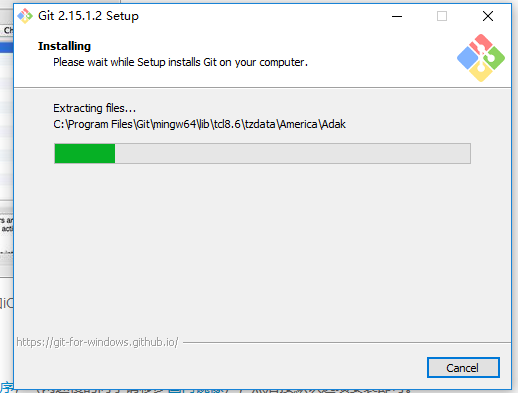
<https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/00137396287703354d8c6c01c904c7d9ff056ae23da865a000>

## 安装git

下载git

<https://git-scm.com/downloads>

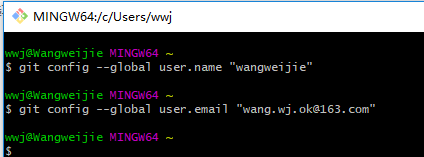
默认勾选安装



安装完成后，还需要最后一步设置，在命令行输入：

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

$ git config --global user.email "email@example.com"



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。

## 创建版本库

新建文件夹learngit,进入该文件夹，通过git init命令把这个目录变成Git可以管理的仓库



可以新建一个readme.txt的文件，内容如下：

Git is a version control system.

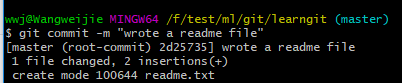
Git is free software.

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；



1. 使用命令git commit -m <message>，完成。



## 基本操作

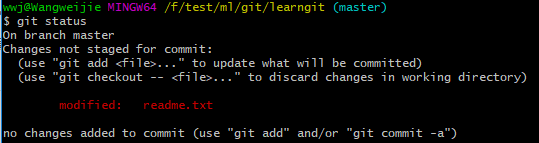
### 状态查看

修改readme.txt文件，改成如下内容：

Git is a distributed version control system.

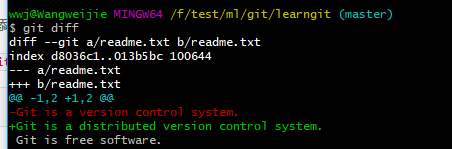
Git is free software.

git status命令可以让我们时刻掌握仓库当前的状态

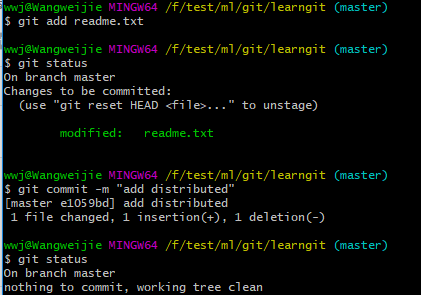


如果git status告诉你有文件被修改过，用git diff可以查看修改内容。

git diff顾名思义就是查看difference，显示的格式正是Unix通用的diff格式



提交修改和提交新文件是一样的两步，同时可以用git status查看状态变化



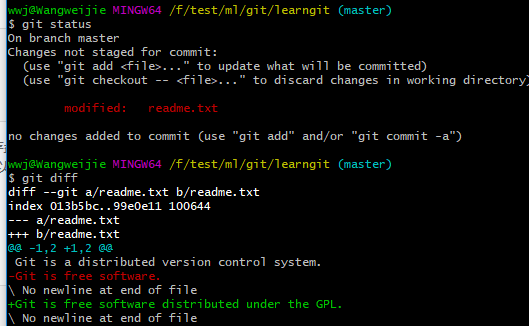
### 版本回退

修改readme.txt文件如下：

Git is a distributed version control system.

Git is free software distributed under the GPL.

可以查看一下状态和变化

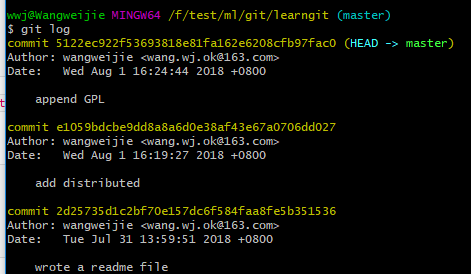


进行添加和提交：

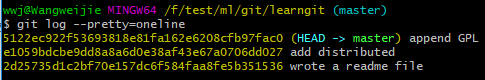




git log命令显示从最近到最远的提交日志



如果嫌输出信息太多，看得眼花缭乱的，可以试试加上--pretty=oneline参数：



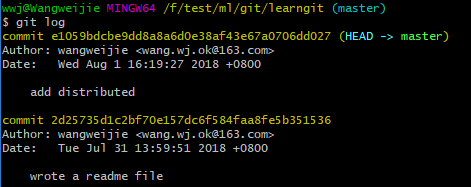
在Git中，用HEAD表示当前版本，也就是最新的提交1094adb...（注意我的提交ID和你的肯定不一样），上一个版本就是HEAD^，上上一个版本就是HEAD^^，当然往上100个版本写100个^比较容易数不过来，所以写成HEAD~100。

要把当前版本append GPL回退到上一个版本add distributed，就可以使用git reset命令：



看看readme.txt的内容是不是版本add distributed：

用git log再看看现在版本库的状态：



Git的版本回退速度非常快，因为Git在内部有个指向当前版本的HEAD指针，当你回退版本的时候，Git仅仅是把HEAD从指向append GPL：

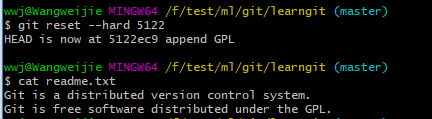


改为指向add distributed：

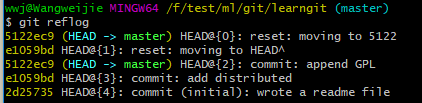


然后顺便把工作区的文件更新了。所以你让HEAD指向哪个版本号，你就把当前版本定位在哪。

找到那个append GPL的commit id是5122...，于是就可以指定回到未来的某个版本：



Git提供了一个命令git reflog用来记录你的每一次命令：



从而可以看到那个append GPL的commit id是5122...

### 工作区和暂存区

#### 工作区（Working Directory）

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，还有Git为我们自动创建的第一个分支master，以及指向master的一个指针叫HEAD。

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，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。

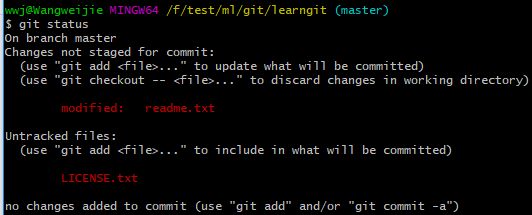
再练习一遍，先对readme.txt做个修改，比如加上一行内容：

Git is a distributed version control system.

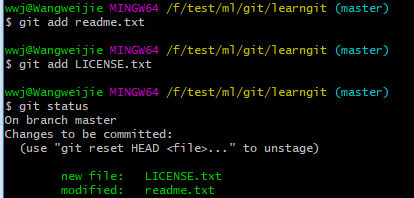
Git is free software distributed under the GPL.

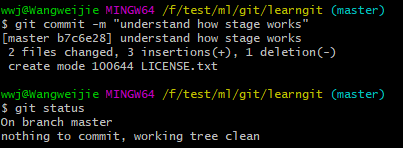
Git has a mutable index called stage.

然后，在工作区新增一个LICENSE文本文件（内容随便写）。

先用git status查看一下状态：

使用两次命令git add，把readme.txt和LICENSE都添加后，用git status再查看一下：



执行git commit就可以一次性把暂存区的所有修改提交到分支

### 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到暂存区，那就不会加入到commit中。

对readme.txt做一个修改，比如加一行内容：

$ cat readme.txt

Git is a distributed version control system.

Git is free software distributed under the GPL.

Git has a mutable index called stage.

Git tracks changes.

进行添加：



再修改readme.txt：

$ cat readme.txt

Git is a distributed version control system.

Git is free software distributed under the GPL.

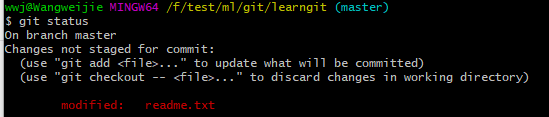
Git has a mutable index called stage.

Git tracks changes of files.

提交：



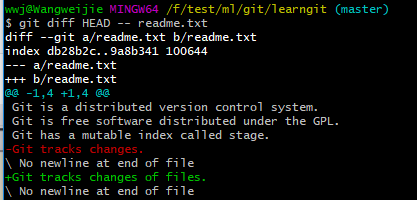
查看状态：



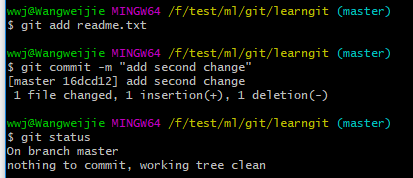
操作过程：

第一次修改 -> git add -> 第二次修改 -> git commit。git commit只负责把暂存区的修改提交了，也就是第一次的修改被提交了，第二次的修改不会被提交。

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：



再次添加并提交，即可：



### 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。

在readme.txt中添加了一行：

$ cat readme.txt

Git is a distributed version control system.

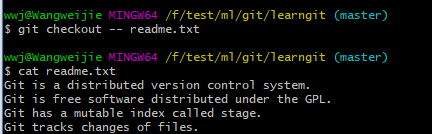
Git is free software distributed under the GPL.

Git has a mutable index called stage.

Git tracks changes of files.

My stupid boss still prefers SVN.

git checkout -- file可以丢弃工作区的修改：

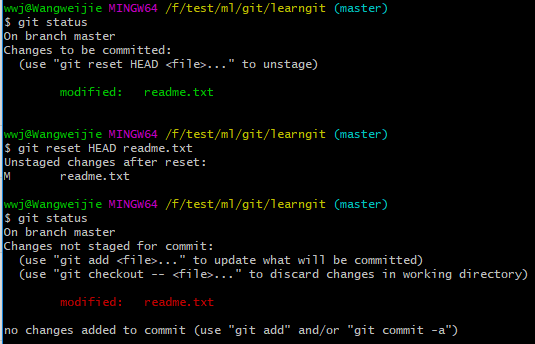


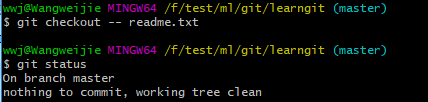
命令git checkout -- readme.txt意思就是，把readme.txt文件在工作区的修改全部撤销，这里有两种情况：

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，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。

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。

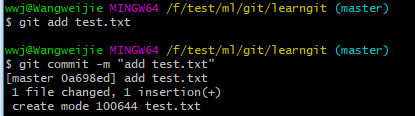




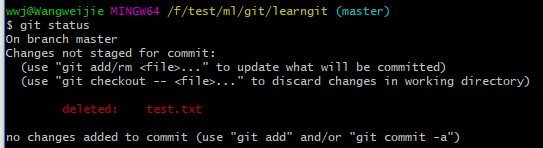
场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考版本回退小节。

### 删除文件

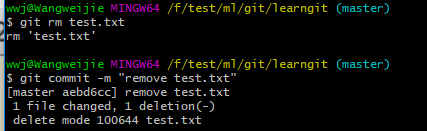
先添加一个新文件test.txt到Git并且提交：



在文件管理器中把test.txt文件删了，git status命令会立刻告诉你哪些文件被删除了：



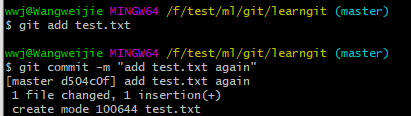
如果确实要从版本库中删除该文件，那就用命令git rm删掉，并且git commit：



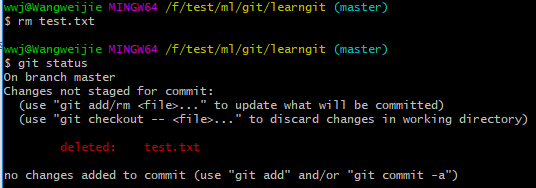


如果删错了，因为版本库里还有呢，所以可以很轻松地把误删的文件恢复到最新版本：

重新添加一个新文件test.txt到Git并且提交：



在文件管理器中把test.txt文件删了



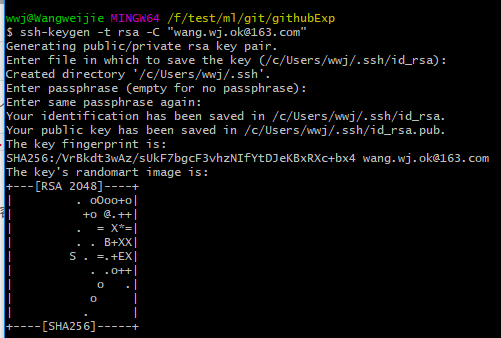
git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。

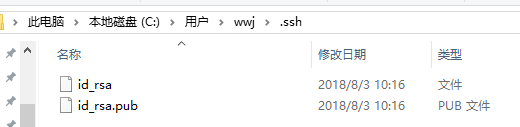


## 远程仓库

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：

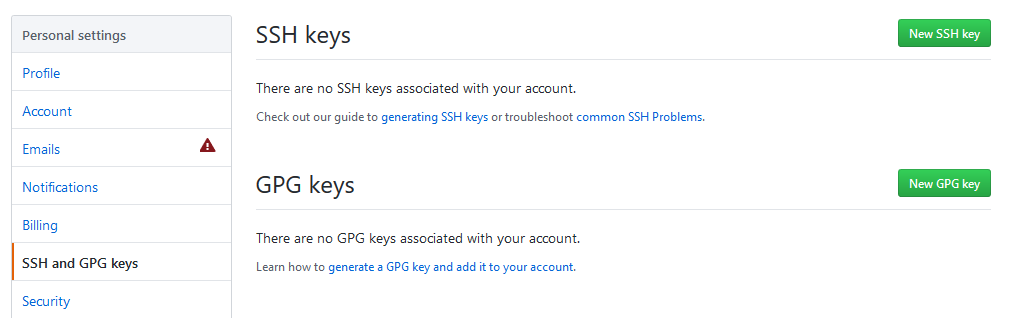
$ ssh-keygen -t rsa -C "youremail@example.com"

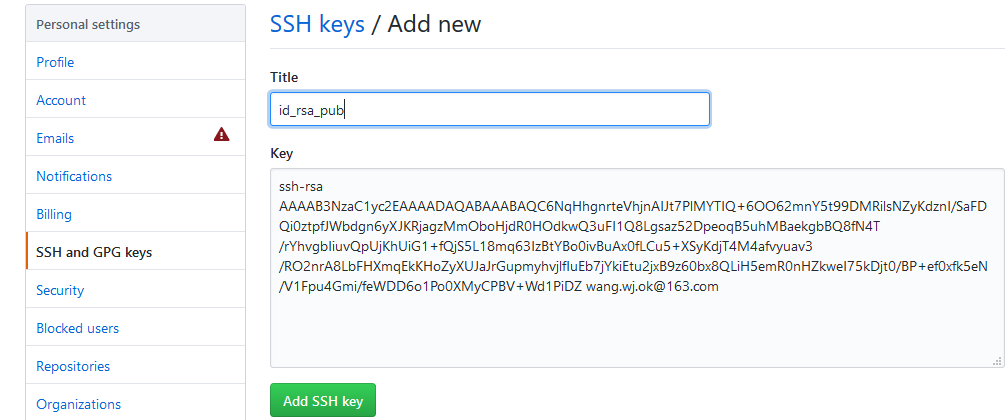




第2步：登陆GitHub，打开“Account settings”，“SSH Keys”页面：

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：





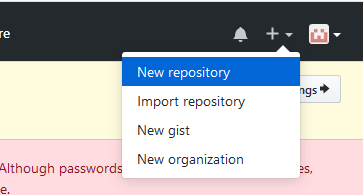
Git支持SSH协议，所以，GitHub只要知道了你的公钥，就可以确认只有你自己才能推送。

在GitHub上免费托管的Git仓库，任何人都可以看到。

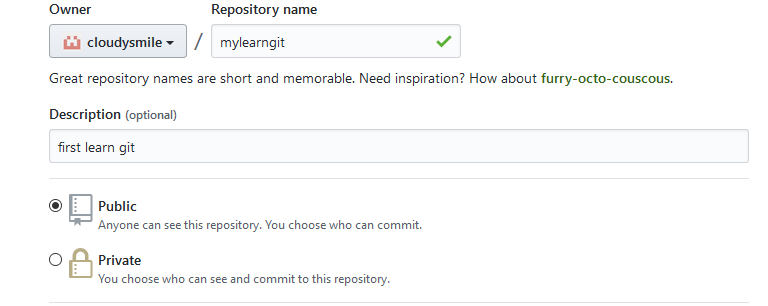
### 添加远程库

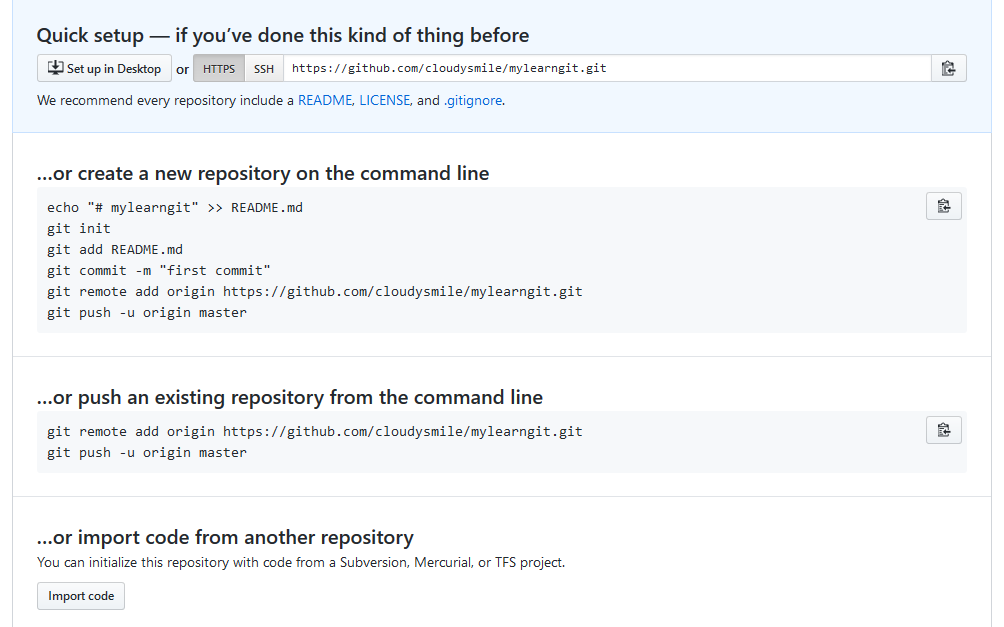
在本地创建了一个Git仓库后，又想在GitHub创建一个Git仓库，并且让这两个仓库进行远程同步，这样，GitHub上的仓库既可以作为备份，又可以让其他人通过该仓库来协作。

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mylearngit，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：





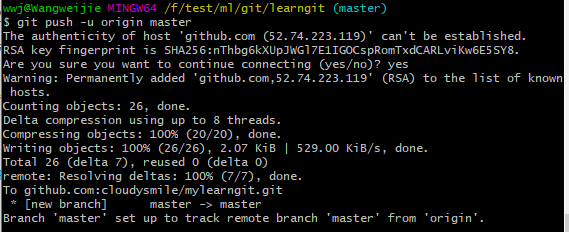
根据GitHub的提示，在本地的learngit仓库下运行命令：

$ git remote add origin [git@github.com:michaelliao/learngit.git](mailto:git@github.com:michaelliao/learngit.git)



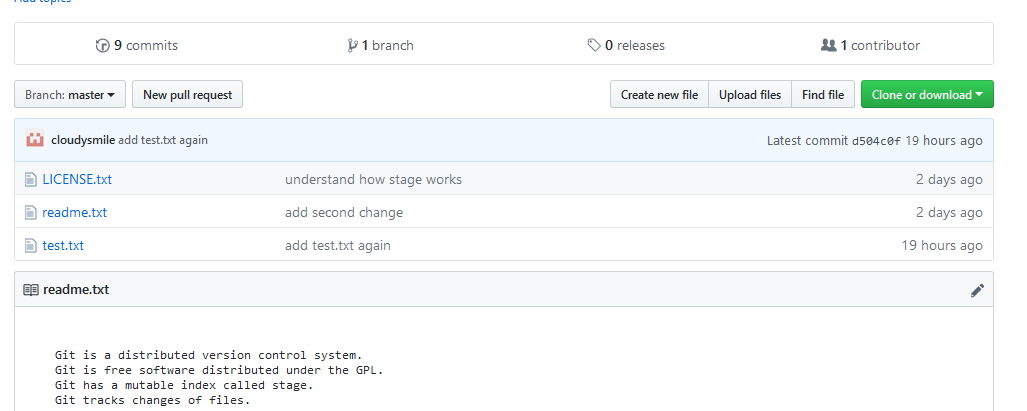
添加后，远程库的名字就是origin，这是Git默认的叫法，也可以改成别的，但是origin这个名字一看就知道是远程库。

下一步，就可以把本地库的所有内容推送到远程库上：



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。

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；

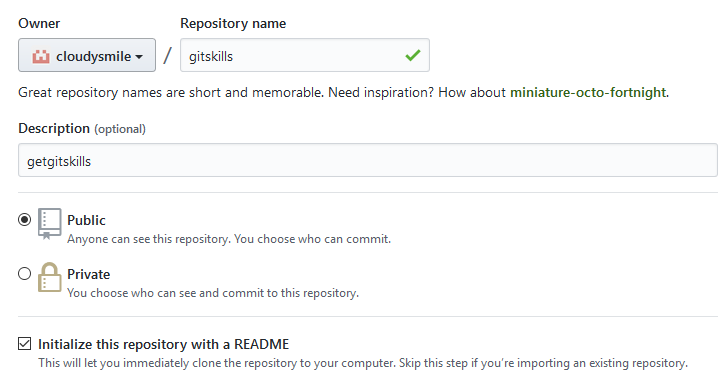
此后，每次本地提交后，只要有必要，就可以使用命令git push origin master推送最新修改；

### 从远程库克隆

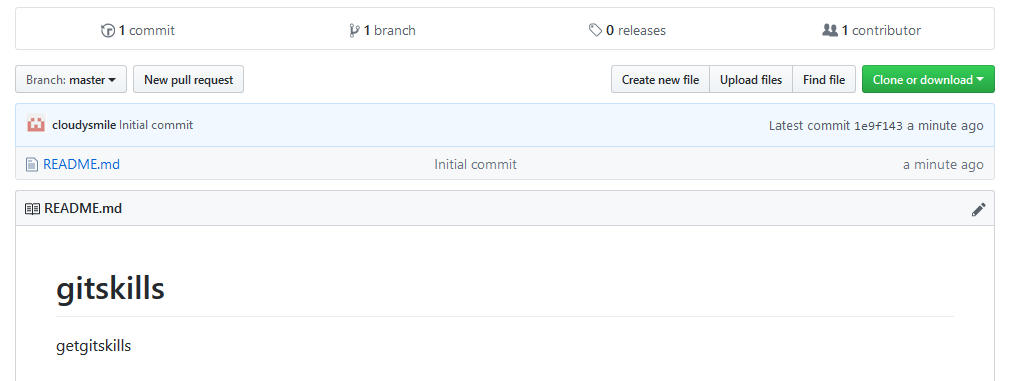
上次我们讲了先有本地库，后有远程库的时候，如何关联远程库。

现在，假设我们从零开发，那么最好的方式是先创建远程库，然后，从远程库克隆。

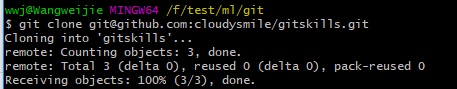
首先，登陆GitHub，创建一个新的仓库，名字叫gitskills：



勾选Initialize this repository with a README，这样GitHub会自动为我们创建一个README.md文件。创建完毕后，可以看到README.md文件：



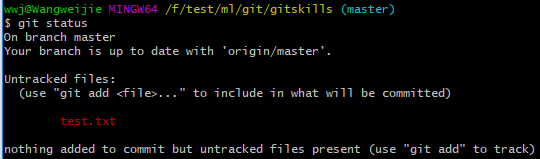
现在，远程库已经准备好了，下一步是用命令git clone克隆一个本地库：



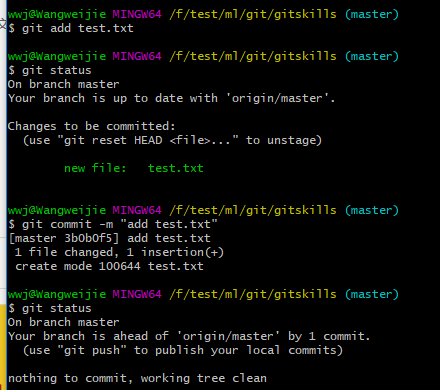
Git支持多种协议，默认的git://使用ssh，但也可以使用https等其他协议。

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。

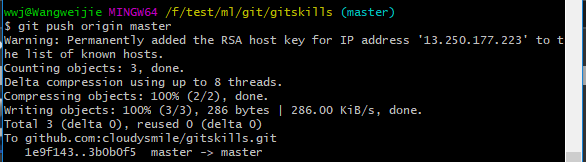
可以试着在本地创建test.txt文件



添加并提交：



推送最新修改到github上：



1. 分支管理