Github客户端安装和使用文档

（以苹果系统为例）

1 去官网注册一个账号，地址<https://github.com>

2 客户端下载

GitHub Desktop <https://desktop.github.com>

sourceTree <https://www.sourcetreeapp.com/>

GitX(L) <http://gitx.laullon.com/>

Tower <https://www.git-tower.com/>

GitBox

3 如果以前用过 xcode command line tools 里面有git最低版本，如果需要最新的git 需要下载 git 安装包

下载地址：<http://git-scm.com/download/mac>.

http://git-scm.com/download/mac 镜像安装失败、、、

3.1 用另一种方式安装git command line 是 用命令 安装Homebrew 里面有git 包

步骤是：

a 用命令控制器 安装homebrew

ruby -e "$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/master/install>)"

b brew --version 如果不是最新的就就 brew update

c brew install git 安装，如果提示xcode-select –install 按照步骤走得先安装xcode 命令工具，再去 安装git（我觉得可以把xcode卸载 直接 brew install git 就没问题version 2.15.1）

github for windows 使用教程 非常齐全的网址

1 基本的使用 (建库 修改内容 上传 本地同步等功能) http://youngxhui.github.io/2016/05/03/GitHub-for-Windows%E4%BD%BF%E7%94%A8%E6%95%99%E7%A8%8B(%E4%B8%80)/

2 建分支 使用分支 http://youngxhui.github.io/2016/05/13/GitHub-for-Windows%E4%BD%BF%E7%94%A8%E6%95%99%E7%A8%8B(%E4%BA%8C)/

3 合并分支 提交 合作http://youngxhui.github.io/2016/05/15/GitHub-for-windows%E4%BD%BF%E7%94%A8%E6%95%99%E7%A8%8B%EF%BC%88%E4%B8%89%EF%BC%89/

4查询等其他功能

http://youngxhui.github.io/2016/08/28/GitHub-for-Windows%E4%BD%BF%E7%94%A8%E6%95%99%E7%A8%8B%EF%BC%88%E5%9B%9B%EF%BC%89/

最基本的git 命令

建库： git init

检出仓库(把仓库克隆到本地): git clone /path/to/repository …….

添加文件/改动：git add <filename>

提交工程/代码/文件/资源等：git commit -m “提交代码信息”

同步到服务器：git push origin master(master 是你的分支的名称)

把你本地的文件上传到git服务器 (以前没有上传过的): git remote add origin <server>

分支管理

创建分支：git checkout –b feature（feature是你的分支名称）

新建分支同步到服务器：git push origin <branch>

合并分支：git merge <branch> 合并的时候可能遇到冲突

切换到主分支：git checkout master （master是你建库默认的主分支）

删除分支：git branch –d feature

更新本地库至最新版本：git pull

冲突：conflicts

遇到冲突要手动修改文件，预览差异：git diff <source\_branch> <target\_branch>手动修改后 执行：git add <filename> 添加文件就可以了

为软件发布或者标识阶段版本添加标签：git tag 1.0.0 1b2e3c4d5f

1b2e3c4d5f 这个是你要标记的提交ID的前10位字符

log：

获取提交ID：git log

获取默认提交的记录：git log --author=wangwenhua

彩色的git输出：git config color.ui true

更多log操作：git log --help

操作失误 替换本地修改

不想要本地的修改了：git checkout --<filename> 注：已经add的内容不会被覆盖还是存在的

所有的改动都不想要了：git fetch origin 然后：git reset --hard origin/master

图解Git: http://marklodato.github.io/visual-git-guide/index-zh-cn.html

git command

**查看、添加、提交、删除、找回，重置修改文件**

git help <command> # 显示command的help

git show # 显示某次提交的内容 git show $id

git co -- <file> # 抛弃工作区修改

git co . # 抛弃工作区修改

git add <file> # 将工作文件修改提交到本地暂存区

git add . # 将所有修改过的工作文件提交暂存区

git rm <file> # 从版本库中删除文件

git rm <file> --cached # 从版本库中删除文件，但不删除文件

git reset <file> # 从暂存区恢复到工作文件

git reset -- . # 从暂存区恢复到工作文件

git reset --hard # 恢复最近一次提交过的状态，即放弃上次提交后的所有本次修改

git ci <file> git ci . git ci -a # 将git add, git rm和git ci等操作都合并在一起做　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　git ci -am "some comments"

git ci --amend # 修改最后一次提交记录

git revert <$id> # 恢复某次提交的状态，恢复动作本身也创建次提交对象

git revert HEAD # 恢复最后一次提交的状态

**查看文件diff**

git diff <file> # 比较当前文件和暂存区文件差异 git dgit diff <id1><id1><id2> # 比较两次提交之间的差异

git diff <branch1>..<branch2> # 在两个分支之间比较

git diff --staged # 比较暂存区和版本库差异

git diff --cached # 比较暂存区和版本库差异

git diff --stat # 仅仅比较统计信息

**查看提交记录**

git log git log <file> # 查看该文件每次提交记录

git log -p <file> # 查看每次详细修改内容的diff

git log -p -2 # 查看最近两次详细修改内容的diff

git log --stat #查看提交统计信息

**tig**

Mac上可以使用tig代替diff和log，brew install tig

**Git 本地分支管理**

**查看、切换、创建和删除分支**

git br -r # 查看远程分支

git br <new\_branch> # 创建新的分支

git br -v # 查看各个分支最后提交信息

git br --merged # 查看已经被合并到当前分支的分支

git br --no-merged # 查看尚未被合并到当前分支的分支

git co <branch> # 切换到某个分支

git co -b <new\_branch> # 创建新的分支，并且切换过去

git co -b <new\_branch> <branch> # 基于branch创建新的new\_branch

git co $id # 把某次历史提交记录checkout出来，但无分支信息，切换到其他分支会自动删除

git co $id -b <new\_branch> # 把某次历史提交记录checkout出来，创建成一个分支

git br -d <branch> # 删除某个分支

git br -D <branch> # 强制删除某个分支 (未被合并的分支被删除的时候需要强制)

**分支合并和rebase**

git merge <branch> # 将branch分支合并到当前分支

git merge origin/master --no-ff # 不要Fast-Foward合并，这样可以生成merge提交

git rebase master <branch> # 将master rebase到branch，相当于： git co <branch> && git rebase master && git co master && git merge <branch>

**Git补丁管理(方便在多台机器上开发同步时用)**

git diff > ../sync.patch # 生成补丁

git apply ../sync.patch # 打补丁

git apply --check ../sync.patch #测试补丁能否成功

**Git暂存管理**

git stash # 暂存

git stash list # 列所有stash

git stash apply # 恢复暂存的内容

git stash drop # 删除暂存区

**Git远程分支管理**

git pull # 抓取远程仓库所有分支更新并合并到本地

git pull --no-ff # 抓取远程仓库所有分支更新并合并到本地，不要快进合并

git fetch origin # 抓取远程仓库更新

git merge origin/master # 将远程主分支合并到本地当前分支

git co --track origin/branch # 跟踪某个远程分支创建相应的本地分支

git co -b <local\_branch> origin/<remote\_branch> # 基于远程分支创建本地分支，功能同上

git push # push所有分支

git push origin master # 将本地主分支推到远程主分支

git push -u origin master # 将本地主分支推到远程(如无远程主分支则创建，用于初始化远程仓库)

git push origin <local\_branch> # 创建远程分支， origin是远程仓库名

git push origin <local\_branch>:<remote\_branch> # 创建远程分支

git push origin :<remote\_branch> #先删除本地分支(git br -d <branch>)，然后再push删除远程分支

**Git远程仓库管理**

*[GitHub](http://blog.jobbole.com/6492/" \o "GitHub如何运作：时间并不决定一切" \t "_blank)*

git remote -v # 查看远程服务器地址和仓库名称

git remote show origin # 查看远程服务器仓库状态

git remote add origin git@ github:robbin/robbin\_site.git # 添加远程仓库地址

git remote set-url origin git@ github.com:robbin/robbin\_site.git # 设置远程仓库地址(用于修改远程仓库地址) git remote rm <repository> # 删除远程仓库

**创建远程仓库**

git clone --bare robbin\_site robbin\_site.git # 用带版本的项目创建纯版本仓库

scp -r my\_project.git git@ git.csdn.net:~ # 将纯仓库上传到服务器上

mkdir robbin\_site.git && cd robbin\_site.git && git --bare init # 在服务器创建纯仓库

git remote add origin git@ github.com:robbin/robbin\_site.git # 设置远程仓库地址

git push -u origin master # 客户端首次提交

git push -u origin develop # 首次将本地develop分支提交到远程develop分支，并且track

git remote set-head origin master # 设置远程仓库的HEAD指向master分支