开源项目阅读与管理—版本管理

二、开源项目阅读与管理-版本管理

版本管理

步骤 1： 创建版本库

1、版本库（repository）

又名版本仓库，英文名，它就像是一个目录，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”。

2、创建版本库

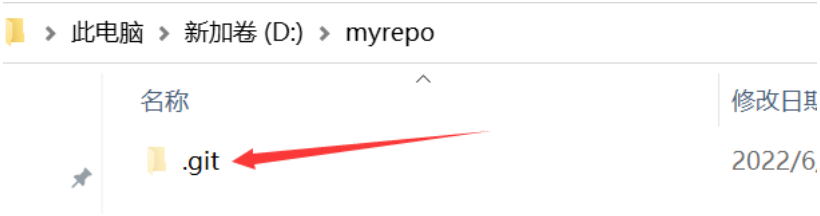
在我的电脑 D 盘下，创建一个名为 myrepo 的空目录，在里面点击鼠标右键，点击菜单里的 Git Bash Here：

输入 pwd 命令，可以显示当前目录为 /d/myrepo

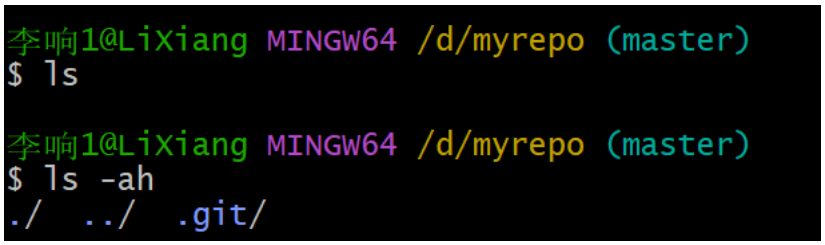
注意：使用 Windows 系统，为了避免遇到各种莫名其妙的问题，请确保目录名（包括父目录）不包含中文。

接下来，我们通过 git init 命令把这个目录变成 Git 可以管理的仓库：

输入完命令，Git 把仓库建好了，并且提示初始化了一个空的仓库（empty Git repository），在 myrepo 目录下多了一个.git 的目录，这个目录是 Git 来跟踪管理版本库的，请注意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。



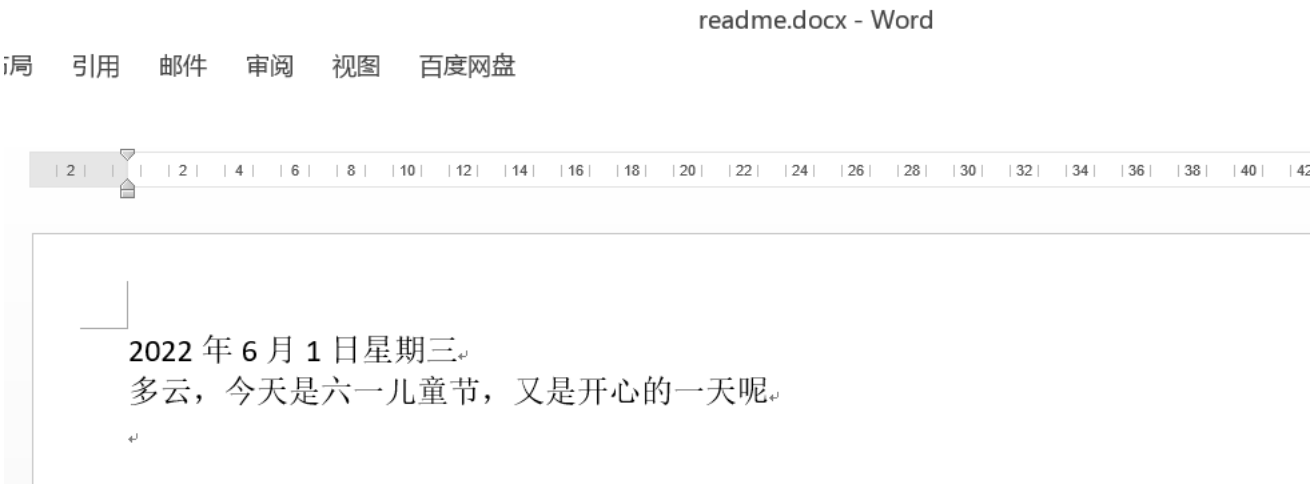
如果你没有看到.git 目录，那是因为这个目录默认是隐藏的，在 windows 下请显示隐藏文件，在 Git Bash 里用 ls -ah 命令就可以看见。



注意：不是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。不过不建议使用实际的项目来学习 Git，初学者不熟悉操作，有造成文件遗漏丢失的风险。

3、把文件添加到版本库

在 D 盘的 myrepo 文件夹下，创建一个 readme.docx 文件，内容如下：



一定要放到 myrepo 目录下（子目录也行），因为这是一个 Git 仓库，放到其他地方 Git 找不到这个文件。

接下来，使用 git add 命令，将 readme.docx 文件添加到 Git 仓库

```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git add readme.docx
```

执行完上面的命令，没有任何显示，代表成功添加 readme.docx 文档

接着用命令 git commit 告诉 Git，把文件提交到仓库：

```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git commit -m "create a readme word file"
[master (root-commit) e2b2795] create a readme word file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 readme.docx
```

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。

git commit 命令执行成功后会告诉你，1 file changed：1 个文件被改动（我们新添加的 readme.doc 文件）；0 insertions 0 deleteions 没有增加或者删除。

Git 添加文件需要先 add 再 commit，你可以先多次 add 不同的文件，再 commit 提交很多文件。比如：

$ git add file1.txt

$ git add file2.txt file3.txt

$ git commit -m "add 3 files."

疑难解答

Q：输入 git add readme.docx，得到错误：fatal: not a git repository (or any of the parent directories)。

A：Git 命令必须在 Git 仓库目录内执行（git init 除外），在仓库目录外执行是没有意义的。

Q：输入 git add readme.docx，得到错误 fatal: pathspec 'readme.docx' did not match any files。

A：添加某个文件时，该文件必须在当前目录下存在，用 ls 或者 dir 命令查看当前目录的文件，看看文件是否存在，或者是否写错了文件名。

小结：

初始化一个 Git 仓库，使用 git init 命令。

添加文件到 Git 仓库，分两步：

- 使用命令 git add <file>，注意，可反复多次使用，添加多个文件；
- 使用命令 git commit -m <message>，完成。

注意：我们的 git 目前只在本地，还没有远程的分支 branch。你只是在本地初始化了一个 Git 仓库，但是你没有将其提交到 GitHub 上。

4、更新文件并提交

readme.docx 文档中有了新内容，我们需要把修改提交到 Git 版本库，现在，再练习一次，修改 readme.docx 文件如下：

执行 git add 命令，将更新后的 readme.docx 文件添加到 Git 仓库

```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git add readme.docx
```

执行 git commit 命令，把文件提交到仓库：

```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git commit -m "add something"
[master 4e29439] add something
 1 file changed, 0 insertions(+), 0 deletions(-)
```

5、再次更新并提交

readme.docx 文档中有了新内容，我们需要把修改提交到 Git 版本库，现在，再练习一次，修改 readme.docx 文件如下：

执行 git add 命令，将更新后的 readme.docx 文件添加到 Git 仓库，接下来，执行 git commit 命令，把文件提交到仓库：

```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git add readme.docx

李响1@LiXiang MINGW64 /d/myrepo (master)
$ git commit -m "The Dragon Boat Festival"
[master eb85296] The Dragon Boat Festival
 1 file changed, 0 insertions(+), 0 deletions(-)
```

步骤 2：版本查看

我们可以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，我们再次修改 readme.docx，

并执行 git add 和 git commit。

```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git add readme.docx

李响1@LiXiang MINGW64 /d/myrepo (master)
$ git commit -m "modify the word"
[master 22a8b87] modify the word
 1 file changed, 0 insertions(+), 0 deletions(-)
```

我们回顾一下 readme.docx 文件一共有几个版本被提交到 Git 仓库里了：

一共 4 个版本。

1、使用 git log 命令

在实际工作中，我们脑子里怎么可能记得一个几千行的文件每次都改了什么内容，不然要版本控制系统干什么。版本控制系统肯定有某个命令可以告诉我们历史记录，在 Git 中，我们用 git log 命令查看：

```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git log
commit 22a8b87443334335dd37e510029c2ce7e56886f8 (HEAD -> master)
Author: lixiang <lixiangood@qq.com>
Date:   Wed Jun 1 15:12:19 2022 +0800

    modify the word

commit eb85296191ce53391a87f15d30429f6b92f0b487
Author: lixiang <lixiangood@qq.com>
Date:   Wed Jun 1 14:50:26 2022 +0800

    The Dragon Boat Festival

commit 4e294398c865b201b0c41ae26ee8e720ae984265
Author: lixiang <lixiangood@qq.com>
Date:   Wed Jun 1 11:30:27 2022 +0800

    add something

commit e2b27950d271e0661520bc9dd498664258b52576
Author: lixiang <lixiangood@qq.com>
Date:   Wed Jun 1 11:18:32 2022 +0800

    create a readme word file
```

git log 命令显示从最近到最远的提交日志，我们可以看到 4 次提交，如果嫌输出信息太多，看得眼花缭乱的，可以试试加上参数 git log --pretty=oneline 。

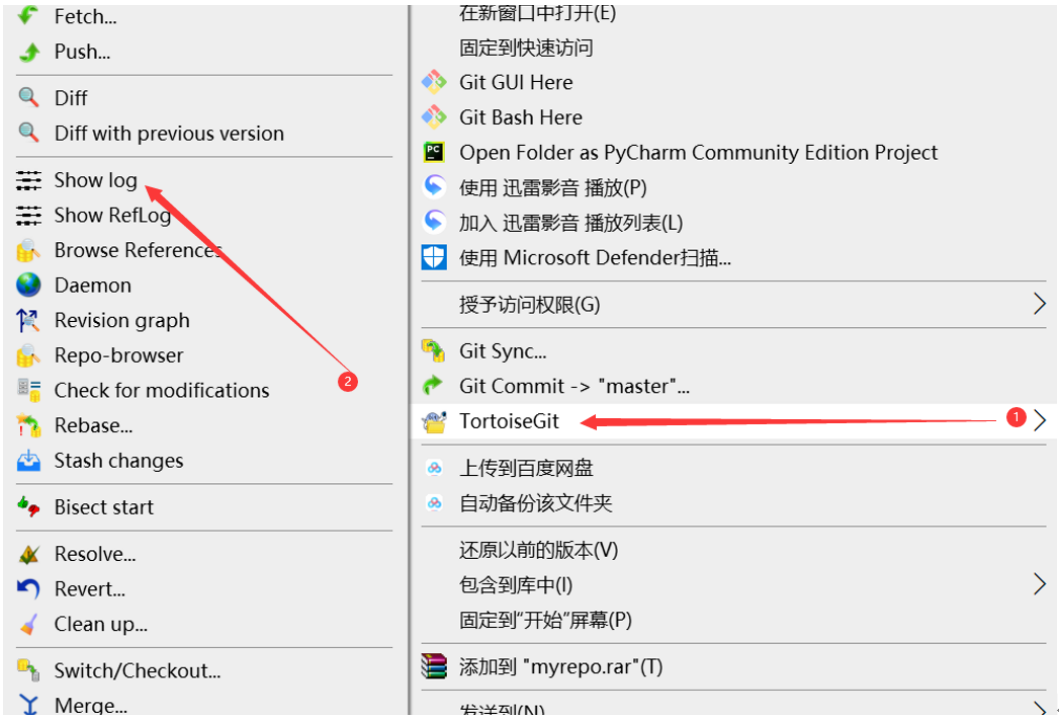
```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git log --pretty=oneline
22a8b87443334335dd37e510029c2ce7e56886f8 (HEAD -> master) modify the word
eb85296191ce53391a87f15d30429f6b92f0b487 The Dragon Boat Festival
4e294398c865b201b0c41ae26ee8e720ae984265 add something
e2b27950d271e0661520bc9dd498664258b52576 create a readme word file
```

你看到的一大串类似 22a8b8...的是 commit id（版本号），和 SVN 不一样，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，后面我们还要研究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。

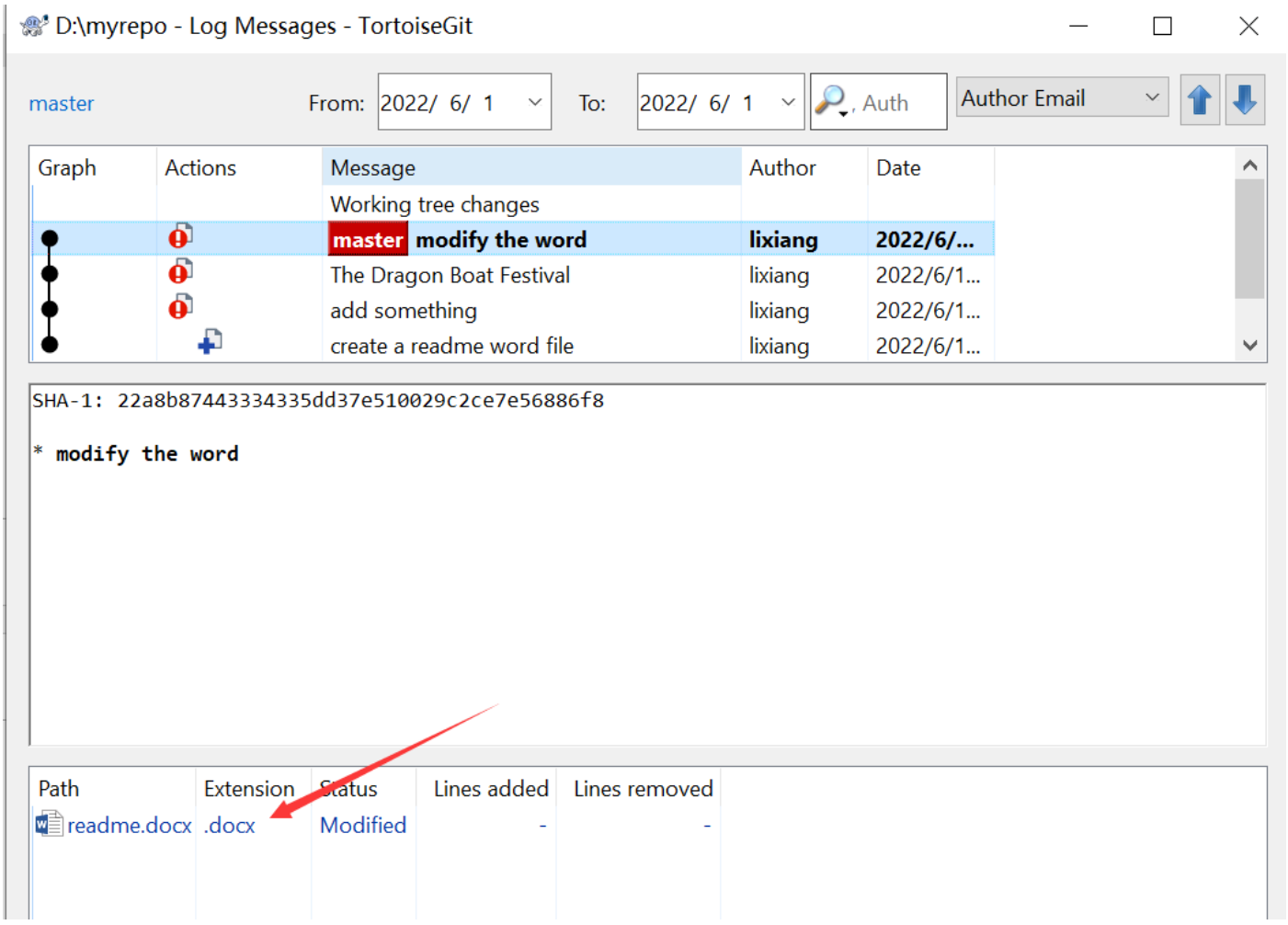
每提交一个新版本，实际上 Git 就会把它们自动串成一条时间线。如果使用可视化工具查看 Git 历史，就可以更清楚地看到提交历史的时间线：git-log-time

2、使用可视化工具 TortoiseGit

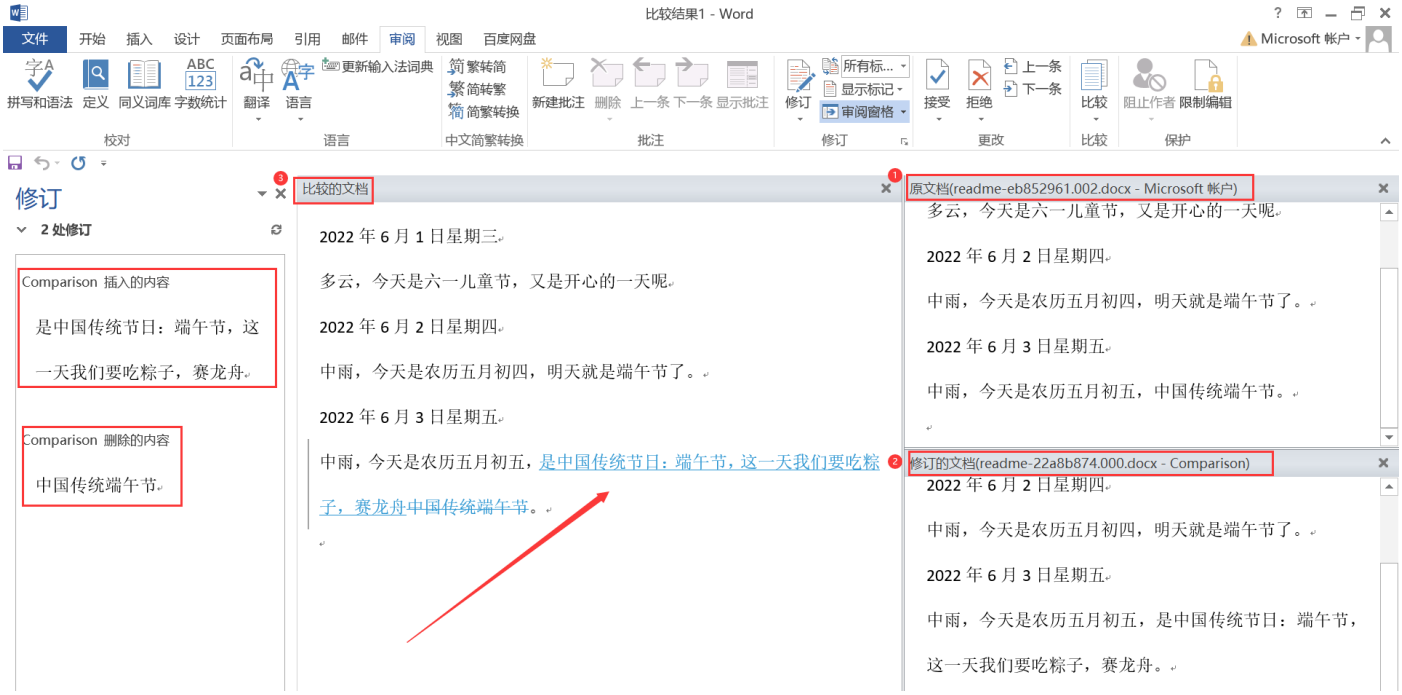
在 myrepo 文件夹上点击鼠标右键，选择“TortoiseGit”—“Show log”



在弹出的界面中，我们可以以图形化的方式看到 readme.docx 文件的所有版本和提交说明：



点击下面箭头的 readme.docx 文件，可以弹出各个版本的比较结果



步骤 3：版本回退

Git 的版本回退，需要启动时光机，把 readme.docx 回退到上一个版本，也就是 The Dragon Boat Festival 的这个版本。

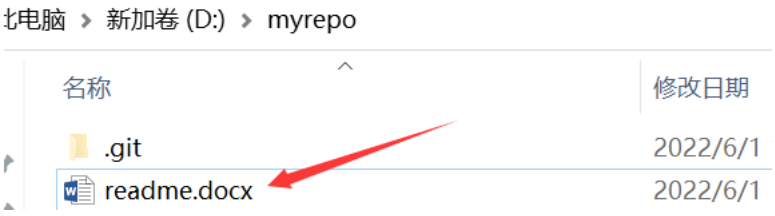
```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git log --pretty=oneline
22a8b87443334335dd37e510029c2ce7e56886f8 (HEAD -> master) modify the word
eb85296191ce53391a87f15d30429f6b92f0b487 The Dragon Boat Festival
4e294398c865b201b0c41ae26ee8e720ae984265 add something
e2b27950d271e0661520bc9dd498664258b52576 create a readme word file
```

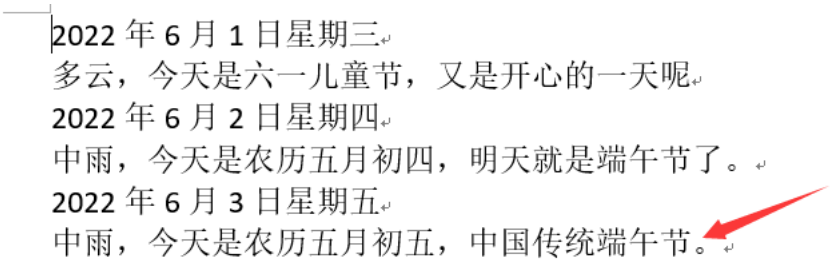
首先，Git 必须知道当前版本是哪个版本，在 Git 中，用 HEAD 表示当前版本，也就是最新的提交 eb85296...（注意我的提交 ID 和你的肯定不一样），上一个版本就是 HEAD^，上上一个版本就是 HEAD^^，当然往上 100 个版本写 100 个^比较容易数不过来，所以写成 HEAD~100。

现在，我们要把当前版本 **modify the word** 回退到上一个版本 **The Dragon Boat Festival**，就可以使用 git reset 命令：git reset --hard HEAD^

```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git reset --hard HEAD^
HEAD is now at eb85296 The Dragon Boat Festival
```

这时，readme.docx 文件已经回退到了上一个版本，我们打开文件验证一下：





文件已经被还原到了上一个版本 The Dragon Boat Festival

使用 git log --pretty=oneline 再次查看版本：

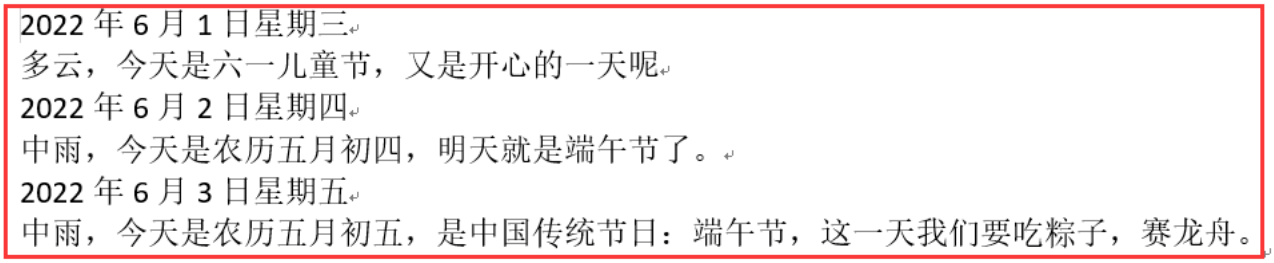
```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git log --pretty=oneline
eb85296191ce53391a87f15d30429f6b92f0b487 (HEAD -> master) The Dragon Boat Festiv
al
4e294398c865b201b0c41ae26ee8e720ae984265 add something
e2b27950d271e0661520bc9dd498664258b52576 create a readme word file
```

最新的那个版本已经看不到了！好比你从 21 世纪坐时光穿梭机来到了 19 世纪，想再回去已经回不去了，怎么办？

办法其实还是有的，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，找到最后那个的commit id是22a8b8...，于是就可以执行命令 **git reset --hard 22a8b8** 指定回到最后那个版本：

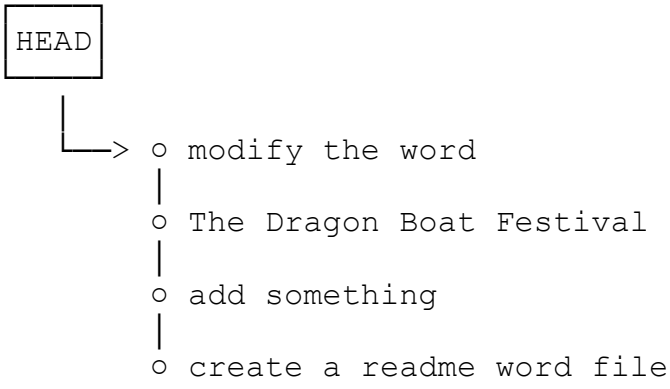
```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git reset --hard 22a8b8
HEAD is now at 22a8b87 modify the word
```

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。我们再打开看 readme.docx 的内容：

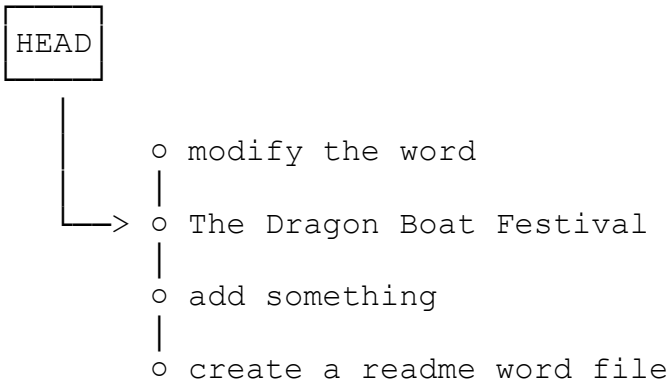


word 文档中 6 月 3 日的内容又回来了！！！

Git 的版本回退速度非常快，因为 Git 在内部有个指向当前版本的 HEAD 指针，当你回退版本的时候，Git 仅仅是把 HEAD 指向 22a8b87443334335dd37e510029c2ce7e56886f8 这个 commit id。



改为指向 The Dragon Boat Festival：



同事把 Git 仓库中的文件也更新了。所以你让 HEAD 指向哪个版本号，你就把当前版本定位在哪。现在，你回退到了某个版本，关掉了电脑，第二天早上就后悔了，想恢复到新版本怎么办？找不到新版本的 commit id 怎么办？

在 Git 中，总是有后悔药可以吃的。当你用$ git reset --hard HEAD^回退到上一个版本时，再想恢复到最后一个版本，就必须找到最后一个版本的 commit id。Git 提供了一个命令 git reflog 用来记录你的每一次命令：

```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git reflog
22a8b87 (HEAD -> master) HEAD@{0}: reset: moving to 22a8b8
eb85296 HEAD@{1}: reset: moving to HEAD^
22a8b87 (HEAD -> master) HEAD@{2}: commit: modify the word
eb85296 HEAD@{3}: commit: The Dragon Boat Festival
4e29439 HEAD@{4}: commit: add something
e2b2795 HEAD@{5}: commit (initial): create a readme word file
```

从记录执行的命令可以看到，modify the word 的 commit id 是 22a8b87，现在，你又可以乘坐时光机回到未来了。

总结：

- HEAD 指向的版本就是当前版本，因此，Git 允许我们在版本的历史之间穿梭，使用命令 git reset --hard commit_id。
- 穿梭前，用 git log 可以查看提交历史，以便确定要回退到哪个版本。
- 要重返未来，用 git reflog 查看命令历史，以便确定要回到未来的哪个版本。

步骤 4：练习

按照上面的步骤，自己建立一个个人简历的 resume.docx 文档，每次修改以后将简历提交到 Git 库中。