# Git&GitHub学习笔记

## 一、安装Git

安装完成后还需要设置名字和地址，需要打开Git Bash，在命令行输入

**$ git config --global user.name "Your Name"**

**$ git config --global user.email** [**email@example.com**](mailto:email@example.com)

注意git config命令的--global参数，用了这个参数，表示你这台机器上所有的Git仓库都会使用这个配置，当然也可以对某个仓库指定不同的用户名和Email地址。

## 二、创建版本库（Repository）

版本库又名仓库（Repository），你可以简单理解成一个目录，这个目录里面的所有文件都可以被Git管理起来，每个文件的修改、删除，Git都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻可以“还原”。

首先创建一个空目录：

**$ mkdir learngit**

**$ cd learngit**

**$ pwd**

**/Users/michael/learngit**

其中，**pwd**命令用于显示当前目录。

接着，通过**git init**命令把这个目录变成Git可以管理的仓库

## 三、把文件添加到版本库

添加文件到Git仓库，分两步：

1、使用命令**git add <file>**，注意，可反复多次使用，添加多个文件；

2、使用命令**git commit -m <message>**，完成。

## 四、查看版本库的状态

要随时掌握工作区的状态，使用**git status**命令。

如果git status告诉你有文件被修改过，用**git diff**可以查看修改内容。

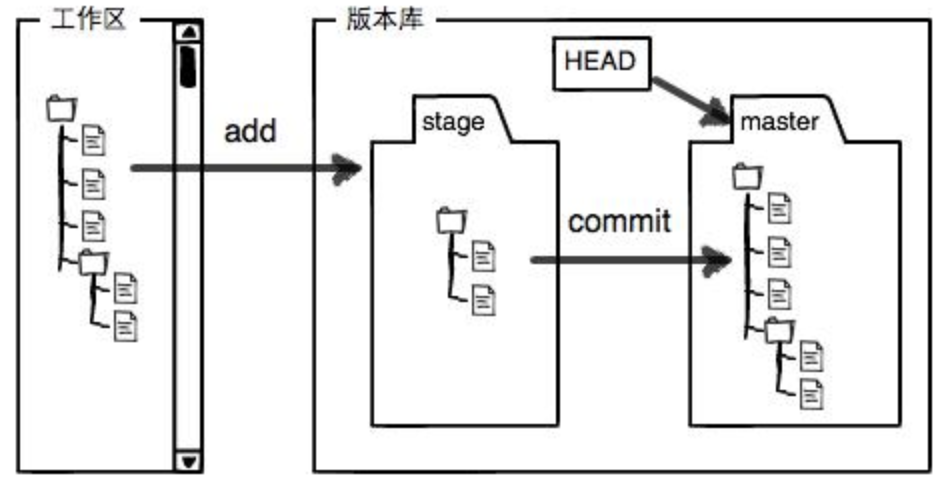
## 五、版本回退

**HEAD**指向的版本就是当前版本，上一个版本就是**HEAD^**，上上一个版本就是**HEAD^^**，当然往上100个版本写100个^比较容易数不过来，所以写成**HEAD~100**。因此，Git允许我们在版本的历史之间穿梭，使用命令**git reset --hard commit\_id**。

穿梭前，可以用**git log**可以查看提交历史，以便确定要回退到哪个版本，，其中一大串类似**1094adb...**的是**commit id**（版本号）。

要重返未来，用**git reflog**查看命令历史，以便确定要回到未来的哪个版本。

## 六、工作区和暂存区



## 七、管理修改

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

## 八、撤销修改

场景1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令**git checkout -- file**。

场景2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令**git reset HEAD <file**>，就回到了场景1，第二步按场景1操作。

场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考[版本回退](https://liaoxuefeng.com/books/git/reset/index.html)一节，不过前提是没有推送到远程库。

## 九、删除文件

命令**git rm**用于删除一个文件。如果一个文件已经被提交到版本库，那么你永远不用担心误删，但是要小心，你只能恢复文件到最新版本，你会丢失最近一次提交后你修改的内容。

## 十、远程仓库

通过GitHub完成

要关联一个远程库，使用命令**git remote add origin git@server-name:path/repo-name.git**；

关联一个远程库时必须给远程库指定一个名字，**origin**是默认习惯命名；

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；

此后，每次本地提交后，只要有必要，就可以使用命令**git push origin master**推送最新修改；

如果添加的时候地址写错了，或者就是想删除远程库，可以用**git remote rm <name>**命令。使用前，建议先用**git remote -v**查看远程库信息

## 十一、从远程库克隆

要克隆一个仓库，首先必须知道仓库的地址，然后使用**git clone**命令克隆。

## 十二、分支管理

1、创建于合并分支

Git鼓励大量使用分支：

查看分支：**git branch**

创建分支：**git branch <name>**

切换分支：**git checkout <name>或者git switch <name>**

创建+切换分支：**git checkout -b <name>**或者**git switch -c <name>**

合并某分支到当前分支：**git merge <name>**

删除分支：**git branch -d <name>**

2、解决冲突

当Git无法自动合并分支时，就必须首先解决冲突。解决冲突后，再提交，合并完成。

解决冲突就是把Git合并失败的文件手动编辑为我们希望的内容，再提交。

用**git log --graph**命令可以看到分支合并图。

3、分支管理策略

合并分支时，加上**--no-ff**参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而**fast forward**合并就看不出来曾经做过合并。

4、bug分支

修复bug时，我们会通过创建新的bug分支进行修复，然后合并，最后删除；

当手头工作没有完成时，先把工作现场**git stash**一下，然后去修复bug，修复后，再**git stash pop**，回到工作现场；

在**master**分支上修复的bug，想要合并到当前**dev**分支，可以用**git cherry-pick <commit>**命令，把bug提交的修改“复制”到当前分支，避免重复劳动。

5、feature分支

开发一个新功能，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。

6、多人协作

查看远程库信息，使用**git remote -v**；

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；

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；

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；

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**；

从远程抓取分支，使用**git pull**，如果有冲突，要先处理冲突。

7、Rebase

**git rebase**操作可以把本地未push的分叉提交历史整理成直线；

rebase的目的是使得我们在查看历史提交的变化时更容易，因为分叉的提交需要三方对比。

## 十三、便签管理

命令**git tag <tagname>**用于新建一个标签，默认为**HEAD**，也可以指定一个commit id；

命令**git tag -a <tagname> -m "blablabla..."**可以指定标签信息；

命令**git tag**可以查看所有标签。

命令**git push origin <tagname>**可以推送一个本地标签；

命令**git push origin --tags**可以推送全部未推送过的本地标签；

命令**git tag -d <tagname>**可以删除一个本地标签；

命令**git push origin :refs/tags/<tagname>**可以删除一个远程标签。

## 十四、使用GitHub

在GitHub上，可以任意Fork开源仓库；

自己拥有Fork后的仓库的读写权限；

可以推送pull request给官方仓库来贡献代码。

## 十五、自定义Git

忽略某些文件时，需要编写**.gitignore**；

**.gitignore**文件本身要放到版本库里，并且可以对**.gitignore**做版本管理！

搭建Git服务器非常简单，通常10分钟即可完成；

要方便管理公钥，用Gitosis；

要像SVN那样变态地控制权限，用Gitolite