

公告

路飞学城作者

最新免费视频：PythonAV.com

Python技术交流群：737658057

限时免费内训课
学不到东西算我输！

- Python开发7天入门特训营
- 七天爬虫实战密训班

抢占免费名额

昵称：武沛齐
园龄：6年5个月
粉丝：6717
关注：45
+加关注

<	2018年12月						>
日	一	二	三	四	五	六	
25	26	27	28	29	30	1	
2	3	4	5	6	7	8	
9	10	11	12	13	14	15	
16	17	18	19	20	21	22	
23	24	25	26	27	28	29	
30	31	1	2	3	4	5	

我的标签

- ASP.NET MVC(15)
- Python(14)
- Tornado(5)
- python之路(4)
- 面试都在问什么？(2)
- Python开源组件 - Tyrion(1)
- Python面试315题(1)

随笔分类

- JavaScript(1)
- MVC(15)
- Python(14)
- 面试都在问什么系列？【图】(2)
- 其他(37)
- 请求响应(6)
- 设计模式(9)
- 微软C#(34)

随笔档案

- 2018年12月 (1)
- 2018年8月 (1)
- 2018年5月 (2)
- 2018年4月 (1)
- 2018年3月 (1)
- 2017年8月 (1)
- 2017年5月 (1)
- 2017年3月 (1)
- 2016年10月 (1)
- 2016年7月 (1)
- 2015年10月 (1)
- 2015年8月 (1)
- 2015年7月 (1)

老铁，这年头不会点Git真不行！！

版本控制

说到版本控制，脑海里总会浮现大学毕业是写毕业论文的场景，你电脑上的毕业论文一定出现过这番景象！

- 1 毕业论文_初稿.doc
- 2 毕业论文_修改1.doc
- 3 毕业论文_修改2.doc
- 4 毕业论文_修改3.doc
- 5 毕业论文_完整版1.doc
- 6 毕业论文_完整版2.doc
- 7 毕业论文_完整版3.doc
- 8 毕业论文_最终版1.doc
- 9 毕业论文_最终版2.doc
- 10 毕业论文_死也不改版.doc
- 11 ...

以上就是使用最原始的方式进行版本控制，但是这种方式有显著缺点：

- 多个文件，保留所有版本时，需要为每个版本保存一个文件...
- 协同操作，多人协同操作时，需要将文件打包发来发去...
- 容易丢失，被删除意味着永远失去...(可以选择网盘)

为了解决以上版本控制存在问题，应运而生了一批版本控制工具：VSS、CVS、SVN、Git等，其中Git属于绝对霸主地位。

注意：一般版本控制工具包含两部分

- 客户端（本地）：本地编写内容以及版本记录
- 服务端（网盘）：将内容和版本记录同时保存在远程（可有可无）

Git介绍

Git 是一个开源的分布式版本控制软件,用以有效、高速的处理从很小到非常大的项目版本管理。 Git 最初是由Linux Torvalds设计开发的，用于管理Linux内核开发。Git 是根据GNU通用公共许可证版本2的条款分发的自由/免费软件，安装参见：http://git-scm.com/

GitHub是一个基于Git的远程文件托管平台（同GitCafe、BitBucket和GitLab等）。

Git本身完全可以做到版本控制，但其所有内容以及版本记录只能保存在本机，如果想要将文件内容以及版本记录同时保存在远程，则需要结合GitHub来使用。使用场景：

- 无GitHub：在本地 .git 文件夹内维护历时文件
- 有GitHub：在本地 .git 文件夹内维护历时文件，同时也将历时文件托管在远程仓库

其他：

- 集中式：远程服务器保存所有版本，用户客户端有某个版本
- 分布式：远程服务器保存所有版本，用户客户端有所有版本

Git使用之小P创业史：初创期

小P是一个年轻有为程序员，从小立志要干出一番大事，某个深夜小P在网上查找**老师主演的学习视频，花了1个小时才找到想要的资源，小P想到和自己一样的有为青年每天花费大量的时间寻找喜欢老师的作品，感觉自己干大事的机会来了，毅然决然选择创业，创建一个**平台，提供**老师的所有资源！！

创业初期，小P独自封闭开发一个月，第一个版本终于上线：



回顾开发过程，其中辛酸只有小P自己知道。上线完成后的某一天，小P猛然看到自己开发目录，卧槽这拓麻也太乱了，加入那天程序出问题回滚到上个版本的时候，自己都找不到确定版本，并且我老子做的这个系统日后是要成千上万人来维护开发，这种通过原始文件来保存版本的形式简直Low到爆啊。

- ▶ pondo_登录注册_Bug修复
- ▶ pondo_登录注册_v1
- ▶ pondo_分页-1
- ▶ pondo_分页-2
- ▶ pondo_首页导航
- ▶ pondo_首页数据填充
- ▶ pondo_太难编了
- ▶ pondo_忘记内容了
- ▶ pondo_我也不知道是什么

文章分类

flask

书籍

git(4)

积分与排名

积分 - 348728

排名 - 597

最新评论

1. Re:最新免费视频放送【冒着被开除的风险】

@xiangxiongflyaa...

--yyxin

2. Re:最新免费视频放送【冒着被开除的风险】

keyi

--yyxin

3. Re:Tyrion中文文档（含示例源码）

@临冬城城主hao...

--yyxin

4. Re:为什么很多IT公司不喜欢进过培训机构的人呢？

我在看武老师讲的视频，对于一个编程爱好者，简直就是福音啊~讲的很不错啊，当然需要自己去课下学习。

--行走的小猫

5. Re:为什么很多IT公司不喜欢进过培训机构的人呢？

@漏网的小鱼请问千锋培训机构怎么样，好迷茫啊。...

--python-1807

开始调研：小P发现了版本控制神奇Git，但是都是道听途说，到底牛逼成什么样子也不清楚，所以抱着试试看的态度，小P开始使用Git进行版本控制。

```
1 MacBook-Pro-4:pondo wupeiqi$ pwd # 进入程序目录
2 /Users/wupeiqi/PycharmProjects/pondo
3 MacBook-Pro-4:pondo wupeiqi$ git init # git初始化
4 Initialized empty Git repository in /Users/wupeiqi/PycharmProjects/pondo/.git/
```

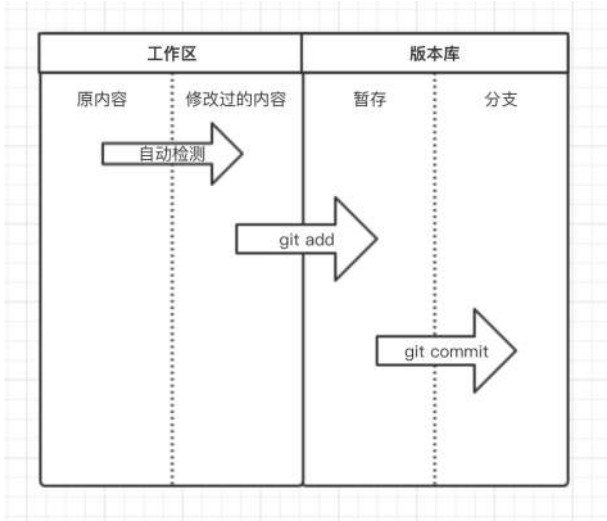
初始化后，会在当前目录自动创建 .git 文件夹，该文件是Git中最重要的文件夹，因为Git相关文件以及版本都将保存在该文件夹中，有了它，妈妈再也不用担心我好多文件来记录版本了，通过Git命令可以将所有版本保存在 .git 文件中，两条命令创建一个版本：

```
1 MacBook-Pro-4:pondo wupeiqi$ git status # 查看当前git状态
2 On branch master
3
4 Initial commit
5
6 Untracked files:
7   (use "git add <file>..." to include in what will be committed)
8
9   .idea/
10  app01/
11  db.sqlite3
12  manage.py
13  pondo/
14  readme
15  templates/
16
17 nothing added to commit but untracked files present (use "git add" to track)
18 MacBook-Pro-4:pondo wupeiqi$ git add . # 添加当前目录下所有文件到版本库
19 MacBook-Pro-4:pondo wupeiqi$ git commit -m '第一次提交' # 提交到版本库，并填写版本说明，以便
20 [master (root-commit) df47fe4] 第一次提交
21 33 files changed, 879 insertions(+)
22 create mode 100644 .idea/dictionaries/wupeiqi.xml
23 create mode 100644 .idea/encodings.xml
24 create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
25 ...
```

注意：执行git commit 命令时，可能会提示进行用户和邮箱的配置，该配置用于记录当前版本由那个用户提交

- git config --local user.name '武沛齐'
- git config --local user.email 'you@example.com'

Git把管理的文件分为了两个区域四个状态。



工作区：当前开发程序所在目录称为工作区，即：工作开发都是在该目录，该区域的文件会有状态的变化且状态由git自动检测，如果程序中文件做任何操作（增、删、改），文件状态均会被检测到，可以使用【git status】命令查看。

View Code

版本库：工作区检测到有文件发生变化，那么意味着较上一个版本之后对程序进行了修改，修改完成之后，可以当做下一版本进行提交，那么就是执行【git add .】将所有文件提交到暂存区，然后再执行【git commit -m '又一个版本'】提交到版本库的分支即可，之后可以使用【git log】命令查看版本记录。

View Code

目前已使用Git的四个命令，这四个命令已经可以代替本地多个文件保存版本的方式：

- git init，初始化，表示即将对当前文件夹进行版本控制。
- git status，查看Git当前状态，如：那些文件被修改过、那些文件还未提交到版本库等。
- git add 文件名，将指定文件添加到版本库的暂存状态。
- git commit -m '提交信息'，将暂存区的文件提交到版本库的分支。
- git log，查看提交记录，即：历史版本记录

调研完，小P好气自己哟，这么6的东西为什么没有早发现，从此小P的版本管理就告别繁杂的文件夹了，赶紧搞起来。

+ View Code

恰好，此时需要开发一个非洲专区的功能，再也不用重新copy一遍文件了，在工作区直接开始搞起来，30分钟开发测试完成，又一个版本完成了咯!!!

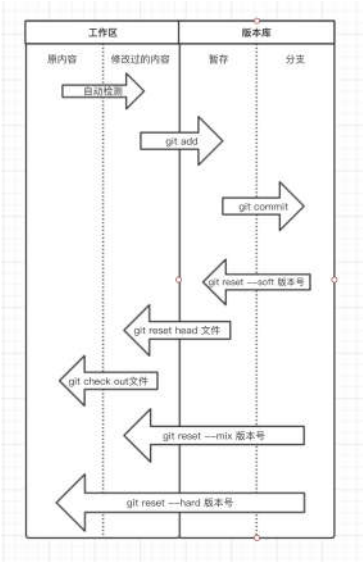
+ View Code

非洲专区上线一个月后，接连收到用户投诉，原来清新脱俗的小P那里去了？怎么变得如此重口味？想回到过去....

小P向来秉承为人民服务的原则，人民不想看那一定要修改。决定：回滚，回到上一个版本。

那么问题来了？

一个月过去了，代码修改的位置早就忘记了，怎么修改，总不能再开发一遍吧。机智的小P猜想Git既然这么牛逼，应该会提供这样的功能，经过一番查找，果不其然Git提供了这个回滚的功能。



回滚到指定版本：

+ View Code

回滚倒是完成了，小P在想如果某一天想要在回有非洲专区功能的版本怎么办呢？来来来，不能像以往通过【git log】来查看记录再回滚了，再回去需要这么搞：

+ View Code

Git使用之小P创业史：成长期

企业想要不被淘汰，就要跟紧时代步伐，近日直播行业日趋火热，小P的也希望自己的平台加入直播功能，已经评估预计2个月开发完成，小P开始没日没夜的干了起来...

一个月过去了，开发任务和按照预期有条不紊的进行着，直播功能也已完成一半，就是在此时线上运行平台出现Bug需要紧急修复，怎么办？怎么办？？怎么办？？？

小P出了几个解决方案：

- 正在开发代码不动，拷贝一份线上运行的代码进行修改 -----> 不行，又踏马回去拷贝去了。
- 把开发了一个月的代码删掉，修改Bug，然后再重头开始 -----> 不行，一定是傻逼才这么干，我不是傻逼。
- 听说git的stash可以 -----> 听说过，没见过
- 听说git的branch可以 -----> 听说过，没见过

方案一：stash

stash用于将工作区发生变化的所有文件获取临时存储在“某个地方”，将工作区还原当前版本未操作前的状态；stash还可以将临时存储在“某个地方”的文件再次拿回到工作区。

+ View Code

特别的：执行 git stash pop 命令时，可能会遇到冲突，因为在紧急修复bug的代码和通过stash存储在“某个地方”的代码会有重合部分，所以执行 git stash pop 时候就会出现冲突，有冲突解决冲突即可。

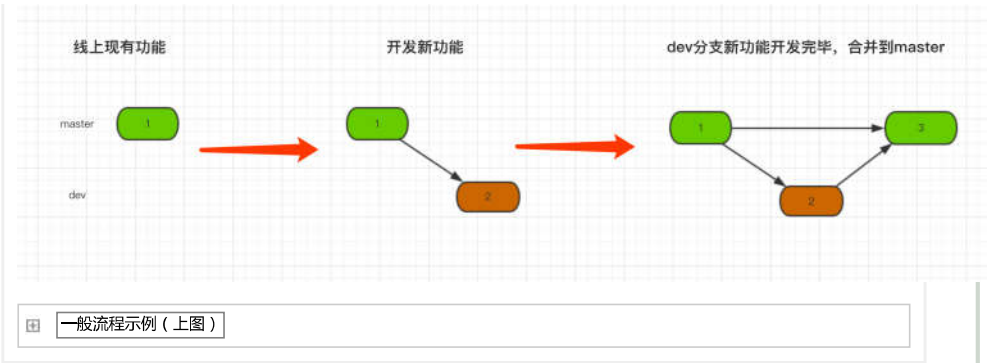
git stash pop 出现冲突

stash相关常用命令：

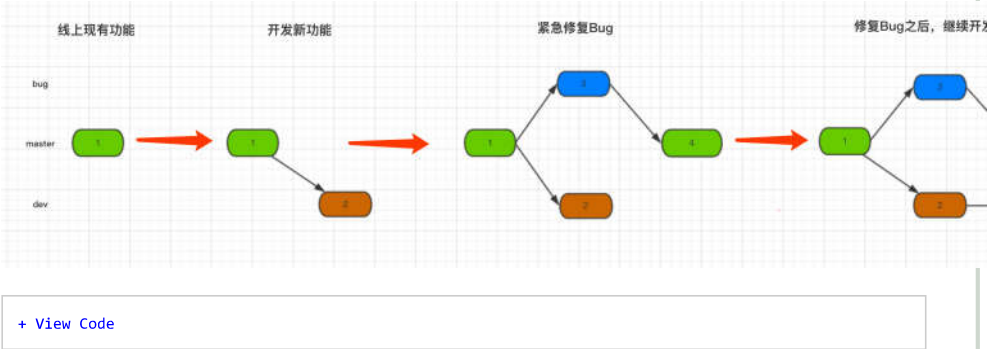
- git stash 将当前工作区所有修改过的内容存储到“某个地方”，将工作区还原到当前版本未修改过的状态
- git stash list 查看“某个地方”存储的所有记录
- git stash clear 清空“某个地方”
- git stash pop 将第一个记录从“某个地方”重新拿到工作区（可能有冲突）
- git stash apply 编号，将指定编号记录从“某个地方”重新拿到工作区（可能有冲突）
- git stash drop 编号，删除指定编号的记录

方案二：branch

分支学习：branch称为分支，默认仅有一个名为master的分支。一般开发新功能流程为：开发新功能时会在分支dev上进行，开发完后再合并到master分支。



学习参考上图，小P也可以按照着这样的流程进行开发，如果遇到上文开发到一般需要临时修复Bug的情况，可以按照下图的流程进行：



注意：git merge 时也可能会出现冲突，解决冲突的方式上述stash相同，即：找到冲突文件，手动修改冲突并提交，此处不再赘述。

branch相关常用命令：

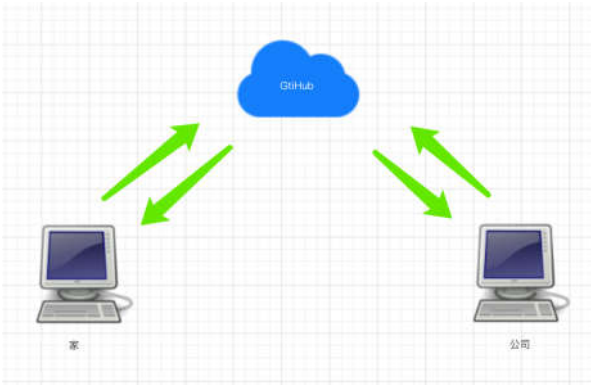
- git branch 分支名称 创建分支
- git checkout 分支名称 切换分支
- git branch -m 分支名称 创建并切换到指定分支
- git branch 查看所有分支
- git branch -d 分支名称 删除分支
- git merge 分支名称 将指定分支合并到当前分支

Git使用之小P创业快速发展期

小P不忘初心始终如一的为广大有为青年提供资源，使得网站的访问量不断攀升，已经出具规模并赚了一些钱，有钱就要造么，索性国贸租了一间写字楼用于办公，并且也完善运营市场团队。。屌丝终究是屌丝，小P还是离不开写代码的习惯，所以开发的任务还是由自己一人承担，小P从此开始了白天在国贸写代码，晚上回天通苑写代码。PS：有钱，公司一台新电脑，家里一台原来老电脑。。。。 妈的，故事怎么才能变得有趣呢？太拓麻烦了。

小P心里开始寻思，我爱写代码，公司写，家里写，如果每天来回带一个U盘拷贝着实麻烦，Git有没有类似于云盘似得东西可以进行数据同步呢？答案肯定是有。 必须有，不然老子真的就编不下去了。

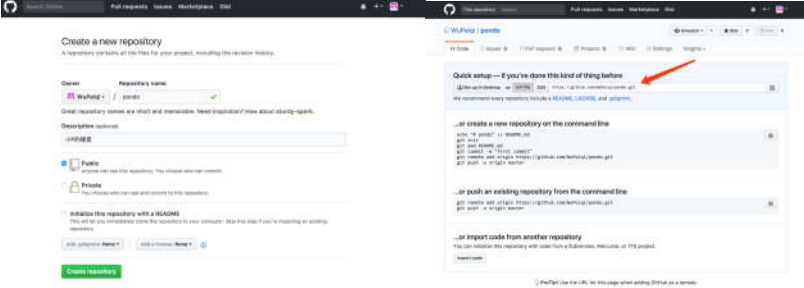
GitHub，一个基于Git实现的代码托管的平台，可以将内容以及版本记录在远程也保存一份，这样就不用U盘咯（类似于云盘）。PS：类似GitHub的产品还有许多，如：GitLab、Bitbucket、码云等。



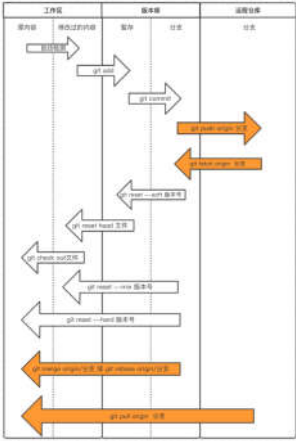
基于GitHub实现代码托管，需要一下步骤：

- 注册GitHub

- 创建仓库，创建完仓库后会会有一个URL代指该仓库，如：



- git可以用该URL进行向远程推送版本信息或获取版本信息



小P学会使用Git和GitHub之后，就可以基于GitHub进行代码远程托管。

在家里，小P开发完毕部分功能将代码推送到GitHub。

+ View Code

在公司，新电脑第一次使用，需要将代码从GitHub中获取并继续开发，开发完事下班就下班回家。

+ View Code

在家里，由于白天在公司已经开发一部分功能并提交到GitHub，家里电脑的代码还是昨晚的版本，所以需要从GitHub拉去最新代码，然后继续开发。

+ View Code

在公司，由于昨天晚上在家已经开发了一部分功能，在公司需要先把昨晚开发的功能从GitHub中拉取，并继续开发。

+ View Code

长此以往，将Git和GitHub结合使用做到避免电脑损坏造成数据丢失以及多地开发的问题，上文执行过程中执行【git pull origin 分支】命令等同于【git fetch origin 分支】+【git merge origin/分支】，并且在执行过程中可能会出现冲突，原因是由于本地代码和获取的最新代码有重合部分，那么就需要自己手动解决冲突然后再继续开发。

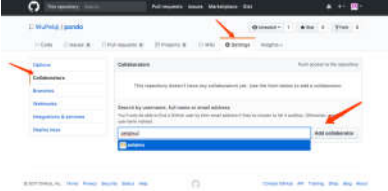
Git使用之小P创业成熟期

小P的公司发展越来越好，但是公司产品单一是严重缺点，经过学习考察小P决定再招聘3个Python程序开发另外一个产品“约P”平台来丰富公司业务线，为用户提供一整套服务。

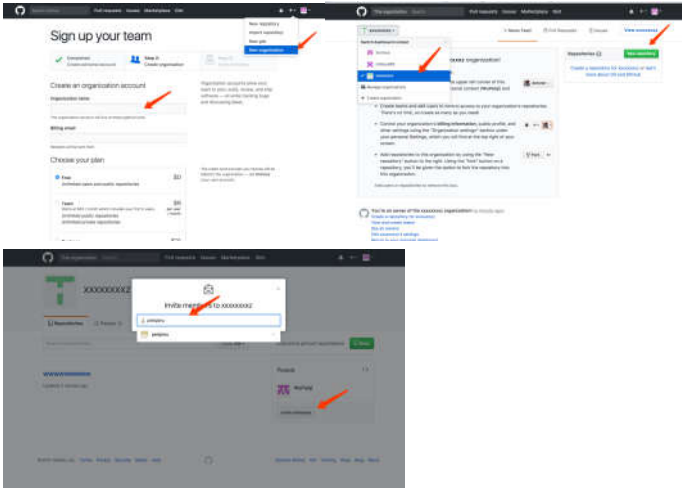
小P的Slogan：看了想要，想要就约。 不要问我怎么想要的，我自己也不知道 哈哈哈哈哈

“约P”平台需要三人协同开发，GitHub中多人协同开发和单人开发还是有点差别，协同开发一般有两种方式：

- 合作者，将其他用户添加到仓库合作者中之后，该用户就具有向当前仓库提交代码。



- 组织，创建一个组织，然后再该组织下可以创建多个项目，组内成员可以向组内所有项目提交代码。PS：也可以对某个项目指定合作者



协同开发命令和以上步骤类似，此处就不再重新写代码，而是使用文件描述三人协同开发整个过程。

- 创建程序
 - 用户A创建程序，提交到GitHub
 - 用户B克隆项目
 - 用户C克隆项目
- 开发功能
 - 用户A开发功能1
 - 用户B开发功能2
 - 用户C开发功能3
- 提交
 - 用户A提交功能1，并push（A用户手速快，先提交。）
 - 用户B提交功能2，无法push，因为GitHub上已经有其他人提交的新代码。
解决方法：从GitHub上获取最新代码并合并到本地，提交自己开发的功能2。
 - 用户C提交功能3，无法push，无法提交，因为GitHub上已经有其他人提交的新代码。
解决方法：从GitHub上获取最新代码并合并到本地，提交自己开发的功能3。
- 获取最新代码
 - 用户A获取最新代码
 - 用户B获取最新代码
 - 用户C获取最新代码

在上面红色标注的解决方法位置可以有三种方式操作，三者都可以完成合并并提交新功能，但是日志记录会有差异，如：前两者版本记录中会出现合并，而第三种可以保证版本记录干净整洁。

- 先 git pull origin master 然后 git push origin master
- 先 git fetch origin master 然后 git merge origin/master 再 git push origin master

[+ View Code](#)

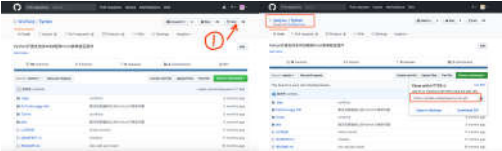
- 先 git fetch origin master 然后 git rebase origin/master 再 git push origin master

[+ View Code](#)

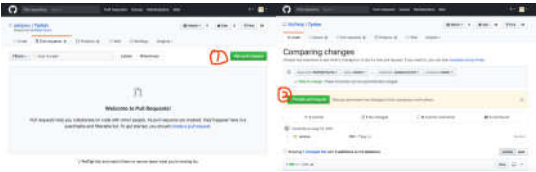
使用Git之小P创业上市期

终于终于小P等到了公司上市实现财务自由，但作为一个技术屌还是脱离不了屌丝的本质，所以每天都是逛逛github，看看别人有什么好的项目，自己可以给他挑挑bug装装逼，但是别人不可能给小P搞成合作者什么的，那怎么才能给别人贡献代码呢？那就是fork了。。。。

- 找到想搞的项目，fork一下，然后这个项目就在自己仓库出现了



- 从自己仓库获取代码并进行编辑提交
- 创建并提交一个pull request，然后等待原作者是否同意这个pull request，如果同意那么在作者的源代码中就推出现小P提交的功能



其他补充

1. 配置文件

Git的配置文件有三个：

- 系统配置：/private/etc/gitconfig
- 用户配置：~/.gitconfig
- 项目配置：.git/config

2. 用户凭证

由于Git和Github交互操作可能会很频繁，那么一定少了用户授权的操作，为了防止每次操作重复输入用户名和密码，Git提供了两种解决方法：

- 秘钥

首先创建一对秘钥 `ssh-keygen -t rsa`，然后将 `id_rsa.pub` (公钥)内容拷贝到github中，日后操作无需再输入用户名和密码。

注意：这种方式需要使用Git中 `git@github.com:WuPeiqi/xxxx.git` 格式地址。

- 密码

Https访问git时，避免每次操作需要输入用户名和密码，可以在配置文件中添加如下配置项：

```
[credential]
helper = store/cache/ 第三方
```

store:

表示将用户名和密码保存在硬盘上

第一次输入过用户名和密码之后，用户名和密码就会保存在当前用户根目录的 `.git-credentials` 文件中，内容格式为：

`https://用户名:密码@github.com`

自动添加配置命令：`git config credential.helper store`

cache:

表示将用户名和密码保存在缓存中

第一次输入过用户名和密码之后，用户名和密码就会保存在缓存中，默认超时时间是 900 秒，缓存相关文件保存在当前用户根目录的 `git-credential-cache` 中

自动添加配置命令：

```
git config credential.helper cache
git config credential.helper 'cache --timeout=300'
```

相关操作：

清除缓存：`git credential-cache exit`

指定超时：

```
[credential]
helper = cache --timeout=300
```

注意：

这种方式需要使用Git中 `https://github.com/WuPeiqi/xxxx.git` 格式地址。

指定用户名和密码：`https://用户名:密码@github.com/wupeiqi/xxx.git`

就酱紫，以后想到再加吧...



- 以星号“*”通配多个字符；
- 以问号“?”通配单个字符
- 以方括号“[]”包含单个字符的匹配列表；
- 以叹号“!”表示不忽略(跟踪)匹配到的文件或目录；



```
git tag -a v1.0 -m '版本介绍'      本地创建Tag
git show v1.0                      查看
git tags -n                        查看本地Tag
git tag -l 'v1.4.2.*'              查看本地Tag, 模糊匹配
git tag -d v1.0                    删除Tag
git push origin :refs/tags/v0.2    更新远程tag
git checkout v.10                  切换tag
git fetch origin tag V1.2

git push origin --tags
git pull origin --tags


git clone -b v0.1
```



```
1  # Byte-compiled / optimized / DLL files
2
3  # pycharm
4  .idea/
5  .DS_Store
6  offline-script/
7  media/
8
9  # database migrations
10 */migrations/*.py
11 !*/migrations/__init__.py
12
13
14 __pycache__/
15 *.py[cod]
16 *$py.class
17
18 # Django stuff:
19 *.log
20 local_settings.py
21 *.sqlite3
22
23
24 # C extensions
25 *.so
26
27 # Distribution / packaging
28 .Python
29 build/
30 develop-eggs/
31 dist/
32 downloads/
33 eggs/
34 .eggs/
35 lib/
36 lib64/
37 parts/
38 sdist/
39 var/
40 wheels/
41 *.egg-info/
42 .installed.cfg
43 *.egg
44 MANIFEST
45
46 # PyInstaller
47 # Usually these files are written by a python script from a template
```

```
48 # before PyInstaller builds the exe, so as to inject date/other infos into it.
49 *.manifest
50 *.spec
51
52 # Installer logs
53 pip-log.txt
54 pip-delete-this-directory.txt
55
56 # Unit test / coverage reports
57 htmlcov/
58 .tox/
59 .coverage
60 .coverage.*
61 .cache
62 nosetests.xml
63 coverage.xml
64 *.cover
65 .hypothesis/
66 .pytest_cache/
67
68 # Translations
69 *.mo
70 *.pot
71
72
73 # Flask stuff:
74 instance/
75 .webassets-cache
76
77 # Scrappy stuff:
78 .scrappy
79
80 # Sphinx documentation
81 docs/_build/
82
83 # PyBuilder
84 target/
85
86 # Jupyter Notebook
87 .ipynb_checkpoints
88
89 # IPython
90 profile_default/
91 ipython_config.py
92
93 # pyenv
94 .python-version
95
96 # celery beat schedule file
97 celerybeat-schedule
98
99 # SageMath parsed files
100 *.sage.py
101
102 # Environments
103 .env
104 .venv
105 env/
106 venv/
107 ENV/
108 env.bak/
109 venv.bak/
110
111 # Spyder project settings
112 .spyderproject
113 .spyproject
114
115 # Rope project settings
116 .ropeproject
117
118 # mkdocs documentation
119 /site
120
121 # mypy
```

```
122 .mypy_cache/  
123 .dmy.py.json  
124 dmy.py.json
```



作者：[武沛齐](#)
出处：<http://www.cnblogs.com/wupeiqi/>
本文版权归作者和博客园共有，欢迎转载，但未经作者同意必须保留此段声明，且在文章页面明显位置给出原文连接。

分类: [Python](#)

标签: [Python](#)

好文要顶

关注我

收藏该文







武沛齐

关注 - 45

粉丝 - 6721

+加关注

511

« 上一篇：[你真的了解WebSocket吗？](#)
» 下一篇：[2. tzl师兄【面试题总结】](#)

posted @ 2017-08-06 17:29 [武沛齐](#) 阅读(29719) 评论(53) 编辑 收藏

[< Prev](#) [1](#) [2](#)

评论列表

# 51楼	2018-09-18 12:13	慕沁	alert(123)	支持(0)	反对(0)
# 52楼	2018-12-07 10:56	LittleGod	很好哦	支持(0)	反对(0)
# 53楼	2018-12-13 12:31	Zjacksparrow	厉害了 武藤兰老师	支持(0)	反对(0)

[< Prev](#) [1](#) [2](#)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

- 【推荐】超50万VC++源码：大型组态工控、电力仿真CAD与GIS源码库！
- 【活动】华为云12.12会员节 云产品1折起 满额送Mate20 点击抢购
- 【推荐】服务器100%基准CPU性能，1核1G首年168元，限时特惠！



腾讯云

腾讯云AMD云服务器

节省IT成本30%

1核1G AMD机型0.57元/天起

立即抢购

- 相关博文：
- [github 【第三章】Github综合](#)
 - [git和GitHub](#)
 - [github\(关键知识点\)](#)

- [Git操作](#)
- [玩转Git](#)

最新新闻：

- [Netflix：没有碰过Facebook用户的私信](#)
 - [贾跃亭又晒“自拍”！没躲在别墅，在预量产车下线现场](#)
 - [Facebook被曝用外部APP采集约会等隐私](#)
 - [土巴兔被曝撤销IPO申请 公司称静默期不便回应](#)
 - [ofo回应是否有钱退押金：商业机密，不便透露](#)
- » [更多新闻...](#)