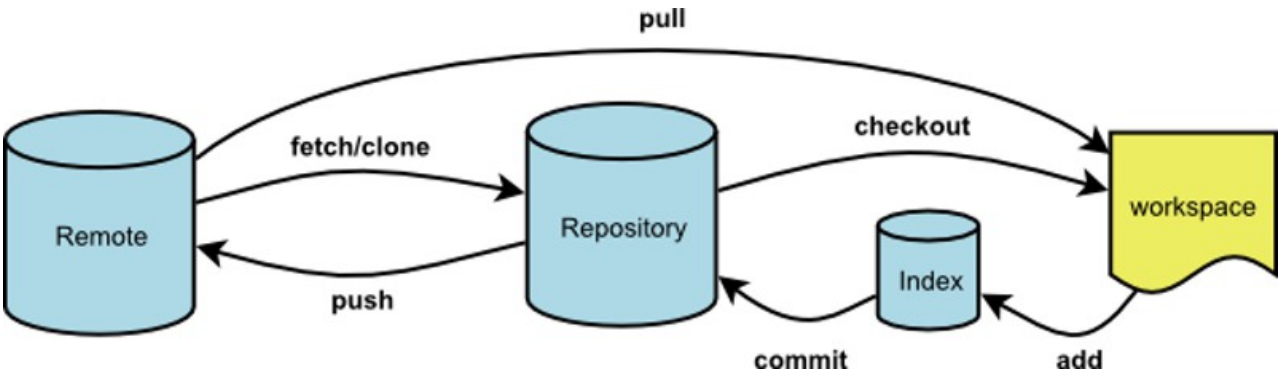
Git与Github之间的那些事儿

作者：*杨旭东*

时间：*8/12/2017 7:31:39 PM*

描述：本文介绍5个Git命令，它们的概念和用法。

- git clone
  - 从远程仓库克隆到本地
- git remote
  - 命令就用于管理主机名
- git fetch
  - 一旦远程主机的版本库有了更新（Git术语叫做commit），需要将这些更新取回本地
- git pull
  - 取回远程主机某个分支的更新，再与本地的指定分支合并。
- git push
  - 将本地分支的更新，推送到远程主机



1.git clone

```
//克隆远程的仓库到本地
$ git clone <远程版本库的网址> <本地的目录>
//
```

- 后面如果不自定义本地目录的名字，系统会默认设定当下目录为本地目录。
- Git协议比http(s)协议下载速度要快，推荐使用，ssh协议用于用户认证的场合。

2.git remote

```
//列出本地主机所关联的所有远程主机的名
$ git remote
//-v选项，可以参看远程主机的网址
$ git remote -v
//添加远程主机
$ git remote add <自定义的远程主机名> <需要设定的远程主机网址>
//删除远程主机
$ git remote rm <远程主机名>
//修改已经设定好的远程主机名
$ git remote rename <原主机名> <新的主机名>
```

- 补充： 绑定远程主机的前提是：

1. 首先将申请的**ssh公钥**注册到github上
    - 申请方法：ssh-keygen -t rsa -C "个人Email地址"
        - 申请后的ssh key会在C:\Users\yangxudong.ssh目录下出现 `id_rsa`和`id_rsa.pub`两个文件，`id_rsa.pub`是公钥，另一个是私钥。
2. 再进行本地主机与远程主机的绑定（否则就算绑定成功，本地文件也无法上传到远程远程库中）

3.git fetch

- 注意：`git fetch`命令通常用来查看其他人的进程，因为它取回的代码对你本地的开发代码没有影响。如图所示拉回来的文件会存储到本地仓库(Repository)，未和本机分支进行合并。如果要合并需要通过`marge`指令去惊醒合并。

```
//取回远程仓库指定分支上的更新（如果语句后面不指定分支名。默认会将某个远程库中更新全部拉回来！）
$ git fetch <远程主机名> <所需要拉下的分支名>
//查看远程分支
$ git branch -r
//查看所有分支
$ git branch -a
//也可以在拉回来的分支上再创建新的分支
$ git checkout -b newBrach <远程主机名/分支名>
//在本地的当前分支上进行合并通过（下面命令表示在当前分支上，合并origin/master）
$ git merge origin/master
# 或者
$ git rebase origin/master
```

- 补充1：fetch拉下来的更新信息在本地主机上要用"远程主机名/分支名"的形式读取。比如origin主机的master，就要用origin/master读取。

4.git pull

```
//取回远程主机某个分支的更新，并且与本地的指定分支合并。
1.$ git pull <远程主机名> <远程分支名>:<本地分支名>
#例如：
//取回origin主机的next分支，与本地的master分支合并，需要写成下面这样。
2.$ git pull origin next:master
//如果远程分支是与当前分支合并，则冒号后面的部分可以省略。
3.$ git pull origin next
#等于
//上面命令表示，取回origin/next分支，再与当前分支合并。实质上，这等同于先做git fetch，再做git merge。
//取下origin的next分支的跟新
$ git fetch origin next
//当前分支合并拉去下来的next
$ git merge origin/next
//在某些场合，Git会自动在本地分支与远程分支之间，建立一种追踪关系（tracking）。比如，在git clone的时候，所有本地分支默认与远程主机的同名分
4.$ git branch --set-upstream master origin/next
//如果当前分支与远程分支存在追踪关系，git pull就可以省略远程分支名。
5.$ git pull origin
//如果当前分支只有一个追踪分支，连远程主机名都可以省略。
6.$ git pull
//如果合并需要采用rebase模式，可以使用--rebase选项。
7.$ git pull --rebase <远程主机名> <远程分支名>:<本地分支名>
//如果远程主机删除了某个分支，默认情况下，git pull 不会在拉取远程分支的时候，删除对应的本地分支。这是为了防止，由于其他人操作了远程主机，导
8.$ git pull -p
# 等同于下面的命令
$ git fetch --prune origin
$ git fetch -p
```

- 补充：git pull默认的合并方式是merge方式，即git merge origin/master.如7指令所示使用--rebase即使用git rebase orgin/master方式。

- 例如：假设现在有两个分支 A B 1. 在B分支上执行 git merge A 后 A就被合到B上了 2. 在B分支上执行 git rebase A 后，效果与merge是一样的，但是 A就没有了，两个分支就合在一起了。 如图：

```
1. D---E test
2. /
3. A---B---C---F master
//merge合并：
```

```
4. D--------E
5. /        \
6.A---B---C---F----G   test, master
//rebase合并
7. A---B---D---E---C'---F'   test, master
```

- 二者的区别（merge 是合并的意思，rebase是复位基底的意思。）：
  - 可以看到，merge操作会生成一个新的节点，之前的提交分开显示。而rebase操作不会生成新的节点，是将两个分支融合成一个线性的提交。
  - merge合并：合并后的分支形成新的节点，原先的节点还存在。而rebase会吞没被合并的节点，并不会形成新的节点。

5.git push

```
//和pull类似。分支推送顺序的写法是<来源地>:<目的地>，所以git pull是<远程分支>:<本地分支>，而git push是<本地分支>:<远程分支>。
$ git push <远程主机名> <本地分支名>:<远程分支名>
//如果省略远程分支名，则表示将本地分支推送与之存在"追踪关系"的远程分支（通常两者同名），如果该远程分支不存在，则会被新建。
$ git push origin master
//如果省略本地分支名，则表示删除指定的远程分支，因为这等同于推送一个空的本地分支到远程分支。
$ git push origin :master
# 等同于
$ git push origin --delete master
//如果当前分支与远程分支之间存在追踪关系，则本地分支和远程分支都可以省略。
$ git push origin
//如果当前分支只有一个追踪分支，那么主机名都可以省略。
$ git push
//如果当前分支与多个主机存在追踪关系，则可以使用-u选项指定一个默认主机，这样后面就可以不加任何参数使用git push。
//下面命令将本地的master分支推送到origin主机，同时指定origin为默认主机，后面就可以不加任何参数使用git push了。
$ git push -u origin master
//不带任何参数的git push，默认只推送当前分支，这叫做simple方式。此外，还有一种matching方式，会推送所有有对应的远程分支的本地分支。Git 2.0版
$ git config --global push.default matching
# 或者
$ git config --global push.default simple
//还有一种情况，就是不管是否存在对应的远程分支，将本地的所有分支都推送到远程主机，这时需要使用--all选项。
//下面命令表示，将所有本地分支都推送到origin主机。
$ git push --all origin
//如果远程主机的版本比本地版本更新，推送时Git会报错，要求先在本地做git pull合并差异，然后再推送到远程主机。这时，如果你一定要推送，可以使用
//下面命令使用--force选项，结果导致远程主机上更新的版本被覆盖。除非你很确定要这样做，否则应该尽量避免使用--force选项。
$ git push --force origin
//最后，git push不会推送标签（tag），除非使用--tags选项。
$ git push origin --tags
```

——>完结