Git 服务器使用说明

一、 创建本地的 Git，并获得 Git 服务器权限

1、 Git 服务器 IP 地址：10.2.1.115
2、 使用前需要安装的程序（假定 Ubuntu 系统已经安装完成）

Git	sudo apt-get install git-all

3、 Git 安装完毕后，必须配置本机上的以下两个全局变量

git config --global user.name “Your name”
git config --global user.email “Your Email”

（注意，global 前面是两个减号--；要用小写的双引号把设置内容括起来）

如希望 Git 输出可以带颜色，也可增加如下设置：
git config --global color.ui “always”

设置完成后，可用下列命令查询设置是否成功
git config --global --list

4、 生成自己的密钥
ssh-keygen -t rsa

创建过程中会显示出一个默认的存储密钥的位置，记住存储密钥的位置，最好不要修改存储密钥的位置，除非你对 SSH 非常了解。
创建密钥过程中会要求输入密码，最好填写密码，并一定记住。
生成后，在存放密钥的位置有两个文件，缺省情况下是：

id_rsa 和 id_rsa.pub

其中，id_rsa.pub 是公钥，可以传递给其他人；id_rsa 自己一定要保存好，不要泄露给其他人。因为这个是验证你本人身份的。

5、 把公钥传给管理员，并通知管理员你希望在 Git 配置系统中出现的名称
6、 经与项目负责人商议后，通知管理员你对哪些 Git 项目有什么样的权限，由管理员在配置文件中分配。

二、 添加本地版本库到 Git 服务器

是否能添加新版本库到 Git 服务器，取决于你对此版本库的权限。

在本地创建一个版本库（如果已有，就使用已有的版本库）
示例代码如下：

cd ??	选择一个自己认为合适的目录
mkdir test cd test	创建一个将要保存此版本库数据的目录
git init	在此目录下，初始化创建一个 Git 版本库
	… 随便写点什么，在这个目录下加个文件
git add .	表示把此目录下（包括子目录）所有的变化添加到版本库暂存区
git commit –m “增加测试文件”	提交此次修改，把暂存区的内容提交到版本库。其中-m 表示后面的字符串是本次提交的留言。留言必须有内容，而且是有效内容，不能太简单
以上为创建一个新的本地版本库的指令，如果已经有，就不需要了，直接到已有本地版本库的目录下即可	
git remote add origin git@10.2.1.115:uiWork.git	表示要把本地版本库 uiWork 添加到 Git 服务器中去
git push origin master	表示把刚才要添加的版本库推送到 Git 服务器中去

注意：执行最后两条命令之前，需要前确认自己是否有权限创建这个项目。

三、 初次获得 Git 服务器上版本库内容

cd ??	进入到一个自己认为合适的目录
git clone git@10.2.1.115:IKEmbX00	将在当前目录下创建 IKEmbX00 目录，并把 Git 服务器上的版本库中数据复制到 IKEmbX00 目录下 回车后，可能会询问该计算机的私钥密码，请输入正确密码。 如果输出显示： ERROR:gitosis.serv.main:Repository read access denied 可能是因为没有把您对该项目权限加入，请与 Git 服务器管理员联系处理此事。

注意：需要对这个项目有读权限

四、 本地版本库中的修改内容提交

（假定本地已有新提交的内容）

git push

五、 从任意指定的远程版本库中取来修改变化，合并到当前本地分支

git pull “origin” “master” origin 即为指定的远程版本库名称；master 是本地版本库上的指定分支

六、 通过 IE 浏览器查询 Git 历史记录

可通过如下网址：

http://10.2.1.115/cgi-bin/gitweb.cgi

查看当前 git 服务器上 Git 项目的更新情况。

七、 Git 常用命令

把自己的邮箱地址添加到 Git 项目更新邮件列表中

如果对某个 Git 项目感兴趣，希望能及时收到该项目中更新情况的信息，可与管理员联系，要求把自己的邮箱地址添加到这个项目的邮件发送列表中。这样，任何人在该项目下有提交内容到 Git 服务器上，你就可以收到提示的邮件。

IKEmbX00 项目的邮件主题总是含有如下内容：

[GIT Notice]IKEmbX00]

你可以根据这个来设置自己的邮箱过滤器，以方便查找相关邮件。

设置 Git 忽略的文件类型（不需要 Git 对这类文件保持跟踪）

方法之一是编辑本地版本仓库的./git/inof/exclude 文件。这个是一个配置文件，写在这里的文件类型不会被 Git 跟踪。

例如，在该文件中写入一下几行

```
# ‘#’开头表示后面是注释
Debug.vc100/        # Debug.vc100 目录下所有文件都被忽略
*.*sdf              # 结尾名为 sdf 的文件都被忽略
*.sln               # 结尾名为 sln 的文件都被忽略
```

之后保存此文件即可。

恢复误删除的文件

因失误，删除了由 Git 管理的某个或某几个文件，还没有提交，希望恢复回来，该怎么做呢？

例如，某个目录下原本有如下文件：

```
ik-admin@IKEMBX00-Srv:~/ik-Git-Srv/IKEmbX00/Code/SocketComm$ ls
autoMachine.cpp  Makefile       socketAPI.h     socketInterfaceAPI.cpp
autoMachine.h    obj            socketBase.cpp  socketInterfaceAPI.h
Client           queueThread.h  socketBase.h    socketInterface.h
common.h         readme.txt     socketcom.h     stdWait.cpp
delegater.h      Server         socketCon.cpp   stdWait.h
lib              socketAPI.cpp  socketCon.h     test
```

之后做了两次删除操作：

```
ik-admin@IKEMBX00-Srv:~/ik-Git-Srv/IKEmbX00/Code/SocketComm$ rm Makefile
ik-admin@IKEMBX00-Srv:~/ik-Git-Srv/IKEmbX00/Code/SocketComm$ rm std*
```

第一次删除了 Makefile 文件，第二次删除了 stdWait.cpp 和 stdWait.h 两个文件。

查看 git 状态，有如下显示：

```
ik-admin@IKEMBX00-Srv:~/ik-Git-Srv/IKEmbX00/Code/SocketComm$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       deleted:    Makefile
#       deleted:    stdWait.cpp
#       deleted:    stdWait.h
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Git 发现有三个文件被删除了。

但实际上，并不想删除这三个文件，该怎么把这三个文件恢复回来呢？

```
ik-admin@IKEMBX00-Srv:~/ik-Git-Srv/IKEmbX00/Code/SocketComm$ git checkout -- std
Wait.cpp
ik-admin@IKEMBX00-Srv:~/ik-Git-Srv/IKEmbX00/Code/SocketComm$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       deleted:    Makefile
#       deleted:    stdWait.h
#
no changes added to commit (use "git add" and/or "git commit -a")
```

如上图所示，输入 git checkout -- 文件名，则可恢复指定文件（上图的指定文件是 stdWait.cpp）。

输入 git checkout .，则可恢复当前目录及子目录下全部被删除的（尚未提交，尚未进入缓冲区）的文件，即使这些文件在不同目录下。

其他 Git 命令

git pull --rebase

git rebase --continue

git pull --rebase 其实就等于 git fetch + git rebase

git diff 查看修改和缓存中的修改信息

git diff cache 查看缓存中的和 commit 中的修改信息

git branch -a 查看所有分支。

git branch new_branch 创建新的分支。

git branch -d branch 删除分支。

git checkout branch 切换当前分支。-f 参数可以覆盖未提交内容。

git branch -v

git remote -v

.gitignore 在规则建立之前的就已经在代码库中的文件不会受到忽略规则的影响，要移除这些文件，只能输入 git rm filename 来移除。

git rm -f *.o 取消跟踪并在工作目录中删除

git rm --cached readme.txt 取消跟踪不在工作目录中删除

git mv file_from file_to 相当于 mv README.txt README $ git rm README.txt $ git add README

git log -p -2 -p 选项展开显示每次提交的内容差异，用 -2 则仅显示最近的两次更新：

git log --stat 简要显示

git log --pretty=format:"%h - %an, %ar : %s"

git log --pretty=format:"%h %s" --graph
git log --since=2.weeks
git log --pretty="%h - %s" --author=gitster --since="2008-10-01" \
--before="2008-11-01" --no-merges -- t/
git reset HEAD benchmarks.rb 取消已经暂存的文件
git checkout -- benchmarks.rb 取消对文件的修改
git remote -v 查看现有的远程仓库
git remote add pb git://github.com/paulboone/ticgit.git 添加一个远程仓库 并用 pb 命名。
git remote rm paul 删除远程仓库
git remote rename pb paul 重名远程仓库 本地也会跟着修改
git remote 查看远程参考
git remote show
git branch 显示所有开发分支
git branch experimental 创建新的开发分支
git checkout experimental 切换到”experimental”分支
git merge experimental 合并分支，如果这个两个分支间的修改没有冲突(conflict), 那么合并就完成了。
如有有冲突，输入下面的命令就可以查看当前有哪些文件产生了冲突:$ git diff，git commit -a
git branch -d experimental 删除分支（合并过才删除）
git branch -D crazy-idea 强制删除分支