Git 使用步骤

1. git 本地服务器安装

- Git（Git 主程序）http://git-scm.com/
- TortoiseGit

2. git 配置

2.1 SSH Key 配置

2.1.1 在 Git Bash 命令行下生成

鼠标右键 -> Git Bash

```
ssh-keygen -t rsa -C "uasername@139.com"
```

生成后的公钥会存放在

C:/Users/You_User_Name/.ssh/id_rsa.pub

> 注：将 id_rsa.pub 拷贝到 github 或者其他服务器，就可以使用ssh 登录了。

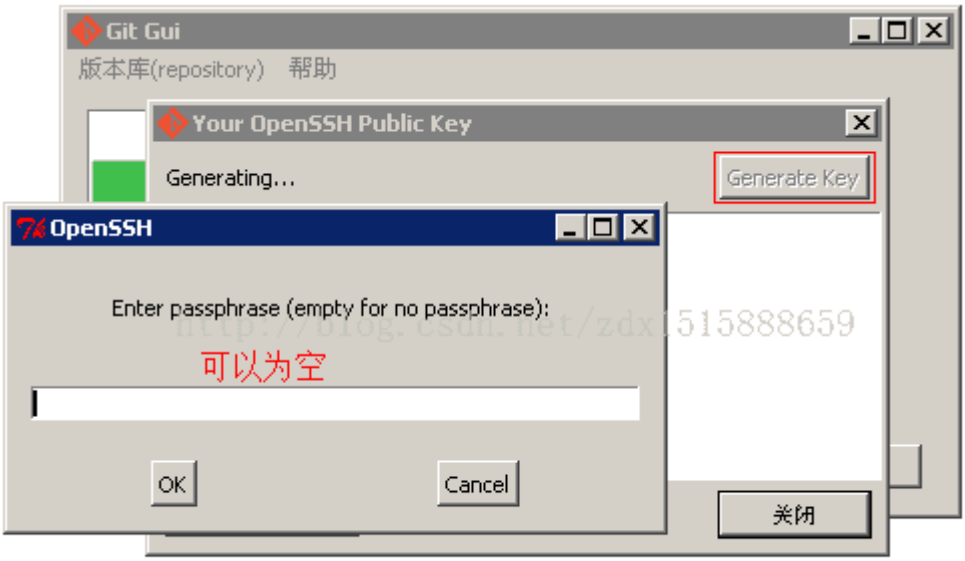
2.1.2 在可视化工具下生成 (推荐)

注意：使用这种方法生成时，默认会用计算机名，作为生成的 SSH Key 的名称，如果计算机名包含中文，则会因编码问题而出错。这时候可以使用在 Git Bash 命令行下生成的方法。

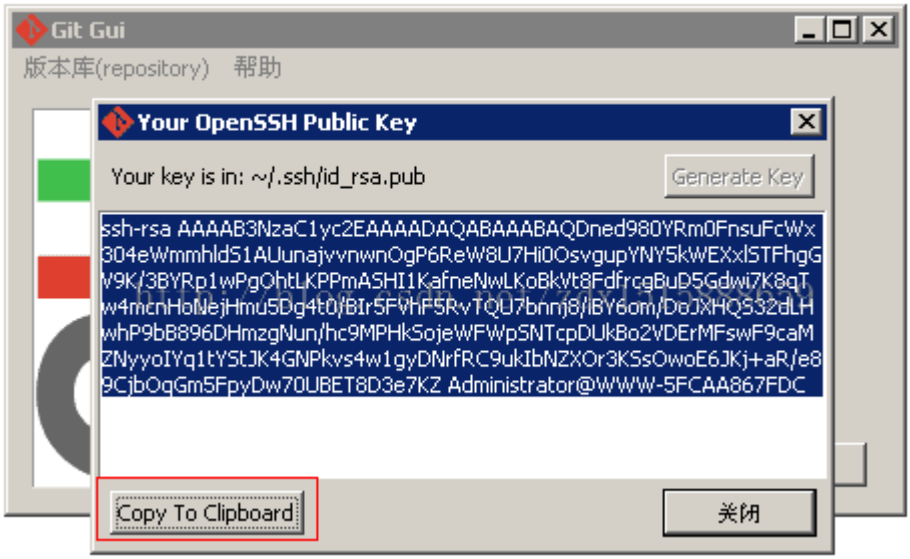
鼠标右键 -> Git Gui

帮助 -> Show SSH Key

点击 Generate Key，弹出一个对话框，提示输入 passphrase（密码短语），需要输入两次。意思就是以后提交数据到服务端，只要输入这个密码短语就可以了。这里可以为空，直接点OK，这样，以后就不需要输入任何密码。但建议还是要输入密码短语。



复制 SSH Key 的公钥



2.2 在 Gitlab 上配置 SSH Key

配置好 SSH Key 以后提交代码，可以不用输入密码。点击右上角的资料设置 -> SSH 密钥 -> 增加 SSH 密钥

粘贴刚刚复制的 SSH Key 公钥，标题为可选，不写会自动生成

2.3 在 TortoiseGit 上配置 SSH Key

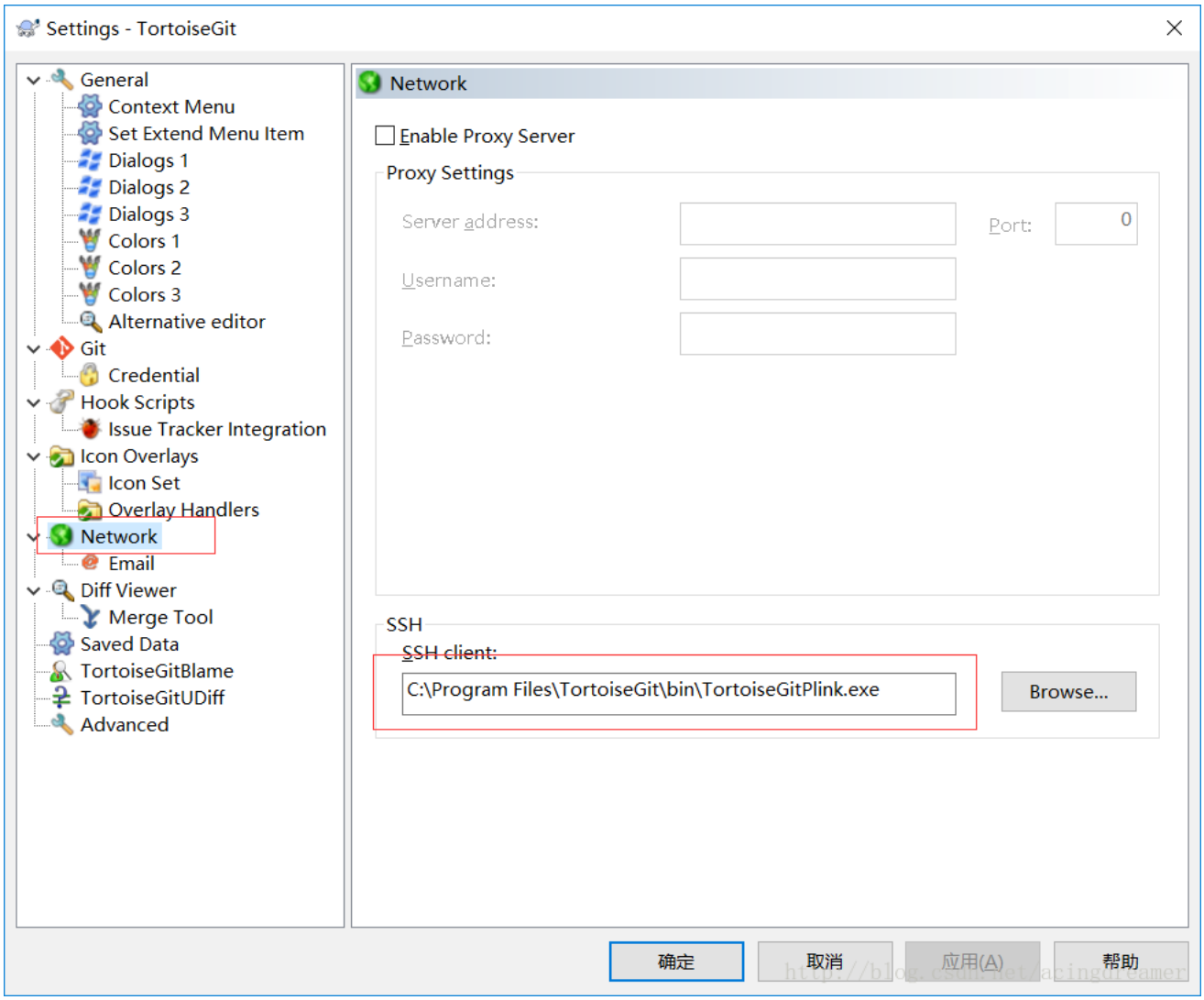
windows下一般会使用TortoiseGit来完成git操作，但是git bash又是必不可少的。

TortoiseGit默认使用putty格式的私钥。

git默认使用的是OpenSsh格式的私钥。

为了使用同一个私钥，将TortoiseGit 的连接方式更改一下。

TortoiseGit提供了这个功能，右键，打开TortoiseGit的setting。将network中的ssh-client改为git目录下的ssh.exe，路径为 `C:\Program Files\Git\usr\bin\ssh.exe`。



3. git 的常用命令

git config配置

```
Git 全局设置
$ git config --global user.name "username"
$ git config --global user.email "username@seemmo.com"

本地设置
$ git config user.name "username"
$ git config user.email "username@seemmo.com"
```

git remote配置远程库

```
# 显示所有远程仓库
$ git remote -v

# 显示某个远程仓库的信息
$ git remote show [shortname]

# 增加一个新的远程仓库，并命名
$ git remote add [shortname] [url]
$ git remote add origin http://192.168.2.3/panpan.nie/mystudy_project.git

# 删除远程仓库
$ git remote rm [shortname]

# 修改ssh 地址
$ git remote set-url origin git@github.com:someaccount/someproject.git

# 修改仓库名称
$ git remote rename origin old-origin
```

基本命令

```
# 新建文件
$ touch README.md

# 添加指定文件
$ git add README.md

# 添加所有文件
$ git add .

# 提交（带注释）到本地服务器
$ git commit -m "add README"

# 从服务器上拉取代码
$ git pull origin master


# 从本地服务器推送到远程服务器
$ git push -u origin master
```

```
$ git push -u origin --tags

# 取回远程仓库的变化，并与本地分支合并
$ git pull [remote] [branch]

# 上传本地指定分支到远程仓库
$ git push [remote] [branch]

# 强行推送当前分支到远程仓库，即使有冲突
$ git push [remote] --force

# 推送所有分支到远程仓库
$ git push [remote] --all
```

添加/删除文件(到暂存区，即add操作）

```
#可以添加一个或多个
$ git add <file1> <file2>...

#添加所有修改的和新添加的
$ git add .
#另一种写法
$ git add -A

#添加指定目录
$ git add <dirname>

#由暂存区恢复到工作区（发现提交错了，退回一步）
$ git reset HEAD <file>

#恢复上一次add提交的所有file
$ git reset HEAD

#撤销修改操作，恢复到修改之前的，撤销add后位于工作区下进行的
$ git checkout -- <file>

#删除文件,并将文件放入暂存区
$ git rm <file1> <file2>
#改文件名，并将修改后的文件放入暂存区
$ git mv <file-original> <file-rename>
```

状态git status

```
git status  显示有变更的文件
git diff  查看不同
git log  查看提交记录
```

强制拉取

```
# git pull时出现冲突 放弃本地修改，使远程库内容强制覆盖本地代码
git fetch --all //只是下载代码到本地，不进行合并操作
git reset --hard origin/master  //把HEAD指向最新下载的版本
```

撤销

撤销未提交的修改
有时候代码写的过于草率，以至于原来正常的功能被我们改出了问题，只要代码还没提交，所有修改的内容就可以撤销，使用checkout命令

```
git checkout src/com/example/activitylifecircletest/MainActivity.java
```

这样，我们对MainActivity.java这个文件所做的修改就可以撤销了 。不过这种撤销方式只适合那些还没执行过add的文件，如果某个人文件已经被添加过了，这种方式是撤销无效的。对于已经被add过的，我们可以先对其取消添加，在撤回提交，使用reset命令

```
git reset HEAD src/com/example/activitylifecircletest/MainActivity.java
```

然后再执行一遍git status命令，就能发现这个文件已经变回未添加状态，此时就可以使用checkout命令来进行撤销了

```
(1) Git checkout
恢复某个已修改的文件（撤销未提交的修改）：
$ git checkout file-name

例如：git checkout src/com/Android/.../xxx.Java

比如修改的都是java文件，不必一个个撤销，可以使用

$ git checkout *.java

撤销所有修改

$ git checkout .

(2) git revert

撤销某次操作，此次操作之前和之后的commit和history都会保留，并且把这次撤销
作为一次最新的提交
    * git revert HEAD                  撤销前一次 commit
    * git revert HEAD^                 撤销前前一次 commit
    * git revert commit-id （比如：fa042ce57ebbe5bb9c8db709f719cec2c58ee7ff）撤销指定的版本，撤销也会作为一次提交进行保存。
git revert是提交一个新的版本，将需要revert的版本的内容再反向修改回去，版本会递增，不影响之前提交的内容
```

gitignore

所有配置文件可以直接在线浏览：https://github.com/github/gitignore

忽略文件的原则是：

1. 忽略操作系统自动生成的文件，比如缩略图等；
2. 忽略编译生成的中间文件、可执行文件等，也就是如果一个文件是通过另一个文件自动生成的，那自动生成的文件就没必要放进版本库，比如Java编译产生的`.class`文件；
3. 忽略你自己的带有敏感信息的配置文件，比如存放口令的配置文件。

例如:Eclipse 项目可以在 .gitignore 中加入下面的内容 来忽略这些文件

```
.settings/
bin/
target/
.classpath
.project
```

4. git 报错与解决方案

4.1 Git pull 冲突解决

1. 问题描述：

王修改了文件A并且push到了git server上，这时李也在修改文件A，但修改没有完成，李希望获得最新的代码，如果李直接pull的话会遇到一下问题：

```
error: Your local changes to the following files would be overwritten by merge:
Please, commit your changes or stash them before you can merge.
```

造成冲突的原因：

很多命令都可能出现冲突，但从根本上来讲，都是merge 和 patch（应用补丁）时产生冲突。

而rebase就是重新设置基准，然后应用补丁的过程，所以也会冲突。

git pull会自动merge，repo sync会自动rebase，所以git pull和repo sync也会产生冲突。当然git rebase就更不用说了。

2. 问题解决

2.1 commit your change

先提交你的改动，在更新代码

2.2 stash

git stash

git pull

git stash pop

git stash的作用是把本地的改动保存起来，、

然后在git pull 这样就不会有冲突的问题

最后git stash pop 就是把之前的改动merge到代码中。

```
1、git stash 将本地代码stash到仓库中。
        可以使用git stash save ***定义自己的标记，方便以后查询
    2、git pull 将远程代码拉取到本地。
    3、git stash pop 将仓库中的代码合到本地最新代码中。
    4、在处理bug的过程中，可能存在多次stash的操作。这时可以使用git stash list查看本地仓库中都存储了几个stash版本。
    5、git stash pop默认将最近一次stash操作合并到本地代码中，也可以通过git stash pop stash@{Number}指定将某次stash的内容合并到本地代码中。
    6、git stash pop命令在合并代码的同时，会把仓库中对应的内容弹出。如果只想查看，而不想弹出内容，可以使用git stash apply命令进行操作。
    7、git stash -h 查看git stash帮助
    8、git stash show 显示stash合并到本地代码后，哪些文件会修改，以及修改的概述
    9、git stash show -p stash@{0} 显示修改的详细内容
```

2.3 merge

使用`git merge --abort`中止merge。merge manual中说，这条命令会尽力恢复到Merge之前的状态（可能失败！）。

merge manual中有一条警告：

> Warning: Running git merge with uncommitted changes is discouraged: while possible, it leaves you in a state that is hard to back out of in the case of a conflict.

4.2 树冲突

文件名修改造成的冲突，称为树冲突。

比如，a用户把文件改名为a.c，b用户把同一个文件改名为b.c，那么b将这两个commit合并时，会产生冲突。

```
$ git status
    added by us:    b.c
    both deleted:   origin-name.c
    added by them:  a.c
```

如果最终确定用b.c，那么解决办法如下：

```
git rm a.c
git rm origin-name.c
git add b.c
git commit
#  执行前面两个git rm时，会告警“file-name : needs merge”，可以不必理会。
```

树冲突也可以用git mergetool来解决，但整个解决过程是在交互式问答中完成的，用d 删除不要的文件，用c保留需要的文件。

最后执行git commit提交即可。

5. 参考

git 常用命令: https://www.cnblogs.com/chenwolong/p/GIT.html

Git & Gitlab 使用指南 https://blog.csdn.net/zdx1515888659/article/details/72954000

git stash https://www.cnblogs.com/huanyou/p/6654813.html

6. 关于文档

文档名	编写人	编写时间	备注
git 使用文档	聂盼盼	20180520	初稿