创建目录，通过git init命令把这个目录变成Git可以管理的仓库：

$ git init

git add filename

git commit -m “commit message”

git status –查看仓库当前状态

git diff

git log /git log --pretty=oneline

git reset --hard HEAD^ --回退到上一版

* 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。

工作区（Working Directory）

就是你在电脑里能看到的目录

**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。



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。



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。

每次修改，如果不add到暂存区，那就不会加入到commit中

本地文件修改或新加，没有提交，需回滚：

git clean -df --只删除所有untracked的文件(如新增加的文件)

git checkout -f --删除tracked文件中新增加的内容(已有文件新增加的内容)  
git reset --hard --只把tracked的文件revert到前一个版本

(实验了下2和3结果类似，不知还有啥区别)

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。

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 file

git commit -m “delete file”

现在，文件就从版本库中被删除了。

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

GITHUB

第1步：创建SSH Key

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：



关联github远程仓库，可看github上的介绍 。

Github上创建axxx仓库

在本地axxx仓库下运行

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

$ git push -u origin master