Git是目前世界上最先进的**分布式版本控制系统**。

所有的版本控制系统只能跟踪文本文件的改动，比如TXT文件，网页，所有的程序代码等等。而图片、视频这些二进制文件，虽然也能由版本控制系统管理，但没法跟踪文件的变化，只能把二进制文件每次改动串起来，也就是只知道图片从100KB改成了120KB，但到底改了啥，版本控制系统就不知道了。

Microsoft的Word格式是二进制格式，因此，版本控制系统是没法跟踪Word文件的改动的。如果要使用版本控制系统，就要以纯文本方式编写文件。文本是有编码的，建议使用标准的UTF-8编码，所有语言使用同一种编码，既没有冲突，又被所有平台所支持。

Windows自带的**记事本**在每个文件开头添加了0xefbbbf（十六进制）的字符，会遇到很多不可思议的问题。建议用[Notepad++](http://notepad-plus-plus.org/" \t "https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/_blank)代替记事本。

#### 一、创建版本库/仓库/****repository：****

1、创建一个空目录：**（**确保目录名（包括父目录）不包含中文。**）**

$ mkdir learngit

$ cd learngit

$ pwd

/Users/michael/learngit

pwd命令用于显示当前目录。

1. 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/

.git目录是Git用来跟踪管理版本库的（默认隐藏）。用ls -ah命令就可以看见。

### 二、把文件添加到版本库：

1、把文件添加到仓库：

$ git add readme.txt

执行上面的命令没有任何显示，说明添加成功。

2、把文件提交到仓库：

$ git commit -m "wrote a readme file"

[master (root-commit) eaadf4e] wrote a readme file

1 file changed, 2 insertions(+)

create mode 100644 readme.txt

git commit命令，-m后面输入的是本次提交的说明。

git commit命令执行成功后会告诉你，1 file changed：1个文件被改动（我们新添加的readme.txt文件）；2 insertions：插入了两行内容（readme.txt有两行内容）。

可以多次add不同文件，一次commit很多文件：

$ git add file1.txt

$ git add file2.txt file3.txt

$ git commit -m "add 3 files."

1. **修改readme.txt文件：**

Git is a distributed version control system.

Git is free software.

1、查看仓库当前的状态：git status命令

1. readme.txt被修改过了，但还没有准备提交的修改。

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: readme.txt

no changes added to commit (use "git add" and/or "git commit -a")

1. git add后：将要被提交的修改包括readme.txt

$ git status

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

modified: readme.txt

1. git add并且git commit后：当前没有需要提交的修改，而且工作目录是干净（working tree clean）的。

$ git status

On branch master

nothing to commit, working tree clean

2、查看具体修改了什么内容：git diff命令

$ git diff readme.txt

diff --git a/readme.txt b/readme.txt

index 46d49bf..9247db6 100644

--- a/readme.txt

+++ b/readme.txt@@ -1,2 +1,2 @@

-Git is a version control system.

+Git is a distributed version control system.

Git is free software.

1. 提交修改和提交新文件是一样的

#### 版本回退

1、查看提交日志（最近到最远）：git log命令

（嫌输出信息太多：--pretty=oneline参数）

$ git log --pretty=oneline

1094adb7b9b3807259d8cb349e7df1d4d6477073 (HEAD -> master) append GPL

e475afc93c209a690c39c13a46716e8fa000c366 add distributed

eaadf4e385e865d25c48e7ca9c8395c3f7dfaef0 wrote a readme file

一大串类似1094adb...的是commit id（版本号）

每提交一个新版本，实际上Git就会把它们自动串成一条时间线。如果使用可视化工具查看Git历史，就可以更清楚地看到提交历史的时间线。

1. 版本回退：git reset命令

在Git中，用HEAD表示当前版本，上一个版本就是HEAD^，上上一个版本就是HEAD^^，往上100个版本写成HEAD~100。

$ git reset --hard HEAD^

HEAD is now at e475afc add distributed

查看文件内容：

$ cat readme.txt

Git is a distributed version control system.

Git is free software.

指定回到某个版本：输入commit id的前几位

$ git reset --hard 1094a

HEAD is now at 83b0afe append GPL

查看命令历史：git reflog命令

#### 五、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。

**暂存区（stage或index）：Git的版本库里存了很多东西，其中最重要的就是暂存区，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。

#### 六、管理修改

Git比其他版本控制系统设计得更优秀是在于Git跟踪并管理的是修改，而非文件。

查看工作区和版本库里面最新版本的区别：git diff HEAD -- readme.txt命令

第一次修改 -> git add -> 第二次修改 -> git commit，这样第二次修改不会被提交。

#### 撤销修改

1. 把readme.txt文件在工作区的修改全部撤销：命令git checkout -- readme.txt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。

2、把暂存区的修改撤销掉（unstage），重新放回工作区：命令git reset HEAD <file>

$ git reset HEAD readme.txt

Unstaged changes after reset:

M readme.txt

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

#### 删除文件

#### 直接在文件管理器中删除文件，或者用rm命令删除：$ rm test.txt

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：

$ git rm test.txt

rm 'test.txt'

$ git commit -m "remove test.txt"

[master d46f35e] remove test.txt

1 file changed, 1 deletion(-)

delete mode 100644 test.txt

二、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

#### 远程仓库：GitHub

本地Git仓库和GitHub仓库之间的传输是通过SSH加密的。

要关联一个远程库，使用命令：

git remote add origin [git@server-name:path/repo-name.git；](mailto:git@server-name:path/repo-name.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。

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

$ git clone [git@github.com:wsywwww/gitskills.git](mailto:git@github.com:michaelliao/gitskills.git)

GitHub给出的地址不止一个，还可以用:

<https://github.com/wsywwww/gitskills.git>

这样的地址。实际上，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。

**分支管理**

HEAD指向的就是当前分支。

master分支即主分支（？master指向主分支的当前分支）

创建并切换到dev分支：

$ git checkout -b dev

Switched to a new branch 'dev'

相当于以下两条命令：

创建dev分支：$ git branch dev

切换到dev分支：$ git checkout dev

然后，用git branch命令查看当前分支：

$ git branch

\* dev

master

git branch命令会列出所有分支，当前分支前面会标一个\*号。

把dev分支的工作成果合并到master分支上：

$ git merge dev

Updating d46f35e..b17d20e

Fast-forward

readme.txt | 1 +

1 file changed, 1 insertion(+)

git merge命令用于合并指定分支到当前分支。

注意到上面的Fast-forward信息，Git告诉我们，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。

合并完成后，就可以放心地删除dev分支了：

$ git branch -d dev

Deleted branch dev (was b17d20e).

删除后，查看branch，就只剩下master分支了：

$ git branch

\* master

#### 解决冲突