revert hello.c 非法操作  
git update hello.c 非法操作  
  
题目3  
每次执行git add时，git会将修改过的文件以一个新object的形式加入到git仓库中，因此，如果此时执行git reset --hard，则这些新加入的object已经不被任何commit引用（包括index），此时运行git fsck会发现，每一次git add操作添加的文件版本都会显示为一个dangling object。  
  
题目4  
git add -A 添加所有文件，包括新添加的文件。  
git add -p 通过逐一检查工作区文件与index文件diff输出来决定是否添加文件。  
git add -i 交互式的选择要git add的文件。  
git add -u 添加所有修改的文件，不包括新添加的文件。  
  
题目5  
git reset --hard 会重置当前branch  
git checkout 只会改变HEAD，不会影响当前branch  
git rebase 会嫁接连续的rev list到new base，进而改变历史  
git commit --amend 会修改最近一次commit  
  
题目6  
这里的主要区别在于使用.gitignore还是.git/info/exclude。两者的主要区别在于.gitignore会被push到server上与其他人共享，而.git/info/exclude完全工作在本地。  
  
题目7  
git config --global core.autocrlf true 版本库中使用LF，而检出时始终使用CRLF。  
git config --global core.autocrlf input 检出时始终使用LF。  
在.gitattributes中设置\* text=auto 如果core.eol未设置，则根据操作系统不同，在windows上使用CRLF，在linux上使用LF。  
  
题目8  
gitolite的写授权可以针对文件路径，分支。但是只有全局的读权限。  
  
题目9  
git revert 不会改变现有历史，而是生成一个新的commit来反转指定commit中的change。  
  
题目10  
git reset --hard 适用于要移除的commit为当前branch上最新的commit，否则会移除其他正确的commit。  
git checkout 只会改变HEAD头，不会对branch历史修改。  
git revert 可以撤销某个commit的change，但是不会在历史中删除该commit。  
  
题目11  
git reset -- filename 用于使用HEAD中的filename覆盖index中的版本。  
  
题目12  
参考git help describe  
  
题目13  
git clone会fetch所有远程branch到本地，但只有远程仓库HEAD指向的branch会在本地checkout。  
  
题目14  
使用git push origin :XX可以删除远程仓库中的XX分支。  
git branch -D xx 无法删除当前所在分支。  
默认执行git fetch，只会fetch远程仓库存在的分支。  
默认执行git push，只会push在远程仓库存在对应分支的本地分支。  
关于git pull/git push的默认操作，请参考我的另一篇文章 http://loveky2012.blogspot.com/2012/08/default-behaviour-of-git-pull-and-git-push.html  
  
题目15  
git rev-list master..maint 会输出所有出现在maint历史中但未出现再master历史中的commit列表。如果列表为空则表示master分支已经包含所有maint上的bugfix。  
git log master..maint 原理同上。  
在maint分支上执行git merge --ff-only master成功表示在master分支的历史中已经包含了maint分支上的change，因此此时merge master分支可以直接移动maint分支指向的commit进行快进合并。  
  
题目16  
当目录merge出现confilct时，可以使用git ls-files -s查看当前staged的object的状态。

[?](http://loveky2012.blogspot.com/2012/08/git-knowledge-test-answer.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | 263 gewang@LM-SHC-00355679@13:55:04:~/test/show  => git merge master  Auto-merging test  CONFLICT (content): Merge conflict in test  Automatic merge failed; fix conflicts and then commit the result.  264 gewang@LM-SHC-00355679@13:55:07:~/test/show  => git ls-files -s  100644 e69de29bb2d1d6434b8b29ae775ad8c2e48c5391 0 aaa  100644 388ddf68417917d4c0e0faa74432c1c2e74d572e 1 test  100644 733205e05f85d7b51b96cbef2938d61bf304798b 2 test  100644 23bc1c902ffde28525f215cd2bfc44b6f4222709 3 test |

可以看到test文件出现了3个stage的状态，分别为1，2，3。而没有出现冲突的文件的stage状态为0.  
针对存在confilict的文件可以：

[?](http://loveky2012.blogspot.com/2012/08/git-knowledge-test-answer.html)

|  |  |
| --- | --- |
| 1  2  3 | git show :1:test  # 查看冲突文件的common ancester版本  git show :2:test  # 查看本地branch中的版本  git show :3:test  # 查看远程branch中的版本 |

题目17  
git config --global push.default upstream 不带参数push是仅推送当前branch  
git config --global pull.rebase true 在执行git pull时，将远程分支获取到本地后使用rebase，而不是merge来将远程change应用到本地分支。  
git config --global receive.denyDeletes true 设置此选项则git-receive-pack不会响应分支删除操作。  
  
题目18  
git gc会调用git reflog expire --all，而git reflog expire默认只会删除90天前的记录。如果想即使生效，则应该使用git reflog expire --expire=now --all。  
  
题目19  
详情请参考submodule文档。  
  
题目20  
用过JIRA，Redmine之类工具的都懂得。。。

## Cherry-pick作用

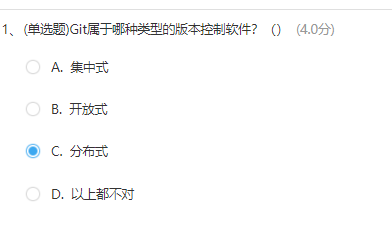
单独挑选某个功能进行合并……针对Commit

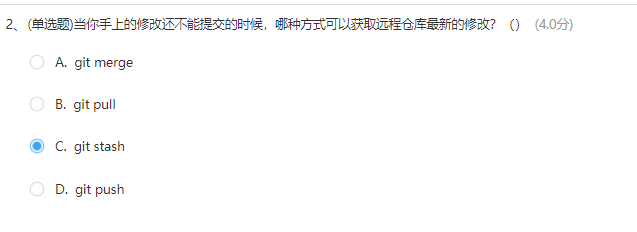
PS:Merge是针对某个文件以及其之后的更改进行处理。

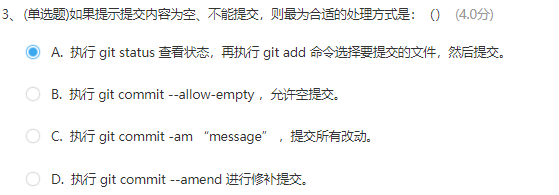
<https://blog.csdn.net/qq_32452623/article/details/79449534>

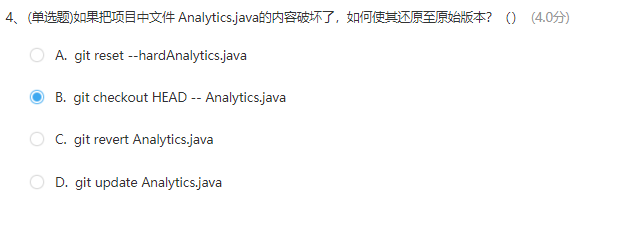
可以将某次其他分支上的Commit 合并到XX分支

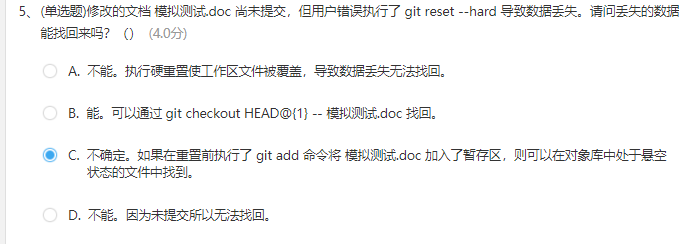
## 七七八八的诡异题目

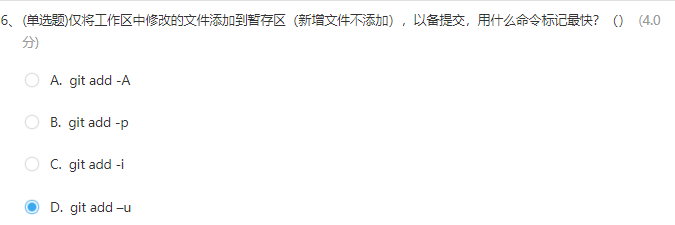


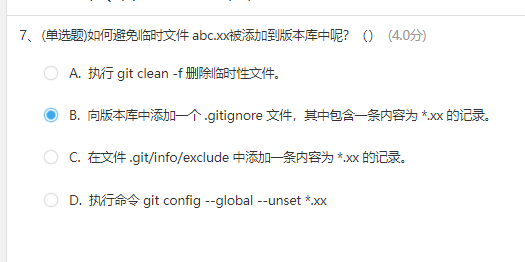




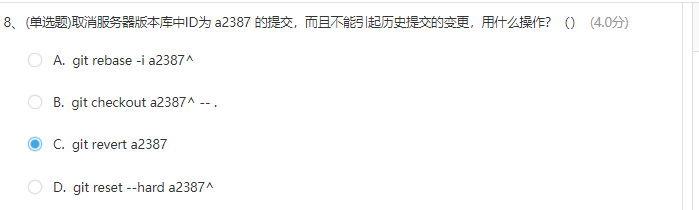


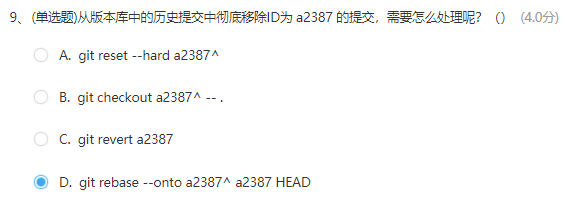


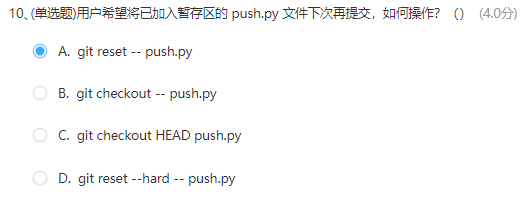


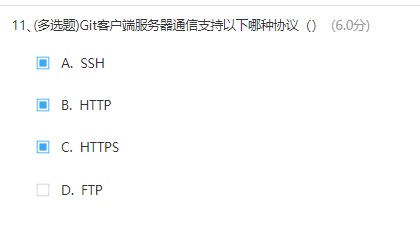


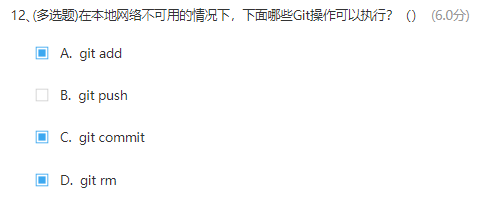
本题选了C，似乎没过。





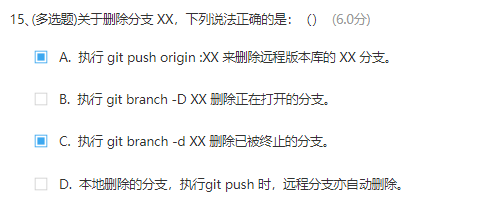




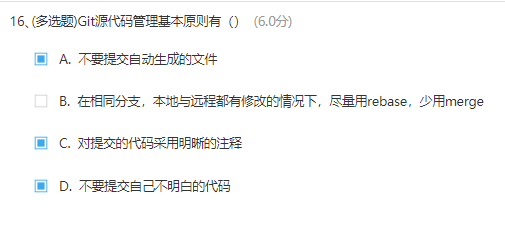




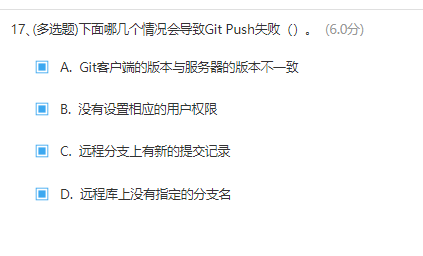




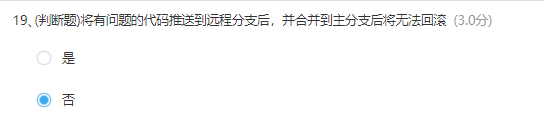
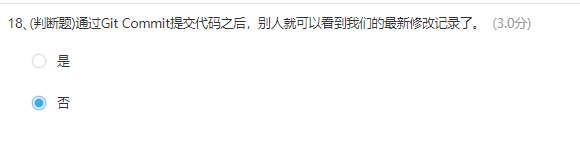
【】



【】



【】



【】

