---下载后先联系到当前的git用户 配置名字和邮箱，这是git服务器上的姓名和邮箱跟github和gitee上的信息无关

shawn@shawn363 MINGW64 /

$ git config --global user.name wanxiao

shawn@shawn363 MINGW64 /

$ git config --global user.email [vito363@sina.com](mailto:vito363@sina.com)

Git config --global --edit 进入到编辑页面直接更改邮箱和姓名

Git config user.name 查看姓名

Git config user.email 查看邮箱

----建立文件夹，然后写文件

------查询当前的状态

$ git status

------添加文件到git版本控制里面

添加一个文件 $ git add 1.txt

添加多个文件 $ git add .

-----提交，及更改备注

$ git commit -m 'init'

工作区->暂存区->版本库

工作区（本地，此时查询状态颜色是红色）-》暂存区（add之后在这里，此时查询状态是new file绿色的,如果是更改后add的会显示modified也是绿色的）-》版本库（commit后在这里，此时查询状态nothing to commit, working tree clean说明提交完成）

------当每次更改时也必须git add 和git commit -m ‘’ 添加和提交一遍

------删除

$ git rm 2.txt

-----删除之后查询状态是绿色的，只提交即可commit ，如果是直接从文件夹里面删除的，查询状态时是红色的，需要先add再commit

3 files changed, 0 insertions(+), 0 deletions(-) 表示3个新创建，0个修改，0个删除

create mode 100644 2.txt

create mode 100644 3.txt

create mode 100644 4.txt

推到远程仓库，例如码云

可以先链接到远程仓库，下面是链接到了远程仓库，且命名为origin

$ git remote add origin <https://gitee.com/vitowan/test.git>

查询当前连接的是哪个远程地址

$ git remote -v

为远程仓库重命名

Git remote rename origin test

从远程仓库退出

Git remote rm origin

把本地内容推送到命名为origin的仓库链接，master意思版本，默认是master

$ git push origin master

克隆一个项目到demo目录里面，目录名不写会默认为原目录名字

$ git clone https://gitee.com/vitowan/test1.git demo

$ git clone https://gitee.com/vitowan/demo1.git ../demo2 //也可以用相对路径，上一级的demo2下面

Git中从远程的分支获取最新的版本到本地有这样2个命令：Git fetch 和 git pull

fetch：相当于是从远程获取最新版本到本地，不会自动merge

用法：git fetch origin master

Git log -p origin master //git diff origin master

Git merge origin master

以上含义：先从远程的origin的master主分支上下载最新的版本到本地分支，然后比较本地master分支和远程的差别，最后进行合并

Pull相当于从远程获得最新的版本并直接merge到本地：git pull origin master,相当于git fetch 和git merge

在实际开发中git fetch 更安全，因为在merge前，我们可查看更新情况，再考虑是否合并

如果想想加入到别人的开源项目，先fork过来后，再更改，然后pull request，之后pull的请求会到开源作者那边，通过后会合并到一块。 分支也可以整合到master里面，也是pull request的方法，把branch分支pull到master里面。反过来master里面也可pull到分支里面

日志查看和版本切换

Git log 显示详细的版本更改的日志信息

$ git log --pretty=oneline //每个版本更改的信息单行显示：显示有本版号和更改时写的备注信息

$ git reset --hard HEAD^^ //强制切换版本到第二个版本，HEAD代表当前最新/最近的版本，从最近的版本开始算的第二个 HEAD^^同理是第三个

$ git reset --hard fcaf //切换版本库也可以用版本号的前4位数进行区别切换到哪个

$ git reflog 显示的操作git的日志信息，不仅包含了版本更改信息，还包含了git操作信息，例如切换版本的操作，然后可以再用上面的版本切换方式进行切换还原

分支：

创建一个分支名字叫为wechat： $ git branch wechat 此时分支跟master原分支里面内容一样

查看本地分支:git branch ,显示如下

\* master

Wechat

查看本地连远程的分支： $ git branch -a 如下显示

master

\* wechat

remotes/demo1/master

型号对着master且颜色绿色说明正在编辑的分支是master的

转换分支：git checkout wechat, 此时查看分支 git branch，显示如下

master

\* wechat

此时编辑wechat分支里面的信息，master里面的文件不会变化

编辑好wechat分支的文件后，测试没问题后需合并到master主分支如下：

先进入到主分支：git checkout master

再合并操作：git merge wechat

此时如果需要另外一个分支，也可以再建另外一个例如ali git branch ali

编辑好之后同样同上方式可以提交，如果wechat分支的合并是在branch分支之前，且两个分支都修改了同一个文件，在提交后会提示冲突，此时进入冲突的文件，进行修改，删除提示符保留两者信息，或选择一个信息保留，然后add,commit

删除分支和恢复分支

如果需要删除的分支不是当前正在打开的分支，使用branch -d直接删除

Git branch -d ali

如果我们在试图删除一个分支时自己还没转移到另外的分支上，Git就会给出一个警告，并拒绝该删除操作。

如果坚持要删除该分支的话，就需要在命令中使用-D选项

Git branch -D ali //强制删除

以上删除都可以换成--delete 两条-

删除本地的远程分支：$ git branch -r --delete demo1/wechat ，可以先用git branch -a 查看所有的分支包括了本地和本地的远程分支，再对应的删除

删除git服务器上的分支：

$ git push demo1 --delete ali //删除了远程demo1上的ali分支

$ git push demo1 -d wechat //删除了wechat分支

恢复被删除的分支:

Git会自行负责分支的管理，所以当我们删除一个分支时，Git只是删除了指向相关提交的指针，但该提交对象依然会留在版本库中。

因此，如果我们知道删除分支时的散列值，就可以将某个删除的分支恢复过来。在已知提交的散列值的情况下恢复某个分支：

git branch <branch\_name> <hash\_val>

如果我们不知道想要恢复的分支的散列值，可以用reflog命令将它找出来。

**reflog命令**：

显示整个本地仓储的commit，包括所有branch的commit，甚至包括已经撤销的commit。

只要HEAD发生了变化， 就会在reflog里面看得到。

这时恢复分支a\_branch分支如下：

$ git branch ali HEAD@{10} //10是带有分支信息的最后一个

推到远程分支上：

可只推主分支，或者只推其它分支，或者挨个都推

先查下远程的是哪个：git remote -v //例如查询到的是demo1

先进入要推的分支: git checkout master //进入到了master分支

然后push: git push demo1 master //把本地的master 推到了远程的master上

如果想推其它分支到远程，跟上面方式一样，先进入当前要推的分支checkout，再push到远程对应的分支名git push demo1 wechat

Git log 和git reflog的区别

log命令可以显示所有提交过的版本信息。git log

如果觉得信息显示太繁琐  可以加上参数  --pretty=oneline[只会留下commit  id (版本号 (用SHA1字串表示))和 提交版本时的描述信息]  git log --pretty=oneline

　如果在回退以后又想再次回到之前的版本，git reflog 可以查看所有分支的所有操作记录（包括commit和reset的操作），包括已经被删除的commit记录，git log则不能察看已经删除了的commit记录