1. **[执行git push出现"Everything up-to-date"](https://www.cnblogs.com/kevingrace/p/6259905.html)**

在github上git clone一个项目，在里面创建一个目录，然后git push的时候，出现报错"Everything up-to-date"。

原因：  
1）没有git add .  
2）没有git commit -m "提交信息"

如果上面两个步骤都成功执行，还出现这个错误是因为创建的目录下是空的，目录下必须有文件才能git push上传成功。

在github上创建文件的时候，在新文件名后加/符号就是文件夹，但是这种方式只支持英文名目录，中文名目录不支持。

要是想创建中文名文件夹，就通过客户端工具或终端命令行实现。如果在github一个文件夹下只有一个文件，那么删除这个文件的同时，它所在的文件夹也一同删除了。

1. 详解git fetch与git pull的区别

<https://blog.csdn.net/riddle1981/article/details/74938111>

git fetch和git pull都可以将远端仓库更新至本地那么他们之间有何区别?想要弄清楚这个问题有有几个概念不得不提。

FETCH\_HEAD： 是一个版本链接，记录在本地的一个文件中，指向着目前已经从远程仓库取下来的分支的末端版本。

commit-id：在每次本地工作完成后，都会做一个git commit 操作来保存当前工作到本地的repo， 此时会产生一个commit-id，这是一个能唯一标识一个版本的序列号。 在使用git push后，这个序列号还会同步到远程仓库。

有了以上的概念再来说说git fetch：git fetch：这将更新git remote 中所有的远程仓库所包含分支的最新commit-id, 将其记录到.git/FETCH\_HEAD文件中

git fetch更新远程仓库的方式如下：

git fetch origin master:tmp

//在本地新建一个temp分支，并将远程origin仓库的master分支代码下载到本地temp分支

git diff tmp

//来比较本地代码与刚刚从远程下载下来的代码的区别

git merge tmp

//合并temp分支到本地的master分支

git branch -d temp

//如果不想保留temp分支 可以用这步删除

（1）如果直接使用git fetch，则步骤如下：

创建并更新本 地远程分支。即创建并更新origin/xxx 分支，拉取代码到origin/xxx分支上。

在FETCH\_HEAD中设定当前分支-origin/当前分支对应，如直接到时候git merge就可以将origin/abc合并到abc分支上。

（2）git fetch origin  
只是手动指定了要fetch的remote。在不指定分支时通常默认为master  
（3）git fetch origin dev  
指定远程remote和FETCH\_HEAD，并且只拉取该分支的提交。

git pull : 首先，基于本地的FETCH\_HEAD记录，比对本地的FETCH\_HEAD记录与远程仓库的版本号，然后git fetch 获得当前指向的远程分支的后续版本的数据，然后再利用git merge将其与本地的当前分支合并。所以可以认为git pull是git fetch和git merge两个步骤的结合。  
git pull的用法如下：

git pull <远程主机名> <远程分支名>:<本地分支名>

//取回远程主机某个分支的更新，再与本地的指定分支合并。

因此，与git pull相比git fetch相当于是从远程获取最新版本到本地，但不会自动merge。如果需要有选择的合并git fetch是更好的选择。效果相同时git pull将更为快捷。

1. Git 里面的 origin 到底代表啥意思?

Watch，就是一楼说的关注，关注后，代码库中有新的commit你都会收到通知；  
Star，与watch不同，star相当于收藏，你可以方便地找到你star过的库，但是不会收到关于那个库的任何通知。 fork的意思是从别人的代码库中复制一份到你自己的代码库，与普通的复制不同，fork包含了原有库中的所有提交记录，fork后这个代码库是完全独立的，属于你自己，你可以在自己的库中做任何修改，当然也可以通过Pull Request向原来的库提交合并请求

<https://www.zhihu.com/question/27712995>

你的代码库(repository)可以存放在你的电脑里，同时你也可以把代码库托管到Github的服务器上。

在默认情况下，origin指向的就是你本地的代码库托管在Github上的版本。

我们假设你首先在github上创建了一个Repository，叫做repository，假设你的Github ID是user1,这个时候指向你的代码库的链接是

<https://github.com/user1/repository>

如果你在terminal里输入

git clone <https://github.com/user1/repository>

那么git就会在本地拷贝一份托管在github上的代码库，这个时候你cd到repository，然后输入git remote –v，你会看到控制台输出

origin https://github.com/user1/repository.git (fetch)

origin https://github.com/user1/repository.git (push)

也就是说git为你默认创建了一个指向远端代码库的origin（因为你是从这个地址clone下来的）

再假设现在有一个用户user2 fork了你个repository，那么他的代码库链接就是这个样子

https://github.com/user2/repository

如果他也照着这个clone一把，然后在他的控制台里输入

git remote -v

他会看的的就是

origin https://github.com/user2/repository.git (fetch)

origin https://github.com/user2/repository.git (push)

可以看的origin指向的位置是user2的的远程代码库

这个时候，如果user2想加一个远程指向你的代码库，他可以在控制台输入

git remote add upstream https://github.com/user1/repository.git

然后再输入一遍 git remote -v

输出结果就会变为

origin https://github.com/user2/repository.git (fetch)

origin https://github.com/user2/repository.git (push)

upstream https://github.com/user1/repository.git (push)

upstream https://github.com/user1/repository.git (push)

增加了指向user1代码库的upstream，也就是之前对指向位置的命名。

总结来讲，顾名思义，origin就是一个名字，它是在你clone一个托管在Github上代码库时，git为你默认创建的指向这个远程代码库的标签，

[@陈肖恩](//www.zhihu.com/people/a5a1cd08e0afc7165ec1f74069541186)

的答案并不准确，origin指向的是repository，master只是这个repository中默认创建的第一个branch。当你git push的时候因为origin和master都是默认创建的，所以可以这样省略，但是这个是bad practice，因为当你换一个branch再git push的时候，有时候就纠结了

1. Git密钥的使用

4.1 确认本地秘钥

SSH 秘钥默认储存在账户的主目录下的 ~/.ssh 目录如：C:\Users\BF100400\.ssh\查看是否包含id\_rsa和id\_rsa.pub(或者是id\_dsa和id\_dsa.pub之类成对的文件)，有.pub 后缀的文件就是公钥，另一个文件则是密钥。如果有这两个文件，则跳过1.2；如果没有这两个文件，甚至.ssh目录也没有，则需要用ssh-keygen 来创建。

4.2密钥的生成

在.ssh 目录下右键打开Git Bash(.ssh目录不存在，则在任一目录下操作，或者手动创建该目录)

生成秘钥：ssh-keygen -t rsa -C "your\_email@youremail.com" ，直接Enter就行，然后会提示输入密码(可输可不输)，说明：命令中的email，就是gitlab中的账号，需要保持一致

执行完成之后，在.ssh 目录下就会生成秘钥文件（没有.ssh目录的会自动生成，手动创建的则不会重复生成）

