

数据结构与算法 (Python)

极简 Git 入门

谢正茂 webg@PKU-Mail

北京大学计算机系

March 17, 2021

什么是 Git

- Git /'git/ 是一个分布式的版本管理和协作工具
 - 版本管理：帮你自动保存、跟踪文档、代码的变化，而不用采用文件重命名的原始方式。文件重命名：
foo, foo.bak, foo.bak.bak, foo.2018, foo.20180703, foo.20200301, ...
 - 分布式的：本地仓库 (repository/repo) 和服务器仓库内容是一样的；服务器的可访问性较好，用它来做文件的中转。
 - 协作：支持多人对同一项目、同一文件进行并行开发、提交、合并。
- Linux 之父 Linus 觉得当时的版本管理系统都不好用，为开发 Linux 内核而创造。
- 开源法宝，程序员最爱：github.com
- Git 不但是一个版本控制系统，它也是个内容管理系统 (CMS)，工作管理系统等。
- 外面的 IT/互联网公司都在用，个人也可以用来管理自己的所有文档、代码、日记。

Git 的基本概念

- **repo/repository/仓库**: 用来存储一组相关文件的文件夹, 文件夹中有'.git' 的隐含目录, 存储了 git 的历史数据。
- **local/本地仓库**: 用户直接在本地仓库上工作, 对文件进行各种操作, 包括创建、编辑, 以及删除和重命名。
 - `git status`
- **commit/提交**: 完成一件任务的多个操作, 把它们装在一起的一个包裹。比如,
 - 修复一个 bug, 改了两个文件: `foo`, `bar`
 - 课程的一次作业包括三个部分: `code.py`, `report.txt`, `figure.png`除了分支和合并以外, **repo** 是由 **commit** 组成的“准”线性机构。
 - `git log`
- **stage/打包台**: 需要打包的东西要先放到打包台上, 对它们的内容检查好了之后, 再打包生成一个 **commit**。
 - `git add code.py report.txt figure.png`
 - `git commit -m "我数算课的第一次作业"`

- 命令行界面

- Linux: 自带, 开箱即用。
- Mac: 首次运行需要先安装, `$xcode-select --install`
- Windows: 下载安装, <https://git-scm.com/download/win>
 - 如果下载失败, 用这个地址
<https://www.zhengmao.ltd/Git-2.31.0-32-bit.exe>

- 图形界面

- <https://git-scm.com/downloads/guis>
- SourceTree, Github Desktop, TortoiseGit, Git Extensions, ...
- 推荐用命令行界面, 图形界面使用 SourceTree 为例子简单介绍。

极简使用流程 - 把仓库克隆到本地

- 从服务器上克隆一个空的仓库到本地
- `git clone stu[学号]@yongfeng.zhengmao.ltd:[学号].git`
 - "git clone" 后面的参数是一个远程仓库的地址;
 - `stu[学号]` 是用于登录服务器的帐号
 - `yongfeng.zhengmao.ltd` 是一台服务器的域名
 - `[学号].git` 是仓库在服务器上的路径名
 - 密码是 'dsa2021spring'
- clone 成功之后, 在当前目录下会出现[学号]的仓库目录
- 仓库中除了'.git' 隐含目录外就是空的, 就用来存放/管理本学期《数算 B》的作业。

```
78035
└── .git
    ├── branches
    ├── config
    ├── description
    ├── HEAD
    ├── hooks
    ├── info
    ├── objects
    └── refs
```

极简使用流程 - 在本地仓库中完成作业

- 首先要进入本地仓库目录
 - `cd [学号]`
- 在本地仓库中按作业号创建子目录
 - 第一次作业 `hw1`，第二次 `hw2`，...
 - 命令行方式: `mkdir hw1`
 - 文件管理器方式: 右键-> 创建新目录
- 在子目录中按要求放入代码文件，说明文件，图片等等。
- 运行"`git status`"，就能看到在仓库下有"`untracked`" 的目录`hw1/`。

```
tree 78035
78035
├── hw1
│   ├── code.py
│   ├── figure.png
│   └── report.txt
```

```
[xzm@yongfeng 78035]$ git status
On branch master

No commits yet

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

hw1/
```

极简使用流程 - 把作业文件放到打包台上

- 用 `git add` 命令把作业文件放到打包台上；在没有打包 `commit` 之前，打包台上的文件可以自由移上移下。
- 用 `git status` 命令，可以看到打包台上的新文件。

```
[xzm@yongfeng 78035]$ git add hw1/report.txt hw1/code.py hw1/figure.png
[xzm@yongfeng 78035]$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   hw1/code.py
    new file:   hw1/figure.png
    new file:   hw1/report.txt
```

极简使用流程 - 把打包台上的文件打成一个包 commit

- 在**第一次**打包之前，需要先设置一下作者/提交者的相关信息。
 - `git config -global user.name "John Doe"`
 - `git config -global user.email johndoe@example.com`
 - 把上面的名字和 email 换成自己的
- 然后运行`git commit -m "第一次作业"`，完成打包提交。打包时需要提供一定的说明信息，比如这里的“第一次作业”；成功的话会显示本次操作的相关信息。

```
[xzm@yongfeng 78035]$ git commit -m "第一次作业"
[master (root-commit) 36bb8d4] 第一次作业
3 files changed, 1 insertion(+)
create mode 100644 hw1/code.py
create mode 100644 hw1/figure.png
create mode 100644 hw1/report.txt
```

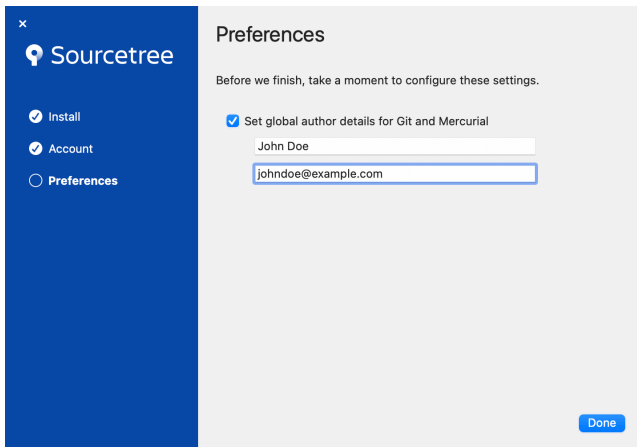

极简使用流程 - 把本地仓库同步到服务器上去

- `git push` # 大功告成！这里是注释。
- 哎呀！发现作业错了一点。
- 把错误改掉，然后对修改了的文件，把上面的 `add`->`commit`->`push` 再来一遍。
- 严格来说，每次打包的不是文件，而是文件的改动内容

```
[xzm@yongfeng 78035]$ git diff
diff --git a/hw1/code.py b/hw1/code.py
index 71f6596..9f69d32 100644
--- a/hw1/code.py
+++ b/hw1/code.py
@@ -1,1 @@
- print("hello, world")
+ print("hello, world")
```

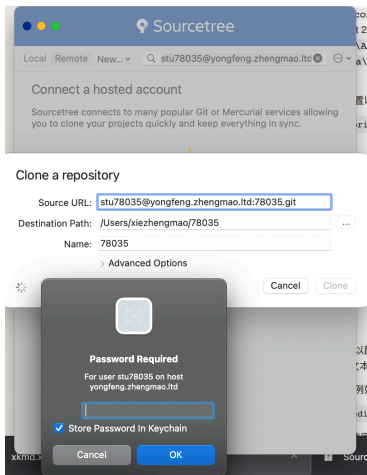
Sourcetree 的对应操作页面-配置用户

- `git config -global user.name "John Doe"`
- `git config -global user.email johndoe@example.com`



Sourcetree 的对应操作页面-克隆仓库

- git clone **stu[学号]**@yongfeng.zhengmao.ltd:**[学号].git**



Sourcetree 的对应操作页面-add/commit/push

- git add hw1/quicksort.py
- git commit -m "增加了一个文件"
- git push

