日期：2019年2月27日

# 创建一个新的版本库：

git init



创建完成，且告诉我们是一个空的仓库(empty Git repository)，而且在当前的目录下多了一个隐藏文件夹.git

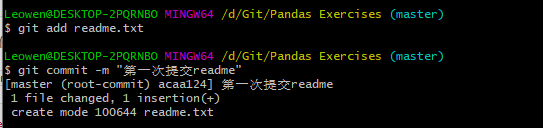
# 把文件添加到版本库中：

git add

这个命令是告诉git，把文件添加到仓库，

git commit -m

这个命令是告诉git，把文件提交到仓库，-m是本次提交的说明



# 小结：

初始化一个git仓库，使用git init命令

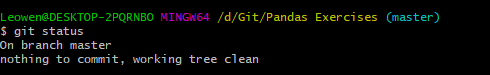
添加文件到git仓库，分为两步：

1. 使用git add <file>，可以反复多次使用，添加多个文件。
2. 使用git commit -m <message>,完成

# 显示工作路径下已经修改的文件：

git status

这个命令让我们可以时刻掌握仓库当前的状态



# 显示与上次提交版本文件的不同：

git diff

# 小结：

要随时掌握工作区的状态，使用git status 命令。

如果git status告诉有文件被修改过，使用git diff可以查看修改内容

# 提交历史

git log

这个命令显示从最近到最远的提交记录(显示所有的提交记录)

显示所有提交(仅显示提交的hash和message)

git log --pretty=oneline



前面一大串数字是**commit id**(版本号)，是一个SHA1计算出来的。

在git中，用**HEAD** 表示当前版本，也就是最新提交的版本，用**HEAD^**表示上一个版本，而上上个版本就是**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**

回退到上一个版本：

git reset –hard HEAD^

git 提供了一个命令

git reflog

用来记录你的每一次命令

# 小结：

HEAD指向的版本就是当前的版本，因此，git允许我们在版本之间穿梭，使用命令git reset –hard commit\_id。

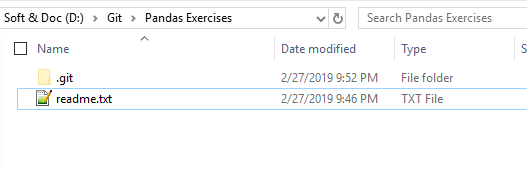
穿梭前，用git log可以查看提交历史，一边确定要回退到哪一个版本

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

# 工作区和暂缓区

## **工作区（Working Directory）**

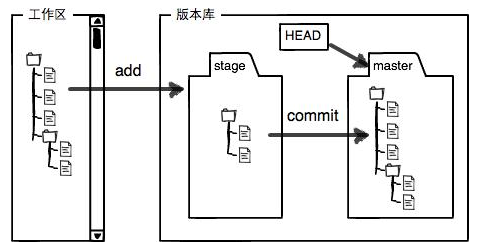
就是我们初始化的目录，这里，Pandas Exercises文件夹就是一个工作区。



## **版本库（Repository）**

工作区里面有一个隐藏的目录.git,这个不算工作区，而是git的版本库。

Git的版本库里面存了很多东西，其中最重要的讲就是称为**state**(或者叫**index**)的暂缓区，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**HEAD**。



前面讲了我们把文件往Git版本库里添加的时候，是分两步执行的：

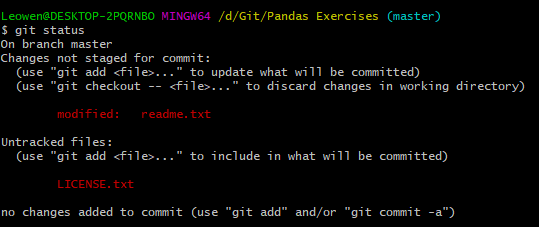
第一步是用git add把文件添加进去，实际上就是把文件修改添加到暂存区；

第二步是用git commit提交更改，其实是把暂存区的所有内容提交到当前分支

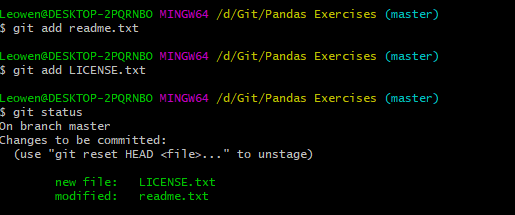
因为我们创建Git版本库时，Git自动为我们创建了唯一一个**master**分支，所以，现在，git commit就是往**master**分支上提交更改。

接下来，readme.txt进行修改，然后在工作区新增一个LICENSE文件

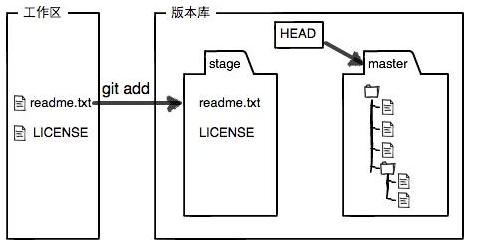
用git status查看一下状态



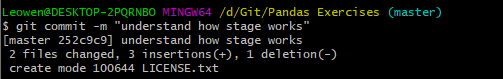
我们可以清楚的看到，readme.txt被修改过，而LICENSE还从来没有被添加过，所以它的状态是Untracked。现在使用两次git add，把readme.txt和LICENSE都添加后，用git status再查看一下：



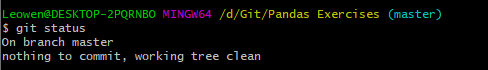
现在，暂存区的状态就变成这样了：



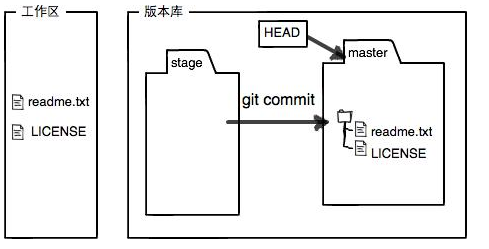
所以，git add命令实际上就是把要提交的所有修改放到暂存区(Stage)，然后，执行git commit就可以一次性把暂存区的所有修改提交到分支。



一旦提交后，如果你又没有对工作区做任何修改，那么工作区就是“干净”的：



现在版本库变成了这样，暂存区就没有任何内容了：



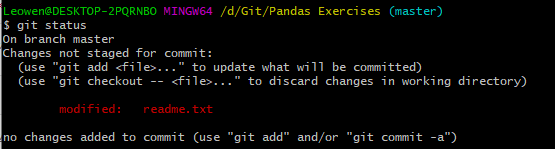
日期：2019年2月28日

# 管理修改

Git跟踪并管理的是修改，而非文件。每次修改，如果不用git add命令将文件添加到暂缓区，那么久不会加入到commit中。

# 撤销修改

在readme.txt中添加了一行“愚笨的人类”，也许发现这并不太好，于是可以删除最后一行，手动将文件恢复到上一个版本。可以用git status查看一下



可以使用

git checkout -- file

该命令可以丢弃工作区的修改



上面的意思是：把readme.txt文件在工作区的修改全部撤销，这里有两种情况：

一种是readme.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态。

一种是readme.txt已经添加到暂存区后，又做了修改，现在，撤销修改就回到添加到暂存区后的状态。

git checkout -- file命令中的--很重要，没有--，就变成了“切换到另一个分支”的命令。

如果使用git add将修改的文件添加到了暂存区，但是还没有commit，这个时候可以用命令git reset HEAD <file>可以把暂存区的修改撤销掉(unstage)，重新放回工作区。

git reset命令既可以回退版本，也可以把暂存区的修改回退到工作区。当我们用HEAD时，表示最新的版本。

# 小结：

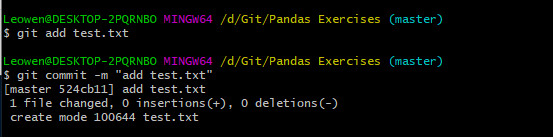
场景1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令git checkout -- file。

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。

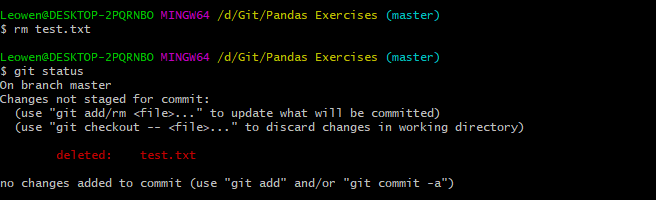
场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考版本回退一节，不过前提是没有推送到远程库。

# 删除文件

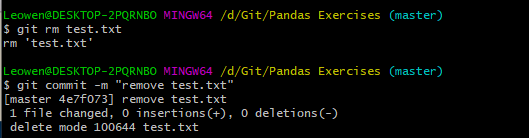
先添加一个新文件test.txt到Git并且提交：



一般情况下，你通常直接在文件管理器中把没用的文件删了，或者用rm命令删了，这个时候，Git知道你删除了文件，因此，**工作区和版本库就不一致了**，git status命令会立刻告诉你哪些文件被删除了：



现在你有两个选择，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：



另一种情况是删错了，因为版本库里还有呢，所以可以很轻松地把误删的文件恢复到最新版本：git checkout – test.txt

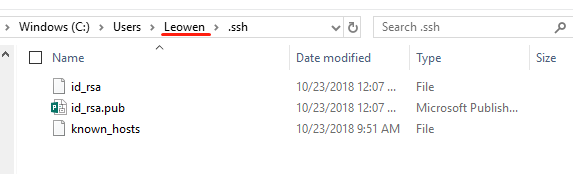
git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。

# 远程仓库

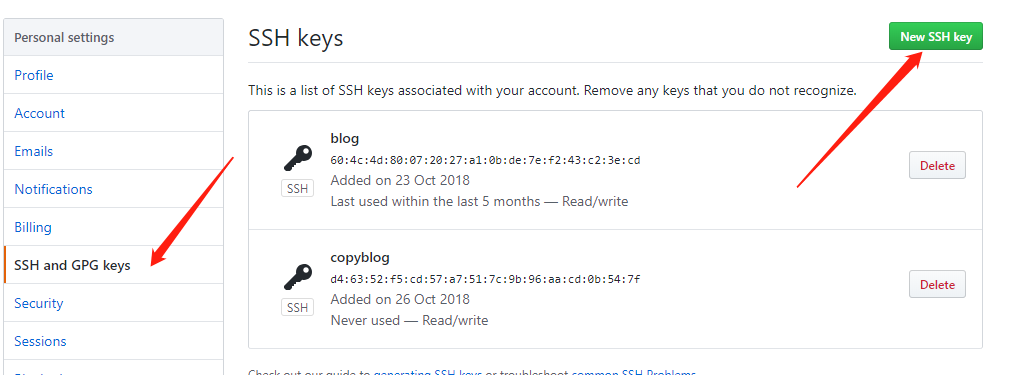
第1步：**创建SSH Key**。在用户主目录下，看看有没有**.ssh**目录，如果有，再看看这个目录下有没有**id\_rsa**和**id\_rsa.pub**这两个文件，如果已经有了，可直接跳到下一步。如果没有，打开Shell（Windows下打开Git Bash），创建SSH Key：

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：

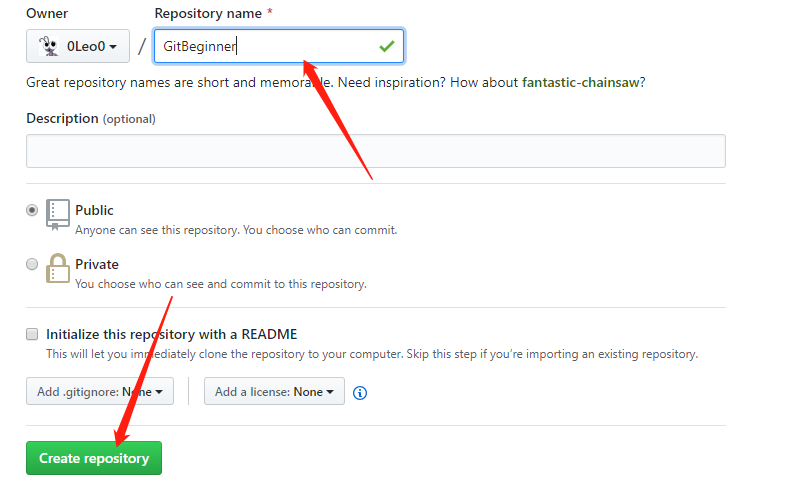


然后，点“Add SSH Key”，填上任意Title，在Key文本框里粘贴id\_rsa.pub文件的内容即可。

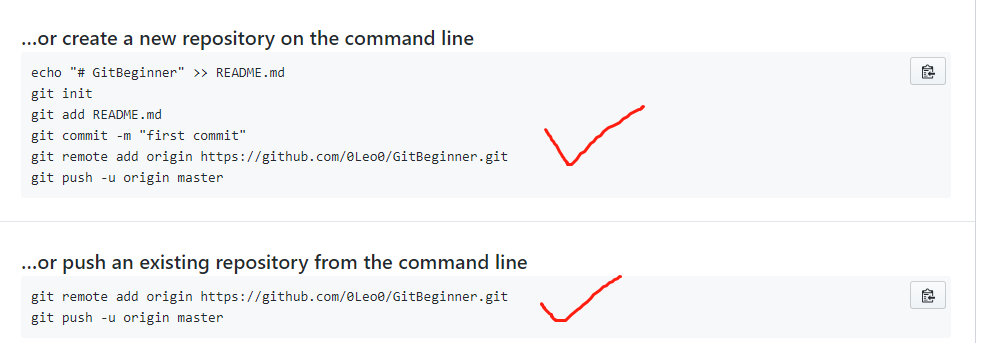
# 添加远程库

现在的情景是，你已经在本地创建了一个Git仓库后，又想在GitHub创建一个Git仓库，并且让这两个仓库进行远程同步，这样，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，真是一举多得。

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：



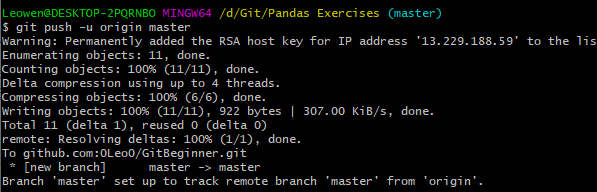
现在这个仓库是空的，我们可以根据提示，将本地仓库的内容推送到GitHub上





添加后，远程库的名字就是origin，这是Git默认的叫法，也可以改成别的，但是origin这个名字一看就知道是远程库。

下一步，就可以把本地库的所有内容推送到远程库上：



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。

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，在以后的推送或者拉取时就可以简化命令。

自第一次提交后，从现在起，只要本地作了提交，就可以通过命令：

git push origin master

把本地master分支的最新修改推送至GitHub，现在，你就拥有了真正的分布式版本库！

# 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；

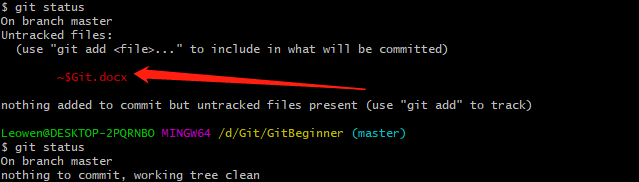
# 使用小记：

日期：2019年2月28日

1、删除本地仓库：删除隐藏的文件夹.git即可。

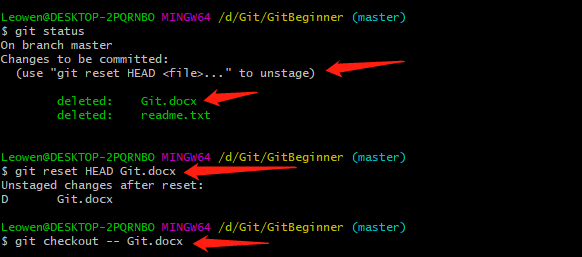
2、第一次推送到远程库后，后面似乎关联上了，不能更改本地与远程仓库的连接。

出现下面的原因：



是因为我打开了word，git检测到了。

3、本地仓库误删除了Git.docx，远程库也删除了，如何还原？



4、远程仓库中的README.md文件不在本地代码目录中

**5**、github上的版本和本地版本冲突的解决方法

<https://blog.csdn.net/shiren1118/article/details/7761203>

git push -u origin master -f