# 获得或创建一个版本库

## 获得远程版本库

git clone git@github.com:gotgithub/helloworld.git

## 1.2 创建本地版本库

# 创建一个版本库

git init

# 本地版本库基本操作

# 每次修改好了后，可以先将修改存入stage(快照/索引)中

git add <modified files>

# 删除文件

git rm <files>

# 修改了大量文件则使用下面这个命令批量存入

git add .

# 使用commit将快照/索引中的内容提交到版本库中

git commit -m "msg"

# 也可以将git add与git commit用一个指令完成

git commit -a -m "msg"

# 若想知道目前工作树的状态，可以輸入

git status

# 还有其它可以查看repo状态的工具

git log #可以查看每次commit的改变

git diff #可以查看最近一次改变的內容，加上参数可以看其它的改变并互相比较

git show #可以看某次的变更

用 oneline 或 format 时结合 --graph 选项，可以看到开头多出一些 ASCII 字符串表示的简单图形，形象地展示了每个提交所在的分支及其分化衍合情况。在我们之前提到的 Grit 项目仓库中可以看到：

git log --pretty=format:"%h %s" –graph

也可以用图形化显示

gitk

# 远程版本库

## 3.1 创建远程版本库

# 把github的repo加入为远程的repo

git remote add origin git@github.com:<你的ID>/test.git

# 把目前的commit状态push并同步到github上面

git push origin master

## 3.2 远程版本库与本地同步

# 将本地的git档案与github(远程)上的同步

git push

# 将github(远程)的git档案与本地的同步(即更新本地端的repo)

git pull

# 例如,pull指令其实包含了fetch(將变更复制回來)以及merge(合并)操作

git fetch origin

git merge origin/master

# 分支管理

## 4.1 本地库分支

# 另外版本控制系統的branch功能也很有意思，若同时修改bug，又要加入新功能，可以fork出一个branch：一个专门修bug，一个专门加入新功能，等到稳定后再merge合并

git branch bug\_fix # 建立branch，名为bug\_fix

git checkout bug\_fix # 切换到bug\_fix

git checkout master #切换到主要的repo

git merge bug\_fix #把bug\_fix这个branch和现在的branch合并

git branch -d bug\_fix #删除这个分支

git branch –D bug\_fix #强制删除未合并的分支

注意：在切换branch之前，将当前branch提交

## 4.2 本地分支与远程分支同步

创建和合并分支

#git branch 显示当前分支是master 本地和远程合并，本地默认分支为master

#git branch new-feature 创建分支

# git checkout new-feature 切换到新分支

合并到远程服务器

# git push origin new-feature 🡨**推送远程分支**

如果new-feature分支成熟了，觉得有必要合并进master

#git checkout master

更新本地master

#git pull origin master

合并new-feature

#git merge new-feature

更新至远程

# git push origin master

则master中也合并了new-feature 的代码

A handy way to push the current branch to the same name on the remote.

git push origin HEAD

# 若有remote的branch，想要查看并checkout

git branch -r # 查看远程branch

🡨**查看远程分支**

git checkout -b new-feature\_local origin/new-feature #获得远程的new-feature branch并命名为new-feature \_local，同时切换到该branch

在把远程分支合并进了远程的 master 分支后，可以用这个非常无厘头的语法来删除一个它：git push [远程名] :[分支名]

$ git push origin :new-feature

# 标签管理

## 5.1 本地标签

创建一个含附注类型的标签非常简单，用 -a （译注：取 annotated 的首字母）指定标签名字即可：

$ git tag -a v1.4 -m 'my version 1.4'

$ git tag

## 5.2 远程标签

默认情况下，git push 并不会把标签传送到远端服务器上，只有通过显式命令才能分享标签到远端仓库。其命令格式如同推送分支，运行 git push origin [tagname] 即可：

$ git push origin v1.5

如果要一次推送所有（本地新增的）标签上去，可以使用 --tags 选项：

$ git push origin --tags