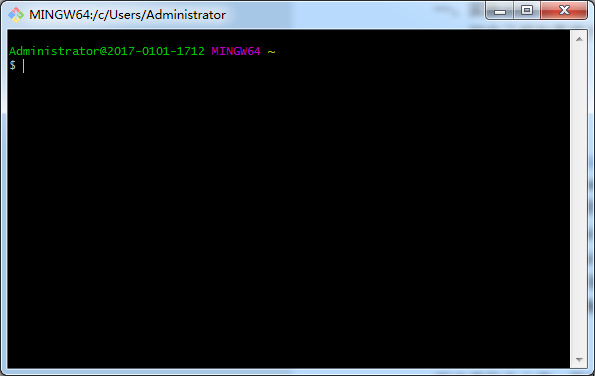
实验二 Git入门

1. 实验目的

初步了解和掌握代码管理工具Git。

1. 实验内容
2. 下载与安装
3. 从 <https://git-scm.com/> 下载并安装Git
4. 设置用户名和Email地址

在开始菜单里找到“Git”->“Git Bash”



输入以下内容：

$ git config --global user.name YourName

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

（$为提示符不用输入，YourName、[email@example.com](mailto:email@example.com)改成你自己的信息）

以下操作如无特殊说明均在Git Bash环境输入

1. 创建库

版本库又名仓库，英文名repository，你可以简单理解成一个目录，这个目录里面的所有文件都可以被Git管理起来，每个文件的修改、删除，Git都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻可以“还原”。

1. 创建一个版本库非常简单，首先，选择一个合适的地方，创建一个空目录：

$ mkdir learngit

$ cd learngit

$ pwd



（pwd命令用于显示当前目录，该位置指C盘的Users/Administrator/learngit文件夹。）

1. 通过git init命令把这个目录变成Git可以管理的仓库：

$ git init



可以发现当前目录下多了一个.git的隐藏目录，这个目录是Git来跟踪管理版本库的，没事千万不要手动修改这个目录里面的文件，不然改乱了，就把Git仓库给破坏了。

1. 把文件添加到版本库

所有的版本控制系统，其实只能跟踪文本文件的改动，比如TXT文件，网页，所有的程序代码等等，版本控制系统可以告诉你每次的改动，比如在第5行加了一个单词“Linux”，在第8行删了一个单词“Windows”。而图片、视频这些二进制文件，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。

千万不要使用Windows自带的记事本编辑任何文本文件，会产生UTF-8编码错误。可以使用Notepad++或Visual Studio Code（VSC）等专业编辑器

在VSC的菜单：文件-打开文件夹选择刚刚创建的版本库目录learngit。菜单：文件-新建文件创建一个readme.txt文件。内容如下：

Git is a version control system.

Git is free software.

把一个文件放到Git仓库只需要两步。

第一步，用命令git add告诉Git，把文件添加到仓库：

git add readme.txt

第二步，用命令git commit告诉Git，把文件提交到仓库：

$ git commit -m "wrote a readme file"

简单解释一下git commit命令，-m后面输入的是本次提交的说明，可以输入任意内容，当然最好是有意义的，这样你就能从历史记录里方便地找到改动记录。

为什么Git添加文件需要add，commit一共两步呢？因为commit可以一次提交很多文件，所以你可以多次add不同的文件，比如：

$ git add file1.txt

$ git add file2.txt file3.txt

$ git commit -m "add 3 files."

1. 版本管理
2. 版本比较

修改readme.txt的内容为：

Git is a distributed version control system.

Git is free software.

运行git status命令看看结果：

$ 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 status命令可以让我们时刻掌握仓库当前的状态，上面的命令输出告诉我们，readme.txt被修改过了，但还没有准备提交的修改。

虽然Git告诉我们readme.txt被修改了，但如果能看看具体修改了什么内容，自然是很好的。比如你休假两周从国外回来，第一天上班时，已经记不清上次怎么修改的readme.txt，所以，需要用git diff这个命令看看：

$ git diff readme.txt

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.

git diff顾名思义就是查看difference，显示的格式正是Unix通用的diff格式，可以从上面的命令输出看到，我们在第一行添加了一个distributed单词。

知道了对readme.txt作了什么修改后，再把它提交到仓库就放心多了，提交修改和提交新文件是一样的两步，第一步是git add：

$ git add readme.txt

同样没有任何输出。在执行第二步git commit之前，我们再运行git status看看当前仓库的状态：

$ git status

On branch master

Changes to be committed:

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

modified: readme.txt

git status告诉我们，将要被提交的修改包括readme.txt，下一步，就可以放心地提交了：

$ git commit -m "add distributed"

[master ab7d5c5] add distributed

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

提交后，我们再用git status命令看看仓库的当前状态：

$ git status

On branch master

nothing to commit, working tree clean

Git告诉我们当前没有需要提交的修改，而且，工作目录是干净（working tree clean）的。

1. 版本回退

再次修改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 349f7ca] append GPL

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

像这样，你不断对文件进行修改，然后不断提交修改到版本库里，就好比玩RPG游戏时，每通过一关就会自动把游戏状态存盘，如果某一关没过去，你还可以选择读取前一关的状态。有些时候，在打Boss之前，你会手动存盘，以便万一打Boss失败了，可以从最近的地方重新开始。Git也是一样，每当你觉得文件修改到一定程度的时候，就可以“保存一个快照”，这个快照在Git中被称为commit。一旦你把文件改乱了，或者误删了文件，还可以从最近的一个commit恢复，然后继续工作，而不是把几个月的工作成果全部丢失。

现在，我们回顾一下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.

在Git中，我们用git log命令查看历史记录：

$ git log

commit 349f7ca7548dfcb996047992d7256510b4cc00bc (HEAD -> master)

Author: xjz <xjzccnu@163.com>

Date: Thu Mar 11 17:30:34 2021 +0800

append GPL

commit ab7d5c51f098047c268ccd980aa77926f65a2191

Author: xjz <xjzccnu@163.com>

Date: Thu Mar 11 17:29:22 2021 +0800

add distributed

commit ced2b11228dc7eb5fe56bb0c06c6952cdd2e76e2

Author: xjz <xjzccnu@163.com>

Date: Thu Mar 11 17:13:33 2021 +0800

wrote a readme file

如果嫌输出信息太多，看得眼花缭乱的，可以试试加上--pretty=oneline参数。

需要友情提示的是，你看到的一大串类似349f7ca...的是commit id（版本号），Git的commit id不是1，2，3……递增的数字，而是一个SHA1计算出来的一个非常大的数字，用十六进制表示，而且你看到的commit id和我的肯定不一样，以你自己的为准。为什么commit id需要用这么一大串数字表示呢？因为Git是分布式的版本控制系统，后面我们还要多人在同一个版本库里工作，如果大家都用1，2，3……作为版本号，那肯定就冲突了。

现在我们启动时光穿梭机，准备把readme.txt回退到上一个版本，也就是add distributed的那个版本，怎么做呢？

首先，Git必须知道当前版本是哪个版本，在Git中，用HEAD表示当前版本，也就是最新的提交349f7ca...（注意我的提交ID和你的肯定不一样），上一个版本就是HEAD^，上上一个版本就是HEAD^^，当然往上100个版本写100个^比较容易数不过来，所以写成HEAD~100。

现在，我们要把当前版本append GPL回退到上一个版本add distributed，就可以使用git reset命令：

$ git reset --hard HEAD^

看看readme.txt的内容是不是版本add distributed：

$ cat readme.txt

还可以继续回退到上一个版本wrote a readme file，不过且慢，让我们用git log再看看现在版本库的状态，最新的那个版本append GPL已经看不到了！

只要上面的命令行窗口还没有被关掉，你就可以顺着往上找啊找啊，找到那个append GPL的commit id是349f7ca...，于是就可以指定回到未来的某个版本：

git reset –hard 349f7ca

版本号没必要写全，前几位就可以了，Git会自动去找。当然也不能只写前一两位，因为Git可能会找到多个版本号，就无法确定是哪一个了。

再小心翼翼地看看readme.txt的内容：

$ cat readme.txt

append GPL版本又回来了。

1. 删除文件

在Git中，删除也是一个修改操作，我们实战一下，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"

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

这个时候，Git知道你删除了文件，因此，工作区和版本库就不一致了，git status命令会立刻告诉你哪些文件被删除了。

现在你有两个选择，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

$ git commit -m "remove test.txt"

另一种情况是删错了，因为版本库里还有呢，所以可以很轻松地把误删的文件恢复到最新版本：

$ git checkout -- test.txt

1. 远程仓库

Git是分布式版本控制系统，同一个Git仓库，可以分布到不同的机器上。怎么分布呢？最早，肯定只有一台机器有一个原始版本库，此后，别的机器可以“克隆”这个原始版本库，而且每台机器的版本库其实都是一样的，并没有主次之分。可以自己搭建一台运行Git的服务器，不过现阶段，为了学Git先搭个服务器绝对是小题大作。好在这个世界上有个叫GitHub的神奇的网站，从名字就可以看出，这个网站就是提供Git仓库托管服务的，所以，只要注册一个GitHub账号，就可以免费获得Git远程仓库。因为国内连接GitHub较慢，可以使用Gitee网站（https://gitee.com/）

1. 注册Gitee账号（自行注册）
2. 创建SSH Key

在用户主目录下，看看有没有.ssh目录，如果有，再看看这个目录下有没有id\_rsa和id\_rsa.pub这两个文件，如果已经有了，可直接跳到下一步。

$ ssh-keygen -t rsa -C [youremail@example.com](mailto:youremail@example.com)

你需要把邮件地址换成你自己的邮件地址，然后一路回车，使用默认值即可

如果一切顺利的话，可以在用户主目录里找到.ssh目录，里面有id\_rsa和id\_rsa.pub两个文件，这两个就是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。

在Gitee上注册账号并登录后，需要先上传自己的SSH公钥。选择右上角用户头像 -> 菜单“修改资料”，然后选择“SSH公钥”，填写一个便于识别的标题，然后把用户主目录下的.ssh/id\_rsa.pub文件的内容粘贴进去：



1. 创建远程仓库

在Gitee网站点击“我的 – 仓库 – +”创建一个远程仓库，项目名称最好与本地库保持一致。



然后，我们在本地库上使用命令git remote add把它和Gitee的远程库关联：

git remote add origin [git@gitee.com:xjzccnu/learngit.git](mailto:git@gitee.com:xjzccnu/learngit.git)

请注意把上面的用户名替换成你自己的账户名。

添加后，远程库的名字就是origin，这是Git默认的叫法，也可以改成别的，但是origin这个名字一看就知道是远程库。

下一步，就可以把本地库的所有内容推送到远程库上：

$ git push -u origin master

可以看到本地库的内容已经被推送到远程库



从现在起，只要本地作了提交，就可以通过命令：

$ git push origin master

把本地master分支的最新修改推送至Gitee，现在，你就拥有了真正的分布式版本库！

也可以把其他人提交到Gitee拉到本地。

创建开源许可，任意选择一种，会发现远程库中多了一个LICENSE文件。



输入命令：

$git pull origin master

会发现在本地的learngit文件夹中也出现了LICENSE文件。

1. 远程克隆

之前介绍了先有本地仓库，创建远程仓库与之关联的方法。另一种更常见的做法是先创建远程仓库，第一次使用前将其克隆到本地，创建本地关联仓库。

登录Gitee，创建一个新的仓库，名字叫gitskills，选中“使用Readme文件初始化这个仓库”。



退出learngit目录，回到用户主目录

$ cd ..

从远程仓库克隆数据

$ git clone [git@gitee.com:xjzccnu/gitskills.git](mailto:git@gitee.com:xjzccnu/gitskills.git)

可以看到用户主目录下多了gitskills文件夹和Readme.md文件。

1. 根据上面的操作，自己练习：
2. 创建一个Java/C/C++的代码文件，开始输出Hello World入库，然后改为其他内容入库，再回退到Hello Word
3. 在Gitee上申请一个账号，创建一个代码仓库，并且将本地仓库与远程仓库关联。

描述你的实验过程并截图

1. 实验过程与结果

四、小结