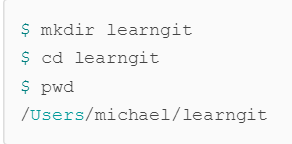
Git：版本控制系统，能允许别人编辑，记录每次文件的改动

局域网是在一个局部的地理范围内将各种计算机，外部设备和数据库等互相联接起来组成的计算机通信网。

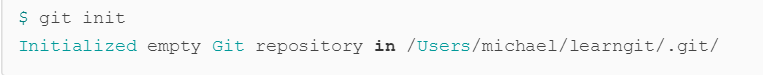
集中式版本控制系统，必须联网才能工作，版本库是集中存放在中央服务器的，

**创建仓库：**

首先，选择一个合适的地方，创建一个空目录：



第二步，通过git init命令把这个目录变成Git可以管理的仓库：



如果你没有看到.git目录，那是因为这个目录默认是隐藏的，用ls -ah命令就可以看见。

版本控制系统没法跟踪Word文件的改动的，要以纯文本方式编写文件。建议你下载[Notepad++](http://notepad-plus-plus.org/" \t "_blank)代替记事本，

**把一个文件放到Git仓库只需要两步。**

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。

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。

**版本回退**

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，使用命令git reset --hard commit\_id。

穿梭前，用git log（git log命令显示从最近到最远的提交日志）可以查看提交历史，以便确定要回退到哪个版本。

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。

需要提交的文件修改通通放到暂存区，然后，一次性提交暂存区的所有修改。

为什么Git比其他版本控制系统设计得优秀，因为Git跟踪并管理的是修改，而非文件。

**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://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/0013744142037508cf42e51debf49668810645e02887691000" \t "_blank)一节，不过前提是没有推送到远程库。

:q 退出 git add \* 添加目录里全部文件到暂存区

，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

git commit -m <message>， git reset --hard commit\_id。

git checkout -- file。 git diff HEAD -- readme.txt

**删除文件**

直接在文件管理器中把没用的文件删了，或者用rm命令删除

（1）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。

（2）如果误删，用git checkout撤销删除

**远程仓库(**GitHub上的仓库既可以作为备份，又可以让其他人通过该仓库来协作**)**

第1步：创建SSH Key

打开Shell（Windows下打开Git Bash）

$ ssh-keygen -t rsa -C [youremail@example.com](mailto:youremail@example.com)

【cd ~(进入用户主目录)

cd .ssh(进入.ssh目录)

ls(列出.ssh目录的文件，可以看到id\_rsa id\_rsa.pub两个文件)

cat ~/.ssh/id\_rsa.pub(可以看到id\_rsa.pub文件的内容，复制即可)】

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：

* $ git remote -v

**添加Ô¶³Ì²Ö¿â**

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填入仓库名（origin），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：

在本地的learngit仓库下运行命令：

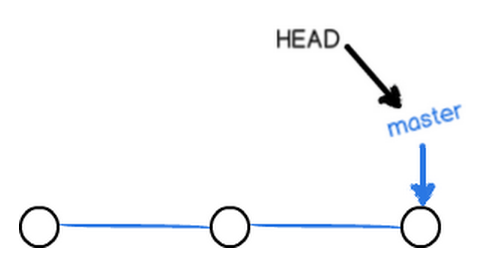
$ git remote add origin [git@github.com:michaelliao/learngit.git(账户名/](mailto:git@github.com:michaelliao/learngit.git(账户名/)仓库名.git) 来关联这个远程库

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；

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

**创建于合并分支**

HEAD严格来说不是指向提交，而才是指向提交的，所以，HEAD指向的就是当前分支，Git创建一个分支，即增加一个指针，切换分支实际上是改变HEAD的指向，使HEAD指向新的指针， 

查看分支：git branch

删除分支：git branch -d <name>（git branch -D <name>强行删除）

创建分支：git branch <name>

切换分支：git checkout <name>

创建+切换分支：git checkout -b <name>

合并某分支到当前分支：git merge <name>

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。

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

**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，你可以多次stash，恢复的时候，先用git stash list查看，然后恢复指定的stash，用命令：$ git stash apply stash@{0}

**多人协作**

可以在另一台电脑（注意要把SSH Key添加到GitHub）或者同一台电脑的另一个目录下克隆

你的小伙伴要在dev分支上开发，就必须创建远程origin的dev分支到本地，于是他用这个命令创建本地dev分支：

git pull失败了，原因是没有指定本地dev分支与远程origin/dev分支的链接，根据提示，设置dev和origin/dev的链接：

小结：

$ git branch --set-upstream-to=origin/dev dev

$ git checkout -b dev origin/dev

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。

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。