对于刚要学习git的人，推荐去看《GitHub入门与实践》这本书，这是一本入门级别的书，里面介绍了git和github的使用方法，其中对于git命令的讲解通俗易懂。   
如果要学习git更高级的用法，可以去：[http://git-scm.com/book/zh/v1](http://git-scm.com/book/zh/v1" \t "_blank)

**基础部分**

**一、用git创建（在E盘的git目录下创建）一个叫myWebSite的目录（用于存储个人网站），并把它放到git版本库里：**

\*cd e:/git————-进入操作目录   
mkdir myWebSite——新建文件夹   
\*cd myWebSite———进入新建的文件夹   
echo 'My website is alive!' > index.html——新建的目录下 新建了一个index.html文件，并初始化了内容   
\*git init—————将所在目录（myWebSite）转化为git版本库（此时的版本库是空的，目录下的文件默认不会被放入版本库中，视为临时文件）

**二、将文件添加到版本库中：**

\*git add index.html——将index.html（指定文件）添加到版本库中   
\*git add .——————–将当前目录及子目录中的文件都添加到版本库里

**三、配置提交作者：**

\*git config user.name "chen shanshan"————配置姓名   
\*git config user.email "123456789@qq.com"——配置email地址

**四、提交文件：（在任何编辑之后，提交之前，都要执行git add命令 去*更新索引*）**

\*git commit -m "Test use git to commit file." （提交先要进行 git add 操作）   
可以设置GIT\_EDITOR环境变量来在交互式编辑器(vim)中编辑内容（vim编辑器的使用方式见扩展1），设置方法：export GIT\_EDITOR=vim   
\* git commit -am "提交信息" 合并git add 和 git commit 操作（适用于比较小的变更）

**五、查看提交：**

\*git status—————显示版本库里 中间状态（已经 add操作 还没 commit操作 / 或commit后修改过）的文件   
\*git log—————— 一系列单独提交的历史。后加 -p 查看前后差异；后加文件/目录 名查看指定文件/目录相关的日志（当前面出现个冒号无法输入命令时，按「q」键）   
git show 提交码(50fd4db21ca586dacf547b23b14374e942b81469)–查看特定提交的更详细的信息（不写提交码默认显示最近一次的）   
git show-branch --more=10（git show-branch）——————–当前开发分支简洁的单行摘要，–more=10表示最新的10个版本（即提交日志）

**六、查看更改之后的差别：**

查看工作树和暂存区的差别：git diff   
查看工作树和最新提交的差别：git diff HEAD

“+”号标出的是新添加的行，被删除的   
行则用“-”号标出：

diff --git a/README.md b/README.md

index e69de29..7ca0b44 100644

**--- a/README.md**

**+++ b/README.md**

**@@ -0,0 +1,2 @@**

+# git test

+# git

\ No newline at end of file

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8

**七、修改（新建）文件：**

1. touch 文件名———直接新建一个文件
2. vi 文件名————–新建一个文件并进入编辑状态（已存在文件直接进入编辑状态）
3. 如果没用过linux和vim，最直接的就是在windows下新建、修改文件吧！！！

**八、版本库内文件的删除和重命名：**

删除文件：（和git add一样。git rm 表示想要删除文件，git commit 实现删除操作）   
\*git rm aaa.html bbb.txt 删除版本库内的aaa和bbb两个文件 （可连续输入：ls / git rm aaa.txt / git commit -m “Remove aaa.txt”）   
\*重命名文件：

git mv index.html aaa.html //（防止git rm命令会将index文件永久删除，必须先执行mv操作）

git rm index.html

git add aaa.html

git commit -m "Moved index to aaa" //在任意情况下，暂存的变更必须随后进行提交

* 1
* 2
* 3
* 4

**九、创建版本库副本：**

git clone website clone\_website（克隆远程仓库：git clone git@47.12.10.11:/option/git/provider1.git provider1）   
（克隆网站上的版本库副本：git clone \（回车） git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git）

**十、其他命令：**

查看日志： git log --pretty=oneline   
回退：到上一版本：git reset --hard HEAD^   
——–到指定版本：git reset --hard 3628164   
迁出： git checkout xxx.txt（未放入暂存区：恢复成版本库；放入暂存区：恢复成暂存区）   
查看和上一版本变化的内容：git diff xxx.txt   
远程版本库：创建公钥：ssh-keygen -t rsa -C “邮箱”

**分支**

查看分支：git branch （列表显示分支名，所在分支前有\*）   
查看分支：git log --graph （以图表形式查看分支）   
新建分支：git branch 分支名   
切换分支：git checkout 分支名   
新建并切换分支：git checkout -b 分支名   
删除分支：git branch -D 分支名   
合并分支：

1. 首先切换回master分支：git checkout master
2. 然后合并子分支（如：feature-A）: git merge --no-ff feature-A（加上 –no-ff参数 创建和并提交，来在历史记录中明确记录下本次分支合   
   并）

**更改提交的操作**

1、查看当前仓库的操作日志：git reflog （所有操作都记录下，可查看回溯历史之前的**哈希值**）   
2、回溯历史版本：git reset --hard 哈希值(81d5c7e) （取一个之后版本的哈希值就可以实现推进历史了）   
3、修改上一次提交信息：git commit --amend （分支的合并提交信息可以为：Merge branch ‘分支名’）   
4、解决冲突：   
打开冲突文件，是这个样子：（等号以上的部分是当前 HEAD 的内容，以下的部分是要合并的分支中的内容）

# Git Test

<<<<<<< HEAD

- feature-A

=======

- fix-B

>>>>>>> fix-B

* 1
* 2
* 3
* 4
* 5
* 6
* 7

可以手动改成下面这样，再执行 git add命令与 git commit命令：

# Git Test

- feature-A

- fix-B

* 1
* 2
* 3

注意：在实际的软件开发中，往往需要删除其中之一，所以各位在处理冲突时，务必要仔细分析冲突部分的内容后再行修改

5、压缩(更改)历史： 提交一个修改到前一个提交中，压缩成一个历史记录（假设最新的一次提交是修改了前一次提交的一个小问题，可将最后一次提交压缩到前一次）

* git rebase -i HEAD~2 会在编辑器中打开两个最新的历史记录，将后一个前面的 pick 改为 fixup

**远程仓库**

这里以github上的一个远程仓库为例，可以在github.com上新建一个远程仓库。

一、把已有仓库推送至远程仓库（以首个开发者的身份）

* 将它设置成本地仓库的远程仓库（与远程仓库建立连接）   
  git remote add origin https://github.com/用户名/仓库名.git   
  也可以写成下面：   
  git remote add origin git@github.com:用户名/仓库名.git
* 推送至远程仓库（加 -u 是为了将 origin 仓库的 master 分支设置为本地仓库当前分支的上游）   
  + 推送至master分支： git push -u origin master
  + 推送至其他分支：git push -u origin 分支名(feature-B) （将本地仓库中的分支 push 给远程仓   
    库并保持分支名称不变）
  + 删除远程仓库上的分支（github上的仓库）：git push origin :feature-B

二、从远程仓库获取（以后来合作开发者的身份）

从远程仓库获取 xxx 分支，在本地仓库中提交更改，再将 xxx 分支推送回远程仓库，通过这一系列操作，就可以与其他开发者相互合作，共同培育 xxx 分支，实现某些功能。

* 获取远程仓库：git clone git@github.com:用户名/仓库名.git （默认处于master分支下）
* 查看获取后的本地仓库和远程仓库的分支信息：git branch -a
* 获取远程分支到本地仓库：git checkout -b 本地新建分支名 origin/远程分支名
* 向本地的分支提交更改（和之前的操作一样），完成后，**推送分支**：git push
* 在推送分支之前，先要获取最新的远程仓库分支：git pull origin 远程分支名

**扩展：**

**1、 vi/vim基本使用方法：（vim是vi的升级版）**

这里（Git Bash）简单用法： （以git commit时的日志内容为例）

* 在Git Bash中输入：git commit，此时会进入vim编辑器
* 按「i」键进入「插入模式」（进入vi[vim]之后，是处于 命令行模式，要切换到 插入模式 才能够输入文字）
* 输入内容，之后按「ESC」键转到 命令行模式
* 按「:」(英文)冒号键进入「Last line mode」输入「wq」，存盘并退出vi （输入「q!」， 不存盘强制退出vi）
* 按回车键，提交成功

具体可参考：<https://www.cnblogs.com/itech/archive/2009/04/17/1438439.html>

**2、在git bash下打开noetpad++等文本编辑器**

* 在Git Bash中输入notepad .bashrc （在 “C:\Users\计算机名” 目录下新建一个没有名字的.bashrc文件）
* 在Git Bash中输入 notepad++.exe 路径，如/C/Notepad++/notepad++.exe，看notepad++是否安装正确，安装正确的话会将它打开
* 用任意文本编辑器打开.bashrc文件，输入’alias 自己起名=”任一个文本编辑器路径’，如：alias notepad++="/C/Notepad++/notepad++.exe"   
  注意：文本编辑器的路径中最好不要有空格，括号等，否则会造成命令无效
* 关闭Git Bash，重新打开，输入notepad++ xxx.txt，在notepad中打开了xxx.txt文件   
  注意：Git Bash进入的目录与xxx.txt所在目录要一致

参考：<https://www.cnblogs.com/BugBrother/p/6963484.html>

**3、在GitHub上发布版本库**

* 创建GitHub的版本库：登陆GitHub账户（任意位置单击logo可跳转到个人dashboard页），单击Start a project
* 为新的版本库命名： 唯一要填的字段是Repository name（项目名），项目将会出现在 <https://github.com/GitHub/>用户名/项目名 处
* 初始化版本库： 现在版本库已经创建了还没有内容，可根据github提示去 推送内容。2步（git remote add origin 地址(git remote add origin https://github.com/yourusername/yourprojectname.git) / git push -u origin master）

webstorm中使用git管理服务器上的代码 可参考：<http://blog.csdn.net/chenxueshanblog/article/details/78722789>