**git安装配置及基本使用**

**安装git**

1）windows

安装msysgit，下载地址：[http://msysgit.github.io/](http://msysgit.github.io/" \t "_blank)

安装的时候，基本选择默认设置，但是：

在Adjusting your PATH environment页，勾选Run Git from the Windows Command Prompt

2）ubuntu

用命令“git --version”查看是否已安装，且版本为1.9.5或更高。若没安装或版本太低：

$ sudo apt-get install git-core git-gui git-doc gitk

3）mac

<http://sourceforge.net/projects/git-osx-installer/>，不仅能装Git本身（选1.9.5或以上版本），还有GUI的安装包

**启动git**

1）windows

Windows：使用Windows自带的命令行界面

可以在Windows自己的命令行界面下可以直接运行Git命令行，比如

D:/test> git help

当命令中有些特殊参数的时候，要加上双引号。比如

D:/test> git log HEAD^

特殊符号^会被Windows误解，所以要加双引号，写成

D:/test> git log "HEAD^"

Windows：使用msysGit自带的Bash

使用Bash就不用像上面那样加双引号了。启动Git Bash的简便方法是，在Windows Explorer里，适当目录的右键弹出菜单，Git Bash。此外，也可以从Windows开始菜单进入。

初次使用时，点击界面右上角，在菜单中选择“属性”项，在弹出对话框中，勾选上“快速编辑模式”和“插入模式”，这样将来copy paste比较方便。

注意，有利有弊，这个Bash对中文的支持不太好。

2）linux

$ git help

**设置git**

不论Windows还是Linux还是Mac，建议至少config下述内容

1. git config --global user.name "test" # 请换成你自己的名字，除非你凑巧也叫wukong.sun
2. git config --global user.email "test@163.com" # 同上
3. git config --global push.default simple # 要是你非要用低版本的Git（比如1.7.x），好吧，那就不设simple设current，否则你的Git不支持
4. git config --global core.autocrlf false # 让Git不要管Windows/Unix换行符转换的事
5. git config --global gui.encoding utf-8 # 避免git gui中的中文乱码
6. git config --global core.quotepath off # 避免git status显示的中文文件名乱码

其中最后两个配置是关于中文乱码的，基本够用了。

Windows上还需要配置：

git config --global core.ignorecase false

**设置SSH**

在Linux的命令行下，或Windos上Git Bash命令行窗口中（总之不要用iOS），键入：

$ssh-keygen -t rsa -C "test@163.com"

然后一路回车，不要输入任何密码之类，生成ssh key pair。然后就生成一个目录.ssh ，里面有两个文件：id\_rsa , id\_rsa.pub

如果在Linux上，需要把其中的私钥告诉本地系统：

$ ssh-add ~/.ssh/id\_rsa

再把其中公钥的内容复制到github上。具体方法是：

显示ssh公钥的内容：

$ cat ~/.ssh/id\_rsa.pub

打开github页面：[https://github.com/settings/profile](https://github.com/settings/profile" \t "_blank)，选择SSH Keys，然后点击Add SSH Key，把刚才ssh公钥id\_rsa.pub（windows下的用户目录找到.ssh文件夹进去就可以看到）的内容paste进去。不需要填title，title会自动生成。

注意：需要copy最开头的“ssh-rsa ”这几个字。

**开始使用**

1）创建新的git仓库

1. $ mkdir git\_repo
2. $ cd git\_repo
3. $ git init
4. $ echo "test" > README.mkd
5. $ git add README.mkd
6. $ git commit -m "add README.mkd file"
7. $ git remote add origin git@github.com:username/test.git
8. $ git push -u origin master

2）使用已存在的git仓库

1. $ cd git\_repo
2. $ git remote add origin git@github.com:username/test.git
3. $ git push -u origin master

注意，如果提示fatal: remote origin already exists.，那么说明该本地仓库已经有远端地址了。你可以先使用git remote rm origin删除origin，或者使用git remote add other\_name git@github.com:username/test.git来添加（提交时记得使用git push -u other\_name master）。

3）一次提交到多个远端仓库

假设现有仓库地址为： git@github.com:username/test.git

1. $ git clone git@github.com/username/test.git
2. $ cd test
3. $ vim .git/config
4. [core]
5. repositoryformatversion = 0
6. filemode = true
7. bare = false
8. logallrefupdates = true
9. [remote "origin"]
10. url = git@github.com:username/test.git
11. url = git@gitshell.com:username/test.git
12. url = git@bitbucket.org:username/test.git
13. fetch = +refs/heads/\*:refs/remotes/origin/\*
14. [branch "master"]
15. remote = origin
16. merge = refs/heads/master

然后第一次提交时需要执行git push -u origin master，再往后就只需要执行git push就能把修改提交到上述三个远端仓库了。

注意：在 Git 2.0 将会更改默认的push动作为【只 push 当前 branch 到远端仓库】。如果想继续使用git push both命令需要手动设置一下git push的默认动作git config --global push.default matching。

push.default有几个简单动作，这里介绍matching和simple，二者意思分别是 push 本地所有的分支到远端仓库和 push 本地当前分支到上游分支。这个解释貌似还不够精确，可以man git-config来查看详细说明。

4）在现有仓库上创建孤儿分支

孤儿分支意思为该分支中没有任何内容，与之前创建的其他分支没有任何关联。

1. $ git clone git@github.com:username/test.git
2. $ cd test
3. $ git checkout --orphan new\_branch
4. Switched to a new branch 'new\_branch'
5. $ git rm -rf . *# 删除旧工作目录树中所有文件*
6. $ rm .gitignore *# 如果有该文件的话就删除*
7. $ echo "orphan branch" > README.mkd
8. $ git add .
9. $ git commit -m "add README.mkd file"
10. $ git push origin new\_branch

5）提交单个分支到远端git仓库

git push命令默认是将所有分支(branch)都提交到git仓库，有时你只想提交某个分支到远端仓库，那么就就需要使用git push origin HEAD。当然也可以使用git config --global push.default tracking命令来改变git push的默认操作，意思是执行git push时默认只提交当前分支到远端git仓库。

**git常用指令**

以下几个是git常用的指令，可以简单了解一下。

1）git config

在使用git前最好先配置一下你的个人信息及使用偏好。以下命令的意思就不用解释了吧，执行完以下命令就会在你的家目录（~）下生成一个文件~/.gitconfig。

1. $ git config --global user.name "username"
2. $ git config --global user.email test@163.com
3. $ git config --global core.editor vim
4. $ git config --global merge.tool vimdiff
5. $ git config --global color.status auto
6. $ git config --global color.branch auto
7. $ git config --global color.interactive auto
8. $ git config --global color.diff auto
9. $ git config --global push.default simple
10. $ git config --global alias.co checkout
11. $ git config --global alias.ci commit
12. $ git config --global alias.st status
13. $ git config --global alias.last *'log -1 HEAD'*
14. $ git config --global alias.unstage *'reset HEAD --'*

2）git add

添加文件内容到索引中去（暂存文件），几个简单示例：

1. $ git add .
2. $ git add --all
3. $ git add \*.txt
4. $ git add directory*/\*.sh*

突然你又不想git add了，那么执行以下命令：

1. $ git reset .
2. $ git reset \*.txt
3. $ git reset directory*/\*.sh*

3）git rm

删除索引和当时工作目录中的文件。

1. $ git rm filename
2. $ git rm -f \*.txt
3. $ git rm -r .

4）git commit

将当前改动记录到仓库中，即提交改动到本地仓库中。

$ git commit -m "add a file and remove a file"

突然你又不想git commit了，那么执行以下命令:

$ git reset HEAD^

你commit之后发现少添加了一个文件：

1. $ git commit -m'msg'
2. $ git add forget\_file
3. $ git commit *--amend*

你的 commit 已经 push 到远程分支(master)了，现在你想反悔了：

1. $ git clone git@github.com:username/test.git
2. $ cd test
3. $ git reset HEAD^
4. $ git push -f master

5）git status

查看当前工作目录的状态，即修改、添加及删除了哪些文件。

$ git status

6）git checkout

检出一个分支和目录到当前工作目录中，可以简单理解为切换分支的命令。

以下命令分别为切换到分支 branch1 和创建一个新的分支 new\_branch 。

1. $ git checkout branch1
2. $ git checkout -b new\_branch

取消本地改动：

$ git checkout *-- file\_name*

7）git branch

列出、创建和删除分支。

以下指令分别为列出本地分支、所有分支、远端分支、创建、删除、强制删除分支。

1. $ git branch --list
2. $ git branch --all
3. $ git branch --remotes
4. $ git branch new\_branch
5. $ git branch --delete branch\_name
6. $ git branch -D branch\_name

删除remote tracking branch，就是git branch -r命令列出的分支。

1. $ git branch -r
2. $ git branch -d -r origin/develop

8）合并分支

如果出现冲突，那么手动解决冲突就可以了。

1. $ git checkout branch\_name
2. $ git checkout master
3. $ git merge branch\_name

9）删除远程分支

合并分支之后如果不再需要以前的分支了，那么可以在本地及远程删除它。

1. $ git branch -d branch\_name
2. $ git branch -D branch\_name
3. $ git push origin :branch\_name

这条命令耐人寻味啊，其中origin是你的远程仓库名字（git remote -v可以查看到）。

10）git diff

查看改动内容。

1. $ git diff filename
2. $ git diff .
3. $ git diff revision1 revision2
4. $ git diff branch1 branch2

DIFF暂存（添加到索引中）的文件：

1. $ git add .
2. $ git diff --cached

View the redundant Tab or Space in your codes:

1. $ git diff *--check*
2. $ git diff *--check --cached*

# 无法推送

1. 撤销所有修改
2. 拉取
3. 同步
4. 进到此文件夹，打开githash
5. git remote rm origin
6. git remote add origin [git@github.com](mailto:git@github.com):你的用户名/mymusicstore.git
7. git push --set-upstream origin master
8. 去创建新页面
9. 提交
10. 推送