git 操作文件

https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/0013743858312764dca7ad6d0754f76aa562e3789478044000

提交文件

$ git add readme.txt　　//把文件添加到仓库（该文件一定要放到git仓库的目录下）

$ git commit -m "wrote a readme file"

//把文件提交到仓库。-m后面输入的是本次提交的说明，可以输入任意内容

```
[master (root-commit) cb926e7] wrote a readme file
 1 file changed, 2 insertions(+)
 create mode 100644 readme.txt
```

为什么Git添加文件需要add，commit两步：

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."
```

查看状态

$ git status　　//文件被修改后运行git status查看结果，该命令可以让我们时刻掌握仓库当前的状态

```
# 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")
```

//readme.txt被修改过了，但还没有准备提交的修改。

$ git diff readme.txt　//如果git status告诉你有文件被修改过，用git diff可以查看具体修改了什么内容

```
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 add readme.txt
$ git commit -m "add distributed"
[master ea34578] add distributed
 1 file changed, 1 insertion(+), 1 deletion(-)
```

//提交

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

//查看仓库的当前状态

//当前没有需要提交的修改，而且，工作目录是干净的。

查看历史提交纪录

$ git log　　//查看历史记录，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

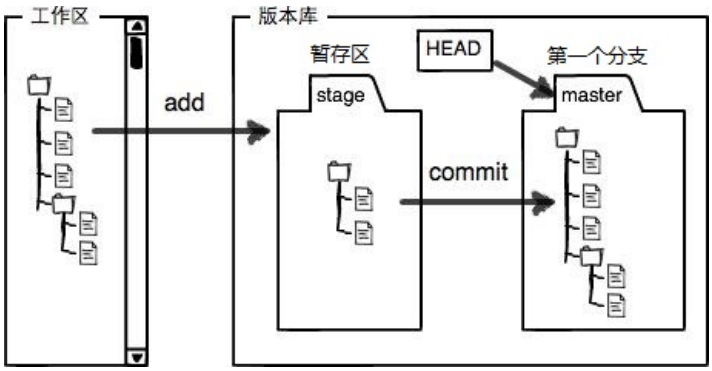
$ git log --pretty=oneline　　//如果嫌输出信息太多，可以加上--pretty=oneline参数，使版本信息只能在一行中显示
3628164fb26d48395383f8f31179f24e0882e1e0 append GPL
ea34578d5496d7dd233c827ed32a8cd576c5ee85 add distributed
cb926e7ea50ad11b8f9e909c05226233bf755030 wrote a readme file
//一大串类似3628164...882e1e0的是commit id（版本号）

版本回退
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，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。
$ git reset --hard HEAD^　//回退到上一个版本
HEAD is now at ea34578 add distributed

版本恢复
$ git reset --hard 3628164　　//用commit id恢复到未来的某个版本，只要命令行窗口没关，就可以往上找到那个版本的commit id，id不用写全
HEAD is now at 3628164 append GPL

$ git reflog　　//若命令行窗口已关找不到要恢复版本的commit id，可以查看命令历史，以便确定要回到未来的哪个版本
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

工作区和暂存区
工作区
就是你在电脑里能看到的目录，比如我的learngit文件夹



版本库

工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。

暂存区

存在在Git的版本库中，称为stage（或者叫index）暂存区，还有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HEAD。

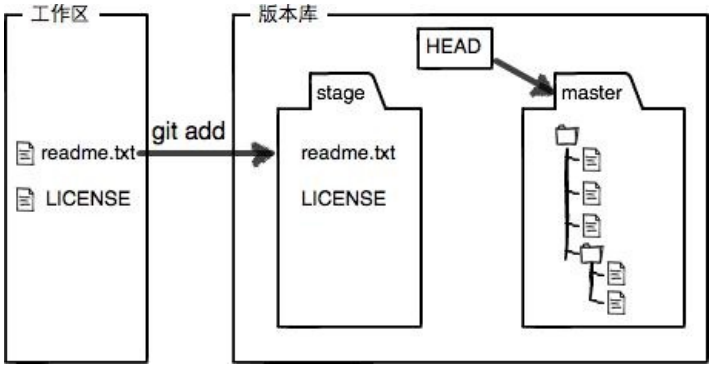
提交命令：

git add 把文件修改添加到暂存区。

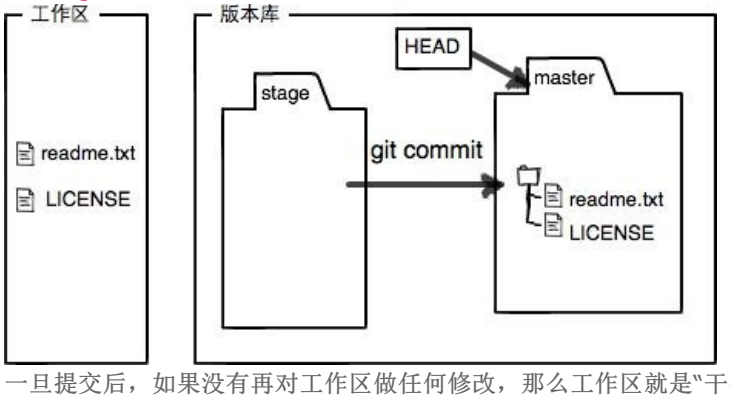
git commit 把暂存区的所有内容提交到当前分支。

在工作区对readme.txt做个修改，然后新增一个LICENSE文本文件。

使用两次命令git add，把readme.txt和LICENSE都添加后，暂存区的状态就变成这样了：



再执行git commit，就可以一次性把暂存区的所有修改提交到分支：



一旦提交后，如果没有再对工作区做任何修改，那么工作区就是“干净”的。

管理修改

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。

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

结果：第一次的修改被提交了，第二次的修改不会被提交。

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.
```

撤销修改

只修改了文件 readme.txt：

```
$ git checkout -- readme.txt        //把 readme.txt 文件在工作区的修改全部撤销
```

修改了文件还git add到暂存区：

```
$ git reset HEAD readme.txt     //把暂存区的修改回退到工作区，HEAD表示最新的版本
$ git checkout -- readme.txt      //丢弃工作区的修改
```

修改了文件还git commit到了版本库：

```
$ git reset --hard HEAD^         //版本回退
```

删除/恢复文件

```
$ git rm test.txt       //从版本库中删除文件
$ git checkout -- test.txt      //把误删文件的最新版本恢复到工作区
```

//git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。