创建版本库

首先，选择一个合适的地方，创建一个空目录：

$ mkdir learngit

$ cd learngit

$ pwd

第二步，通过git init命令把这个目录变成Git可以管理的仓库（初始化Git仓库）：

$ git init

Initialized empty Git repository **in** /Users/michael/learngit/.git/

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；

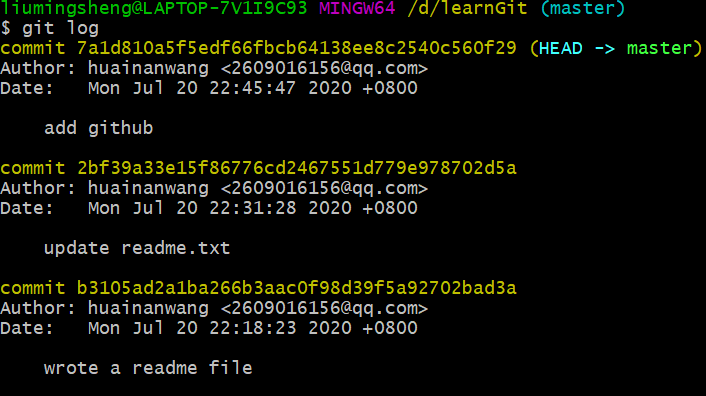
2.使用命令git commit -m <message>，完成。

git status命令可以让我们时刻掌握仓库当前的状态;

用git diff可以查看修改内容。

#### 版本回退

git log命令显示从最近到最远的提交日志;



7a1d810a5f5edf66fbcb64138ee8c2540c560f29 版本号

在Git中，用HEAD表示当前版本，也就是最新版本，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。

回退到上一版本：用 git reset命令：$ git re**set** --hard HEAD^

再返回前一个最新版本：$ git re**set** --hard 版本号（前几位即可）

用git reflog查看命令历史，以便确定要回到未来的哪个版本。

#### 工作区和暂存区

#### 工作区

#### 就是你在电脑里能看到的目录，比如我的learngit文件夹就是一个工作区：

#### 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。

#### git add把文件添加进去，实际上就是把文件修改添加到暂存区;

#### 用git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支。

#### 因为我们创建Git版本库时，Git自动为我们创建了唯一一个master分支，所以，现在，git commit就是往master分支上提交更改。

#### 提交后，用git diff HEAD -- readme.txt命令可以查看工作区和版本库里面最新版本的区别。

#### 撤销修改

#### <https://www.liaoxuefeng.com/wiki/896043488029600/897889638509536>

#### git checkout -- file可以丢弃工作区的修改：

#### 意思就是，把file文件在工作区的修改全部撤销

#### 总之，就是让这个文件回到最近一次git commit或git add时的状态。

#### 删除文件

#### 一是确实要从版本库中删除该文件，那就用命令git rm删掉，并且git commit：

$ git rm test.txt

rm 'test.txt'

$ git **commit** -m "remove test.txt"

#### git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。

$ git checkout -- test.txt

#### 远程仓库

#### 第1步：创建SSH Key。

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

#### 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， id\_rsa.pub是公钥。

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：

#### 添加远程库

#### 首先，登陆GitHub，然后，在右上角找到“Create a new repo”按钮，创建一个新的仓库：

#### 在Repository name填入learngit，其他保持默认设置，点击“Create repository”按钮，就成功地创建了一个新的Git仓库：

$ git remote add origin git@github.com:michaelliao/learngit.git

#### 注意，把上面的michaelliao替换成你自己的GitHub账户名

#### 添加后，远程库的名字就是origin，这是Git默认的叫法

#### 把本地库的内容推送到远程，用git push命令，实际上是把当前分支master推送到远程。

$ git push -u origin master

#### 由于远程库是空的，我们第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来

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

注意：可能会出现这样的错误：fatal: remote origin already exists.

解决方法：

1. 先删除：git remote rm origin
2. 再次执行即可：

$ git remote add origin https://github.com/huainanwang/learngit.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；

### 小结

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

从远程库克隆

$ git clone git@github.com:michaelliao/gitskills.git

### 小结

GitHub给出的地址不止一个，还可以用https://github.com/michaelliao/gitskills.git这样的地址。实际上，Git支持多种协议，默认的git://使用ssh，但也可以使用https等其他协议。

使用https除了速度慢以外，还有个最大的麻烦是每次推送都必须输入口令，但是在某些只开放http端口的公司内部就无法使用ssh协议而只能用https。

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。

Git支持多种协议，包括https，但ssh协议速度最快。

#### 分支管理

#### 创建与合并分支

#### 流程：

#### 在Git里，这个分支叫主分支，即master分支。

#### 每次提交，master分支都会向前移动一步，这样，随着你不断提交，master分支的线也越来越长。

#### 当我们创建新的分支，例如dev时，Git新建了一个指针叫dev，指向master相同的提交，再把HEAD指向dev，就表示当前分支在dev

#### 从现在开始，对工作区的修改和提交就是针对dev分支了，比如新提交一次后，dev指针往前移动一步，而master指针不变

#### 假如我们在dev上的工作完成了，就可以把dev合并到master上。Git怎么合并呢？最简单的方法，就是直接把master指向dev的当前提交，就完成了合并。

#### 合并完分支后，甚至可以删除dev分支。删除dev分支就是把dev指针给删掉，删掉后，我们就剩下了一条master分支。

首先，我们创建dev分支，然后切换到dev分支：

$ git checkout -b dev

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

然后，用git branch命令查看当前分支：

$ git branch

 branch命令会列出所有分支，当前分支前面会标一个\*号。

然后，我们就可以在dev分支上正常提交，比如对readme.txt做个修改，加上一行：

Creating a **new** branch is quick.

然后提交：

$ git add readme.txt

$ git commit -m "branch test"

现在，dev分支的工作完成，我们就可以切换回master分支：

$ git checkout master

切换回master分支后，再查看一个readme.txt文件，刚才添加的内容不见了！因为那个提交是在dev分支上，而master分支此刻的提交点并没有变：

现在，我们把dev分支的工作成果合并到master分支上：

$ git merge dev

git merge命令用于合并指定分支到当前分支

合并完成后，就可以放心地删除dev分支了：

$ git branch -d dev

### switch

我们注意到切换分支使用git checkout <branch>，而前面讲过的撤销修改则是git checkout -- <file>，同一个命令，有两种作用，确实有点令人迷惑。

实际上，切换分支这个动作，用switch更科学。因此，最新版本的Git提供了新的git switch命令来切换分支：

创建并切换到新的dev分支，可以使用：

$ git switch -c dev

直接切换到已有的master分支，可以使用：

$ git switch master

使用新的git switch命令，比git checkout要更容易理解。

### 小结

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>

### 小结

* 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。

标签管理

创建标签

在Git中打标签非常简单，首先，切换到需要打标签的分支上：

$ git branch

\* dev

master

$ git checkout master

Switched to branch 'master'

然后，敲命令git tag <name>就可以打一个新标签：

$ git tag v1.0

可以用命令git tag查看所有标签：

$ git tag

v1.0

可以用git show <tagname>查看标签信息。

还可以创建带有说明的标签，用-a指定标签名，-m指定说明文字：

$ git tag -a v0.1 -m "version 0.1 released" 1094adb

### 小结

* 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。

### 小结

* 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。