github Mac端使用手册

一：Git 简介

Git 是什么？

分布式版本控制系统。

接地气的介绍就是：自动帮我记录每次文件的改动，还可以让同事协作编辑。

二：安装Git ( MacOs 系统)

推荐的方法，就是直接从AppStore安装Xcode，Xcode集成了Git，不过默认没有安装，你需要运行Xcode，选择菜单“Xcode”->“Preferences”，在弹出窗口中找到“Downloads”，选择“Command Line Tools”，点“Install”就可以完成安装了。

三：创建版本库

1:建立空文件夹 mkdir xxx

2:将文件夹git化 git init

备注：1可以通过 ls -ah 查看所有文件目录/隐藏文件

2所有的版本控制系统，其实只能跟踪文本文件的改动，比如TXT文件，网页，所有的程序代码等等

四：把文件添加到版本库

1: git add xxx/ .

2: git commit -m “备注”

备注：commit可以一次提交很多文件，所以你可以多次add不同的文件

五：时光机穿梭

1:git status 查看状态

2:git diff xx/ 查看更改内容

备注：再进行提交如步骤四

六： 版本回退

1:git log 用git log可以查看提交历史，以便确定要回退到哪个版本。

2:git reset --hard commit\_id 穿梭到指定历史版本

3:git reflog 重返回未来的版本，查看命令历史

备注：HEAD指向的版本就是当前版本，因此，Git允许我们在版本的历史之间穿梭

七：工作区和暂存区

工作区（Working Directory）

就是你在电脑里能看到的目录

版本库（Repository）

工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。

Git的版本库里存了很多东西，其中最重要的就是称为stage（或者叫index）的暂存区，还有Git为我们自动创建的第一个分支master，以及指向master的一个指针叫HEAD。

备注：git status 查看的就是工作区和暂存区的状态。

八：管理修改

第一次修改 -> git add -> 第二次修改 -> git add -> git commit

备注：每次修改，如果不用git add到暂存区，那就不会加入到commit中

九：撤销修改

1:git checkout -- file 丢弃工作区的修改，撤销修改后就回到和版本库一模一样的状态

2:git reset HEAD file 可以把暂存区的修改撤销掉（unstage），重新放回工作区；再将工作区的修改撤销掉。

3:已经commit ,但是没有提交远程仓库。 参考六： 版本回退。

十：删除文件

1: git rm file 工作区删去了文件，进而删去版本库的文件。

2: git checkout -- file 工作区误删恢复到最新版本（恢复到与版本库一样的版本）

备注：会丢失最新一次的修改。

十一：远程仓库

第1步：创建SSH Key。在用户主目录下，看看有没有.ssh目录，如果有，再看看这个目录下有没有id\_rsa和id\_rsa.pub这两个文件，如果已经有了，可直接跳到下一步。如果没有，打开Shell（Windows下打开Git Bash），创建SSH Key：

$ ssh-keygen -t rsa -C "youremail@example.com"

你需要把邮件地址换成你自己的邮件地址，然后一路回车，使用默认值即可，由于这个Key也不是用于军事目的，所以也无需设置密码。

如果一切顺利的话，可以在用户主目录里找到.ssh目录，里面有id\_rsa和id\_rsa.pub两个文件，这两个就是SSH Key的秘钥对，id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥，可以放心地告诉任何人。

第2步：登陆GitHub，打开“Account settings”，“SSH Keys”页面：

然后，点“Add SSH Key”，填上任意Title，在Key文本框里粘贴id\_rsa.pub文件的内容

备注：1:Mac的主目录下是看不到 。ssh的。 方法：open ~/.ssh .回车就可以看到了;

2:GitHub允许你添加多个Key

十二：添加远程库

步骤：1:登录github,创建一个新仓库。

2:此时有两种选择，一种是与本地仓库关联；另一种是克隆仓库到本地。第三种选择一般用不到。创建仓库后有提示。

第一种方法：在本地仓库目录下 git remote add origin [git@github.com:David520025/practise.git](mailto:git@github.com:David520025/practise.git) ； 关联后，使用命令git push -u origin master第一次推送master分支的所有内容。

备注：SSH警告，当你第一次使用Git的clone或者push命令连接GitHub时，会得到一个警告，输入yes就行。

十三：从远程库克隆

步骤：1:登录github,创建一个新仓库。

2:在本地选好的目录下执行终端指令：git clone URL

十四：分支管理

Git的分支是与众不同的，无论创建、切换和删除分支，Git在1秒钟之内就能完成！无论你的版本库是1个文件还是1万个文件.

十五：创建与合并分支

1: git checkout -b dev 创建dev分支，然后切换到dev分支：

备注：git checkout命令加上-b参数表示创建并切换，相当于以下两条命令：

$ git branch dev

$ git checkout dev

2: git branch命令查看当前分支

备注：git branch命令会列出所有分支，当前分支前面会标一个\*号

3: git merge <name> 命令用于合并指定分支到当前分支。

备注：$ git merge dev 就是将dev分支合并到当前分支

4:git branch -d dev 删除dev分支

总结：

1. 查看分支：git branch
2. 创建分支：git branch <name>
3. 切换分支：git checkout <name>
4. 创建+切换分支：git checkout -b <name>
5. 合并某分支到当前分支：git merge <name>
6. 删除分支：git branch -d <name>

十六：解决冲突

1:git merge dev 合并分支到当前分支，会自动爆出冲突，并在文档上显示出冲突。

2:git status 显示冲突的文件

3:直接查看冲突文档，并修改后，提交。

4: git log --graph --pretty=oneline --abbrev-commit 带参数的git log也可以看到分支的合并情况

5:git branch -d <name> 删除分支。 工作完成。

十七：分支管理策略

通常，合并分支时，如果可能，Git会用Fast forward模式，但这种模式下，删除分支后，会丢掉分支信息。

如果要强制禁用Fast forward模式，Git就会在merge时生成一个新的commit，这样，从分支历史上就可以看出分支信息。

合并dev分支，请注意--no-ff参数，表示禁用Fast forward模式

$ git merge --no-ff -m "merge with no-ff" dev

备注：因为本次合并要创建一个新的commit，所以加上-m参数，把commit描述写进去

合并后，用git log看看分支历史：可以看到合并历史。

分支策略

在实际开发中，我们应该按照几个基本原则进行分支管理：

首先，master分支应该是非常稳定的，也就是仅用来发布新版本，平时不能在上面干活；

那在哪干活呢？干活都在dev分支上，也就是说，dev分支是不稳定的，到某个时候，比如1.0版本发布时，再把dev分支合并到master上，在master分支发布1.0版本；

你和你的小伙伴们每个人都在dev分支上干活，每个人都有自己的分支，时不时地往dev分支上合并就可以了。

十八：Bug分支

1: git stash 隐藏当今分支在版本库的修改

2: git status 查看状态（工作区应该是干净的，除非有没有被git管理起来的文件）

3: 创建新的分支，完成bug修复，提交。

4: git checkout xx 切回工作分支，

5: git stash list 查看隐藏文件

6: git stash pop 恢复最近一次的文件并删除隐藏目录。

7: git stash list 查看确认是否还有么隐藏文件，持续操作67步骤直到没有。

备注：之所以需要隐藏的原因：1 因为内容还没完成，不想有脏commit; 2 如果不隐藏也不提交，切回到其它分支，也能看到暂存区的修改，一不注意就给搞乱了。

十九：Feature分支

开发一个新feature，最好新建一个分支；

如果要丢弃一个没有被合并过的分支，可以通过git branch -D <name>强行删除。

二十：多人协作

从远程仓库克隆时，实际上Git自动把本地的master分支和远程的master分支对应起来了，并且，远程仓库的默认名称是origin。

1:git push origin <branch-name> 推送分支 推送自己的修改

2:如果推送失败，则因为远程分支比你的本地更新，需要先用git pull试图合并；

3:如果合并有冲突，则解决冲突，并在本地提交；

4:没有冲突或者解决掉冲突后，再用git push origin <branch-name>推送就能成功！

备注：如果git pull提示no tracking information，则说明本地分支和远程分支的链接关系没有创建，用命令git branch --set-upstream-to <branch-name> origin/<branch-name>。

小结

1. 查看远程库信息，使用git remote -v；
2. 本地新建的分支如果不推送到远程，对其他人就是不可见的；
3. 从本地推送分支，使用git push origin branch-name，如果推送失败，先用git pull抓取远程的新提交；
4. 在本地创建和远程分支对应的分支，使用git checkout -b branch-name origin/branch-name，本地和远程分支的名称最好一致；
5. 建立本地分支和远程分支的关联，使用git branch --set-upstream branch-name origin/branch-name；
6. 从远程抓取分支，使用git pull，如果有冲突，要先处理冲突