1创建版本库(添加、修改文件)：

$ 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:

$ git init

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

添加文件到Git仓库，分两步：

* 第一步，使用命令git add <file>，注意，可反复多次使用，添加多个文件；
* 第二步，使用命令git commit，完成。

git status命令可以让我们时刻掌握仓库当前的状态。

git diff顾名思义就是查看两次提交的difference

cat txt文件名 查看.txt的内容

2 版本回退

git log 或 git log –pretty=oneline命令显示从最近到最远的提交日志。

把当前版本“append GPL”回退到上一个版本：git reset –hard HEAD^

当你用$ git reset --hard HEAD^回退到add distributed版本时，再想恢复到append GPL，就必须找到append GPL的commit id。Git提供了一个命令git reflog用来记录你的每一次命令：以便确定要回到未来的哪个版本。

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。每次修改，如果不add到暂存区，那就不会加入到commit中。

3 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)一节，不过前提是没有推送到远程库。

3 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命令删了：

$ rm test.txt

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"

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

4 远程仓库

创建SSH 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"

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：



点“Add Key”，你就应该看到已经添加的Key：



本地仓库与GitHub上的仓库进行关联

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

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，但通过ssh支持的原生git协议速度最快。

5 分支管理

创建与合并分支

查看分支：git branch

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> 这次合并是“快进模式”，也就是直接把master指向dev的当前提交，所以合并速度非常快。（master中有，dev中没有，合并后master中也会丢失）

删除分支：git branch -d <name>

解决冲突

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。

用git log --graph命令可以看到分支合并图。