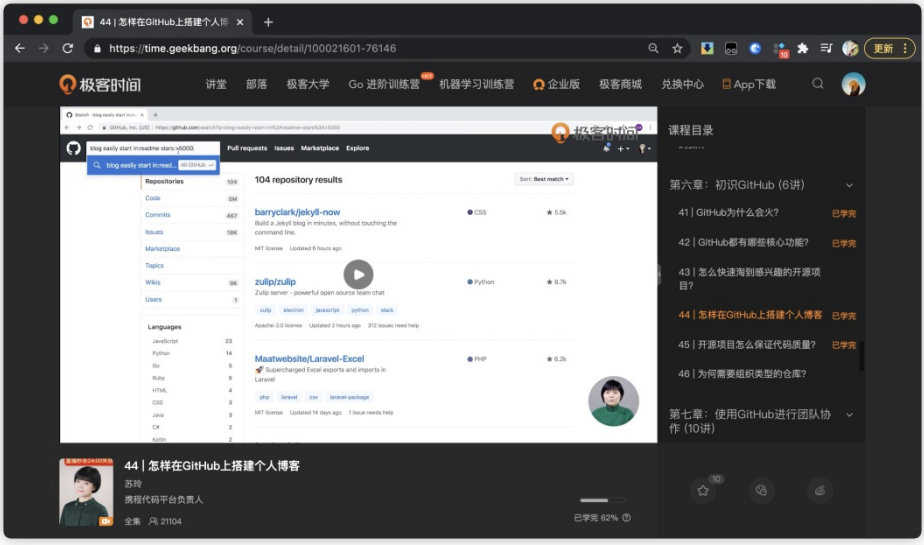
搭建博客可能会用到的 Git 命令 | 学习笔记

原创 彭宏豪 效率工具指南

收录于话题
#2021 93 #Git 2 #博客 7 #GitHub 12

题图：来自 Wallhaven

2019 年为了学习使用 GitHub Pages 搭建博客，我在一个 App 上买了一门和 GitHub 有关的课，但主要看了其中一讲：



学完这一讲之后，我还写了一篇简短的搭建博客的文章：

不懂技术，如何搭建个人博客？

课程剩下的其他内容，就被我晾在一边了。上个月看到这个 App 提醒我好久没学习了，提醒的文案也很皮，「**21 天了，似乎养成了不学习的习惯**」，哈哈哈哈哈哈。

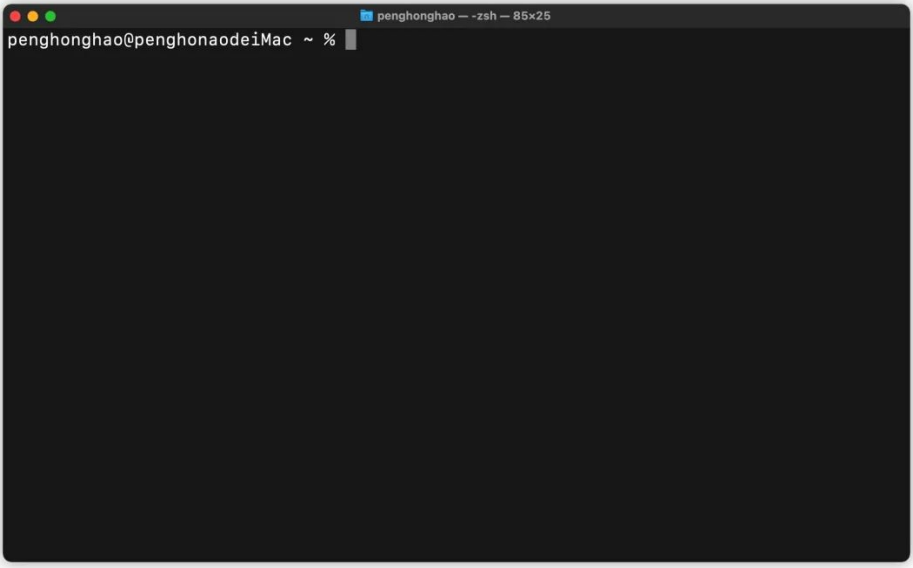
买了课程没看，心里有点愧疚，就开始捡起来看一些，顺便做了一下笔记~~（别说了，主要是不知道今天发点啥，强行用学习笔记来凑数）~~。

以下内容主要来自「极客时间」的课程《Git 三剑客》，这里的三剑客是指：

- Git
- GitHub
- GitLab

我整理了一些在课程中用到的 Git 命令，对程序员可能是小菜一碟，可能对想搭建博客的朋友有帮助：

运行这些命令，需要有一个**环境**，这个环境就是电脑的「**终端**」，Windows 上打开「运行」，输入「cmd」，同样可以打开终端。



Git 下载地址：

https://git-scm.com/downloads

安装 Git 之后，查看是否正确安装

git --version

这个命令也可以简写成 git --v

创建一个新的仓库并初始化

git init hexo_blog

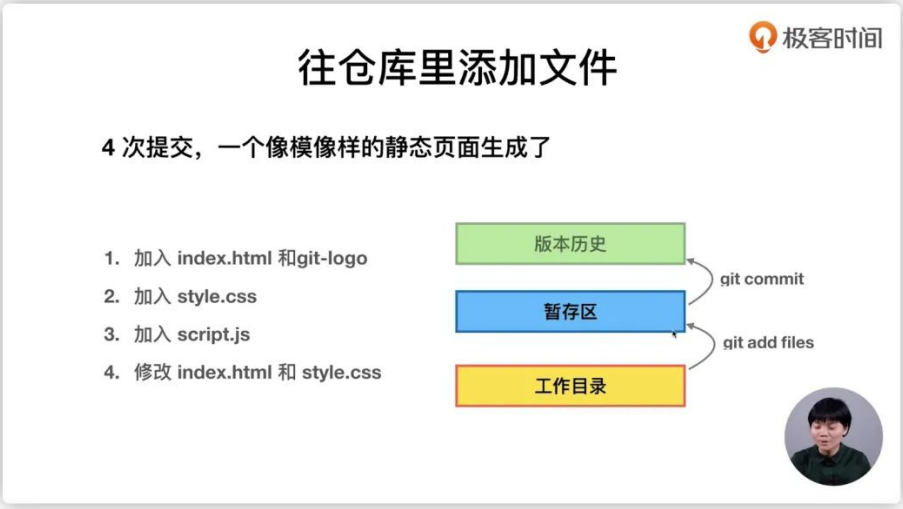
添加到暂存区

git add index.html

注：add 后面加上做出变更的文件

这里简单说一下 Git 的工作方式（或者说工作流程）：

- 当前正在编辑的文件，处于工作目录（工作区）
- 编辑好但待提交的文件，可以暂时添加到暂存区，这相当于是一个中间态，既可以向前一步作提交（commit），也可以向后一步，回退或覆盖工作区，这个方式有点像是 PS 里的「历史记录」面板，可以比较灵活地进行变更。



将编辑好的内容 push 到 GitHub

git add .

git commit -m"附带的信息，例如在某个时间点编辑了博客"

git push

这三个命令，是搭建好博客之后，每次发布文章时，都需要用到的命令。看完这篇文章，其他的 Git

命令都可以忘记，除了这三条。

查看 Git 工作目录和暂存区的状态

命令：git status

遇到未被 Git 管理的文件或文件夹，可以使用 add 命令将其纳入到 Git 的管理之下。

例如：git add index.html images

注：add 后面可以加文件，也可以加文件夹，上面的 index.html 是文件名，images 是文件夹。

查看 Git 变更的记录

这个有点像是 PS 里面的「历史记录面板」，记录了你做出的所有变更。

命令：git log

清理一下终端当前显示的内容

命令：clear

新建文件夹的命令

命令：mkdir+空格+文件夹名称

创建一个新的文件

echo "hello,world" > readme

创建一个 readme 文件，里面的内容为 hello,world

查看 Git 最近 N 次的日志

git log -n

如果之前提交过很多次，git log 不附带参数的话，会返回一长串日志，不便于观看

在终端中查看文件内容或编辑文件

命令：vi+空格+文件名

例如你想在终端中编辑一个名为 `style.css` 的文件，可以运行命令

vi `style.css`

在 Vim 编辑器中编辑好文件，该如何退出呢？

按下 ESC 键，在终端底部输入不同的命令，命令对应的含义如下：

:w!　保存
:wq!　保存并退出编辑(w 代表写入，q 代表退出)

从某个子路径直接退回到根目录下

命令：cd ../

查看当前路径已有的文件和文件夹

命令：ls -al

注：这个命令可以列出隐藏的文件。

在 Git 中给文件快速重命名的方法

命令：git mv

举个例子，如果你想把 readme 重命名为 `readme.md`，只需要执行命令：

git mv readme `read.md`

删除某个文件

git rm filename

这个命令会直接将处于暂存区的文件删除，这样就不需要先在工作目录下删除文件，再到暂存区清理文件。

切换分支

git checkout master

注：从当前分支切换到 master 分支上，这个命令需要在工作路径下运行

新建分支并切换到新分支上

git checkout -b 新分支的名称 旧分支名称

注：基于旧分支的基础上，创建一个新的分支

查看当前工作在哪个分支下边（查看当前有哪些分支）

git branch -av

注：运行返回的带有星号 * 的分支，就是当前的工作分支

查看本地的 git 配置

git config --local --list：显示所有配置信息，包含 `user.name`、user.email 等。

git config --local `user.name`：仅显示 `user.name` 信息。

更改 `user.name` 信息

git config --local `user.name` 'angola'

末尾的单引号内的名字 angola 就是更改之后的 `user.name`

cat 命令

命令含义：主要用来查看文件内容，创建文件，文件合并，追加文件内容等功能。

cat+空格+文件名：将文件内容打印显示

查看 HEAD 指针目前指向哪个分支

cat .git/HEAD

假设运行返回的结果是 ref: refs/heads/fix_readme

这意味着 HEAD 目前指向分支 fix_readme

查看路径下边类型是文件的个数

find .git/objects -type file

注：.git/objects 路径下类型是文件的数量

比较两次 commit 的区别

git diff 3d4731d80eb 415c5c8086e1

注：diff 后面跟的是两次 commit 对应的哈希值

比较 HEAD 与 HEAD 父级（即 HEAD 的前一个版本）的不同：

git diff HEAD `HEAD^1`

最末尾也可以不加数字，单纯使用：

git diff HEAD `HEAD^`

HEAD 与 HEAD 父级的父级（即 HEAD 的爷爷）的不同：

git diff HEAD `HEAD^^`

等同于 git diff HEAD `HEAD~2`

图形界面工具

gitk
gitk --all

删除不需要的分支

git branch -d 待删除的分支名称

修改最新 commit 的 message

git commit --amend

运行之后，会打开编辑窗口，顶部可编辑最近一次提交的 message 信息，编辑好之后按 ESC，输入 :wq! 退出编辑

消除最近的几次提交

git log --graph
git reset --hard 5df3fd1900

第一个命令是，在终端中以图形化的方式显示之前提交的记录，记录中会显示每一次 commit 对应的哈希值。

注：5df3fd1900 是恢复到的某一次 commit 对应的哈希值

比较暂存区和 HEAD 所含文件的差异

git diff --cached

注：cached 代表暂存区

比较工作区和暂存区所含文件的差异

git diff

diff 后面不加参数，默认比较的是工作区和暂存区的差别

如果 diff 添加了文件名，就是比较这个文件在工作区和暂存区的差别，例如

git diff --readme.md

添加两个文件名的话，就可以同时查看两个文件在工作区和暂存区的差别：

git diff --readme.md styles/style.css

如何指定不需要 Git 管理的文件?

不需要 Git 管理的文件，可以在 `.gitignore` 文件中列出。

先使用命令 vi `.gitignore`

创建一个 `.gitignore` 文件，文件名必须为 `.gitignore`，在文件中列明不纳入 Git 管理的文件和文件夹。

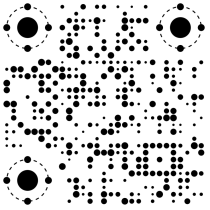
在输入文件和文件名时需要注意：

- doc 代表 doc 文件不纳入 Git 管理
- doc/ 末尾如果带有斜杠，代表文件夹 doc 下的所有文件都不纳入 Git 管理

如果先将文件或文件夹添加到了暂存区，`.gitignore` 对其也不起作用。

其他内容

这门课我还没看完，其他有些单纯靠文字无法理解的内容，我在记笔记的时候还附上了一些截图，放在幕布中，感兴趣的朋友可以扫描下方的二维码查看：



本文也同步发布在我的个人博客上，你可以点击底部左下角的「**阅读原文**」，查看本文在博客上的表

现。

以上。

阅读原文