Week 3: git

Mayrain

2023 年 11 月 27 日

1 Git

1.1 What is git

一个分布式的版本控制系统。分布式代表不需要联网，版本控制代表可以回溯文件修改历史。
有趣的部分：git 是自托管的，也就是说 git 他自己的代码就是放在 git 仓库里的。现在你甚至可以在 github 上看到 git 的源代码。实现是用 1000 多行代码完成的。
git 自带的 git bash 是一个命令行工具，可以用来操作 git。他也很有用。

1.2 How git works

$$working\quad directory \underset{add}{->} staging\quad area \quad -> gitrepository \underset{commit}{}$$

他妈的这一段真丑啊我去，我得搞明白为什么排版会变成这样，但是不是现在。
push: 本地仓库 -> 远程仓库
pull: 远程仓库 -> 本地仓库
在这里课程讲的很简单，最好参见网络教程。

利用 git init folder 可以新建一个文件夹并将其转化为 git 仓库。
git 文件有三种状态：

- untracked: 未跟踪
- modified: 已修改
- staged: 已暂存
- ignored: 已忽略

这里的 ignored 是指 git 不会跟踪这个文件，也就是说这个文件不会出现在 git status 的结果中。他被存储在 gitignore 文件。一般我们都在这里加一些规则。
github/gitignore: 这里有很多 gitignore 的规则，可以直接复制。

commit message standards:
angular/angular:CONTRIBUTING.md

作者的话：
老天，连 git 的 commit 的 message 都在 github 上有标准化的规定，不得不说这就是程序员的思维：机械而规范。让我想起了 github 上有名的“how to ask questions wisely”系列。很有意思。

1.2.1 ADDITION:Commit Message Format

*It is totally copy of angular/CONTRIBUTING.md.

<header>

Format: -> <Commit Type>(<Commit Scope>): <Short summary>

Commit Type:
build: Changes that affect the build system or external dependencies (example scopes: gulp, broccoli, npm)
修改项目构建系统的代码或者外部依赖的改动，例如构建脚本，Dockerfile，package.json，webpack 配置等等
ci: Changes to our CI configuration files and scripts (examples: CircleCi,

SauceLabs)
修改项目继续集成流程的提交，例如 Travis，Jenkins，GitLab CI，Browser-Stack，SauceLabs 等等
docs: Documentation only changes
仅仅修改了文档，比如 README，CHANGELOG，CONTRIBUTE 等等
feat: A new feature
新功能
fix: A bug fix
修复 bug
perf: A code change that improves performance
提升性能的代码改动
refactor: A code change that neither fixes a bug nor adds a feature
代码重构
test: Adding missing tests or correcting existing tests
测试用例的变动

Commit Scope:
indicate the place of the commit change.Should be the name of the npm package affected.Better to see this:
https://github.com/angular/angular/blob/main/CONTRIBUTING.md

<Blank Line>

<content>

<Blank Line>

<footer>

1.2.2 Version Name change rules

version a.b.c[-d]

a: major version（主版本号，大改，不兼容的 API 修改。0 表示开发阶段，不保证完整性）

b: minor version（次版本号，添加新功能，保持兼容）

c: patch version（修订号，兼容更改以及修正不正确的行为）

d: pre-release version（预发布版本号，代表这个程序是预发布的，实际上只是尝鲜版。顺序是 alpha（内测）, beta（公测）, rc.1, rc.2, rc 开头的都是预发布。）

1.3 Git Branch

Branch，也就是分支，是相当于一个岔路指向标。在 git 中，我们可以创建分支，然后在分支上进行修改，最后将分支合并到主分支上。

如果我们在某个历史版本上做出一个修改，而不添加 branch 的话，那么当我们想要回到修改的版本时，就会发现我们的指针依然在 master 这个大 branch 上移动，而这个修改的版本，除非你记住了他的 commit id，否则就无法回到这个版本了。（因为没有路标通向他，所以当到达分叉路口时系统会自动以为只有一条路，也就是 master 分支）

有两种创建分支的方法：

- git branch <branch name>（基于当前的 header，也就是分岔路口）
- git branch <branch name> <branch id>（基于当前所在分支的 id 提交，也就是根据这条路提交）

2 github

GithubCLI 是一个命令行程序，通过这个程序可以使用命令行操控 github。现在很多没能注意的就是，github 的 commit 需要用签名去验证。在 git 中会出现一个 verfied 的图标，证明我使用我的私钥签名，并可以用公钥去验证。对比图如下：

基于此我设置了 github 的公钥验证，在本地的 git 上打开了 GPG 签名。

具体如何实现请详见 b 站 tonycrane 的录播。

另外，github 还给每个用户提供了二级域名，可以用仓库的组织形式去编写这一个域名作为自己的主页。对于每个仓库，他们也有自己的主页。

github 的 action 则是 github 提供的 CI（持续集成）和 CD（持续交付）工作。相当于配置自动化任务，在特定时间执行。
可以通过配置文件，文件名为.github/workflows/workflow_name.yml。编写可以看见 github 官方文档。

3 open-source software

3.1 oss versus free software?

所谓开源软件，意思就是公开源代码。将其源代码放在 github 上并设置 repo 为 public 就是开源。
自由软件则遵循四大原则：

- 自由运行
- 自由修改
- 自由分发 copy
- 自由分发 modify

注意开源并不等效于完全自由。自由和开源完全不一样，即便是开源，也可以通过规定开源证书来约束用户如何使用源代码。

copyright: 版权所有，一切权利归软件作者所有
书写方式：copyright© 最初发表年-最初更新年所有者. All rights reserved.
copyleft: 版权归作者所有，其他一切权利归任何人所有。他一定是自由软件。GPL 就是一种 copyleft 许可证。

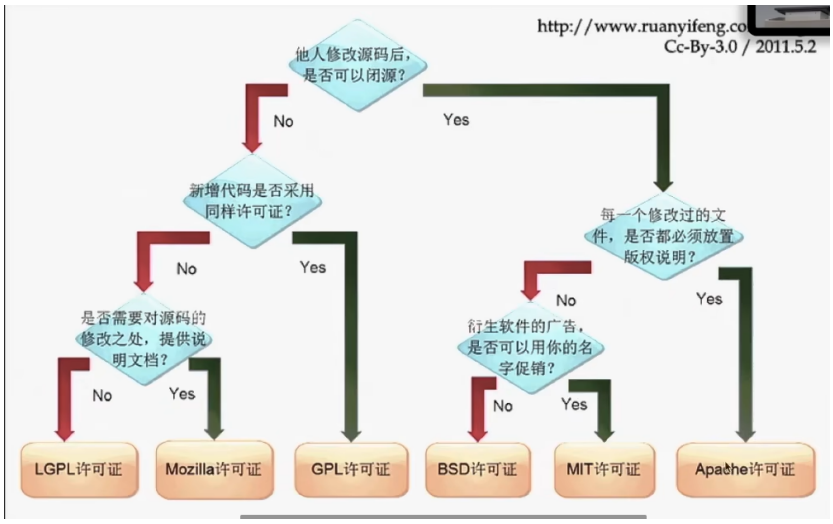
3.2 license

3.2.1 No license?

没有许可证意味着：
原作者保留所有权利，不允许复制、分发、修改。如果需要使用，请联系原作者。（这个很反直觉，算是某种“最小权利”的约束吧。）

3.2.2 Yes it has license!

也有一些变体，例如 AGPL 和 LGPL



3.2.3 Unlicense!

这个概念需要和没有许可证（我没有特意设置）的情况区分，unlicense 意味着放弃所有权利，该软件进入公共领域。

最后，详细的部分请见：
choosealicense.com

3.3 How to use a license?

在根目录下包含文件 LICENSE 即可，其中附上许可证内容。github 可以通过一些模板生成许可证，也会根据内容识别并显示许可证。

如果涉及多张许可证，都要放而且要声明许可证作用范围。

3.4 License unrelevent with software

一些许可证并不是用于软件的……而是用于知识。例如 cc（creative commons）许可证，用于知识共享。一般是用于知识型的内容，比如笔记。
详见：
creativecommons.org/share-your-work/cclicense/
官网上直接查看许可证内容，后缀改成 txt 就可以直接复制了。
注意 github 目前只识别 cc 0（公共领域），cc by（需要标明原作者），cc by-sa（需要采用相同许可证）。

4 github community

github 的项目贡献，一般先看这些：

- README
- CONTRIBUTING
- CODE_OF_CONDUCT

另外，提出 issue 之前，必须先查看是否有 issue 已经提出你的疑问！
And that's all. Thanks.