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：

* 第一步，使用命令git add <file>，注意，可反复多次使用，添加多个文件；
* 第二步，使用命令git commit –m message，完成。

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 <file>这个命令看看.

小结

* 要随时掌握工作区的状态，使用git status命令。
* 如果git status告诉你有文件被修改过，用git diff可以查看修改内容。

现在总结一下：

* 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。$ git re**set** --hard HEAD^
* 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。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：
* 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。在Git中，总是有后悔药可以吃的。当你用$ git reset --hard HEAD^回退到add distributed版本时，再想恢复到append GPL，就必须找到append GPL的commit id。Git提供了一个命令git reflog用来记录你的每一次命令：

Merge:

Git checkout release

Git pull release

Git status

Git checkout dev-20170523

Git merge release

Git status

Git push

多人协同工作之分支管理：

创建分支：git branch testing

Git log –oneline –decorate :--decorate可以使用git log命令查看各个分支当前所指的对象

Git checkout testing 分支切换

Git checkout –b testing 创建并切换分支

Git branch –d testing 删除分支

Git log –graph –pretty=oneline –abbrev-commit 看分支的合并情况

Git merge –no-ff –m “merge with no-ff”dev 准备合并dev分支到当前分支，--no-ff参数表示禁用Fast forward模式，所以合并后会产生一个新的commit，所以加上-m参数，把commit描述写进去

Git remote 查看远程库的信息

Git remote –v 显示更详细的信息

Git push origin dev推送当前本地分支到远程dev分支

Git fetch origin 同步远程服务器上的数据

多人协作的工作模式通常是这样的：

* 1. 首先，可以试图用git push origin branch-name推送自己的修改；
  2. 如果推送失败，则因为远程分支比你的本地更新，需要先用git pull试图合并
  3. 如果合并有冲突，则解决冲突，并在本地提交
  4. 没有冲突或解决了冲突后，再用git push origin branch-name推送就能成功

Git pull 提示”no tracking information”,则说明本地分支和远程分支的链接关系没有创建，用命令git branch –set-upstream-to branch-name origin/branch-name

提交错了想撤销或者回退版本：

Git checkout – [file]/. 撤销操作 没有提交到暂存区的情况下，即没有git add

Git reset HEAD [file] 撤销操作 文件在工作修改了，并且也执行了git add提交给暂存区了，

Git checkout HEAD [file] 是git checkout – [file] 和 git reset HEAD的合成体，直接用HEAD覆盖工作区，暂存区。

版本回退以及切换

Git reset –hard HEAD^ 回退到上个版本

Git reset –hard HEAD~100 回退到往上100个版本

Git reset –hard commit\_id

删除操作

Git rm 执行它会同时删除工作区和暂存区中的指定文件，要慎重处理

意外删除恢复：

Git checkout HEAD [file] 还未执行git commit提交到HEAD

Git reset –hard HEAD^ 执行commit之后，只能回退上一个版本

git stash用于保存和恢复工作进度

git stash

保存当前的工作进度。会分别对暂存区和工作区的状态进行保存

git stash save "message..."

这条命令实际上是第一条 git stash 命令的完整版

git stash list

显示进度列表。此命令显然暗示了git stash 可以多次保存工作进度，并用在恢复时候进行选择

git stash pop [--index] [<stash>]

如果不使用任何参数，会恢复最新保存的工作进度，并将恢复的工作进度从存储的工作进度列表中清除。

如果提供参数（来自 git stash list 显示的列表），则从该 <stash> 中恢复。恢复完毕也将从进度列表中删除<stash>。

选项--index 除了恢复工作区的文件外，还尝试恢复暂存区。

git stash apply [--index] [<stash>]

除了不删除恢复的进度之外，其余和 git stash pop 命令一样

git stash clear

删除所有存储的进度