

## Github进行fork后如何与原仓库同步

问题场景：

公司要求所有的代码提交都要先通过自己的库提交到主repo上去，所以登录自己的github网页上fork公司的仓库到自己github，在本地克隆自己github仓库代码到本地工作区，修改代码后提交到远程自己库上，然后在自己github网页上发起一个merge request请求，然后等待主repo主人review，同意之后合入。

问题是同时也有其他同事在主repo合入代码，所以我要定期和主repo保持同步。

整体思路如下：

在自己的本地添加公司的仓库为主repo为上游代码库，注意只是配置原仓库的路径，并没有真正clone原仓库，然后将远程公司主repo同步到自己本地的机器，然后本地的机器再push到自己的远程的fork仓库所有的操作都要在本地命令行完成

### 步骤一

- 1) 首先登陆到自己的远程仓库，fork主仓库；这样自己的远程仓库与公司主仓库一致
- 2) 本地仓库有两种方式下载自己远程仓库（或公司的远程仓库）

```
git https(ssh):xxxxxx.git //远程主仓库链接
git https(ssh):xxxxxx.git //自己的远程仓库
```

- 3) 然后在本地的机器修改代码再push到自己的远程的fork库

```
git push
```

以上操作只是完成从公司仓库到个人远程仓库的过程，在协同开发的时候还需要下面的神操作

照着这个操作，完全没问题：

### 步骤二

我们在进行Github协同开发的时候，往往会去fork一个仓库到自己的Github中，过一段时间以后，原仓库可能会有各种提交以及修改，很可惜，Github本身并没有自动进行同步的机制，这个需要我们手动去执行，现在我来演示一下如何进行自己的仓库和原仓库进行Github同步的操作。

(1) 我使用终端 命令行操作。首先在终端中配置原仓库的位置。进入项目目录，执行如下命令：查看你的远程仓库的路径。

```
$ git remote -v
origin https://github.com/YOUR_USERNAME/YOUR_FORK.git (fetch)
origin https://github.com/YOUR_USERNAME/YOUR_FORK.git (push)
```

- (2) 配置原仓库的路径：

下面这步操作即添加公司主repo为上游代码库

注意一定要cd到你自己的fork出来的库里面去，比如工程名叫luoluo，那要先cd到luoluo中去，然后才能操作

```
$ git remote add upstream https://github.com/ORIGINAL_OWNER
/ORIGINAL_REPOSITORY.git
```

<http://blog.csdn.net/>

(3) 再次查看远程目录的位置:

```
$ git remote -v
origin    https://github.com/YOUR_USERNAME/YOUR_FORK.git (fetch)
origin    https://github.com/YOUR_USERNAME/YOUR_FORK.git (push)
upstream  https://github.com/ORIGINAL_OWNER/ORIGINAL_REPOSITORY.git (fetch)
upstream  https://github.com/ORIGINAL_OWNER/ORIGINAL_REPOSITORY.git (push)
```

(4) 抓取原仓库的修改文件:

```
$ git fetch upstream
remote: Counting objects: 75, done.
remote: Compressing objects: 100% (53/53), done.
remote: Total 62 (delta 27), reused 44 (delta 9)
Unpacking objects: 100% (62/62), done.
From https://github.com/ORIGINAL_OWNER/ORIGINAL_REPOSITORY
 * [new branch]      master    -> upstream/master
```

执行效果如下所示

```
$ git fetch upstream
warning: Permanently added the RSA host key for IP address '54.255.254.175' to t
he list of known hosts.
remote: Counting objects: 193, done.
remote: Compressing objects: 100% (137/137), done.
remote: Total 193 (delta 72), reused 102 (delta 35)
Receiving objects: 100% (193/193), 36.90 KiB | 169.00 KiB/s, done.
Resolving deltas: 100% (72/72), done.
From git.liebaopay.com:nest/pegasi
 * [new branch]      2018_Branding_index -> upstream/2018_Branding_index
 * [new branch]      619css              -> upstream/619css
 * [new branch]      Jira-TIANMA-223     -> upstream/Jira-TIANMA-223
 * [new branch]      Jira-TIANMA-286-FRONT -> upstream/Jira-TIANMA-286-FRONT
 * [new branch]      TIANMA-619_CSS      -> upstream/TIANMA-619_CSS
 * [new branch]      TIANMA-703_appLock  -> upstream/TIANMA-703_appLock
 * [new branch]      TIANMA-715_AdNetwork -> upstream/TIANMA-715_AdNetwork
 * [new branch]      TIANMA-748_format   -> upstream/TIANMA-748_format
 * [new branch]      branch_fe           -> upstream/branch_fe
 * [new branch]      dateCon              -> upstream/dateCon
 * [new branch]      feature              -> upstream/feature
 * [new branch]      master               -> upstream/master
 * [new branch]      streaming            -> upstream/streaming
 * [new branch]      unit                 -> upstream/unit
```

(5) 切换到master分支。

```
$ git checkout master
Switched to branch 'master'
```

<http://blog.csdn.net/>

(6) 合并本地的master分支:

下面这行代码执行结束之后, 本地代码会立刻和主库保持同步, 非常神奇

```
$ git merge upstream/master
Updating a422352..5fdff0f
Fast-forward
 README          | 9 -----
 README.md       | 7 ++++++
2 files changed, 7 insertions(+), 9 deletions(-)
delete mode 100644 README
create mode 100644 README.md
```

(7) 此时，你的本地库已经和原仓库已经完全同步了。但是注意，此时只是你电脑上的本地库和你的远程的github原仓库同步了，你自己的github仓库还没有同步，此时需要使用“git push”命令把你本地的仓库提交到github中。

-----这是华丽的分割线-----

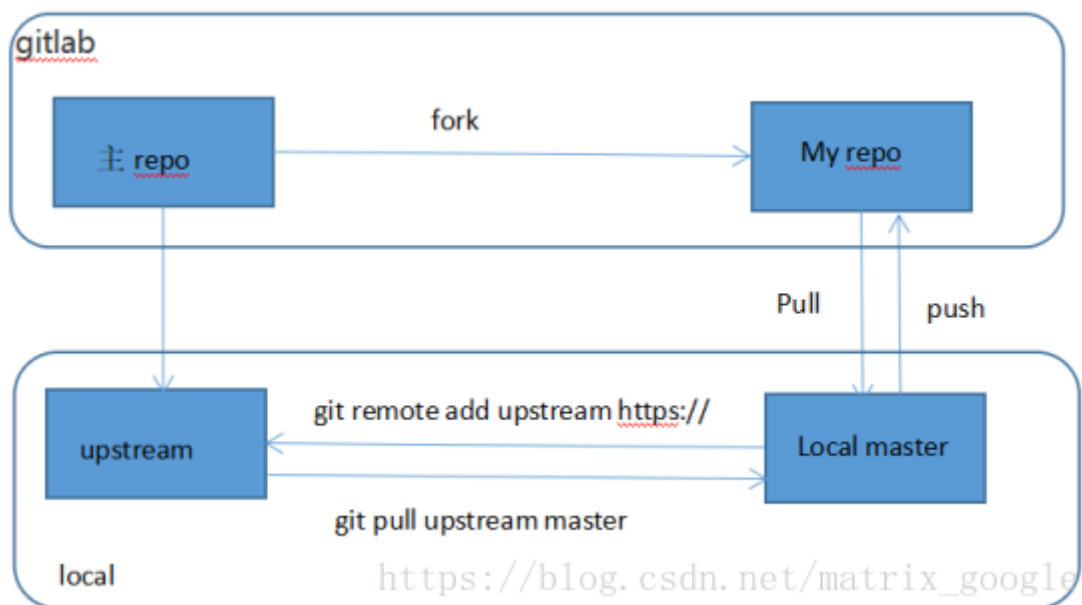
其实(4)(5)(6)可以合并成一条命令

git pull upstream master/(或其他分支)

第一个参数upstream 表示远程主repo

第二个参数master 表示自己fork库的master分支

为了说清楚这个问题，亲自做了一张图



**注：每天上班就要做至少一次上面的操作**

**如何git 同步代码到另外一个分支**

例如：将master分支的代码同步到develop分支（从同步本地仓库到推送至远程仓库）

（以下指令都是在本地电脑的仓库目录操作）

```
git checkout develop    //转换到develop分支
git merge master        //更新到与master一致
git push
```

上面三个指令操作，就完成了master分支的代码同步到develop分支并推送到自己的远程develop的仓库

#### 删除远程分支指令(也是在电脑终端上执行)

```
git push origin -delete 分支名    //删除远程仓库的分支
git branch -r            //查看执行删除分支指令后的分支列表
```

#### 在本地仓库建立了一个分支（Develop），并把这分支推送自己的远程仓库

```
git checkout -b Develop //在本地建立一个Develop分支
git push --set-upstream origin Develop //把本地的Develop分支直接推送到远程主仓库，自己已有操作远程主仓库的权限
```

#### 从远程主仓库克隆代码之后的修改或更新并推送到自己的远程仓库

```
//执行更新远程代码之后 -->git fetch upstream/(对应的分支)
git add .                //提交到本地缓冲区
git commit -m "xxx"      //添加备注"xxx",提交到本地文档
```

完成以上两个步骤；就可以用以下指令推送到自己的远程仓库了

```
git push    //吧自己改动或更新的代码推送到自己的远程仓库
```

**请求与主仓库合并，是要登录到自己的远程仓库的操作界面进行PR请求，并等候审核**