Git:linux下的版本控制系统

Windows下的是msysgit和Cygwin

windows创建仓库步骤：可参照网站：

<http://artori.us/git-github-usage/> 安装好后，重点参照这篇文档

http://www.cnblogs.com/yjzhu/archive/2014/07/21/3858188.html

1. 注册账号：<https://github.com/>，这里我创建了一个免费的账号lengyuyan,.密码：lengyuyan123

在网站上创建仓库,这里我创建了一个创库： g\_testcode

1. 安装客户端msysgit

MsysGit(Git for Windows) 3264位下载 v2.6.3官方版 .rar

github 是服务端，要想在自己电脑上使用 git 还需要一个 git 客户端，这里选用 msysgit，这个只是提供了 git 的核心功能，而且是基于命令行的。如果想要图形界面的话只要在 msysgit 的基础上安装 TortoiseGit 即可。

装完 msysgit 后右键鼠标会多出一些选项来，然后我们在本地新建个文件夹（比如叫 github），右键选择 Git Init Here，这样 github 文件夹内会多出来一个 .git 文件夹，这就表示本地 git 创建成功。右键 Git Bash 进入 git 命令行就可以把刚刚新建的仓库克隆到本地，当然我们还需要配置下 ssh key

1. 配置git:

首先在本地创建 ssh key：

ssh-keygen -t rsa -C "your\_email@youremail.com"

后面的 your\_email@youremail.com 改为自己的邮箱，之后会要求确认路径和输入密码，这里使用默认的一路回车就行。成功的话会在 ~/ 下生成 .ssh 文件夹，打开 id\_rsa.pub，复制里面的 key，回到 github，进入 settings，左边选择 SSH keys，Add SSH Key，title 随便填，粘贴 key。为了验证是否成功，在 git bash 下输入：

ssh -T git@github.com

如果是第一次的会提示是否 continue，输入 yes 就会看到：You've successfully authenticated, but GitHub does not provide shell access，这就表示已成功连上 github。

接下来我们要做的就是把 github 上面建立的仓库克隆到本地，在此之前还需要设置 username 和 email，因为 github 每次 commit 都会记录他们。

git config --global user.name "your name"

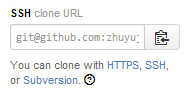
git config --global user.email "your\_email@youremail.com"

克隆到本地（比如克隆 css 的项目）：

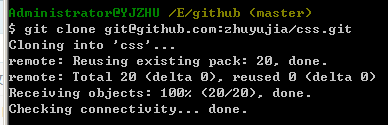
git clone [git@github.com:zhuyujia/css.git](mailto:git@github.com:zhuyujia/css.git)

这种格式的地址有可能会有问题（我这里是公司限制了吗），可以用hhtps

需要注意的是：github 提供了 3 种 url 路径（HTTPS，SSH，Subversion），一般如果账号处于登录状态，那么我们可以用 SSH，就像上面的代码，如果没有登录的话，只能用 HTTPS 的 url 了，如图所示：



克隆成功，如下所示：



1. 修改，提交，上传

我们可以修改克隆到本地的项目，修改完成后先要 add 修改的文件（. 表示全部），然后填写 commit，最后在 push 到 github。

git add .

git commit -m 'update'

git push

按照上面的例子不能连接到我的github服务器，不知道是不是公司限制了；

例子：

1,上传

在家目录：

   使用git在本地创建一个项目的过程

    $ makdir ~/ g\_test    //创建一个项目hello-world  
    $ cd ~/ g\_test       //打开这个项目  
    $ git init             //初始化   
    $ touch README  
    $ git add README        //更新README文件  
    $ git commit -m 'first commit'     //提交更新，并注释信息“first commit”  
    $ git remote add origin [https:@github.com:lengyuyan/g\_test.git](mailto:git@github.com:defnngj/hello-world.git)     //连接远程github项目    
    $ git push origin master     //将本地项目更新到github项目上去

git init 产生的目录解释

error: src refspec master does not match any.

引起该错误的原因是，目录中没有文件，空目录是不能提交上去的

error: insufficient permission for adding an object to repository database ./objects  
服务端没有可写目录的权限

错误提示：fatal: remote origin already exists.

解决办法：$ git remote rm origin

错误提示：error: failed to push som refs to ........

解决办法：$ git pull origin master //先pull 下来 再push 上去

git init //在当前项目工程下履行这个号令相当于把当前项目git化，变身！

git add .//把当前目次下代码参加git的跟踪中，意思就是交给git经管，提交到本地库

git add <file> //把当前文件参加的git的跟踪中，交给git经管，提交到本地库

git commit -m “…”//相当于写点提交信息

git remote add origin git＠github.com:ellocc/gittest.git //这个相当于指定本地库与github上的哪个项目相连

git push -u origin master //将本地库提交到github上。

git clone git＠github.com:ellocc/gittest.git  //将github上的项目down下来。

git fetch origin //取得长途更新，这里可以看做是筹办要取了

git merge origin/master //把更新的内容归并到本地分支/master

下面是删除文件后的提交

git status //可以看到我们删除的哪些文件

git add .   //删除之后的文件提交git经管。

git rm a.c //删除文件

git rm -r gittest //删除目次

git reset --hard HEAD 回滚到add之前的状态

git diff比较的是跟踪列表中的文件和文件系统中文件的差别

注意：要用https形式的url ， git形式的没用，且f:/pools/linux-stable要是不存在的文件夹；

可以先进入git.kernel.org，找到git地址；

git clone https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git f:/pools/linux-stable

如果输入$ git remote add origin [git@github.com:djqiang（github帐号名）/gitdemo（项目名）.git](mailto:git@github.com:djqiang/gitdemo.git)

    提示出错信息：fatal: remote origin already exists.

    解决办法如下：

    1、先输入$ git remote rm origin

    2、再输入$ git remote add origin [git@github.com:djqiang/gitdemo.git](mailto:git@github.com:djqiang/gitdemo.git) 就不会报错了！

    3、如果输入$ git remote rm origin 还是报错的话，error: Could not remove config section 'remote.origin'. 我们需要修改gitconfig文件的内容

    4、找到你的github的安装路径，我的是C:\Users\ASUS\AppData\Local\GitHub\PortableGit\_ca477551eeb4aea0e4ae9fcd3358bd96720bb5c8\etc

    5、找到一个名为gitconfig的文件，打开它把里面的[remote "origin"]那一行删掉就好了！

    如果输入$ ssh -T [git@github.com](mailto:git@github.com)  
    出现错误提示：Permission denied (publickey).因为新生成的key不能加入ssh就会导致连接不上github。

    解决办法如下：

    1、先输入$ ssh-agent，再输入$ ssh-add ~/.ssh/id\_key，这样就可以了。

    2、如果还是不行的话，输入ssh-add ~/.ssh/id\_key 命令后出现报错Could not open a connection to your authentication agent.解决方法是key用Git Gui的ssh工具生成，这样生成的时候key就直接保存在ssh中了，不需要再ssh-add命令加入了，其它的user，token等配置都用命令行来做。

    3、最好检查一下在你复制id\_rsa.pub文件的内容时有没有产生多余的空格或空行，有些编辑器会帮你添加这些的。

    如果输入$ git push origin master

    提示出错信息：error:failed to push som refs to .......

    解决办法如下：

    1、先输入$ git pull origin master //先把远程服务器github上面的文件拉下来

    2、再输入$ git push origin master

    3、如果出现报错 fatal: Couldn't find remote ref master或者fatal: 'origin' does not appear to be a git repository以及fatal: Could not read from remote repository.

    4、则需要重新输入$ git remote add origin[git@github.com:djqiang/gitdemo.git](mailto:git@github.com:djqiang/gitdemo.git)

    使用git在本地创建一个项目的过程

    $ makdir ~/hello-world    //创建一个项目hello-world  
    $ cd ~/hello-world       //打开这个项目  
    $ git init             //初始化   
    $ touch README  
    $ git add README        //更新README文件  
    $ git commit -m 'first commit'     //提交更新，并注释信息“first commit”  
    $ git remote add origin [git@github.com:defnngj/hello-world.git](mailto:git@github.com:defnngj/hello-world.git)     //连接远程github项目    
    $ git push -u origin master     //将本地项目更新到github项目上去

    gitconfig配置文件

         Git有一个工具被称为git config，它允许你获得和设置配置变量；这些变量可以控制Git的外观和操作的各个方面。这些变量可以被存储在三个不同的位置：   
         1./etc/gitconfig 文件：包含了适用于系统所有用户和所有库的值。如果你传递参数选项’--system’ 给 git config，它将明确的读和写这个文件。   
         2.~/.gitconfig 文件 ：具体到你的用户。你可以通过传递--global 选项使Git 读或写这个特定的文件。  
         3.位于git目录的config文件 (也就是 .git/config) ：无论你当前在用的库是什么，特定指向该单一的库。每个级别重写前一个级别的值。因此，在.git/config中的值覆盖了在/etc/gitconfig中的同一个值。  
        在Windows系统中，Git在$HOME目录中查找.gitconfig文件（对大多数人来说，位于C:\Documents and Settings\$USER下）。它也会查找/etc/gitconfig，尽管它是相对于Msys 根目录的。这可能是你在Windows中运行安装程序时决定安装Git的任何地方。

        配置相关信息：

        2.1　当你安装Git后首先要做的事情是设置你的用户名称和e-mail地址。这是非常重要的，因为每次Git提交都会使用该信息。它被永远的嵌入到了你的提交中：

　　$ git config --global user.name "John Doe"

　　$ git config --global user.email [johndoe@example.com](mailto:johndoe@example.com)

       2.2    你的编辑器(Your Editor)

　　现在，你的标识已经设置，你可以配置你的缺省文本编辑器，Git在需要你输入一些消息时会使用该文本编辑器。缺省情况下，Git使用你的系统的缺省编辑器，这通常可能是vi 或者 vim。如果你想使用一个不同的文本编辑器，例如Emacs，你可以做如下操作：

　　$ git config --global core.editor emacs

      2.3 检查你的设置(Checking Your Settings)

　　如果你想检查你的设置，你可以使用 git config --list 命令来列出Git可以在该处找到的所有的设置:

　　$ git config --list

      你也可以查看Git认为的一个特定的关键字目前的值，使用如下命令 git config {key}:

　　$ git config user.name

      2.4 获取帮助(Getting help)

　　如果当你在使用Git时需要帮助，有三种方法可以获得任何git命令的手册页(manpage)帮助信息:

　　$ git help <verb>

　　$ git <verb> --help

　　$ man git-<verb>

　　例如，你可以运行如下命令获取对config命令的手册页帮助:

　　$ git help config