

从入门到工作「方方」

①上个任务 | 下个任务②

课程简介

学习资源

视频
命令行速记视频
命令行详解视频
vim&git 简单用法文章
讲义文章
Git 讲义文章
使用 GitHub Pages 预览 HTML文章
苹果系统命令行入门文章
使用 Hexo 建立自己的博客文章
命令行技巧 (可不看, 目前用不到)

课后习题

测试
基本的命令行测试
Git 测试题测试
命令行相关博客

返回课程

任务已完成

Git 讲义

首先告诉你真相：新人无法理解 git 的原理，你只能背命令。等你用 git 用一个月，再谈原理，用 git 三个月，你就自然理解 git 了。不要试图一开始就理解 git！

Git 资料 (可不看)：<http://www.runoob.com/git/git-tutorial.html>

配置 GitHub

1. 进入 <https://github.com/settings/keys>
2. 如果页面里已经有一些 key，就点「delete」按钮把这些 key 全删掉。如果没有，就往下看
3. 点击 New SSH key，你需要输入 Title 和 Key，但是你现在没有 key，往下看
4. 打开 Git Bash
5. 复制并运行 `rm -rf ~/.ssh/*` 把现有的 ssh key 都删掉，这句命令行如果你多打一个空格，可能就要重装系统了，建议复制运行。
6. 运行 `ssh-keygen -t rsa -b 4096 -C "你的邮箱"`，注意填写你的邮箱！
7. 按回车三次
8. 运行 `cat ~/.ssh/id_rsa.pub`，得到一串东西，完整的复制这串东西
9. 回到上面第 3 步的页面，在 Title 输入「我的第一个 key」
10. 在 Key 复制刚刚你复制的那串东西
11. 点击 Add SSH key
12. 回到 Git Bash
13. 运行 `ssh -T git@github.com`，你可能会看到这样的提示：

```
52620@DESKTOP-EA7GML1 MINGW64 ~
$ ssh -T git@github.com
The authenticity of host 'github.com (192.30.255.113)' can't be established.
RSA key fingerprint is SHA256:nThbgkXUpJM17E1GOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)? |
```

图片

输入 yes 回车.....问你话你就答，别傻在那

14. 然后如果你看到 Permission denied (publickey)。就说明你失败了，请回到第 1 步重来；是的，回到第 1 步重来；如果你看到 Hi FrankFang! You've successfully authenticated, but GitHub does not provide shell access。就说明你成功了！

好了，终于 TMD 添加了一个无聊的 SSH key，不要问我这个有什么用，因为一会儿你就会用到它，你想了解原理就看这篇 [文章](#)

如果要讲清楚，太浪费时间了，我们只是想用用 GitHub 而已。

- 一台电脑只需要一个 SSH key
- 一个 SSH key 可以访问你的所有仓库，即使你有 1000000 个仓库，都没问题
- 如果你新买了电脑，就在新电脑上重新生成一个 SSH key，把这个 key 也上传到 GitHub，它可以和之前的 key 共存于 GitHub 上
- 如果你把 key 从电脑上删除了，重新生成一个 key 即可，替换之前的 key

配置 git

```
git config --global user.name 你的英文名
git config --global user.email 你的邮箱
git config --global push.default matching
git config --global core.quotePath false
git config --global core.editor "vim"
```

#此英文名不需要跟GitHub账号绑定
#此邮箱不需要跟GitHub账号绑定

五句话，依次在命令行中运行（其中前两句要把中文改成对应的内容）。

一定要执行这五行！！！
一定要执行这五行！！！
一定要执行这五行！！！

使用 git

使用 git 有三种方式，请按照你的需求选择

1. 只在本地使用
2. 将本地仓库上传到 GitHub
3. 下载 GitHub 上的仓库

1 只在本地使用

1.1 初始化

1. 创建目录作为我们的项目目录：`mkdir git-demo-1`
2. 进入目录 `cd git-demo-1`
3. `git init`，这句话会在 `git-demo-1` 里创建一个 `.git` 目录
4. `ls -la` 你就会看到 `.git` 目录，它就是一个「仓库」，不要进去看，这仓库里面有毒，别进去！
5. 在 `git-demo-1` 目录里面添加任意文件，假设我们添加了两个文件，分别是 `index.html` 和 `css/style.css`
 - i. `touch index.html`
 - ii. `mkdir css`
 - iii. `touch css/style.css`
6. 运行 `git status -sb` 可以看到文件前面有 ?? 号


```
## Initial commit on master
?? css/
?? index.html
```

这个 ?? 表示 git 一脸懵逼，不知道你要怎么对待这些变动。

7. 使用 `git add` 将文件添加到「暂存区」
 - i. 你可以一个一个地 `add`
 - a. `git add index.html`
 - b. `git add css/style.css`
 - ii. 你也可以一次性 `add`
 - a. `git add .`，意思是把当前目录（. 表示当前目录）里面的变动都加到「暂存区」
8. 再次运行 `git status -sb`，可以看到 ?? 变成了 A


```
## Initial commit on master
A  css/style.css
A  index.html
```

A 的意思就是添加，也就是说你告诉 git，这些文件我要加到仓库里

9. 使用 `git commit -m "信息"` 将你 `add` 过的内容「正式提交」到本地仓库（`.git` 就是本地仓库），并添加一些注释信息，方便日后查阅

- i. 你可以一个一个地 `commit`
- a. `git commit index.html -m "添加index.html"`
- b. `git commit css/style.css -m "添加 css/style.css"`
- ii. 你也可以一次性 `commit`
- a. `git commit . -m "添加了几个文件"`

10. 再再次运行 `git status -sb`，发现没有文件变动了，这是因为文件的变动已经记录在仓库里了。

11. 这时你使用 `git log` 就可以看到历史上的变动：

```
commit f0d95058cd32a32b98967f6c0a701c64a00810a
Author: frankfang <frankfang1990@gmail.com>
Date: Thu Sep 28 22:30:43 2017 +0800

    添加几个文件
```

12. 以上就是 `git add / git commit` 的一次完整过程，可以看到，挺复杂的。原则上，你错了任何一步，就给我从头来一遍，做到你不会再手抖为止。

1.2 文件变动

如果我想继续改文件，应该怎么做呢？

1. start `css/style.css` 会使用默认的编辑器打开 `css/style.css`（macOS 上对应的命令是 `open css/style.css`）

2. 然后我们在 `css/style.css` 里写入 `body {background: red;}`，保存退出

3. 运行 `git status -sb`，发现提示中有一个 M

```
## master
M css/style.css
```

这个 M 的意思就是 `Modified`，表示这个文件被修改了

4. 此时你如果想让改动保存到仓库里，你需要先 `git add css/style.css` 或者也可以 `git add .`

注意：由于这个 `css/style.css` 以前被我们 `add` 过，你往文章上面看，我们是 `add` 过 `css/style.css` 的，所以此处的 `git add` 操作可以省略，但我建议你使用 `git` 的前一个月，不要省略 `git add`。

换句话说，每一次改动，都要经过 `git add` 和 `git commit` 两个命令，才能被添加到 `.git` 本地仓库里。

5. 再次运行 `git status -sb`，发现 M 有红色变成了绿色，红色和绿色有啥区别呢？别管它们的区别，记住我说的，先 `add`，再 `commit`，等你熟练之后再去理解区别。

先形成肌肉记忆，在去形成大脑记忆！

6. 运行 `git commit -m "更新 css/style.css"`，这个改动就被提交到 `.git` 本地仓库了。再说一次，不要去 `.git` 目录里面，那里东西你一无所知。

7. 再再次运行 `git status -sb`，会发现没有变更了，这说明所有变动都被本地仓库记录在案了。

这里来透露一下 `git status -sb` 是什么意思：`git status` 是用来显示当前的文件状态的，哪个文件变动了，方便你进行 `git add` 操作。`-sb` 选项的意思就是，SB 都能看懂，哈，这是开玩笑，`-s` 的意思是显示总结（summary），`-b` 的意思是显示分支（branch），所以 `-sb` 的意思是显示总结和分支。

1.3 总结

至此，我们来总结一下用到的命令

1. `git init`，初始化本地仓库 `.git`

2. `git status -sb`，显示当前所有文件的状态

3. `git add` 文件路径，用来将变动加到暂存区

4. `git commit -m "信息"`，用来正式提交变动，提交至 `.git` 仓库

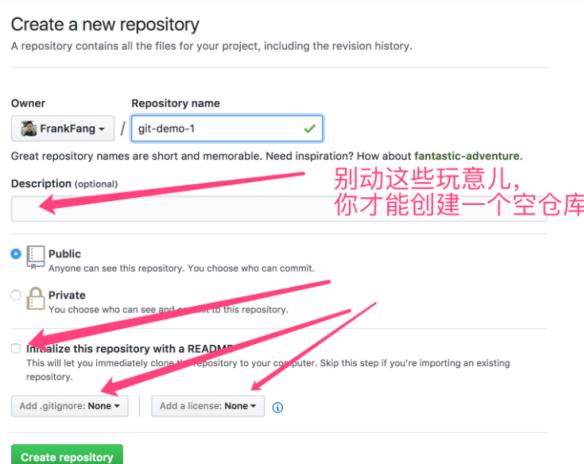
5. 如果有新的变动，我们只需要依次执行 `git add xxx` 和 `git commit -m 'xxx'` 两个命令即可。别看本教程废话那么多，其实就这一句有用！先 `add` 再 `commit`，行了，你学会 `git` 了。

6. `git log` 查看变更历史

2 将本地仓库上传到 GitHub

如何将我们这个 `git-demo-1` 上传到 GitHub 呢？

1. 在 GitHub 上新建一个空仓库，名称随意，一般可以跟本地目录名一致，也叫做 `git-demo-1`

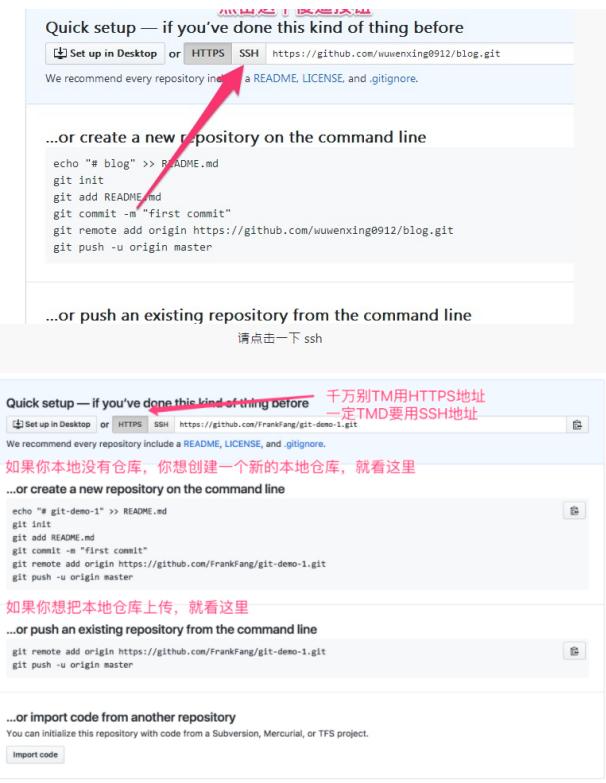


图片

按照截图所示，除了仓库名，其他的什么都别改，其他的什么都别改，其他的什么都别改，其他的什么都别改，这样你才能创建一个空仓库

2. 点击创建按钮之后，GitHub 就会把后续的操作全告诉你，如图





图片

3. 看图，点击 SSH 按钮，点击 SSH 按钮，点击 SSH 按钮，我想你现在肯定不会忘了点击 SSH 按钮了吧~~~~如果不点这个按钮，你就会使用默认的 HTTPS 地址，但是千万不要使用 HTTPS 地址，因为 HTTPS 地址使用起来特别麻烦，每次都要输入密码，而 SSH 不用输入用户名密码。
为什么 SSH 不用密码呢，因为你已经上传了 SSH public key，还记得吗？如果还不记得，翻到本文第一部分「配置 GitHub」章节。
4. 由于我们已经有本地仓库了，所以看图，图中下面半部分就是你需要的命令，我们一行一行拷贝过来执行
i. 找到图中的「...or push an existing repository from the command line」这一行，你会看到 `git remote add origin https://github.com/xxxxxxxxxx/git-demo-1.git`，如果你发现这个地址是 https 开头的，那你就做错了，还记得吗？我们要使用 SSH 地址，GitHub 的 SSH 地址是以 `git@github.com` 开头的。
ii. 再次点击 SSH 按钮，不管我强调多少遍，总会有人忘记点击 SSH 按钮，为什么呢？我也不知道，为了防止你忘了点击 SSH 按钮，我最后再说一遍：「点击 SSH 按钮」，点击之后，整个世界就会变得美好起来。
iii. 得到新的命令 `git remote add origin git@github.com:xxxxxxxxxxxxxxxxxxxxxxxxxxxxx/git-demo-1.git`，复制并运行它
iv. 复制第二行 `git push -u origin master`，运行它
v. 刷新当前页面，你的仓库就上传到 GitHub 了！是不是特别简单？只要你按照我说的做，一丝不苟，即可。

3 直接在 GitHub 创建一个仓库，然后下载到本地

上面两步讲了

1. 在本地创建仓库
2. 将本地仓库上传到 GitHub

这里将第三种用法，那就是直接在 GitHub 创建一个仓库，然后下载到本地。

1. 在 GitHub 上新建一个仓库 `git-demo-2`，这次就不创建空仓库了，而是自带 README 和 License 的仓库，创建截图如下：

Create a new repository

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

Owner: FrankFang / Repository name: git-demo-2

Great repository names are short and memorable. Need inspiration? How about [musical-waffle](#).

Description (optional): 这是一个示例仓库

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: Node | Add a license: MIT License

Create repository

图片

- 请按图中所示，填写一模一样的内容，然后点击创建按钮。
2. 这样一来，这个仓库就会自动拥有三个文件：

This screenshot shows a GitHub repository page for a repository named 'git-demo-2'. At the top, it displays basic statistics: 1 commit, 1 branch, 0 releases, 1 contributor, and the license as MIT. Below this, a commit history is shown for 'FrankFang' with three commits: '.gitignore', 'LICENSE', and 'README.md', all made 'just now'. The 'README.md' file is expanded, showing its content: 'git-demo-2' and '这是一个示例仓库' (This is a sample repository). A red arrow points to the 'Clone or download' button at the bottom right of the main content area.

3. 这三个文件的作用请自行了解：[.gitignore 的作用](#)、[README.md 的作用](#) 以及 [LISENCE 的作用](#)
4. 好了，现在远程仓库已经创建好了，怎么下载到我们的本地（也就是我们的电脑上）呢？答案是使用 git clone 命令
5. 点击页面中唯一的绿色按钮「clone or download」，会看到一个弹出层

This screenshot shows a modal window titled 'Clone with HTTPS' from GitHub. It includes options for 'Use SSH' and 'Use Git or checkout with SVN using the web URL'. The URL listed is <https://github.com/FrankFang/git-demo-2.git>. Below the modal are two buttons: 'Open in Desktop' and 'Download ZIP'. A large red arrow points from the previous screenshot's red arrow towards this modal.

图片

6. 请确保弹出层里的地址是 SSH 地址，也就是 `git@github.com` 开头的地址，如果不是，就点击 Use SSH 按钮，就点击 Use S SH 按钮，就点击 Use SSH 按钮。然后复制这个地址。
7. 打开 Git Bash，找一个安全的目录，比如 `~/Desktop` 桌面目录就很安全；`cd ~/Desktop`，运行。
8. 运行 `git clone` 你刚才得到的以`git@github.com`开头的地址，运行完了你会发现，桌面上多出一个 `git-demo-2` 目录。我再说一遍，桌面上多出一个 `git-demo-2` 目录，我再说一遍，桌面上多出一个 `git-demo-2` 目录。这个细节很重要，很多小白发现不了这个细节，我也不知道他们是眼睛还是怎么了.....
9. 进入这个多出来的目录，对的，你肯定会忽略这一步。
10. 进入这个多出来的目录，对的，你肯定会忽略这一步。
11. 进入这个多出来的目录，对的，你肯定会忽略这一步。
12. 好了你进入了这个目录了，如果没有，我就要吐血了，因为我的提示很明显。
13. 运行 `ls -la` 你会看到，远程目录的所有文件都在这里出现了，另外你还看到了 `.git` 本地仓库，这是你就可以添加文件，`git add`，然后 `git commit` 了。

三种方式都讲完了，它们分别是：

1. 在本地创建仓库
2. 将本地仓库上传到 GitHub
3. 下载 GitHub 上的仓库到本地

其实呢，我还可以讲很多种不同的方式，但是，你记住这几种就行了，够你用的了。我们并不想要了解 git 的所有高级用法，我们的目的很明确：能通过 Git 命令使用 GitHub 就行。

我们再回顾一遍已经学到的命令：（这次只多了一个 `git clone` 命令）

0. `git clone git@github.com:xxxxx`，下载仓库
1. `git init`，初始化本地仓库`.git`
2. `git status -sb`，显示当前所有文件的状态
3. `git add` 文件路径，用来将变动加到暂存区
4. `git commit -m "信息"`，用来正式提交变动，提交至 `.git` 仓库
5. 如果有新的变动，我们只需要依次执行 `git add xxx` 和 `git commit -m 'xxx'` 两个命令即可，别看本教程废话那么多，其实这一句有用！先 `add` 再 `commit`，行了，你学会 `git` 了。
6. `git log` 查看变更历史

如何上传更新

你在本地目录有任何变动，只需按照以下顺序就能上传：

1. `git add` 文件路径
2. `git commit -m "信息"`
3. `git pull`（相信我，你一定会忘记这一个命令）
4. `git push`

下面是例子

1. `cd git-demo-1`
2. `touch index2.html`
3. `git add index2.html`
4. `git commit -m "新增 index2.html"`
5. `git pull`
6. `git push`

然后你去 `git-demo-1` 的 GitHub 页面，就能看到 `index2.html` 出现在里面了。是不是很.....简.....单.....呢.....

git ignore

在项目目录创建 `.gitignore` 文件就可以指定「哪些文件不上传到远程仓库」，比如

```
.gitignore
```

```
/node_modules/  
.vscode/
```

这样就可以避免 node_modules/ 和 .vscode/ 目录被上传到 github 了。

记住一句话：永远都不要上传 node_modules 到 github。

如果你想防止自己手贱上传 node_modules 到 github，可以：

1. 在项目根目录 touch .gitignore
2. 在 .gitignore 里添加一行 /node_modules/
3. git add .gitignore; git commit -m 'ignore'

其他

还有一些有用的命令

- git remote add origin git@github.com:xxxxxx.git 将本地仓库与远程仓库关联
- git remote set-url origin git@github.com:xxxxx.git 上一步手抖了，可以用这个命令来挽回
- git branch 新建分支
- git merge 合并分支
- git stash 通灵术
- git stash pop 反转通灵术
- git revert 后悔了
- git reset 另一种后悔了
- git diff 查看详细变化

学 git 命令都够你们学一周的，所以别妄想现在就掌握它，切记。

如果你发现 git 下载速度很慢

你可以看这篇教程 <https://jscode.me/t/topic/789>

资源

- 常用 Git 命令清单
- 读懂 diff - 阮一峰
- 搭建一个免费的，无限流量的Blog----github Pages和Jekyll入门
- Git 菜鸟教程
- 廖雪峰的 Git 教程

① 上一节: 讲义

下一节: 使用 GitHub Pages 预览 HTML ②