Git 学习笔记

```
$ git config --global user.name "Your Name"
$ git config --global user.email "email@example.com"
```

创建版本库

```
$ mkdir learngit
$ cd learngit
$ pwd
/Users/michael/learngit

$ git init
>>>Initialized empty Git repository in /Users/michael/learngit/.git/
```

初始化一个Git仓库，使用git init命令。

添加文件到Git仓库，分两步：

1. 使用命令git add ，注意，可反复多次使用，添加多个文件；
2. 使用命令git commit -m ，完成。

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。

时间机器

版本回退

HEAD指向的版本就是当前版本，

```
$ git reset --hard HEAD^ #HEAD^^ ...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，用`git log`可以查看提交历史，以便确定要回退到哪个版本。

```
**$ git log --pretty=oneline
```

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。

```
$ cat readme.txt #to read the txt
---------------------------------------
$ git diff HEAD -- readme.txt
```

撤下修改

- 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
  or
  $git restore <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。

  ```
  $git reset HEAD <file>
  or
  $git restore --staged <file>
  ```

- 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，参考版本回退一节，不过前提是没有推送到远程库。

删除

```
$ git rm test.txt

git remote add origin https://github.com/lu-kong/learngit.git
git push -u origin master
```

远程仓库

Github 远程库的准备

---

1. 创建SHH—key
   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"
   ```

2. 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。

3. 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。

4. 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文件的内容可以直接txt打开

关联远程库

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；

分布式版本系统的最大好处之一是在本地工作完全不需要考虑远程库的存在， 也就是有没有联网都可以正常工作，而SVN在没有联网的时候是拒绝干活的！ 当有网络的时候，再把本地提交推送一下就完成了同步，真是太方便了！

从远程库克隆

---

1. 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。

2. 进到要存放库的目录上层`\Documents\Github\`

3. 使用`git clone` 命令， 加上远程库的地址

4. Git支持多种协议，包括https，但通过ssh支持的原生git协议速度最快。

分支管理

Git鼓励大量使用分支：

- 查看分支：git branch
- 创建分支：git branch
- 切换分支：git checkout 或者git switch
- 创建+切换分支：git checkout -b 或者git switch -c
- 合并某分支到当前分支：git merge
- 删除分支：git branch -d

解决冲突（merge)

需要手动修改，把当前分支下出现conflict的文件手动合并修改。 Git用`<<<<<<<`，`=======`，`>>>>>>>`标记出不同分支的内容，我们修改如下后保存： 然后提交， 则当前分支merge到修改后的分支，另一个冲突分支留在原处。 *用带参数的`git log`也可以看到分支的合并情况：*

```
$ git log --graph --pretty=oneline --abbrev-commit
```

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。

准备合并dev分支，请注意--no-ff参数，表示禁用Fast forward：

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

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。

bug 分支

对于bug处理， 可以把当前分支`stash` 起来 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；

  ```
  Or,
  $git stash list
  $git stash apply
  $git stash drop
  ```

- 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。

出现vi界面

It's not a Git error message, it's the editor git uses your default editor.

To solve this:

- press "i"
- write your merge message
- press "esc"
- write ":wq"
- then press enter

> If it helps anyone, the way you remember this is that "i" is for "insert", "esc" is the exit the insertion, and ":wq" is just "write" and "quit".