**三.创建版本库(仓库 repository)**

1.一个目录，目录里的文件都可被Git管理，每个文件的修改、删除，Git都能跟踪

2.步骤

(1)选E盘,右键Git bash here

|  |
| --- |
| $ mkdir gitroot  $ cd gitroot  $ pwd //显示当前目录  /e/gitroot |

注:Windows系统，目录名（包括父目录）不要包含中文。

(2)

|  |
| --- |
| $ git init //这个目录变成Git可以管理的仓库  Initialized empty Git repository in E:/gitroot/.git/ |

该目录下多了.git目录，是Git用来跟踪管理版本库的，不要手动修改

若没有看到.git目录，是因为该目录默认隐藏，用ls -ah命令就可看见。

也不一定必须在空目录下创建Git仓库，选择一个已经有东西的目录也是可以的。

(3)把文件添加到版本库

1)只能跟踪文本文件的改动，如TXT文件，网页，程序代码等。

2)图片、视频、Word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，也没法知道。

3)文本有编的，建议使用UTF-8

不要使用Windows自带的记事本编辑任何文本文件。Microsoft开发记事本的团队在每个文件开头添加了0xefbbbf（十六进制）的字符，比如，网页第一行可能会显示一个“?”，明明正确的程序一编译就报语法错误，等等，

Notepad++代替记事本，默认编码设置为UTF-8 without BOM

4)编写一个readme.txt文件，内容：

|  |
| --- |
| Git is a version control system.  Git is free software. |

放到gitroot目录下（子目录也行），因为是Git仓库

|  |
| --- |
| $ git add readme.txt //把文件添加到仓库(可写多次add多个)：没有任何显示，就对了  $ git commit -m "wrote a readme file" //把文件提交到仓库, -m后输入提交说明  结果:  [master (root-commit) cb926e7] wrote a readme file  1 file changed, 2 insertions(+)  create mode 100644 readme.txt |

**四.时光机穿梭**

1.(1)修改readme.txt文件：

|  |
| --- |
| Git is a distributed version control system.  Git is free software. |

(2) 查看仓库当前状态

|  |
| --- |
| $ git status  # On branch master  # Changes not staged for commit:  # (use "git add <file>..." to update what will be committed)  # (use "git checkout -- <file>..." to discard changes in working directory)  # modified: readme.txt  no changes added to commit (use "git add" and/or "git commit -a") |

(3)

|  |
| --- |
| $ git diff readme.txt //查看difference  diff --git a/readme.txt b/readme.txt  index 46d49bf..9247db6 100644  --- a/readme.txt  +++ b/readme.txt  @@ -1,2 +1,2 @@  -Git is a version control system.  +Git is a distributed version control system.  Git is free software. |

(4)$ git add readme.txt

$ git status

# On branch master

# Changes to be committed:

# (use "git reset HEAD <file>..." to unstage)

# modified: readme.txt

$ git commit -m "add distributed"

[master ea34578] add distributed

1 file changed, 1 insertion(+), 1 deletion(-)

$ git status

# On branch master

nothing to commit (working directory clean) //当前没有需要提交的修改，且工作目录干净

**五.时光机穿梭**

**5.1版本回退**

(1)再修改readme.txt文件如下：

|  |
| --- |
| Git is a distributed version control system.  Git is free software distributed under the GPL. |

然后尝试提交：

|  |
| --- |
| $ git add readme.txt  $ git commit -m "append GPL"  [master 3628164] append GPL  1 file changed, 1 insertion(+), 1 deletion(-) |

(2)回顾一下readme.txt文件一共有几个版本被提交到Git仓库里了：

版本1：wrote a readme file

Git is a version control system.

Git is free software.

版本2：add distributed

Git is a distributed version control system.

Git is free software.

版本3：append GPL

Git is a distributed version control system.

Git is free software distributed under the GPL.

(3) 提交日志

|  |
| --- |
| $ git log //显示从最近到最远的提交日志  commit 3628164fb26d48395383f8f31179f24e0882e1e0  Author: Michael Liao <askxuefeng@gmail.com>  Date: Tue Aug 20 15:11:49 2013 +0800  append GPL  commit ea34578d5496d7dd233c827ed32a8cd576c5ee85  Author: Michael Liao <askxuefeng@gmail.com>  Date: Tue Aug 20 14:53:12 2013 +0800  add distributed  commit cb926e7ea50ad11b8f9e909c05226233bf755030  Author: Michael Liao <askxuefeng@gmail.com>  Date: Mon Aug 19 17:51:55 2013 +0800  wrote a readme file |

(4)

|  |
| --- |
| $ git log --pretty=oneline //减少输出信息  3628164fb26d48395383f8f31179f24e0882e1e0 append GPL // commit id（版本号）  ea34578d5496d7dd233c827ed32a8cd576c5ee85 add distributed  cb926e7ea50ad11b8f9e909c05226233bf755030 wrote a readme file |

(5)版本回退

1)当前版本：HEAD

上一个版本：HEAD^，

上上一个版本：HEAD^^

往上100个版本：HEAD~100

2)把当前版本“append GPL”回退到上一个版本“add distributed”：

|  |
| --- |
| $ git reset --hard HEAD^  HEAD is now at ea34578 add distributed |

3)

|  |
| --- |
| $ cat readme.txt //看看readme.txt的内容  Git is a distributed version control system.  Git is free software. |

4)

|  |
| --- |
| $ git log //查看提交历史  commit ea34578d5496d7dd233c827ed32a8cd576c5ee85  Author: Michael Liao <askxuefeng@gmail.com>  Date: Tue Aug 20 14:53:12 2013 +0800  add distributed  commit cb926e7ea50ad11b8f9e909c05226233bf755030  Author: Michael Liao <askxuefeng@gmail.com>  Date: Mon Aug 19 17:51:55 2013 +0800  wrote a readme file |

最新的append GPL版本已经看不到了

只要上面的命令行窗口还没有关掉，就可顺着往上找，找到append GPL的commit id是3628164...，就可指定回到未来的某个版本：

5)

|  |
| --- |
| $ git reset --hard 3628164 //版本号没必要写全，Git会自动去找  HEAD is now at 3628164 append GPL |

6)再查看

|  |
| --- |
| $ cat readme.txt  Git is a distributed version control system.  Git is free software distributed under the GPL. |

Git的版本回退速度非常快，

7)

|  |
| --- |
| $ git reflog //查看命令历史，即使关闭过窗口  ea34578 HEAD@{0}: reset: moving to HEAD^  3628164 HEAD@{1}: commit: append GPL  ea34578 HEAD@{2}: commit: add distributed  cb926e7 HEAD@{3}: commit (initial): wrote a readme file |

**5.2工作区和暂存区**

1. Git和其他版本控制系统的不同之处就是有暂存区的概念。

2.工作区（Working Directory）

电脑里能看到的目录，如learngit文件夹

3.版本库（Repository）

(1)工作区的隐藏目录.git，不算工作区，而是Git的版本库。

版本库里存了很多东西，最重要的就是称为stage（或叫index）的暂存区，还有Git自动创建的第一个分支master，以及指向master的一个指针叫HEAD。



(2)前面讲了把文件往Git版本库里添加：

git add把文件添加进去，就是把文件修改添加到暂存区；

git commit提交更改，把暂存区的所有内容提交到当前分支。

注:创建Git版本库时，Git自动创建了唯一一个master分支，git commit就是往master分支上提交更改。

需要提交的文件修改通通放到暂存区，然后，一次性提交暂存区的所有修改。

(3)先对readme.txt做个修改，比如加上一行内容：

|  |
| --- |
| Git is a distributed version control system.  Git is free software distributed under the GPL.  Git has a mutable index called stage. |

然后，在工作区新增一个LICENSE文本文件（内容随便写）。

|  |
| --- |
| $ git status  # On branch master  # Changes not staged for commit:  # (use "git add <file>..." to update what will be committed)  # (use "git checkout -- <file>..." to discard changes in working directory)  # modified: readme.txt  # Untracked files:  # (use "git add <file>..." to include in what will be committed)  # LICENSE  no changes added to commit (use "git add" and/or "git commit -a") |

Git告诉我们,readme.txt被修改了,而LICENSE还从来没有被添加过，状态是Untracked。

使用两次命令git add，把readme.txt和LICENSE都添加后，用git status再查看一下：

|  |
| --- |
| $ git status  # On branch master  # Changes to be committed:  # (use "git reset HEAD <file>..." to unstage)  # new file: LICENSE  # modified: readme.txt |

现在，暂存区的状态就变成这样了：



git add命令实际上是把要提交的所有修改放到暂存区（Stage），然后执行git commit就可以一次性把暂存区的所有修改提交到分支。

|  |
| --- |
| $ git commit -m "understand how stage works"  [master 27c9860] understand how stage works  2 files changed, 675 insertions(+)  create mode 100644 LICENSE |

|  |
| --- |
| $ git status  # On branch master  nothing to commit (working directory clean) |

现在版本库变成了这样，暂存区就没有任何内容了：



**5.3管理修改**

1.为什么Git比其他版本控制系统设计得优秀，因为Git跟踪并管理的是修改，而非文件。

解说:

对readme.txt做一个修改，比如加一行内容：

|  |
| --- |
| $ cat readme.txt  Git is a distributed version control system.  Git is free software distributed under the GPL.  Git has a mutable index called stage.  Git tracks changes. |

然后，添加：

|  |
| --- |
| $ git add readme.txt  $ git status  # On branch master  # Changes to be committed:  # (use "git reset HEAD <file>..." to unstage)  # modified: readme.txt |

然后，再修改readme.txt：

|  |
| --- |
| $ cat readme.txt  Git is a distributed version control system.  Git is free software distributed under the GPL.  Git has a mutable index called stage.  Git tracks changes of files. |

提交：

|  |
| --- |
| $ git commit -m "git tracks changes"  [master d4f25b6] git tracks changes  1 file changed, 1 insertion(+) |

提交后，再看看状态：

|  |
| --- |
| $ git status  # On branch master  # Changes not staged for commit:  # (use "git add <file>..." to update what will be committed)  # (use "git checkout -- <file>..." to discard changes in working directory)  # modified: readme.txt  no changes added to commit (use "git add" and/or "git commit -a") |

怎么第二次的修改没有被提交？

第一次修改 -> git add -> 第二次修改 -> git commit

解释:在工作区的第二次修改并没有放入暂存区，git commit只负责把暂存区的修改提交了，也就是第一次的修改被提交了，第二次的修改不会被提交。

提交后，用git diff HEAD -- readme.txt命令可以查看工作区和版本库里面最新版本的区别：

|  |
| --- |
| $ git diff HEAD -- readme.txt  diff --git a/readme.txt b/readme.txt  index 76d770f..a9c5755 100644  --- a/readme.txt  +++ b/readme.txt  @@ -1,4 +1,4 @@  Git is a distributed version control system.  Git is free software distributed under the GPL.  Git has a mutable index called stage.  -Git tracks changes.  +Git tracks changes of files. |

可见，第二次修改确实没有被提交。

第一次修改 -> git add -> 第二次修改 -> git add -> git commit

总结:每次修改，如果不add到暂存区，那就不会加入到commit中。

**5.4撤销修改**

1.在readme.txt中添加了一行：

|  |
| --- |
| $ cat readme.txt  Git is a distributed version control system.  Git is free software distributed under the GPL.  Git has a mutable index called stage.  Git tracks changes of files.  My stupid boss still prefers SVN. |

可以删掉最后一行，手动把文件恢复到上一个版本的状态。如果用git status查看一下：

|  |
| --- |
| $ git status  # On branch master  # Changes not staged for commit:  # (use "git add <file>..." to update what will be committed)  # (use "git checkout -- <file>..." to discard changes in working directory)  # modified: readme.txt  no changes added to commit (use "git add" and/or "git commit -a") |

git checkout -- file可以丢弃工作区的修改：

|  |
| --- |
| $ git checkout -- readme.txt //没有--，就变成了“切换到另一个分支”的命令 |

把readme.txt文件在工作区的修改全部撤销，有两种情况：

(1)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

(2)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

就是让这个文件回到最近一次git commit或git add时的状态。

现在，看看readme.txt的文件内容：

|  |
| --- |
| $ cat readme.txt  Git is a distributed version control system.  Git is free software distributed under the GPL.  Git has a mutable index called stage.  Git tracks changes of files. |

文件内容果然复原了。

2. 现在假定是凌晨3点，你不但写了一些胡话，还git add到暂存区了：

|  |
| --- |
| $ cat readme.txt  Git is a distributed version control system.  Git is free software distributed under the GPL.  Git has a mutable index called stage.  Git tracks changes of files.  My stupid boss still prefers SVN.  $ git add readme.txt |

在commit之前，发现了这个问题。用git status查看一下，修改只是添加到了暂存区，还没有提交：

|  |
| --- |
| $ git status  # On branch master  # Changes to be committed:  # (use "git reset HEAD <file>..." to unstage)  # modified: readme.txt |

(1)用命令git reset HEAD file可以把暂存区的修改撤销掉（unstage），重新放回工作区：

|  |
| --- |
| $ git reset HEAD readme.txt  Unstaged changes after reset:  M readme.txt |

git reset命令既可以回退版本，也可以把暂存区的修改回退到工作区。

用HEAD时，表示最新的版本。

再用git status查看一下，现在暂存区是干净的，工作区有修改：

|  |
| --- |
| $ git status  # On branch master  # Changes not staged for commit:  # (use "git add <file>..." to update what will be committed)  # (use "git checkout -- <file>..." to discard changes in working directory)  # modified: readme.txt  no changes added to commit (use "git add" and/or "git commit -a") |

(2)丢弃工作区的修改:

|  |
| --- |
| $ git checkout -- readme.txt  $ git status  # On branch master  nothing to commit (working directory clean) |

一旦你把“stupid boss”提交推送到远程版本库，你就真的惨了……

小结

场景1：改乱了工作区某文件内容，想直接丢弃工作区的修改，git checkout -- file。

场景2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令git reset HEAD file，就回到了场景1，第二步按场景1操作。

场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考版本回退一节，不过前提是没有推送到远程库

**5.5删除文件**

删除也是一个修改操作

1.先添加一个新文件test.txt到Git并且提交：

|  |
| --- |
| $ git add test.txt  $ git commit -m "add test.txt"  [master 94cdc44] add test.txt  1 file changed, 1 insertion(+)  create mode 100644 test.txt |

一般情况下，你通常直接在文件管理器中把没用的文件删了，或者用rm命令删了：

|  |
| --- |
| $ rm test.txt |

Git知道你删除了文件，工作区和版本库就不一致了：

|  |
| --- |
| $ 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: test.txt  no changes added to commit (use "git add" and/or "git commit -a") |

(1)两个选择，一是确实要从版本库中删除该文件，那就用命令git rm删掉，并且git commit：

|  |
| --- |
| $ git rm test.txt  rm 'test.txt'  $ git commit -m "remove test.txt"  [master d17efd8] remove test.txt  1 file changed, 1 deletion(-)  delete mode 100644 test.txt |

现在，文件就从版本库中被删除了。

(2)另一种情况是删错了，因为版本库里还有，所以可以把误删的文件恢复到最新版本：

|  |
| --- |
| $ git checkout -- test.txt //用版本库里的版本替换工作区的版本, 修改和删除都可一键还原 |

小结

git rm用于删除一个文件。如果一个文件已经被提交到版本库，那么你永远不用担心误删，但是要小心，你只能恢复文件到最新版本，你会丢失最近一次提交后你修改的内容。

**六.远程仓库**

1.实际情况往往是这样，找一台电脑充当服务器的角色，每天24小时开机，其他每个人都从这个“服务器”仓库克隆一份到自己的电脑上，并且各自把各自的提交推送到服务器仓库里，也从服务器仓库中拉取别人的提交。

2.GitHub网站，就是提供Git仓库托管服务的

第1步：创建SSH Key:

用户主目录下，.ssh目录， id\_rsa和id\_rsa.pub文件

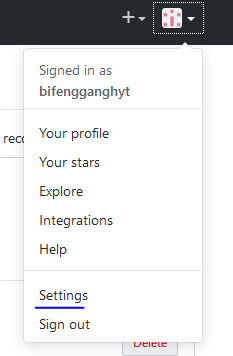
如果没有，打开Shell（Windows下打开Git Bash），创建SSH Key：

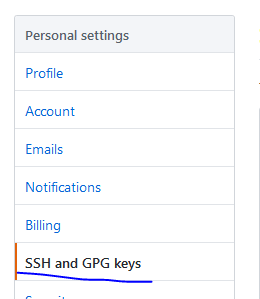
|  |
| --- |
| $ ssh-keygen -t rsa -C [418239445@qq.com](mailto:418239445@qq.com)  Generating public/private rsa key pair.  Enter file in which to save the key (/c/Users/S43W/.ssh/id\_rsa):  Enter passphrase (empty for no passphrase):  Enter same passphrase again:  Your identification has been saved in /c/Users/S43W/.ssh/id\_rsa.  Your public key has been saved in /c/Users/S43W/.ssh/id\_rsa.pub.  The key fingerprint is:  SHA256:23j659b6QP4RGo2z1CtdT0W5DrhhuhvRqIi1EMAZsT0 418239445@qq.com  The key's randomart image is:  +---[RSA 2048]----+  |.++ o|  | o+ o |  | . E . o|  | o o+ = ..|  | . . Soo.O \* o|  | + o .=.= \* \*.|  | . o .o.+ \*.+ .|  | +. o+.. |  | .oo+ooo |  +----[SHA256]-----+ |

一路回车，使用默认值即可，也无需设置密码。

在用户主目录里找到.ssh目录，id\_rsa和id\_rsa.pub两个文件，是SSH Key的秘钥对，id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥，可以放心地告诉任何人。

第2步：登陆GitHub





填上任意Title，在Key文本框里粘贴id\_rsa.pub文件的内容

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。

GitHub允许你添加多个Key。

任何人都可以看到喔（但只有你自己才能改）。如果不想让别人看到Git库，一个是交点保护费，让GitHub把公开的仓库变成私有的。另一个办法是自己动手，搭一个Git服务器.

s

git clone [git@github.com:bifengganghyt/gitroot.git](mailto:git@github.com:bifengganghyt/gitroot.git)

每次本地提交后，只要有必要，就可使用命令git push origin master推送最新修改；

**十二.附录**

**12.1新电脑克隆远程库**

**1.安装git**

在Windows上安装Git

(1)Windows下使用很多Linux/Unix的工具时，需Cygwin模拟环境。

msysgit是Windows版的Git，把模拟环境和Git都打包好了，从https://git-for-windows.github.io下载，默认安装。

“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！

(2)安装完成后，设置，：

|  |
| --- |
| $ git config --global user.name "heyutong"  $ git config --global user.email "418239445@qq.com" |

因为Git是分布式版本控制系统，所以，每个机器都必须自报家门：你的名字和Email地址。

--global参数，表示这台机器上所有的Git仓库都会使用这个配置，当然也可以对某个仓库指定不同的用户名和Email地址。

**2.远程仓库**

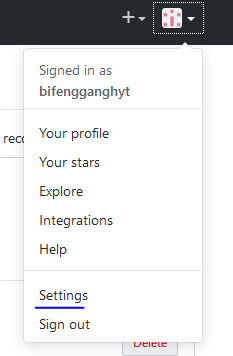
第1步：创建SSH Key:

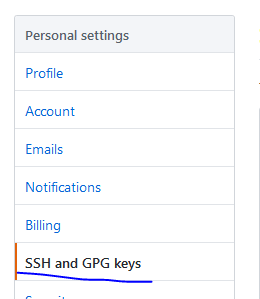
|  |
| --- |
| $ ssh-keygen -t rsa -C [418239445@qq.com](mailto:418239445@qq.com)  Generating public/private rsa key pair.  Enter file in which to save the key (/c/Users/S43W/.ssh/id\_rsa):  Enter passphrase (empty for no passphrase):  Enter same passphrase again:  Your identification has been saved in /c/Users/S43W/.ssh/id\_rsa.  Your public key has been saved in /c/Users/S43W/.ssh/id\_rsa.pub.  The key fingerprint is:  SHA256:23j659b6QP4RGo2z1CtdT0W5DrhhuhvRqIi1EMAZsT0 418239445@qq.com  The key's randomart image is:  +---[RSA 2048]----+  |.++ o|  | o+ o |  | . E . o|  | o o+ = ..|  | . . Soo.O \* o|  | + o .=.= \* \*.|  | . o .o.+ \*.+ .|  | +. o+.. |  | .oo+ooo |  +----[SHA256]-----+ |

一路回车，使用默认值即可，也无需设置密码。

在用户主目录里找到.ssh目录，id\_rsa和id\_rsa.pub两个文件，是SSH Key的秘钥对，id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥，可以放心地告诉任何人。

第2步：登陆GitHub





填上任意Title，在Key文本框里粘贴id\_rsa.pub文件的内容

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。

**3.克隆远程仓库到本地**

|  |
| --- |
| $ git clone git@github.com:bifengganghyt/gitroot.git  Cloning into 'gitroot'...  remote: Counting objects: 26, done.  remote: Compressing objects: 100% (3/3), done.  remote: Total 26 (delta 0), reused 3 (delta 0), pack-reused 23  Receiving objects: 100% (26/26), 1.05 MiB | 387.00 KiB/s, done.  Resolving deltas: 100% (3/3), done. |