

# 上手 1：新公司用 Git 管理代码，怎么快速上手？

已经有 Git 经验的可以跳过这一节

刚进入新公司，被告知团队是用 Git 管理项目代码的，而你却从来没用过 Git。纵然同事告诉你「没事，先自学一下」「有问题可以问我」，但你肯定不想一进公司就花太多时间在自我学习上，也不想过多地打扰这些刚认识的同事。怎么办？

对你来说，最重要的是，**先知道 Git 怎么用**。先把最基本的掌握了，至于正规团队使用 Git 有什么额外要求、Git 有什么高级用法、Git 的原理这些，都可以放在这之后。万事开头难，你先把 Git 最基本的使用掌握了，后面的再一步一步来。

## 安装 Git

点击[这里 \(https://git-scm.com/\)](https://git-scm.com/)去下载个 Git，安装到你的机器上。或者如果你喜欢用 Homebrew 或 apt 什么的来安装都好，总之，把它安装好。

装好以后，就可以正式开始上手 Git 了。

## 先建个练习项目

学习的时候最好别拿团队的正式项目练手，先在 GitHub 上建一个自己的练习项目。


1. 访问 [GitHub \(https://github.com\)](https://github.com)（用别的平台比如 bitbucket 什么的也行）

2. 注册或登录您的账号
3. 点击右上角的「New Repository」来新建远程仓库
4. 进入仓库设置页面填写信息：其中 ① 是你的仓库名，这个仓库名同样会被 GitHub 设置为你的仓库的根目录的名称；② 是为 .gitignore 设置项目类型，.gitignore 是 Git 仓库中的一个特殊的文本文件，它里面记录了你不希望提交到仓库的目录和文件的名称或类型，例如你的 /build 目录；把 ① 和 ② 填好之后，就可以点 ③ 来完成远程仓库的创建了

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

 rengwuxian ▾

Repository name

git-practice ✓

①

Great repository names are short and memorable. Need inspiration? How about **psychic-system**.

Description (optional)

A practice repo for learning

☒ Public

Anyone can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: Android ▾

②

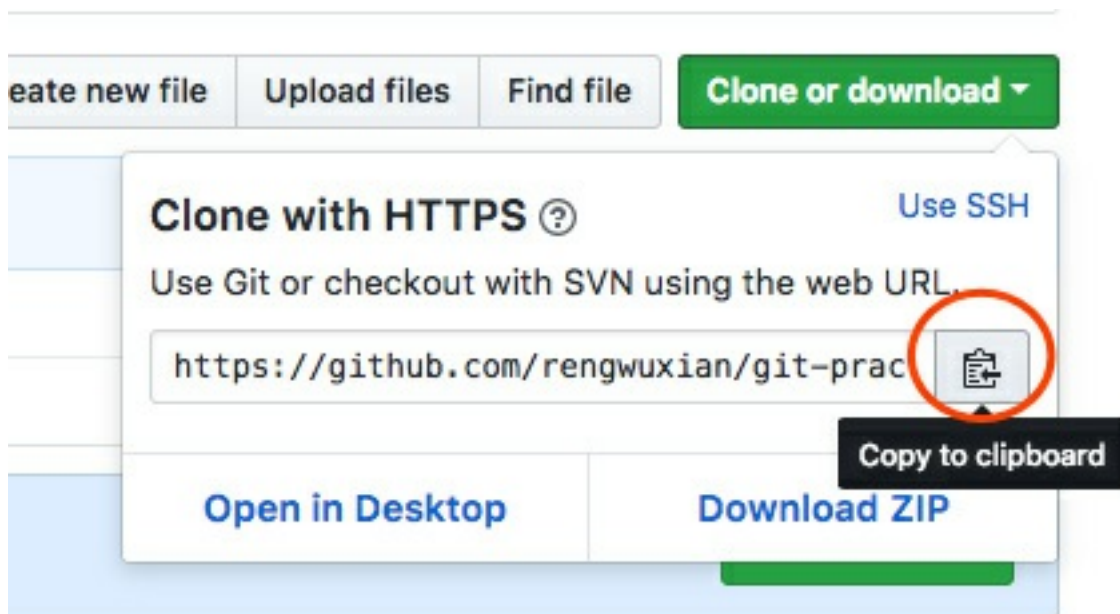
Add a license: Apache License 2.0 ▾ ⓘ

Create repository

③

创建完的远程仓库大概长这样：

点击右边的「Clone or download」，然后把仓库的 clone 地址复制到剪贴板：



clone 地址是什么？下面马上就说啦！

## 把远程仓库取到本地

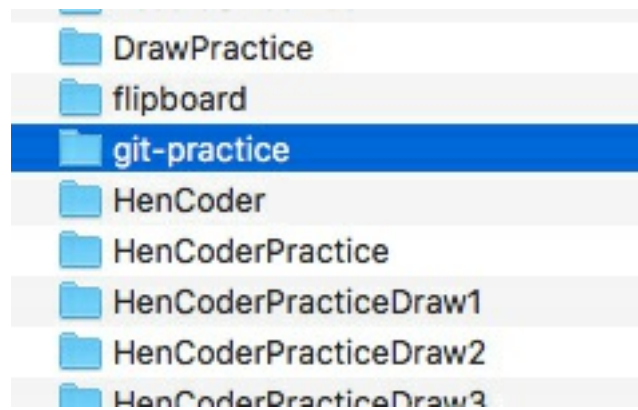
接下来就可以把远程仓库取下来了。取的方式很简单：在 Terminal 或 cmd 中切换到我希望放置项目的目录中，然后输入：

```
git clone 你刚复制的地址
```

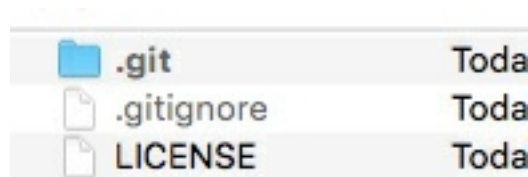
Git 就会把你的远程仓库 clone 到本地。在这个过程中，你可能会需要输入你的 GitHub 用户名和密码：

```
→ projects git clone https://github.com/rengwuxian/git-practice.git
Cloning into 'git-practice'...
warning: templates not found /Users/rengwuxian/bin/git-hooks
Username for 'https://github.com': rengwuxian
Password for 'https://rengwuxian@github.com':
```

输入正确的用户名和密码以后，你会看到你的当前目录下多了一个新的子目录，它的名字和刚才新建的 GitHub 仓库名一致：



进入这个目录，你会发现这里除了你刚才添加的 LICENSE 和 .gitignore 文件外，还有一个叫做 .git 的隐藏目录。



这个 .git 目录，就是你的**本地仓库 (Local Repository)**，你的所有版本信息都会存在这里。而 .git 所在的这个根目录，称为 Git 的**工作目录 (Working Directory)**，它保存了你当前从仓库中签出 (checkout) 的内容。现在你在项目的目录下输入：

```
git log
```

```
commit 0977323580be59378cf84fea791bb2e87ef21411 (HEAD -> master, origin/master,
origin/HEAD)
Author: Kai Zhu <rengwuxian@gmail.com>
Date:   Sun Nov 19 09:51:11 2017 +0800

    Initial commit
(END)
```

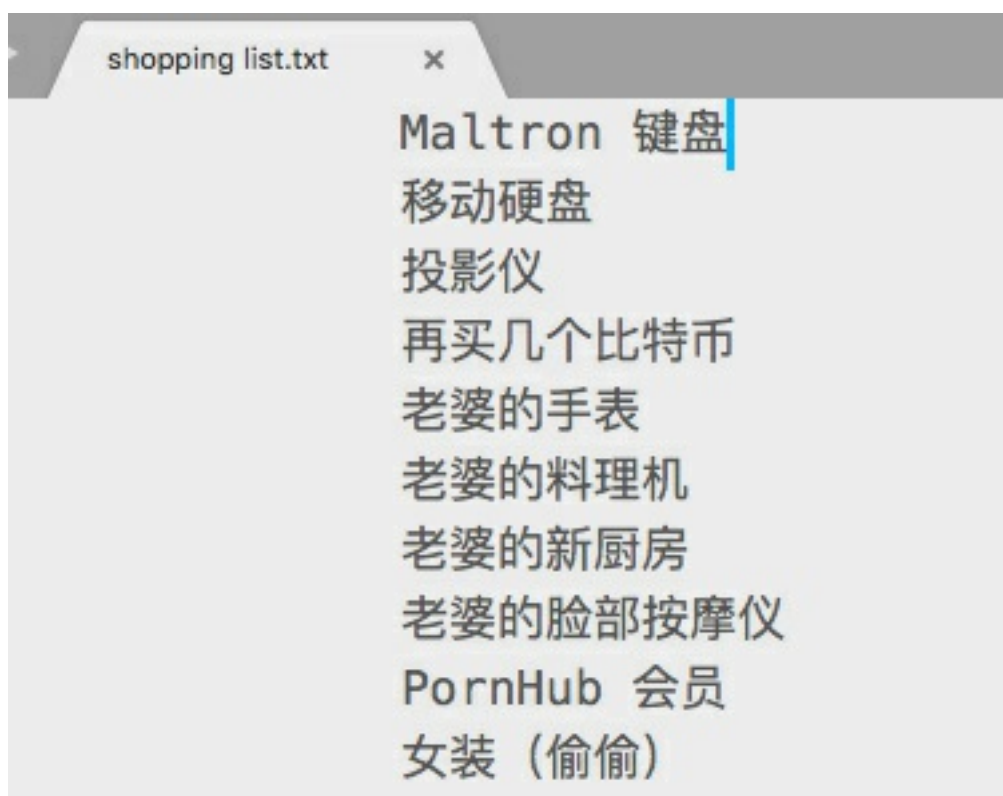
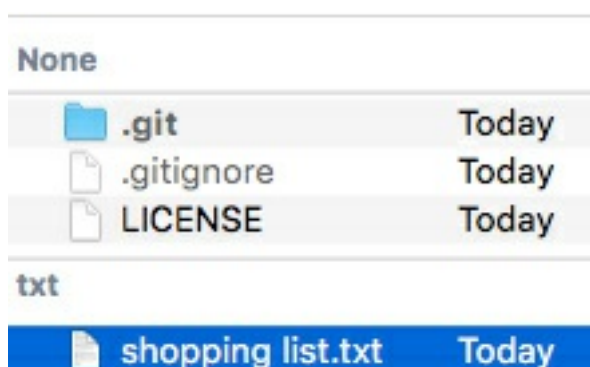
在这里你只能看到一个提交，这个提交是 GitHub 帮你做的，它的内容是创建你的初始 .gitignore 和 LICENSE 这两个文件。图中第一行中的 commit 右边的那一大串字符 (09773235...21411)，是这个 commit 的 SHA-1 校验和 (如果不知道什么是 SHA-1，你可以暂时把它简单理解为这个 commit 的 ID)；后面括号里的内容 (HEAD -> master ...) 稍后再讲；第一行的下面，依次是这个

commit 的作者、提交日期和提交信息，其中提交信息记录了这个提交做了什么，是提交者填写的（当然，这条提交信息是 GitHub 帮你写的）。

简单看一下这些信息，然后按 q 键退出吧，往下继续。

## 自己写个提交试试

把远程仓库取到本地之后，你就可以开始尝试提交代码了。不过为了方便，你不必真的写代码，你可以在工作目录下创建一个文本文件，例如 shopping list.txt。





Git 的提交是用的 `commit` 指令。不过.....你现在还不能直接提交。你现在在 Terminal 输入：

```
git status
```

`status` 是用来查看工作目录当前状态的指令：

```
→ git-practice git:(master) ✕ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        shopping list.txt

nothing added to commit but untracked files present (use "git add" to track)
```

这段文字表述了很多项信息：

1. 你在 `master` branch
2. 当前 branch 没有落后于 `origin/master`
3. 你有 `untracked files`（未追踪的文件），文件名是 `shopping list.txt`。
4. 你可以使用 `git add` 来开始追踪文件。

其中前两条你可以暂时先不理，branch 的东西我在后面会讲。关于后两条，简单说一下：

从上面的信息可以看出，`shopping list.txt` 这个文件目前属于 `"untracked"` 状态，它的意思是 Git 仓库对它没有进行任何记录，你在提交的时候不会把它提交上去，查看提交历史也不会看到它。总之，对于 Git 仓库来说，它是不存在的。

而你现在想提交这个文件，所以首先，你需要用 `add` 指令来让 Git 开始跟踪它：

```
git add shopping\ list.txt
```

输入这行代码，Terminal 不会给你反馈信息。但这时你再执行一次 `git status`，你会发现显示内容变了：

```
→ git-practice git:(master) X git add shopping\ list.txt
→ git-practice git:(master) X git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

       new file:   shopping list.txt
```

可以看到，`shopping list.txt` 的文字变成了绿色，它的前面多了「new file:」的标记，而它的描述也从 "Untracked files" 变成了 "Changes to be committed"。这些都说明一点：`shopping list.txt` 这个文件的状态从 "untracked"（未跟踪）变成了 "staged"（已暂存），意思是这个文件中被改动的部分（也就是这整个文件啦）被记录进了 staging area（暂存区）。

解释一下 "stage" 这个词，这个词对我们中国人可能有一些理解难度。按我们英语课本上的内容来看，stage 是一个名词，它的意思是「舞台」。可是不论从词性还是词义，「舞台」都不太能解释 "stage" 或 "staging area" 的意思。实质上，Git 中的 stage 取自这个词的另一个意思：组织和准备某个事件。而 "staging area" 的意思也并不是「舞台区域」，而是「用来汇集物品或材料以备使用的区域」的意思。

所以 stage 这个词在 Git 里，是「集中收集改动以待提交」的意思；而 staging area，就是一个「汇集待提交的文件改动的地方」。简称「暂存」和「暂存区」。至于 staged 表示「已暂存」，就不用再解释了吧？

所谓的 staging area，是 .git 目录下一个叫做 index 的文件（嗯，它的文件名并不叫 stage）。你通过 add 指令暂存的内容，都会被写进这个文件里。

现在文件已经放进了暂存区，就可以提交了。提交的方式是用 commit 指令：

```
git commit
```

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Your branch is up-to-date with 'origin/master'.
#
# Changes to be committed:
#   new file:   shopping list.txt
#
```

然后你会进入这样一个界面，这个界面是用来填写提交信息（commit message）的。根据操作系统以及设置的不同，这个界面的编辑器可能是 nano 或者 vi 或者别的什么，总之如果你不会用它，那么建议你尽快上网搜搜它的用法，因为 Git 的操作会经常用到它。在这里我只简单说一下它的最基本用法：

- 在初始状态下，你是在命令模式，不能编辑这个文件，你需要按一下 "i"（小写）来切换到插入模式，然后就可以输入你的提交信息了：

```
Add "shopping list.txt"
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Your branch is up-to-date with 'origin/master'.
#
# Changes to be committed:
#   new file:   shopping list.txt
#
```



- 在输入完成后别按回车，而是要按 ESC 键返回到命令模式，然后连续输入两个大写的 "Z"（用 Shift 键或 Capslock 键都可以），就保存并退出了。

```
→ git-practice git:(master) ✕ git commit
[master 79c353d] Add "shopping list.txt"
1 file changed, 10 insertions(+)
create mode 100644 shopping list.txt
```

这样，一次提交就完成了。如上图这样，从界面中你可以看到这次提交的简单信息。这时如果你再执行一次刚才执行过的 `git log`（还记得这个指令是什么意思吗？它会列出你的提交历史）：

```
git log
```

```
commit 79c353da8263fd9e93d7eb68dfbd1c8190d3450c (HEAD -> master)
Author: Kai Zhu <rengwuxian@gmail.com>
Date:   Sun Nov 19 14:31:54 2017 +0800

    Add "shopping list.txt"

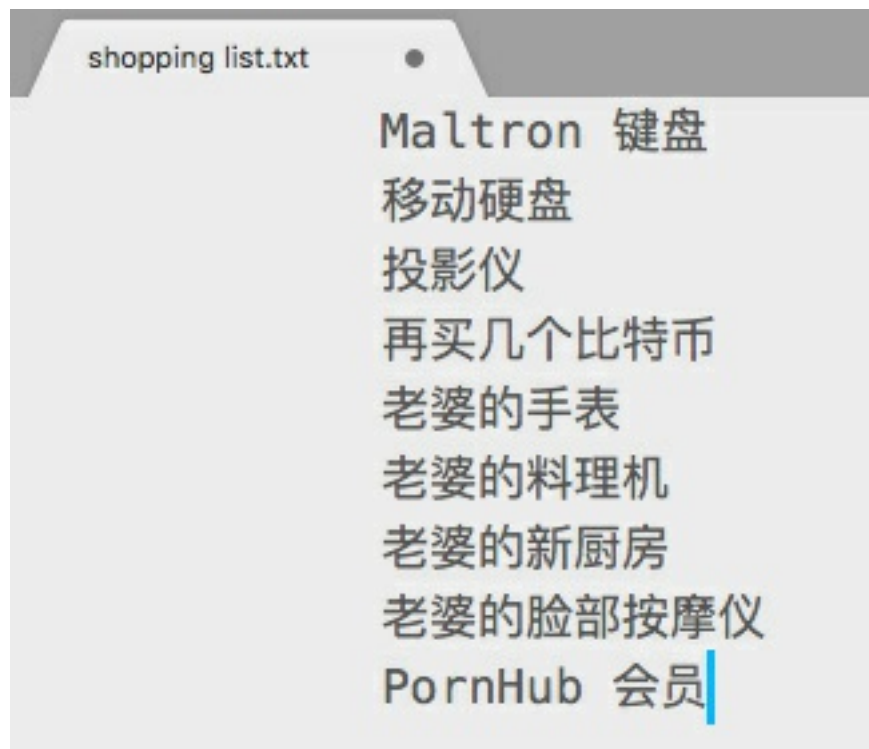
commit 0977323580be59378cf84fea791bb2e87ef21411 (origin/master, origin/HEAD)
Author: Kai Zhu <rengwuxian@gmail.com>
Date:   Sun Nov 19 09:51:11 2017 +0800

    Initial commit
(END)
```

可以看到，你的这条提交被列在了最上面，现在你的提交历史中有两条记录了。这说明，你已经成功做了一次提交到本地仓库，它已经被保存在了 `.git` 这个目录里的某个地方了。

## 再来个提交

想来想去，我还是觉得把「女装」列在购物清单有点太羞耻了，所以还是把它删掉吧：



嗯删掉以后脸不烫了，赶紧把它提交上去。提交之前先看看文件状态是个好习惯：

```
git status
```

```
→ git-practice git:(master) git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

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:   shopping list.txt
```

可以看到，shopping list.txt 又变红了，不过这次它左边的文字不是 "New file:" 而是 "modified:"，而且上方显示它的状态也不是 "Untracked" 而是 "not staged for commit"，意思很明确：Git 已经认识这个文件了，它不是个新文件，但它有了一些改动。所以虽然状态的显示有点不同，但处理方式还是一样的：

```
git add shopping\ list.txt
```

这时再 status 一下，就会看到 shopping list.txt 已经 staged ("to be committed") 了：

```
git status
```

```
→ git-practice git:(master) ✕ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   shopping list.txt
```

好，最后一步，commit：

```
git commit
```

```
Remove a shameful part
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#   (use "git push" to publish your local commits)
#
# Changes to be committed:
#       modified:   shopping list.txt
#
```

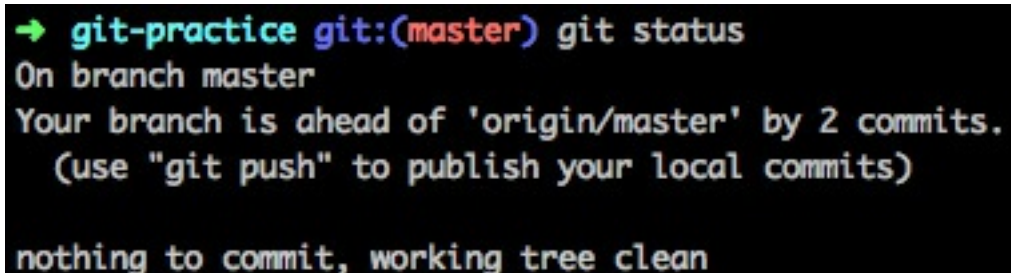
这时再看看 log，你已经有三条 commits 了：

```
git log
```

## 把提交推送到中央仓库

到现在为止，已经写了两条提交，但它们都还在本地仓库。为了把代码分享出去，你还需要把这些提交上传到中央仓库。如果你现在再看一下 status：

```
git status
```

A terminal window with a black background and green text. The command 'git status' is entered. The output shows the current branch is 'master', it is ahead of 'origin/master' by 2 commits, and the working tree is clean.

```
→ git-practice git:(master) git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

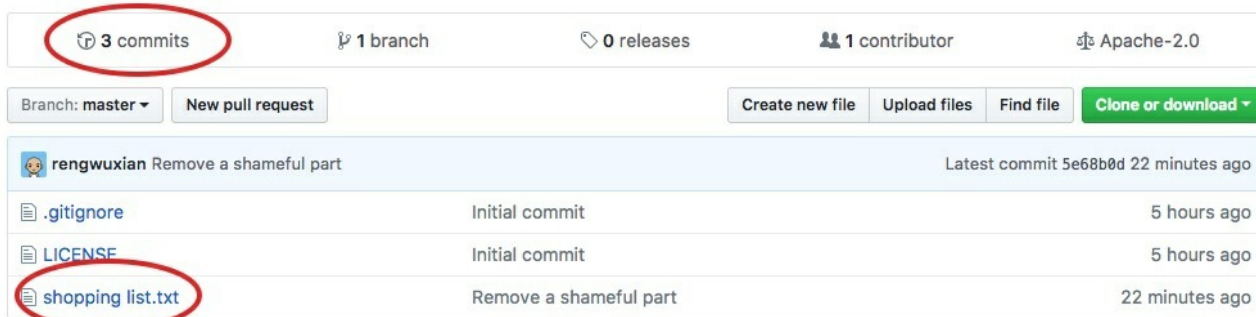
nothing to commit, working tree clean
```

你会看到，Git 提示你的当前 branch 已经领先于（"ahead of"）'origin/master' 两个提交了。就像上面我说的，branch 这个概念我到后面再讲，origin/master 中的 origin 是远端仓库的名称，是你在用 clone 指令初始化本地仓库时 Git 自动帮你起的默认名称；master 是 origin 上的分支名称。不过对于现在来说，可以暂时把 origin/master 简单理解为「中央仓库」。也就是说，这句话是告诉你，你的本地仓库已经领先中央仓库两个提交了。这两个提交很明显就是指的刚才那两个关于 shopping list.txt 的提交。

而这句话的下面也说明了，你可以使用 git push 来把你的本地提交发布（即上传到中央仓库）。所以很简单，照做吧：

```
git push
```

由于这是联网操作，所以在这个过程 GitHub 会再次向你索要账户和密码。填入正确的账户和密码，push 操作就完成了。这时你再去你的 GitHub 仓库页面看一下：



你会发现 `shopping list.txt` 文件已经在上面了，并且 `commits` 的数量从一个变成了 3 个。这说明你已经成功把本地仓库的提交推送到了服务器。

如果觉得一遍遍地输入密码很烦，可以按照[这个页面 \(https://help.github.com/articles/caching-your-github-password-in-git/\)](https://help.github.com/articles/caching-your-github-password-in-git/) 提供的方案来把密码保存起来。

另外还有一个更简单但安全性低一些的方案。执行这行代码：

```
git config credential.helper store
```

在这之后你只需要再输入一次密码，Git 就会把你的密码保存下来，这之后就再也不用输入了。说它「安全性低」，是因为这条指令会让 Git 把你的密码以明文形式保存在你的电脑上。具体这两种保存密码的方案选择哪个，看你自己了。

## 小结

在这一节里，描述了 Git 的最基本的工作模型：

1. 从 GitHub 把中央仓库 clone 到本地（使用命令：`git clone`）



2. 把写完的代码提交（先用 `git add` 文件名 把文件添加到暂存区，再用 `git commit` 提交）

- 在这个过程中，可以使用 `git status` 来随时查看工作目录的状态
- 每个文件有 "changed / unstaged"（已修改），"staged"（已修改并暂存），"committed"（已提交） 三种状态，以及一种特殊状态 "untracked"（未跟踪）

3. 提交一次或多次之后，把本地提交 `push` 到中央仓库（`git push`）

下一节将会讲一下在团队开发中会遇到的问题，从而以这种工作模型为基础，延伸到团队开发的基本工作模型。

但在这之前，强烈建议你先按照这节的内容照做一遍，并尝试理解这一节中所说的工作模型。不然的话，可能会出现「小册看完了，却还是不知道怎么做」的情况。