**1.创建key, 根据自己的邮箱,将key贴到web中**

//生成key

ssh-keygen -t rsa -C "houxl@git.com"

//验证key

ssh -T git@github.com

git config --global user.name "houxl@git.com" //设置用户名

git config --global user.email "houxl@git.com"  //设置邮箱

1.1 在git clone 远程库的时候,直接在本地创建.git ,创建文件夹并且名称与远程项目名称一致;

1.2 如果先 git init 创建.git ,那么clone的时候创建子仓库

1.3 在clone的时候会自动创建远程项目别名origin;并且HEAD 指向master;

1.4 生成的key默认存储在C:\Users\Administrator\.ssh,并且所有的项目都会读取这里的key与远程库进行比较;

1.5 生成的key,只能被一个远程库使用;只能添加一次;

1.6 生成的key,在加入ssh keys,可以被远程库所有项目使用,加入某个项目的deploy,则只能这个项目使用;

**2.在clone服务器时,容易出现连接问题,主义https manager tools**

2.1 用https 生成的时候,会提示你输入github的帐号密码,但不需要生成key;

**3.缓存的使用 add ,**

注意添加和删除文件的操作;

经常 查看status ,可以看出哪些还没有添加到缓存,

**4.提交 commit,**

提交只是创建快照,如果要推送到服务器,用push将分支推送到服务器;

**5.分支 分支的创建**

用branch ,也可以用checkout -b,如果服务器上有,pull下来,如果没有创建分支,创建分支时clone 的分支是当前分支的所有已经提交的操作的副本;

 5.1服务端的分支可以有多个,每个代表一个项目或是独立模块,互不想干,这样多个分支可以并行;

5.2服务端的分支有多个,但只有一个项目,那么,其它分支都是master的子分支,在与主分支合并时不会产生冲突,当然,不相关的分支也可

以合并,但这是5.1情况;

**6.本地分支的使用:**

6.1直接拷贝主分支,建立临时分支,修改完毕,提交分支,创建快照;

6.2 git fetch  test\_ssh:temp从指定的服务器中clone 指定的head分支,temp为新建;

6.3 临时分支建立并修改 快照完毕,需要切换到服务端存在的主分支中,然后将这个临时子分支合并,然后 git push 分支到服务器,修改

完毕;

6.4 本地的分支都可以看做是临时的,不管服务端有没有,;当然也可以git push origin <branch>,将临时分支push到服务端;因为在push的时候本地当前分支只能push当前分支名称到服务端;在服务端,是不是合并到默认分支,看具体情况;

6.5 删除本地分支 git branch --delete ...;删除远端分支 git push origin --deleted ...

6.6 指定在本地创建心分支的时候copy的当前分支

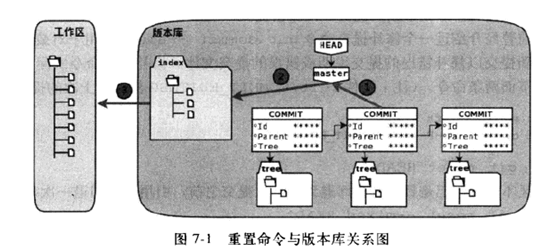
**7.比较不同,使用diff**

7.1默认比较本地的未提交修改的与服务器不同 git diff

7.2比较缓存中的与服务器的不同 diff -cached

7.3比较前两者与服务器的不同 diff -hard

**8. 恢复操作,reset**



8.1追加修改到最后一次快照

git commit -m 'initial commit'

git add forgotten\_file

git commit --amend

8.2 删除缓存中的修改,

git reset HEAD benchmarks.rb

8.3删除硬盘中的修改

git checkout -- benchmarks.rb

8.3 恢复多次commit

git reset --hard HEAD~3  从0开始删除0,1,2,3 commit;

--hard  硬盘修改,并且硬盘,暂存区,远程库 统一;

--soft 参数只执行1， 不进行暂存区和工作区的覆盖

--mixed或不使用参数，执行1,2覆盖暂存区，但不覆盖工作区

head 当前指针

**9.HEAD操作**

head:当前正在编写的分支;

检查当前head指向

cat .git/HEAD

检查commit id

cat .git/refs/heads/mastertemp

## 10.sourcetree ssh 相关问题

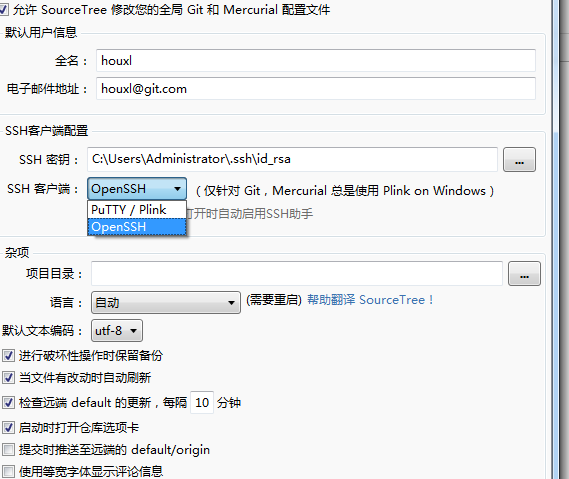
### windows系统ssh的远程连接设置(PUTTY)

原因是首次连接一个使用 SSH 协议的 Git 服务器的时候，因为远程SSH服务器的公钥没有经过确认而导致 git 命令执行失败。  
  
解决方法是在命令行下找到SourceTree安装目录里的PuTTy工具，  
例如" D:\Program Files (x86)\Atlassian\SourceTree\tools\putty\plink.exe"（注意路径包含空格时要加引号），  
然后运行命令 plink.exe root@192.168.1.2 -P port（根据自己的Git仓库URL自行更改），

plink.exe git@git.imyyl.com -P 8899 //端口改变后无法连接  
遇到确认服务器公钥时，输入"y"即可。

### 项目相关连接远程仓库的设置

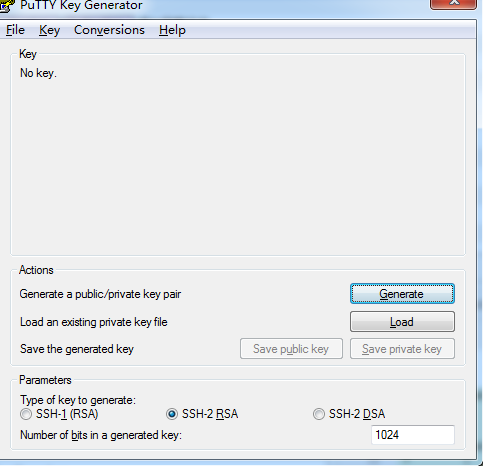
使用openSSH时,(github可生成生成本地公钥和获取远程公钥)



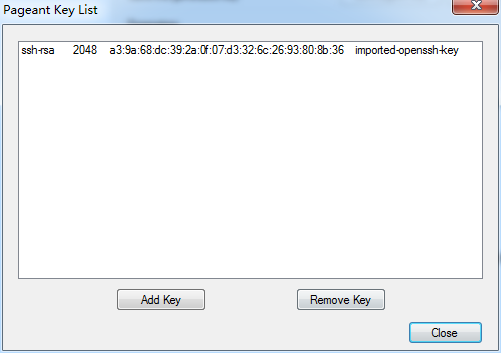
使用putty

可以导入私钥,然后转换成ppk格式的文件

也可以自己生成私钥;



生成后,加入



并将公钥传给远程服务器;

## 11.ssh相关介绍

http://www.ruanyifeng.com/blog/2011/12/ssh\_remote\_login.html

### 登录对方主机

$ ssh user@host

显示远程服务器的公钥,

核实公钥的真实性(到官网公布的公钥中查找),

输入密码

保存公钥到本地/etc/ssh/ssh\_known\_hosts

### 设置自己的公钥,省去每次登录输入密码的繁琐

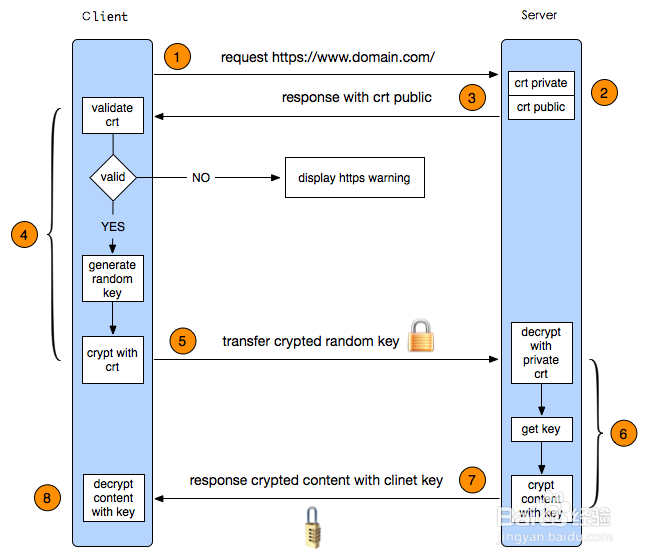
$ ssh-keygen

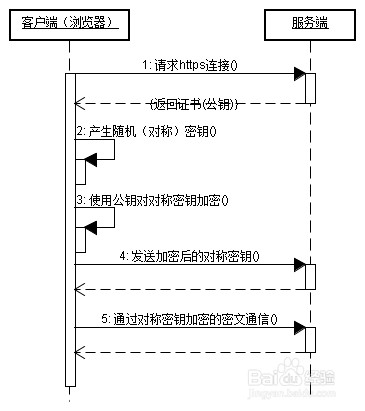
生成私钥-公钥(可以添加密码)

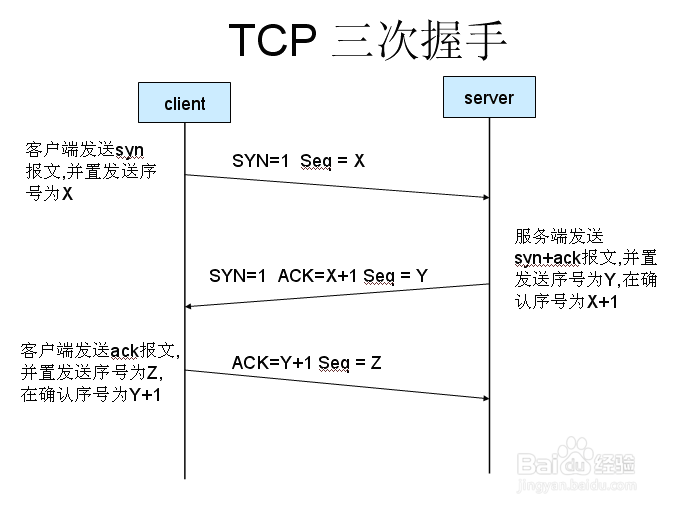
$ ssh-copy-id user@host

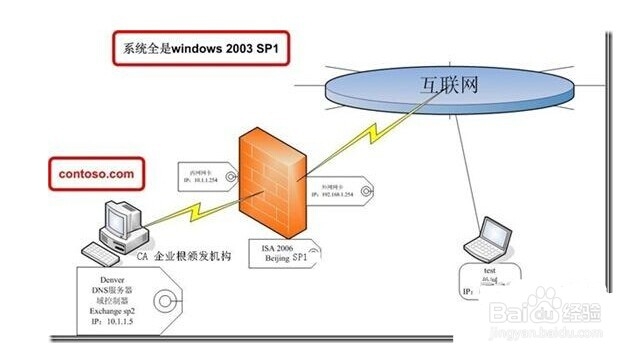
保存公钥到远程主机

https连接相关介绍









**服务器修改端口后**

直接修改URL为SSH://开头

git@git.imyyl.com:egameClient/EGameClient.git

git remote set-url origin ssh://git@git.imyyl.com:8899/~/egameClient/EGameClient.git

=====================

修改配置

vi  ~/.ssh/configvi

写入

# 映射一个别名

host egame

hostname git.imyyl.com

port 8899

# ctrl+D

vim .git/config

[remote "origin"]

    url = git@git.imyyl.com:egameClient/EGameClient.git

    fetch = +refs/heads/\*:refs/remotes/origin/\*

修改p1.git项目下的git配置文件

git remote add/rm origin 首先添加远程地址缩写

再设置url,也可以在上一部直接添加;

git remote set-url origin git@egame:egameClient/EGameClient.git

git remote set-url orign git@egame:egameClient/EGameClient.git

========================

**执行Git命令时出现各种 SSL certificate problem 的解决方法**

http://www.myexception.cn/open-source/1868332.html

git config --global http.sslVerify false

1.本地

2.缓存

3.快照

4.head

5.远程库

存在五个操作区域,最终达到本地文件操作与远程文件操作的合并而不相互冲突;

并且存储每一次的快照,可以用来使用;

6.分支

6.1 主分支 6.2 临时分支

所有的临时分支,都是主分支的副本,不管临时分支包含什么,最终首先建立主分支副本,然后合并;

构建共同操作主分支的临时分支,通过合并主分支,完成操作的合并;

可以建立多个长久性的临时分支,建立不同的并行操作分支;

代码合并：

首先本地未提交代码提交，然后pull 分支到本地，并进行合并，合并完毕push到远程分支上；

sourcetree停止跟踪：

直接删除本地和服务器上的操作的文件；在这个分支上的版本上，以后不再添加，但本地还是可以操作此文件；

？停止跟踪是忽略的批量操作：首先从版本中移除文件，然后添加到忽略名单中。

忽略文件：\*.ignore配置

客户端添加忽略，一个客户端添加，所有客户端都使用；

如果是需要忽略的，那么客户端都是需要忽略的本地文件或是配置

如果是要跟踪的，那么所有客户端可以选择xiu

 忽略的文件是未添加到版本中的文件

添加到版本中的文件要先删除，再添加忽略；

忽略有仅限于这个仓库和整个仓库的区别；