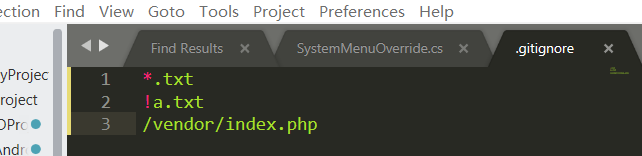
# Git教程

## Git命令注意

1. git log
   1. 说明：查看库的操作日志
   2. 如何退出：按q键 或者 ctrl+c
2. 细节命令
   1. ctrl +l 清除屏幕
   2. ctrl+ins 复制 shift + ins 粘贴
   3. 文本编辑命令
      1. subl xxx.xxx 指的是sublime Text工具
         1. 如果命令没用，设置环境变量：系统变量->Path->设置sublime的安装路径
      2. vim xxx.xxx 指的是vim文本编辑工具
         1. 按键I:进行输入
         2. Esc:回到左下角目录
         3. cat xxx.xxx 查看文本文件内容
      3. :wq :退出

## Git注意事项

1. git仓库，个人公共仓库总的不能超过1个G,单个文件不能超过100M
2. git 国内镜像地址 <https://npm.taobao.org/mirrors/git-for-windows/>
3. github 个人仓库只允许开发1000次仓库
4. git 不能跟踪空的文件夹
5. git 修改过后文件 最后使用git add 具体文件名，而不是.
6. .gitignore 文件，管理文件检测忽略
   1. subl .gitignore 打开该文件

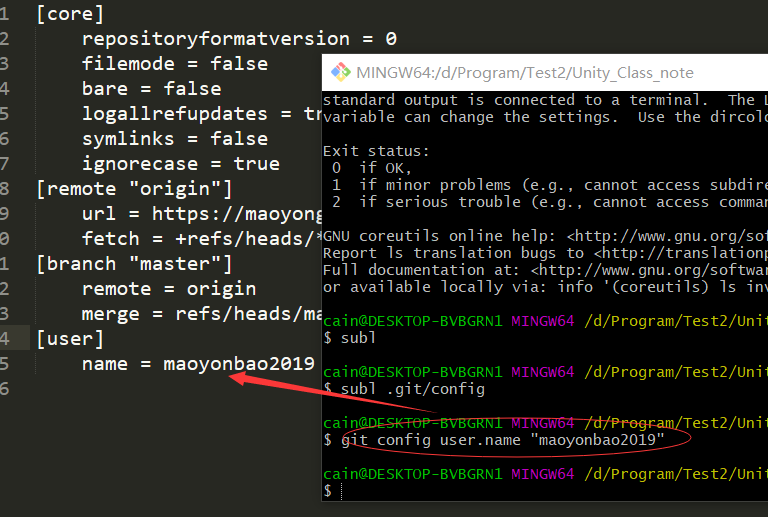


* 1. 例如加上\*.txt 不在检测txt文件的变化,!a.txt表示检测a.txt
  2. /vendor 表示不检测该目录下的文件

1. 命令行修改文件名可能git检测不到大小写的变化，可以在外部进行修改或者删除替换

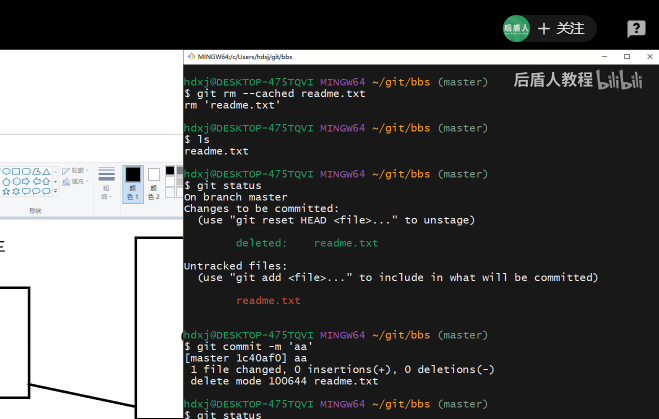
## Git命令

1. 基础命令
   1. git --version 显示git的版本
   2. git config 显示git config 下的所有子命令说明
   3. git config --list 显示当前的git配置
   4. git config --global user.name “name” 设置git全局用户名
   5. git config --global user.email “email” 设置git 全局用户邮箱
   6. git status 查看当前仓库状态
2. git中包含的linux命令
3. pwd 显示当前目录
4. mkdir xxx 创建目录
5. touch xx.xx 创建空白的文件
6. ls 显示列表
7. ls -a 显示列表 包括隐藏目录
8. ll 显示详细列表
9. dir 显示所有文件以及目录
10. rm -rf \*删除所有
11. git rm -rf xxx/xxx.xx 删除目录或文件
12. git mv c.php xxx.php 修改名称
13. cd xx/xx 进入到每个目录
14. cd 进入到盘符根目录
15. cd .. 进入到上一级目录
16. 关键命令
17. git config user.name “name” 在配置文件中标识用户名

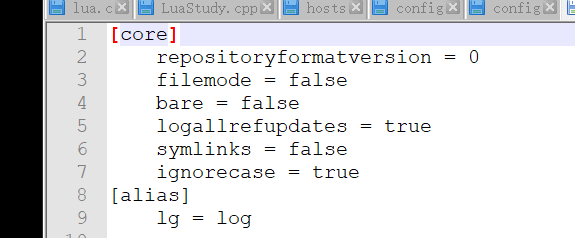


四. 流程命令

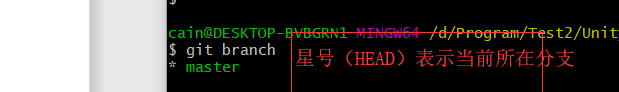
1. git init 初始化仓库命令，该命令会生成一个隐藏.git的文件目录用于通知git把该文件目录下的文件做为仓库进行管理
2. git add # 将工作区的修改提交到暂存区（包括添加，修改，删除）
3. git add . 将工作区所有的修改提交到暂存区
4. git rm x.xxx 本地删除文件
5. git commit -m ”说明” 将暂存区的修改提交到当前分支
6. git commit --amend 修改最近一次的提交的说明信息
7. git log 显示日志信息
   1. git log -p 查看具体日志信息
   2. git log -p –[number] 查看最近的n次提交
   3. git log --oneline 一行显示所有提交日志
   4. git log --oneline -p 一行显示所有的具体信息
   5. git log --name-only 查看有哪些文件发生了变化
   6. git log --name-status 查看文件的变化状态信息
8. 特殊命令(撤销操作)
   1. git rm –cached x.xxx从版本库中删除文件
   2. 这样删除文件会有个问题，因为在本地还未删除



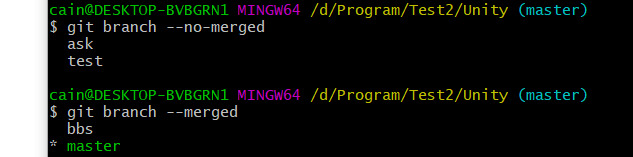
* 1. git commit --amend 修改最近一次的提交的说明信息, 并且修改最近一次提交
  2. git reset HEAD a.php 撤销对文件最近一次的修改
  3. git checkout -- xx.xx 将文件恢复到最近一次操作前的内容
  4. 修改操作符
     1. git config --global/local alias.a add
     2. 或直接编辑配置.gitconfig 文件



1. 分支命令(撤销操作)
   1. 说明：
      1. master主分支
      2. (ask)xxx 支线分支
      3. HEAD指针，它所在的分支为当前工作分支
      4. 各个分支之间的资源不共享包括主分支
   2. git branch 查看分支

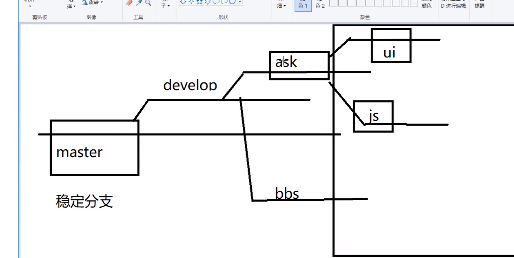


* + 1. git branch xxx 创建其他分支
    2. git checkout xxx 切换到其他分支
    3. git checkout -b xxx 创建分支并切换到相应的分支
  1. git merge xxx 合并分支到主分支
  2. git branch --merged 查看已经合并的分支
     1. 刚刚创建的分支，提交点一样所以会显示为已经提交的分支，下一次提交才不会有

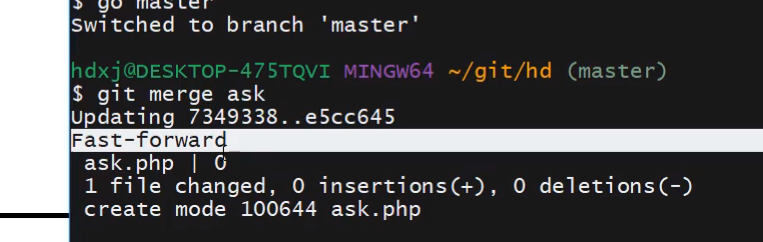


* 1. git branch --no-merged 查看还未合并的分支
  2. git branch -d xxx 删除已经合并的分支xxx
  3. git branch -D xxx 无论分支是否合并都一律删除
  4. 解决分支产生的简单冲突
     1. 冲突发生原因：其他分支创建了相同名称的一个资源，在合并的时候产生冲突
     2. 例子：ask 创建并修改t.txt文件内容为ask, bbs创建并修改t.txt文件内容为bbs，分别合并到master产生冲突
     3. 解决：把产生冲突的文件再次进行修改即可

1. 标准分支操作流程
   1. 第一个自动创建的分支master为稳定分支
   2. 一般会在master分出一个新的分支develop
   3. 创建的分支ask从develop分出
   4. ask下根据模块又分出了分支ui、js
   5. ask写好就往下合并，并删除
   6. develop可能又会分出其他功能分支
   7. 完成好的分支可以一直往上回溯

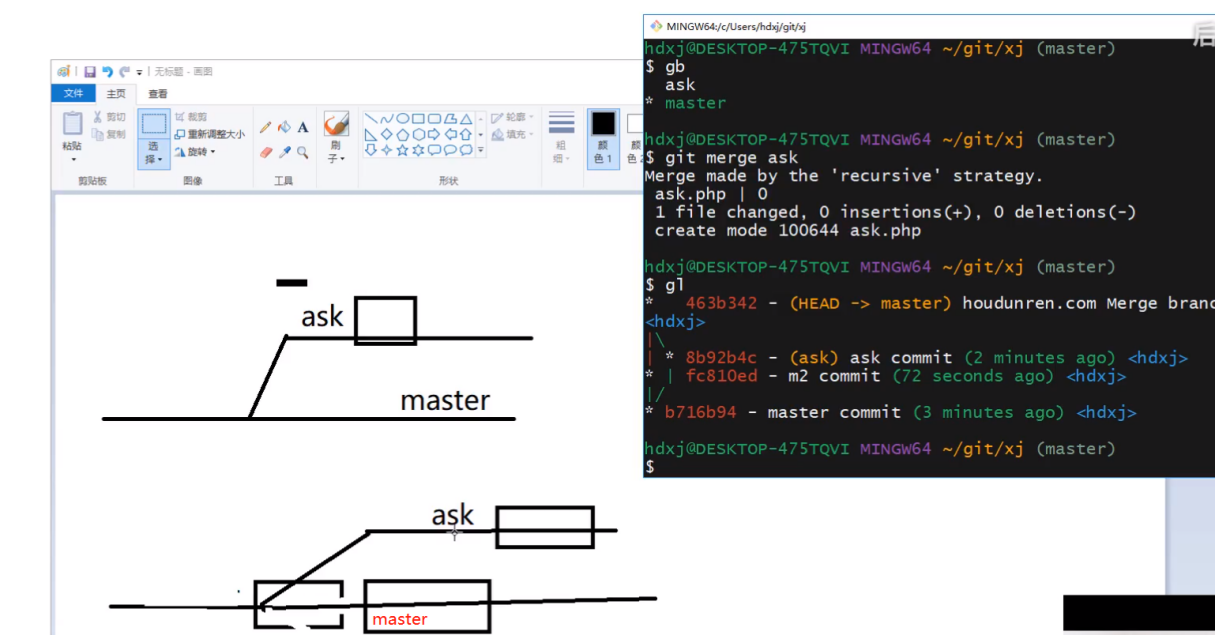


* 1. 合并主分支与支分支操作冲突解决
     1. 正常的合并操作只是指针的移动，把ask分支与master分支合并如下图

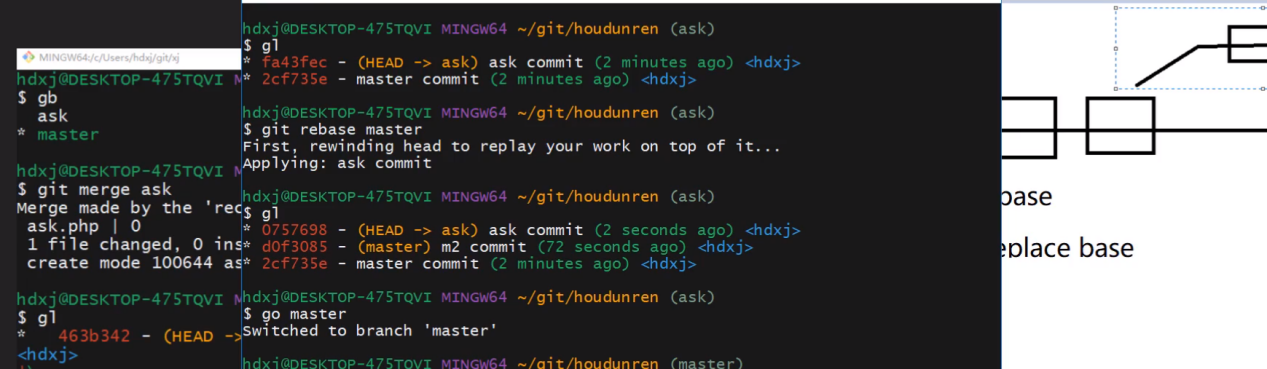


这个fast-forward 提示的就是指指针的移动->快速合并

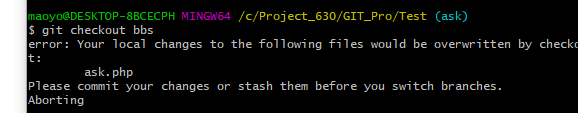
1. 出现的问题：正常的合并，主分支没有加东西，合并支线分支时就是指针的简单移动，如果主分支向后多很多修改，在合并的时候会发生分支冲突，这时的冲突需要分支来解决



1. 解决思路：把支线分支总体往后挪，挪到主线分支最新提交点，git rebase
2. 步骤：先去要合并的支分支移动，git checkout xx, git rebase master,在回到主分支进行合并，git checkout master ,git merge xxx



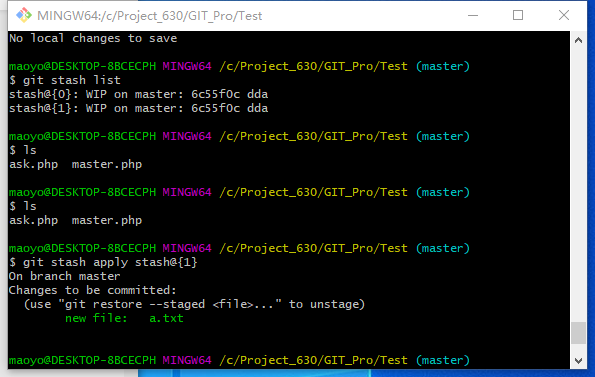
1. 暂存区操作
   1. 解决问题背景:发生在当支线分支对文件修改后加入暂存区但未提交，然后切换到其他分支出切换不了出现下面提示：



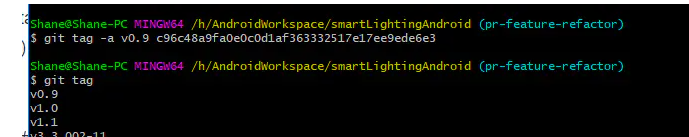
* 1. 解决思路：将当前暂存区的工作状态暂时存储起来
  2. git statsh 存储当前暂存区的工作状态
  3. git statsh list 查看当前存储的所有暂存区的状态



* 1. 其他分支做完了再切换回当前分支，恢复之前保存了的工作状态
  2. git stash apply 恢复最近的暂存区的状态
  3. git stash apply [statsh@{0}] 恢复到指定的暂存区的状态
  4. git stash drop [statsh@{0}] 恢复的暂存区是还在的，所以还要删除暂存区
  5. git stash pop = git stash apply + git stash drop xx 恢复栈顶状态并删除它

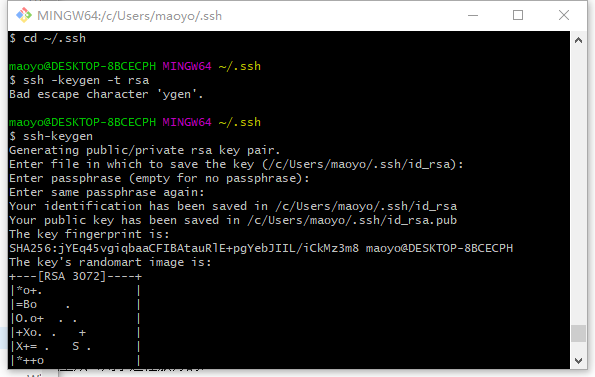


1. TAG操作
   1. 意义：标签指的是对某个阶段完成的稳定的项目的总结，别名
   2. 通常会在发布软件的时候打一个tag,tag会记录版本的commit号，方便后期回溯
   3. 查看tag:
      1. git tag 列出已有标签
      2. git tag -l “v3.0.\*” 加上-l可以使用通配符过滤掉tag
      3. git show tag 查看tag详细信息
   4. 新建tag:
      1. git tag [name] 创建一个tag（为最近的提交打上标签）
      2. git tag -a [name] -m”sdadad” 创建一个带标注的标签
      3. git tag -a [name] 提交号 给指定的commit号加上tag

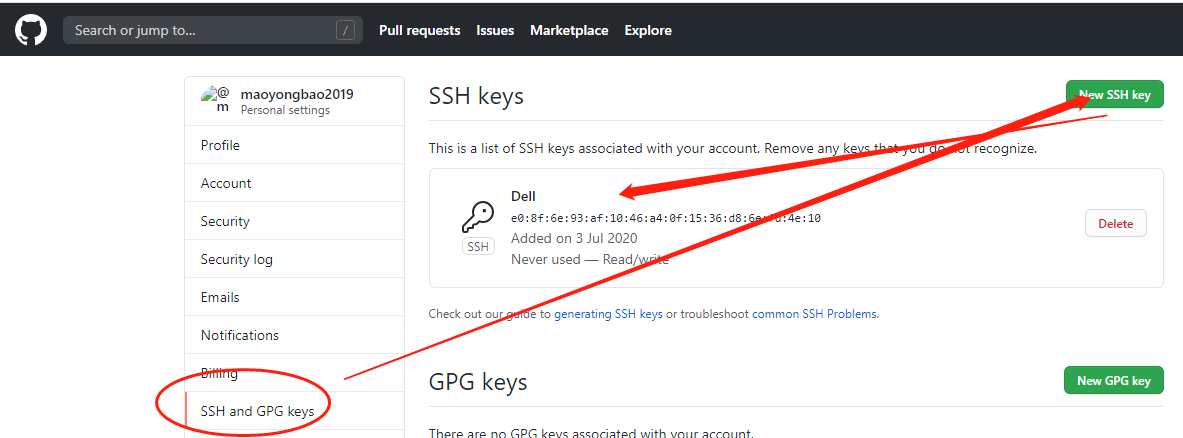


* 1. 其他命令:
     1. git push origin [name]推送当前分支的tag到远程服务器
     2. git push origin --tags 推送当前仓库所以的tag到远程服务器
     3. git checkout [name] 切换到某个tag
     4. git tag -d[name] 删除某个tag

1. 打包操作
   * 1. git archive -list/l 查看git 支持的文件压缩格式
     2. git archive master --prefix='[hdcms]/' --format=zip > hdcms.zip使用指定格式打包到指定目录下的指定名称
     3. 打包的操作目录应该在根目录，不然会出错
2. SSH密钥生成（用于远程服务的）第一种方式有时会出错
   1. 先检查有么有.ssh目录，没有可以在cmd的用户目录下敲击命令 md .ssh
   2. 进入.ssh 目录下，在git bash 中敲击 cd ~/.ssh
   3. 敲击ssh -keygen 根据提示进行操作，一般不理会直接敲击回车三次



* 1. 敲击subl id\_rsa.pub命令，或者直接打开文件rsa.pub中的内容，直接全选复制
  2. 进入github个人settings界面，这样远程仓库就能安全的记住你了



* 1. 不安全的方式：直接修改config 文件 进入目录.git/config 进行修改

[remote “origin”]

url = https://用户名：[密码@github.com/用户名/远程仓库名.\_git](mailto:密码@github.com/用户名/远程仓库名._git)

这种方式就只能争对当前本地仓库有效，换一个又要设置一次

* 1. 重新生成密匙连接不上的问题，试试清楚 ~/.ssh下know\_hosts文件的内容

1. 远程操作
2. git clone 地址命名