**[git 常用命令(含删除文件)](http://www.cnblogs.com/cosiray/archive/2012/04/10/2440994.html)**

**Git常用操作命令收集：**

**1) 远程仓库相关命令**

检出仓库：$ **git** clone **git**://github.com/jquery/jquery.**git**

查看远程仓库：$ **git** remote -v

添加远程仓库：$ **git** remote add [name] [url]

删除远程仓库：$ **git** remote rm [name]

修改远程仓库：$ **git** remote set-url --**push**[name][newUrl]

拉取远程仓库：$ **git** pull [remoteName] [localBranchName]

推送远程仓库：$ **git push** [remoteName] [localBranchName]

**2）分支(branch)操作相关命令**

查看本地分支：$ **git** branch

查看远程分支：$ **git** branch -r

创建本地分支：$ **git** branch [name] ----注意新分支创建后不会自动切换为当前分支

切换分支：$ **git** checkout [name]

创建新分支并立即切换到新分支：$ **git** checkout -b [name]

删除分支：$ **git** branch -d [name] ---- -d选项只能删除已经参与了合并的分支，对于未有合并的分支是无法删除的。如果想强制删除一个分支，可以使用-D选项

合并分支：$ **git** merge [name] ----将名称为[name]的分支与当前分支合并

创建远程分支(本地分支**push**到远程)：$ **git push** origin [name]

删除远程分支：$ **git push** origin :heads/[name]

我从master分支创建了一个issue5560分支，做了一些修改后，使用**git push** origin master提交，但是显示的结果却是'Everything up-to-date'，发生问题的原因是**git push** origin master 在没有track远程分支的本地分支中默认提交的master分支，因为master分支默认指向了origin master 分支，这里要使用**git push** origin issue5560：master 就可以把issue5560推送到远程的master分支了。

    如果想把本地的某个分支test提交到远程仓库，并作为远程仓库的master分支，或者作为另外一个名叫test的分支，那么可以这么做。

$ **git push** origin test:master         // 提交本地test分支作为远程的master分支 //好像只写这一句，远程的github就会自动创建一个test分支

$ **git push** origin test:test              // 提交本地test分支作为远程的test分支

如果想删除远程的分支呢？类似于上面，如果:左边的分支为空，那么将删除:右边的远程的分支。

$ **git push** origin :test              // 刚提交到远程的test将被删除，但是本地还会保存的，不用担心

**3）版本(tag)操作相关命令**

查看版本：$ **git** tag

创建版本：$ **git** tag [name]

删除版本：$ **git** tag -d [name]

查看远程版本：$ **git** tag -r

创建远程版本(本地版本**push**到远程)：$ **git push** origin [name]

删除远程版本：$ **git push** origin :refs/tags/[name]

**4) 子模块(submodule)相关操作命令**

添加子模块：$ **git** submodule add [url] [path]

如：$ **git** submodule add **git**://github.com/soberh/ui-libs.**git** src/main/webapp/ui-libs

初始化子模块：$ **git** submodule init ----只在首次检出仓库时运行一次就行

更新子模块：$ **git** submodule update ----每次更新或切换分支后都需要运行一下

删除子模块：（分4步走哦）

1)$ **git** rm --cached [path]

2) 编辑“.gitmodules”文件，将子模块的相关配置节点删除掉

3) 编辑“.**git**/config”文件，将子模块的相关配置节点删除掉

4) 手动删除子模块残留的目录

**5）忽略一些文件、文件夹不提交**

在仓库根目录下创建名称为“.gitignore”的文件，写入不需要的文件夹名或文件，每个元素占一行即可，如

target

bin

\*.db

**git操作-删除文件**

日期:2012-05-20 来源: bg090721 分享至:

[git](http://www.linuxso.com/command/git.html)删除文件

[rm](http://www.linuxso.com/command/rm.html) a[dd](http://www.linuxso.com/command/dd.html)2.txt

git rm add2.txt

git commit -m "rm test"

git push web

-----------at server

[cd](http://www.linuxso.com/command/cd.html) /var/www/foo.git;[su](http://www.linuxso.com/command/su.html)do git up[date](http://www.linuxso.com/command/date.html)-server-info

------------检查删除效果

cd;rm foo3 -rf;git clone http://[某ip]/foo.git foo3

------------更新已经存在的local code

cd;cd foo2

git remote add web [某user]@[某ip]:/var/www/foo.git/

git pull web master

Git是分布式版本控制系统，

创建新的git仓库：创建新文件夹，打开然后执行git init来创建新的git仓库

检出仓库：执行git clone /project/soft创建一个本地仓库的克隆版本，执行git clone username@host://project/soft检出远程服务器上的仓库

本地仓库工作流：工作目录（实际文件）、暂存区（缓存区域临时保存改动的代码）、head（指向最后一次提交的结果）

添加到暂存区：git add <filename> 或git add \*

提交到本地仓库的head中：实际提交改动git commit -m “代码提交信息”，这时改动已提交到head，但还未到远端仓库

提交到远端仓库：git push origin master（master可以换成想要推送的任何分支）

未克隆现有仓库但想将本地仓库连接到某个远程服务器，则使用git remote add origin <server>，这样就能将改动推送到所添加的服务器上去了

分支：用来将特性开发绝缘开来，在创建仓库是master为默认的分支，在其他分支上开发完成后合并到主分支上

创建并切换分支：git checkout -b master（master可换成分支名）

切换到主分支：git checkout master

把创建的分支删掉：git branch -d master(master可换成分支名)

把分支推送到远端仓库(若不推送到远端则其他git用户看不到此分支)：git push origin <branch>

更新本地仓库到最新改动：git pull

将其他分支合并到当前分支：git merge <branch>

合并之后将其标记为合并成功：git add <filename>

查看差异：git diff <source\_branch> <target\_branch>

为软件发布创建标签：git tag 1.0.0 1ldedl63ff（要标记的提交id的前10位字符，也可以少一点，只要指向具有唯一性）

获取提交id：git log

替换本地改动：git checkout -- <filename>，此命令会使用head中的最新内容替换掉你的工作目录中的文件，已添加到暂存区的改动以及新文件都不会受到影响

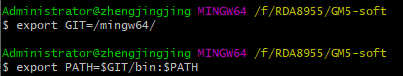
丢弃本地所有改动与提交，可以到服务器获取最新的版本历史并将本地主分支指向它：git fetch origin / git reset --hard origin/master

工作区：就是在电脑里能看到的目录

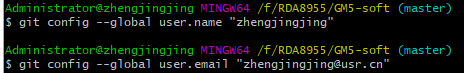
暂存区一般存放在.git目录下的index文件中(。git/index)，所以暂存区有时又被叫成索引

版本库：工作区有个隐藏目录.git，为git的版本库

由于始终配置不了git-bash中直接使用git命令，所以只能每次都开始就加上如下操作（远程桌面使用git时，先找到git目录，打开git-bash应用程序。然后在远程电脑上找到本地需要使用到的目录，使用“$ cd \\tsclient/f/RDA8955/GM5-soft”到对应的目录下进行以下操作即可）



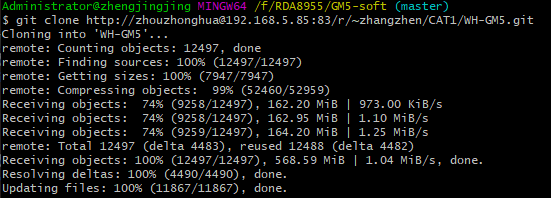
配置自己的用户名和邮箱



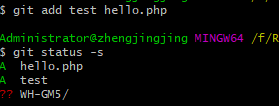
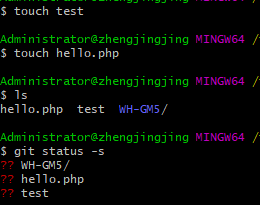
先使用git init创建一个本地git仓库，执行完后会生成一个.git目录，该目录包含了资源的所有元数据，跟踪管理版本库，其他项目目录保持不变(不像svn会在每个子目录生成.svn目录，git只在仓库会生成.git目录)

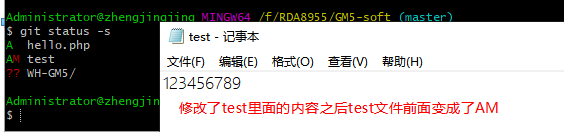


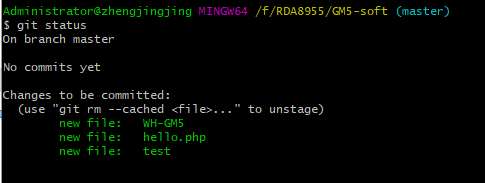
git clone：将远端仓库的项目检出到本地仓库中，默认情况下git会根据所指示的项目的名称创建本地项目目录，通常url最后一个 / 之后的项目名称，若想改名称则在该命令后加上要改的名称



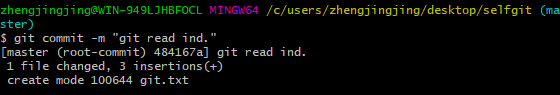
git add：将文件添加到缓存，git status命令用于查看项目的当前状态，git add添加文件。AM是这个文件在将它添加到缓存之后又有改动，改动后在执行git add将其添加到缓存中：即当要将修改包含到即将提交的快照里的时候需要执行git add。git status查看上次提交之后是否有修改（加了 -s 表示以简短的结果输出）





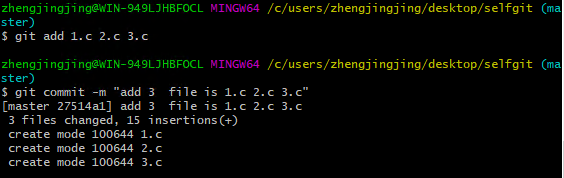


Git commit把文件提交到仓库，-m后面输入的是本次提交的说明，可以输入任意内容，1 file changed为一个文件被改动，2 insertions为插入了3行内容

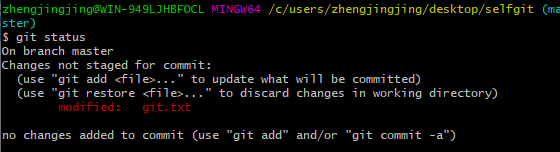


若不小心提交了错误的或想本地仓库撤销：git reset HEAD~

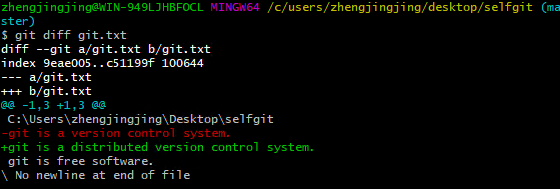
可一次提交多个文件，如下

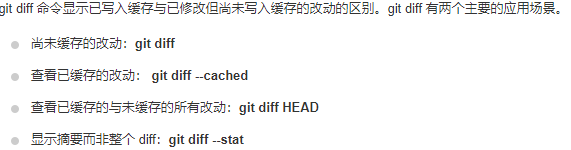


Git status：查看仓库当前状态，这里为git.txt文件被修改过但还没有准备提交修改

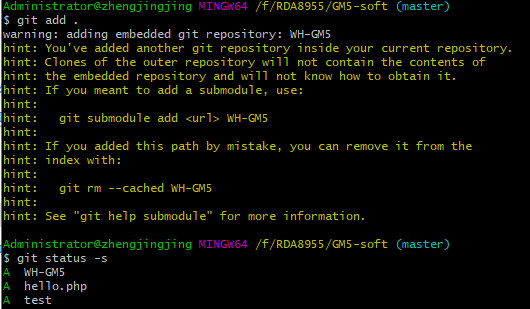


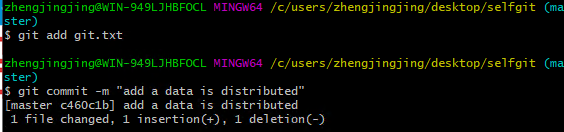
git diff：查看执行git status的结果的详细信息，



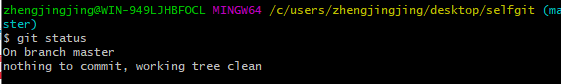


提交修改和提交新文件是一样的两步骤，git add和git commit

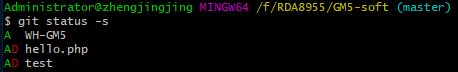




提交后再用git status查看当前仓库状态，这里显示当前没有需要提交的修改，工作目录是干净的



若之后删除文件则会显示成如下，后面有个D



git checkout：切换分支，使用-b参数为创建分支并切换到此分支上去

git push：将代码推送到远程分支上去git push origin develop\_xxx

git log：上传远程仓库时上传的修改日志

git pull：下拉代码，git pull origin develop\_gm5

git clean -d -fx为删除一些没有git add的文件，若切换时切换不过去，可以使用这个命令

git reset --hard && git clean -dxf

图形化git：gitk

彩色的git输出：git config color.ui true

显示历史记录时每个提交信息只显示一行：git config format.pretty oneline

交互式添加文件到暂存区：git add -i

图形化客户端：gitx(l)开源软件、tower、source tree免费、github for mac免费、gitbox

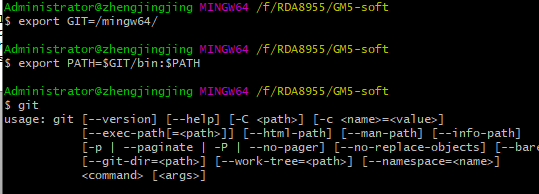
Git 完整命令手册地址：[http://git-scm.com/docs](http://git-scm.com/docs" \t "https://www.runoob.com/git/_blank)

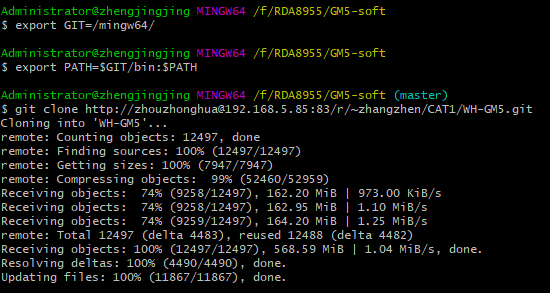
Github：付费可建立私库，一般免费用户只能使用公共仓库，也就是代码要公开

Git安装之后git-bash只能在指定的目录运行git命令问题解决如下：（其中mingw64/bin/git.exe这里是可以直接执行git命令的）

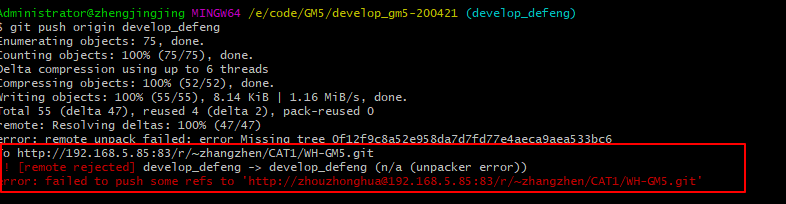
export GIT=/mingw64/

export PATH=$GIT/bin:$PATH

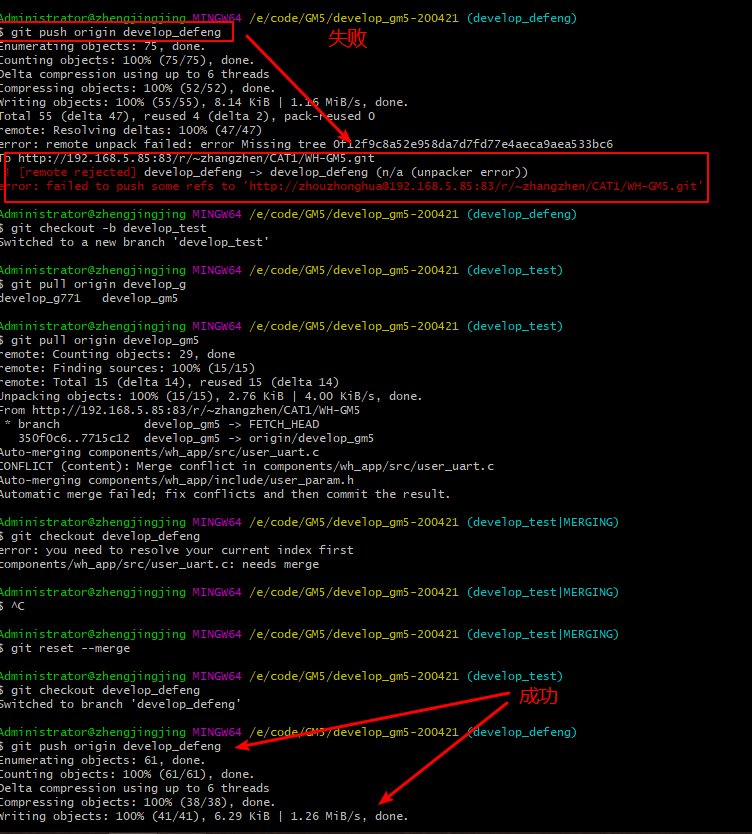




注意：重新创建分支之后先git push上传一个基本的代码，即默认分支develop\_gm5，然后再在此代码上修改



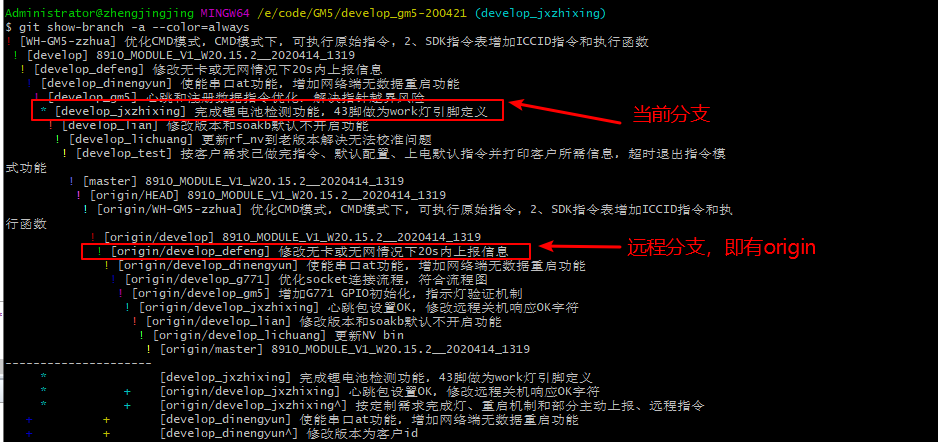
解决方式为：在本地重新创建一个新的分支，然后pull远程代码，之后使用git reset --merge之后再切换到本地分支上去

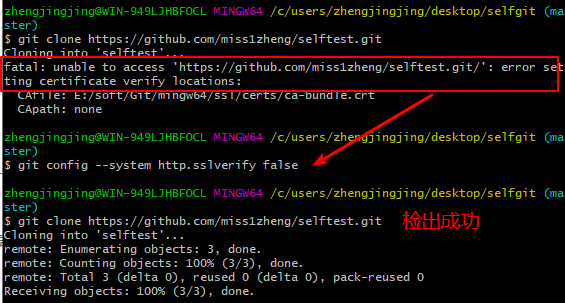


切换到某一版本



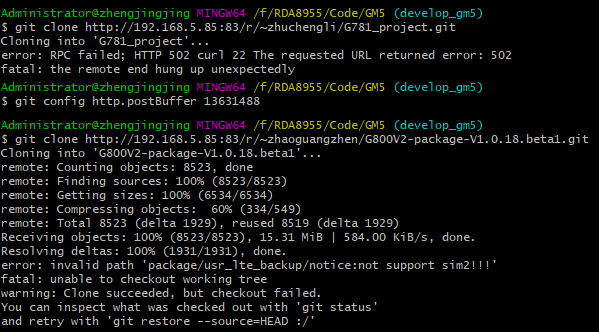
查看是否提交到远程分支上去了。有若干短划线”-”分隔，几个分支就会有几个短划线。其中短划线分割两部分，上部分为层次缩进的分支列表，下半部分为commit列表。上部分\*标识当前分支，！标识其他分支，不同分支颜色不同。下部分commit列表的\*和+,\*标识这列中当前分支由此commit，而+标识这列中的非当前分支由此commit。颜色用于区分分支





解决git仓库过大导致克隆到本地仓库时出错：

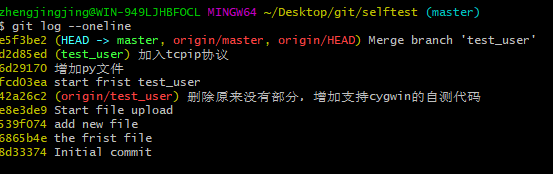
git config http.postBuffer 13631488（此为远程仓库的大小，以字节为单位）



git rm：删除文件，若文件修改过则需要加-f选项，若想将文件从暂存区删除但仍然想保留在工作目录中，则需要加--cached选项，使用-r \*则会删除整个目录文件和子目录



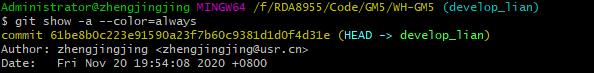
git log：查看提交日志，使用--oneline可以查看简洁版的log



若不小心将代码add ./并且commit -m了之后想撤回提交，执行git reset --soft HEAD^即可。此时仅撤回了commit而代码仍然保留。



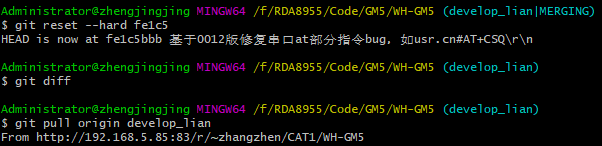
若commit之后还想再看下提交前后之间的有哪些地方改变了，则可以使用如下命令查看git show -a --color=always



Git的时候出现如下问题：

解决方式为：git rebase --abort，然后再git reset --hard xxxxx回到以前版本，再git pull origin xxx获取最新版本。（原因可能是本地提交版本和远程冲突导致）





远程版本回退实现（牧原定制中不小心加入了信源定制信息）：



