是什么：版本控制系统，开发软件中记录每个开发者提交修改代码的情况等
工作区：当前目录
暂存区：在.git隐藏文件中
分支和指向分支的指针文件提交：working space(工作区)--working tree (暂存区)--git repository(git仓库)

git常用命令
git add .
git diff
(1)
无参数：看已经修改还没*add的文件的修改内容（2）参数--staged：看暂存区的文件和上一个版本文件的差异

git commit -m
git status查看暂存区，工作区，仓储区状态
以前commit过的文件必须先add到暂存区才能commit
可以添加多个文件到暂存区一起commit
恢复以前版本
git reset到指定版本，之前之后都可以，回到过去与，穿梭未来，只需要知道commit id
参数--hard HEADˆ(~n)
还可以指定返回到之前某个版本，参数为commit后地址前4位
git log查看当前版本库更新记录
查看后退出需要按q
git reflog检查回退
（1）找到删除版本的commit id(所以不要轻易clear)
（2）找不到id 用git reflog查看所有commit和reset记录，可以看到原来所有commit id
删除文件
git rm<file>从当前版本库删除文件再commit成为新版本，文件同时从repository和working space 中删除，如果要恢复，需要先reset到上一版本，再用git checkout --file
丢弃对该文件的修改(删除文件)
远程提交
git remote add 远端仓库名+远端仓库地址
git push -u 远端仓库名+master

分支管理

一个仓库（远程/本地）里有多个分支

master分支 ：稳定的一条线，记录每次提交

特点：代码安全且稳定，可以直接发布或被项目之外的人使用

其他分支：新特性测试和bug修改

新建分支

git branch 分支名 新建一个指针指向最近的一个commit

将操作转移到当前分支

git checkout 分支名将head指向新分支

指向新分支后，对工作区的修改都是在新分支上的修改

合并分支

（1）快速合并fast forward

git checkout master先将操作转移到master上

git merge 新分支名 将master指向新分支的最后一次提交，顺着新分支知道新分支最后一次提交

git branch -d 新分支名 在合并完成之后删除新分支

```
          A---B---C feature
         /         master
D---E---F
```

（2）--no-ff合并

创建一个新提交，和新分支最后一次提交相同

```
          A---B---C feature
         /         \
D---E---F-----------G master
```

```
  git merge --no-ff feature
```

远程仓库

添加远程仓库

```
  git remote add <shortname>（ 自定义的远程仓库名，用来代替远程仓库的url）
<url>
```

查看远程仓库

从远程仓库中抓取与拉取

就如刚才所见，从远程仓库中获得数据，可以执行：

`$ git fetch <remote>`这个命令会访问远程仓库，从中拉取所有你还没有的数据。

如果你使用 `clone` 命令克隆了一个仓库，命令会自动将其添加为远程仓库并默认以“origin” 为简写。

推送到远程仓库

`git push <remote> <branch>`。

远程仓库的重命名与移除