@author 毕胜华

@email net.bsh@gmail.com

## Git及GitHub简介

GIT：

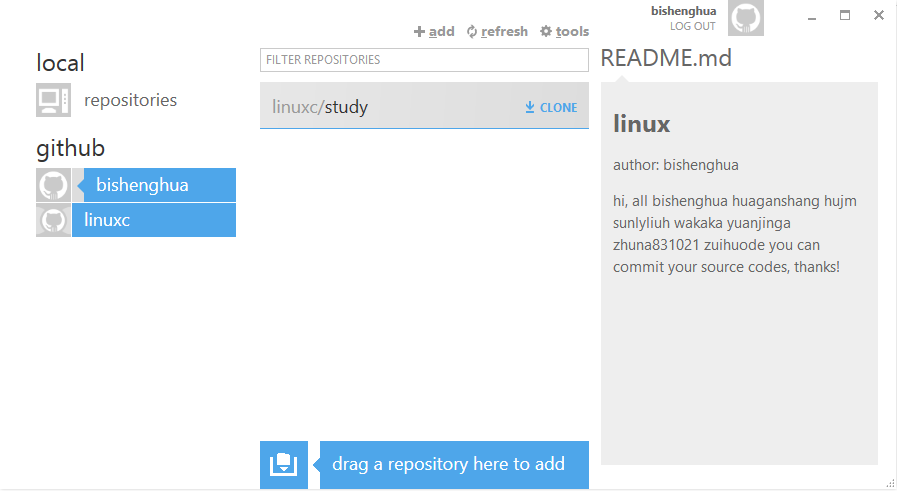
Git是一个分布式的版本控制系统，最初由Linus Torvalds编写，用作[Linux内核](http://baike.baidu.com/view/573460.htm" \t "_blank)代码的管理。在推出後，Git在其它项目中也取得了很大成功，尤其是在Ruby社区中。目前，包括[Rubinius](http://baike.baidu.com/view/1772201.htm" \t "_blank)和Merb在内的很多知名项目都使用了Git。Git同样可以被诸如[Capistrano](http://baike.baidu.com/view/7283319.htm" \t "_blank)和Vlad the Deployer这样的部署工具所使用。

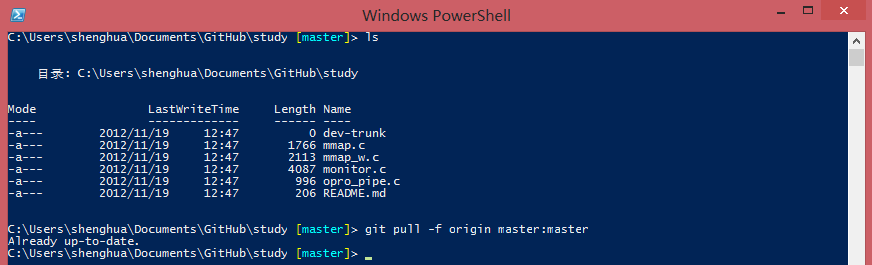
GitHub：

GitHub可以托管各种[git](http://baike.baidu.com/view/1531489.htm" \t "_blank)库， 并提供一个web界面，但与其它像 SourceForge或Google Code这样的服务不同，GitHub的独特卖点在于从另外一个项目进行分支的简易性。为一个项目贡献代码非常简单：首先点击项目站点的“fork”的按 钮，然后将代码检出并将修改加入到刚才分出的代码库中，最后通过内建的“pull request”机制向项目负责人申请代码合并。已经有人将GitHub称为代码玩家的MySpace。

GitHub For Window：

GitHub 使用 git 分布式版本控制系统，而 git 最初是 Linus Torvalds 为帮助 Linux 开发而创造的，它针对的是 Linux 平台，因此 git 和 Windows 从来不是最好的朋友，因为它一点也不像 Linux。现在，GitHub 发布了GitHub for Windows，为 Windows 平台开发者提供了一个易于使用的 Git 图形客户端。





## Git开发流程（例子）

$git clone <https://github.com/linuxc/study.git> #（克隆(获取)代码）

$git config --global --list #查看git配置

$git config --global user.email [net.bsh@gmail.com](mailto:net.bsh@gmail.com) #设置git配置

$git config --global user.name bishenghua #设置git配置

$git config --global core.editor vim #设置git默认编辑器

$git branch dev #基于当前所在的分支创建dev分支

$git branch trunk #基于当前所在的分支创建trunk分支

$git checkout dev #切换到dev分支

$touch test.txt #创建test.txt文件

$git add test.txt #添加文件

$git commit -m 'test' #提交到dev分支

//$git push -u origin dev:master #将本地分支local推到远程master上（一般不会这么做）

#下面准备合并分支

$git checkout trunk #先切换到本地trunk上（一般不在本地master上）

$git pull -f origin master: trunk #从服务器的master拉取代码到本地的trunk分支（如果有冲突则需要解决）

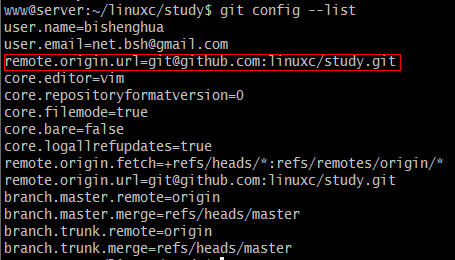
$git merge dev #将dev合并到trunk

$git push -u origin trunk:master #将本地分支trunk推到远程master上

## 避免每次push都输入用户名密码

1. 查看配置
2. $git config --list
3. 如果remote.origin.url为<https://github.com/linuxc/study.git>，那么无论如何设置都不走ssh-key，所以需要修改这个配置项为：

git config --global remote.origin.url git@github.com:linuxc/study.git



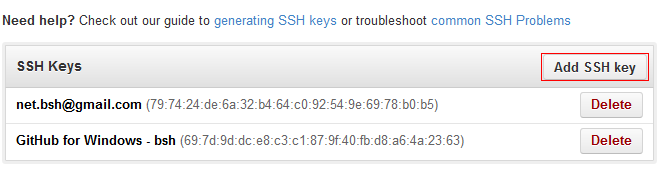
1. 生成ssh-key
2. 生成密钥：

$ssh-keygen -t rsa -C [net.bsh@gmail.com](mailto:net.bsh@gmail.com)

连续敲三次回车

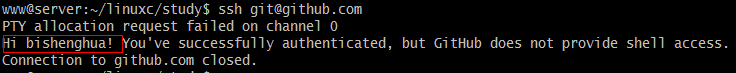
1. $vim ~/ .ssh/id\_rsa.pub

将里面的内容添加到服务器上：<https://github.com/settings/ssh>



1. 验证：

$ssh [git@github.com](mailto:git@github.com)



## Git常用命令

config:

开发人员需要为git仓库配置相关信息，这样在提交代码时，这些信息会自动

反映在git仓库的日志中。

git config user.name "your name"

git config user.email yourname@email\_server

git config core.editor vim

git config core.paper "less -N"

git config color.diff true

git config alias.co checkout

git config alias表示，可以用git co代表git checkout。git var -l可以查看已经设置的配置。

diff:

开发人员在本地进行开发后，可以使用git diff查看改动。

除了直接比较当前开发后的改动外，git diff还可以：

git diff tag 比较tag和HEAD之间的不同。

git diff tag file 比较一个文件在两者之间的不同。

git diff tag1 tag2 比较两个tag之间的不同。

git diff SHA11 SHA12 比较两个提交之间的不同。

git diff tag1 tag2 file or

git diff tag1:file tag2:file 比较一个文件在两个tag之间的不同。

ORIG\_HEAD用于指向前一个操作状态，因此在git pull之后如果想得到pull的内容就可以：

git diff ORIG\_HEAD

git diff --stat 用于生成统计信息。

git diff --stat ORIG\_HEAD

apply:

git apply 相当于patch命令。--check 检查能否正常打上补丁，-v verbose模式， -R reverse模式，反打补丁。

log:

git log file 查看一个文件的改动。

git log -p 查看日志和改动。

git log tag1 tag2 查看两个tag之间的日志。

git log -p tag1 tag2 file 查看一个文件在两个tag之间的不同。

git log tag 查看tag和HEAD之间的不同。

commit:

git commit -a -e 提交全部修改文件，并调用vim编辑提交日志。

git reset HEAD^ or

git reset HEAD~1 撤销最后一次提交。

git reset --hard HEAD^ 撤销最后一次提交并清除本地修改。

git reset SHA1 回到SHA1对应的提交状态。

add/delete/ls:

git add -a 添加所有文件。除了.gitignore文件中的文件。

git rm file 从git仓库中删除文件。

git commit 添加或是删除后要提交。

git ls-files -m 显示修改过的文件。

git ls-files 显示所有仓库中的文件。

git中有四种对象：blob、tree、commit、tag。blob代表文件，tree代表目录，commit代表提交历史，tag代表标签。这四种对象都是由SHA1值表示的。在仓库的.git目录中保存了git管理仓库所需要的全部信息。

git ls-tree HEAD file 显示file在HEAD中的SHA1值。

git cat-file -t SHA1 显示一个SHA1的类型。

git cat-file type SHA1 显示一个SHA1的内容。type是blob、tree、commit、tag之一。

patch:

git format-patch -1 生成最后一个提交对应的patch文件。

git am < patch 把一个patch文件加入git仓库中。

git am --resolved 如果有冲突，在解决冲突后执行。

git am --skip 放弃当前git am所引入的patch。

conflict:

git merge 用于合并两个分支。

git diff 如果有冲突，直接使用diff查看，

冲突代码用<<<和>>>表示。手动修改冲突代码。

git update-index 更新修改后的文件状态。

git commit -a -e 提交为解决冲突而修改的代码。

branch:

git branch -a 查看所有分支。

git branch new\_branch 创建新的分支。

git branch -d branch 删除分支。

git checkout branch 切换当前分支。-f参数可以覆盖未提交内容。

daemon:

有时更新公共代码仓库使用patch的方式，或者直接用git pull git://ip/repo branch的方式更新每个人的代码。使用git pull的方式需要提交代码的机器运行：

git daemon --verbose --export-all --enable=receive-pack --base-path=/repo

request-pull:

git request-pull start url 用于产生本次pull请求的统计信息。

clean:

git clean -dxf 用于清除未跟踪文件。

git clean -dnf 可以显示需要删除的文件，但不包括被.gitignore忽略的。

git reset --hard HEAD 用于清除跟踪文件的修改。