创建一个版本库非常简单，首先，选择一个合适的地方，创建一个空目录：

$ mkdir learngit

$ cd learngit

$ pwd

/Users/michael/learngit

pwd命令用于显示当前目录

如果你使用Windows系统，为了避免遇到各种莫名其妙的问题，请确保目录名（包括父目录）不包含中文。

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/

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。

千万不要使用Windows自带的**记事本**编辑任何文本文件。原因是Microsoft开发记事本的团队使用了一个非常弱智的行为来保存UTF-8编码的文件，他们自作聪明地在每个文件开头添加了0xefbbbf（十六进制）的字符，你会遇到很多不可思议的问题，比如，网页第一行可能会显示一个“?”，明明正确的程序一编译就报语法错误，等等，都是由记事本的弱智行为带来的。建议你下载Notepad++代替记事本，不但功能强大，而且免费！记得把Notepad++的默认编码设置为UTF-8 without BOM即可。

言归正传，现在我们编写一个readme.txt文件，内容如下：

Git is a version control system.

Git is free software.

一定要放到learngit目录下（子目录也行），因为这是一个Git仓库，放到其他地方Git再厉害也找不到这个文件。

第一步，用命令git add告诉Git，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，这就对了，Unix的哲学是“没有消息就是好消息”，说明添加成功。

第二步，用命令git commit告诉Git，把文件提交到仓库：

$ git commit -m "wrote a readme file"

[master (root-commit) cb926e7] wrote a readme file

1 file changed, 2 insertions(+)

create mode 100644 readme.txt

简单解释一下git commit命令，-m后面输入的是本次提交的说明，可以输入任意内容，当然最好是有意义的，这样你就能从历史记录里方便地找到改动记录。

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

为什么Git添加文件需要add，commit一共两步呢？因为commit可以一次提交很多文件，所以你可以多次add不同的文件，比如：

$ git add file1.txt

$ git add file2.txt

$ git add file3.txt

$ git commit -m "add 3 files."

我们已经成功地添加并提交了一个readme.txt文件，现在，是时候继续工作了，于是，我们继续修改readme.txt文件，改成如下内容：

Git is a distributed version control system.

Git is free software.

现在，运行git status命令看看结果：

$ 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")

git status命令可以让我们时刻掌握仓库当前的状态，上面的命令告诉我们，readme.txt被修改过了，但还没有准备提交的修改。

虽然Git告诉我们readme.txt被修改了，但如果能看看具体修改了什么内容，自然是很好的。需要用git diff这个命令看看：

$ 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.

提交修改和提交新文件是一样的两步，第一步是git add：

$ git add readme.txt

同样没有任何输出。在执行第二步git commit之前，我们再运行git status看看当前仓库的状态：

$ git status

# On branch master

# Changes to be committed:

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

#

# modified: readme.txt

#

git status告诉我们，将要被提交的修改包括readme.txt，下一步，就可以放心地提交了：

$ git commit -m "add distributed"

[master ea34578] add distributed

1 file changed, 1 insertion(+), 1 deletion(-)

提交后，我们再用git status命令看看仓库的当前状态：

$ git status

# On branch master

nothing to commit (working directory clean)

Git告诉我们当前没有需要提交的修改，而且，工作目录是干净（working directory clean）的。

版本控制系统肯定有某个命令可以告诉我们历史记录，在Git中，我们用git log命令查看。

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：

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。

我们要把当前版本回退到上一个版本，可以使用git reset命令：

$ git reset --hard HEAD^

HEAD is now at ...

你让HEAD指向哪个版本号，你就把当前版本定位在哪。

Git提供了一个命令git reflog用来记录你的每一次命令。

**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。

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，实际上就是把暂存区的所有内容提交到当前分支。

git checkout -- file可以丢弃工作区的修改：

$ git checkout -- readme.txt

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。

总之，就是让这个文件回到最近一次git commit或git add时的状态。

git checkout -- file命令中的--很重要，没有--，就变成了“创建一个新分支”的命令，我们在后面的分支管理中会再次遇到git checkout命令。

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 HEAD readme.txt

Unstaged changes after reset:

M readme.txt

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。

再用git status查看一下，现在暂存区是干净的，工作区有修改：

$ 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")

还记得如何丢弃工作区的修改吗？

$ git checkout -- readme.txt

$ git status

# On branch master

nothing to commit (working directory clean)

整个世界终于清静了！

在Git中，删除也是一个修改操作，我们实战一下，先添加一个新文件test.txt到Git并且提交：

$ git add test.txt

$ git commit -m "add test.txt"

[master 94cdc44] add test.txt

1 file changed, 1 insertion(+)

create mode 100644 test.txt

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：

$ rm test.txt

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：

$ git status

# On branch master

# Changes not staged for commit:

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

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

#

# deleted: test.txt

#

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

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 d17efd8] remove test.txt

1 file changed, 1 deletion(-)

delete mode 100644 test.txt

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

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”。