**Git学习总结**

时间：2017/07/03 参考博客：http://www.liaoxuefeng.com/

## 1、前言：

Linus在1991年创建了开源的Linux，从此，Linux系统不断发展，已经成为最大的服务器系统软件了。到了2002年，Linux系统已经发展了十年了，代码库之大让Linus很难继续通过手工方式管理了，于是Linus选择了一个商业的版本控制系统BitKeeper，BitKeeper的东家BitMover公司出于人道主义精神，授权Linux社区免费使用这个版本控制系统。安定团结的大好局面在2005年就被打破了，原因是Linux社区牛人聚集，不免沾染了一些梁山好汉的江湖习气。开发Samba的Andrew试图破解BitKeeper的协议（这么干的其实也不只他一个），被BitMover公司发现了（监控工作做得不错！），于是BitMover公司怒了，要收回Linux社区的免费使用权。

Linus花了两周时间自己用C写了一个分布式版本控制系统，这就是Git！一个月之内，Linux系统的源码已经由Git管理了！

## 2、版本控制系统分类

版本控制系统分为分布式版本控制系统和集中式版本控制系统。

集中式：代表：CVS、SVN；版本库是集中存放在中央服务器的，而干活的时候，用的都是自己的电脑，所以要先从中央服务器取得最新的版本，然后开始干活，干完活了，再把自己的活推送给中央服务器。

缺点：速度慢，必须联网才可以使用！

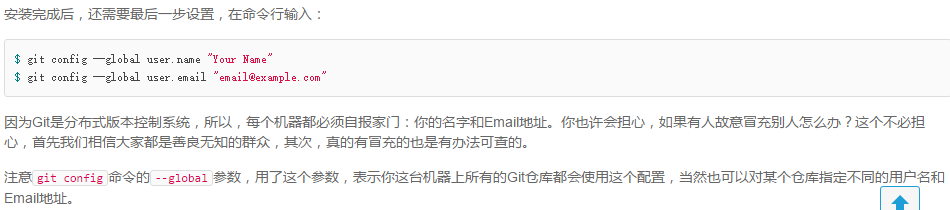
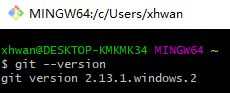
分布式：Git；，分布式版本控制系统根本没有“中央服务器”，每个人的电脑上都是一个完整的版本库，这样，你工作的时候，就不需要联网了，因为版本库就在你自己的电脑上。既然每个人电脑上都有一个完整的版本库，那多个人如何协作呢？比方说你在自己电脑上改了文件A，你的同事也在他的电脑上改了文件A，这时，你们俩之间只需把各自的修改推送给对方，就可以互相看到对方的修改了。

优点：安全性更高

## 3、git安装

下载地址：<https://git-for-windows.github.io/> 下载最新版本，安装即可

安装完成后，在开始菜单里找到“Git”->“Git Bash”，蹦出一个类似命令行窗口的东西，就说明Git安装成功！

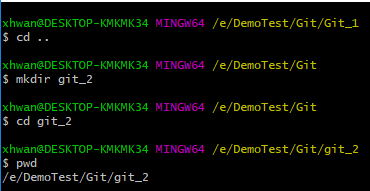


## 4、创建版本库

创建一个版本库非常简单，首先，选择一个合适的地方，创建一个空目录

第二步，通过git init命令把这个目录变成Git可以管理的仓库：

如果你没有看到.git目录，那是因为这个目录默认是隐藏的，用ls -ah命令就可以看见。



把文件添加到版本库：

所有的版本控制系统，其实只能跟踪文本文件的改动，比如TXT文件，网页，所有的程序代码等等，Git也不例外。版本控制系统可以告诉你每次的改动，比如在第5行加了一个单词“Linux”，在第8行删了一个单词“Windows”。而图片、视频这些二进制文件，虽然也能由版本控制系统管理，但没法跟踪文件的变化，只能把二进制文件每次改动串起来，也就是只知道图片从100KB改成了120KB，但到底改了啥，版本控制系统不知道，

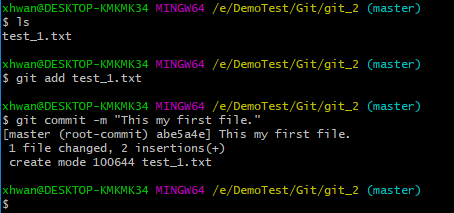
记得把Notepad++的默认编码设置为UTF-8 without BOM即可：在“格式”里面

总结：

初始化一个Git仓库，使用git init命令。

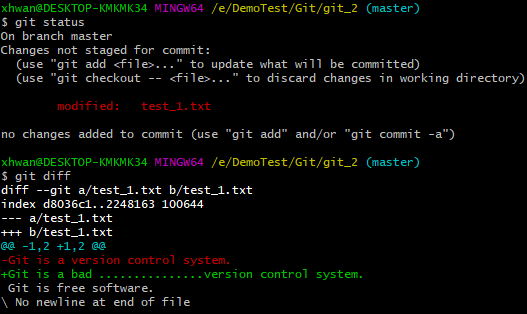
添加文件到Git仓库，分两步：

* 第一步，使用命令git add <file>，注意，可反复多次使用，添加多个文件；
* 第二步，使用命令git commit，完成。



## 5、时光穿梭机

### a)提交版本，并查看修改状态。



git status: 命令可以让我们时刻掌握仓库当前的状态，上面的命令告诉我们，readme.txt被修改过了，但还没有准备提交的修改。如果想看具体修改了那些内容，需要用到 git diff

### b)版本回退

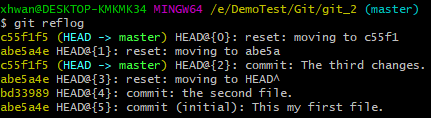
xhwan@DESKTOP-KMKMK34 MINGW64 /e/DemoTest/Git/git\_2 (master)

$ git log -----------查看日志

$ git log --pretty=oneline -----------查看日志

$ git reset --hard HEAD^ -----------回退代码

$ git reset --hard abe5a



abe5a4e9919f24813283bcf33baae5058e4e2829 (HEAD -> master) This my first file.

### c)工作区和暂缓区

工作区; 就是你在电脑里能看到的目录，比如我的learngit文件夹就是一个工作区

版本库：工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。

注意：git管理的是修改，不是文件！

### d)撤销修改

命令git checkout -- readme.txt意思就是，把readme.txt文件在工作区的修改全部撤销，这里有两种情况：

一种是readme.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；

一种是readme.txt已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。

总之，就是让这个文件回到最近一次git commit或git add时的状态。

$ git add test\_2.txt ---------增加文件

$ git commit -m "add the second file." ---------提交文件

$ ls ---------查看文件数目

$ rm test\_2.txt ---------删除文件

$ git checkout -- test\_2.txt ---------恢复文件

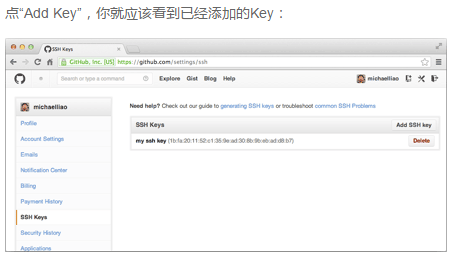
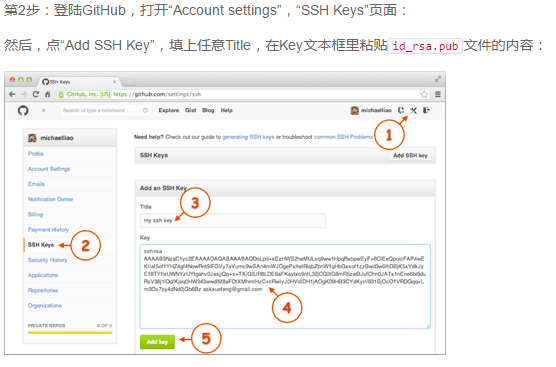
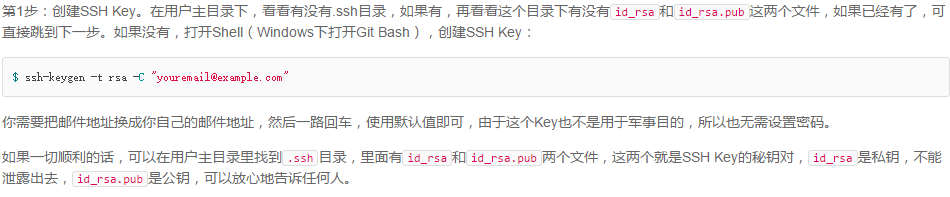
另一种恢复文件方式是：

$ git log --pretty=oneline ---------查询提交的id

$ git reset --hard 07e4 ---------恢复文件到指定id

## 6、远程仓库

为什么GitHub需要SSH Key呢？因为GitHub需要识别出你推送的提交确实是你推送的，而不是别人冒充的，而Git支持SSH协议，所以，GitHub只要知道了你的公钥，就可以确认只有你自己才能推送。



## 7、远程仓库各种报错

a)说没有权限远程仓库，原因是IP地址不对

$ ssh -T [git@github.com](mailto:git@github.com) ------- 看是否可以连上github库

$ cd C:/Users/xhwan/.ssh ------- 检查计算机SSH秘钥

$ ls ------- 查看文件目录

$ mkdir key\_backup ------- 创建备份文件夹

$ cp id\_rsa\* keyey\_backup/ ------- copy进行备份

报错：Hi JamesWang2017! You've successfully authenticated, but GitHub does not provide shell access.

解决方案：

地址1：http://www.cnblogs.com/fnng/archive/2011/08/25/2153807.html

xhwan@DESKTOP-KMKMK34 MINGW64 ~/.ssh ------- 用户名

$ git config --global github.user JamesWang2017

xhwan@DESKTOP-KMKMK34 MINGW64 ~/.ssh ------- token信息

$ git config --global github.token 18b83af4055a39f7878af245023ae320b77fd5bc

xhwan@DESKTOP-KMKMK34 MINGW64 /e/DemoTest/Git/Git\_1 ------- 从远程库copy信息

$ git clone git@github.com:JamesWang2017/learningit.git

地址2：https://github.com/blog/1509-personal-api-tokens

设置token地址：

You can now [create your own personal API tokens](https://github.com/settings/tokens) for use in scripts and on the command line. Be careful, these tokens are like passwords so you should guard them carefully. The advantage to using a token over putting your password into a script is that a token can be revoked, and you can generate lots of them. [Head on over to your settings](https://github.com/settings/tokens) to manage personal API tokens.

https://github.com/settings/tokens

## 8、分支管理

在[版本回退](http://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/0013744142037508cf42e51debf49668810645e02887691000)里，你已经知道，每次提交，Git都把它们串成一条时间线，这条时间线就是一个分支。截止到目前，只有一条时间线，在Git里，这个分支叫主分支，即master分支。HEAD严格来说不是指向提交，而是指向master，master才是指向提交的，所以，HEAD指向的就是当前分支。

$ git checkout -b dev 相当于git branch dev git checkout dev

git checkout命令加上-b参数表示创建并切换，相当于以下两条命令



$ git checkout master -----合并分支

$ git branch -----查看分支

$ git branch -d dev -----删除分支

查看分支合并图：

$ git log --graph --pretty=oneline --abbrev-commit

\* 30f453e (HEAD -> master) This is to test merge.

\* 07e4daa add the second file.

\* c55f1f5 The third changes.

\* abe5a4e This my first file.

合并有区别：

合并分支时，加上--no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并。

Bug分支：

修复bug时，我们会通过创建新的bug分支进行修复，然后合并，最后删除；

当手头工作没有完成时，先把工作现场git stash一下，然后去修复bug，修复后，再git stash pop，回到工作现场。

## 9、打标签

$ git tag v1.0 -----打标签

$ git tag -----查询标签

$ git log --pretty=oneline --abbrev-commit -----查询提交日志

git show v1.0 -----展示登记记录

命令git tag <name>用于新建一个标签，默认为HEAD，也可以指定一个commit id；

git tag -a <tagname> -m "blablabla..."可以指定标签信息；

git tag -s <tagname> -m "blablabla..."可以用PGP签名标签；

命令git tag可以查看所有标签。

命令git push origin <tagname>可以推送一个本地标签；

命令git push origin --tags可以推送全部未推送过的本地标签；

命令git tag -d <tagname>可以删除一个本地标签；

命令git push origin :refs/tags/<tagname>可以删除一个远程标签。