学习参照：廖雪峰的Git教程

> 注：本笔记仅仅截取重要内容

Git简介

创建版本库

使用Windows的童鞋要特别注意：

千万不要使用Windows自带的**记事本**编辑任何文本文件。原因是Microsoft开发记事本的团队使用了一个非常弱智的行为来保存UTF-8编码的文件，他们自作聪明地在每个文件开头添加了0xefbbbf（十六进制）的字符，你会遇到很多不可思议的问题，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，都是由记事本的弱智行为带来的。建议你下载Visual Studio Code代替记事本，不但功能强大，而且免费！

把一个文件放到Git

第一步，用命令`git add`告诉Git，把文件添加到仓库：

```
$ git add readme.txt
```

执行上面的命令，没有任何显示，这就对了，Unix的哲学是“没有消息就是好消息”，说明添加成功。

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"
[master 4f5b01f] wrote a readme file
 1 file changed, 5 insertions(+)
 create mode 100644 readme.txt
```

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。

时光机穿梭

```
$ git diff readme.txt
diff --git a/readme.txt b/readme.txt
index 5bd0573..0a14813 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1,5 +1,5 @@
-Git is a version control system.
+Git is a distributed version control system.
 Git is free software.


-    --Edited by LRL52 on Jan 19th 2022
\ No newline at end of file
+    --Edited by LRL52 on Jan 19th 2022
```

`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。

版本回退

`git log`命令显示从最近到最远的提交日志

```
$ git log
commit 6f462d544bd68fa294ceea7c212201c905e6e2d7 (HEAD -> master)
Author: LRL52 <525687841@qq.com>
Date:   Wed Jan 19 18:05:51 2022 +0800

    append GPL

commit 4ebe5f7dd1ddb9828425714d4a529fb98f85acb8
Author: LRL52 <525687841@qq.com>
Date:   Wed Jan 19 18:01:32 2022 +0800

    add distributed

commit 4f5b01fcf4de737f117db76f4b6778ced5192211
Author: LRL52 <525687841@qq.com>
Date:   Wed Jan 19 17:50:16 2022 +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

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`1094adb...`（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`。

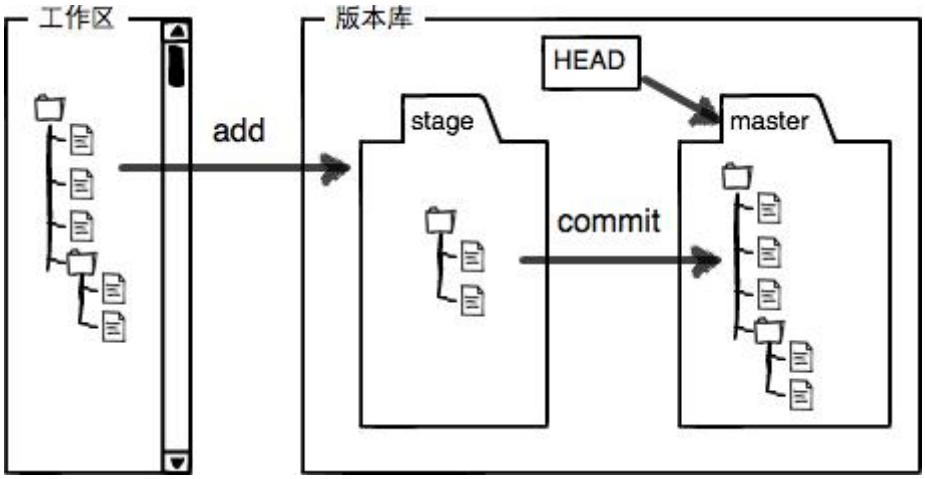
```
┌────┐
│HEAD│
└────┘
   │
   │    ○ append GPL
   │    │
   └──> ○ add distributed
        │
        ○ wrote a readme file
```

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 --hard commit_id`命令

Git提供了一个命令`git reflog`用来记录你的每一次命令

```
$ git reflog
6f462d5 (HEAD -> master) HEAD@{0}: reset: moving to 6f46
4ebe5f7 HEAD@{1}: reset: moving to HEAD^
6f462d5 (HEAD -> master) HEAD@{2}: commit: append GPL
4ebe5f7 HEAD@{3}: commit: add distributed
4f5b01f HEAD@{4}: commit: wrote a readme file
3d71e0a (origin/master, origin/HEAD) HEAD@{5}: commit: 4.txt
df67be7 HEAD@{6}: commit: test
14e2198 HEAD@{7}: clone: from https://gitee.com/lrl52/git-study.git
```

工作区和暂存区



Git为我们自动创建的第一个分支`master`，以及指向`master`的一个指针叫`HEAD`。

`git add`命令实际上就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 diff 几种用法的比较

1. 缓存区↔ 库(HEAD)：`git diff --cached`
2. 工作区↔ 缓存区：`git diff`
3. 工作区↔ 库(HEAD)：`git diff HEAD -- filename`
4. 库↔ 库：`git diff 243550a 24bc01b`

下面中的stage.txt已经被add到缓存区，但还未commit到库

```
LENOVO@LAPTOP-V351JDD8 MINGW64 /d/LRL52/gitstudy (master)
$ git diff --cached
diff --git a/stage.txt b/stage.txt
new file mode 100644
index 0000000..f87b8ac
--- /dev/null
+++ b/stage.txt
@@ -0,0 +1 @@
+Fisrt modified!
```

总结：diff参数后啥都没有就是工作区和缓存区比较，加了库的`commit id`或者`HEAD HEAD^`就是工作区和对应的库比较，加了`--cached`才是缓存区和库比较

管理修改

（没啥新内容

撤销修改

`git checkout -- file`可以丢弃工作区的修改：

命令`git checkout -- readme.txt`意思就是，把`readme.txt`文件在工作区的修改全部撤销，这里有两种情况：

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；

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。

总之，就是让这个文件回到最近一次`git commit`或`git add`时的状态。

```
LENOVO@LAPTOP-V351JDD8 MINGW64 /d/LRL52/gitstudy (master)
$ cat stage.txt
Fisrt modified!
Second modified!

LENOVO@LAPTOP-V351JDD8 MINGW64 /d/LRL52/gitstudy (master)
$ git checkout -- stage.txt

LENOVO@LAPTOP-V351JDD8 MINGW64 /d/LRL52/gitstudy (master)
$ cat stage.txt
Fisrt modified!
```

Git同样告诉我们，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：

（注意这个命令是清除缓存区的file）

小结

又到了小结时间。

场景1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令`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。

删除文件

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：

```
$ 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")
```

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
rm 'test.txt'

$ git commit -m "remove test.txt"
[master d46f35e] 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。

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
```

小结

命令`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。**

远程仓库

```
#查看系统config
git config --system --list

#查看当前用户（global）配置
git config --global  --list

#设置用户名与邮箱
git config --global user.name "kuangshen"  #名称
git config --global user.email 24736743@qq.com   #邮箱
```

要关联一个远程库，使用命令`git remote add origin git@server-name:path/repo-name.git`；

关联一个远程库时必须给远程库指定一个名字，`origin`是默认习惯命名；

关联后，使用命令`git push -u origin master`第一次推送master分支的所有内容；

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

远程仓库有本地没有的东西时：

```
LENOVO@LAPTOP-V351JDD8 MINGW64 /d/LRL52/gitstudy (master)
$ git push origin master
To github.com:LRL52/gitskills.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'github.com:LRL52/gitskills.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

LENOVO@LAPTOP-V351JDD8 MINGW64 /d/LRL52/gitstudy (master)
$ git pull --rebase origin master
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (2/2), 595 bytes | 85.00 KiB/s, done.
From github.com:LRL52/gitskills
 * branch            master     -> FETCH_HEAD
   bcbc29a..6999276  master     -> origin/master
Successfully rebased and updated refs/heads/master.

LENOVO@LAPTOP-V351JDD8 MINGW64 /d/LRL52/gitstudy (master)
$ git push origin master
Everything up-to-date
```

分支管理

创建与合并分支

Git鼓励大量使用分支：

查看分支：`git branch`

创建分支：`git branch <name>`

切换分支：`git checkout <name>`或者`git switch <name>`

创建+切换分支：`git checkout -b <name>`或者`git switch -c <name>`

合并某分支到当前分支：`git merge <name>`

删除分支：`git branch -d <name>`