简介
Git 分为本地仓库和远程仓库，如果是个人独立开发，可不用远程仓库。而远程仓库其实就是***别人的本地仓库***。因为 Git 是分布式的，项目成员在本地都有自己的仓库，可以自己自由地创建分支、修改、合并，而且 Git 鼓励多创建分支。需要时自己的分支可以和任意一个成员的分支合并。所以 Git 总结起来就是： 1. 创建本地分支，修改代码，合并本地分支，解决冲突，形成本地最终分支； 2. 获取其它成员仓库里的最终分支成为本地分支，并将此分支合并到本地最终分支里，解决冲突，形成最终项目发布分支。

本地仓库		
初始化本地仓库	mkdir git_test_project	新建一个目录或找一个已经存在目录
	cd git_test_project	
	git init	初始化此目录本地仓库，隐藏目录.git 就是仓库信息
	git add ./	添加当前目录及其子文件子目录
	git commit -a -m "the init commit"	提交修改到仓库,必须要提交了才会保存到本地仓库里
修改内容提交到本地仓库	vi a.c	修改了一些东西
	git commit -a -m "the 1st commit"	提交修改到本地仓库
添加文件	touch d.c	新建文件或一个已存在的文件
	git add d.c	添加文件
	git commit -a -m "add new file"	提交刚才的添加到本地仓库
添加目录	mkdir new_floder	新建或者找一个现成的目录
	touch new_floder/test.c	建立目录下的文件
	git add new_floder	添加了目录及目录下所有子目录及子文件
	git commit -a -m "add new floder"	提交刚才的添加修改到本地仓库
删除文件/目录	rm -f a.c new_floder	先执行删除命令
	git commit -a -m "delete d.c"	提交删除到本地仓库,虽然当前版本删除了，但历史版本中还有
创建分支	git branch	显示本地仓库所有分支，带*为当前工作的分支
	git branch testing	在本地仓库创建一个名叫 testing 的分支
	git checkout testing	切换到 testing 分支工作，以后的 commit 只对 testing 分支有效
删除分支	git branch -d testing	删除 testing 分支
查看历史版本	git log	在 commit 字段后面的那串数字就是所谓的”commit id”
回滚到历史版本继续开发	git checkout <commit id>	这样获取的版本只能看，就算改了也不能提交，逻辑上不允许
	git branch fix_bug	要想在这个版本上工作，必须在 checkout 之后创建一个分支
	git checkout -b fix_bug <commit id>	或在的获取历史版本的时候直接创建分支
比较分支差异	git diff release	比较***当前分支*** 与 release 的差异并在终端上输出(patch 格式)
合并本地分支	git merge release	合并 release 分支到***当前分支*** 上,注意： 1.如果是某个文件内容有冲突： <<<<<<< HEAD 此处为当前分支的内容 ======= 此处为 release 分支的内容 >>>>>>> new 2. 合并是将“操作”合并。假设初始时有三个文件 a.c b.c c.c: 如果当前分支添加 d.c, release 删除 b.c,合并结果: a.c c.c d.c 如果当前分支删除 b.c, release 删除 c.c,合并结果: a.c 如果当前分支添加 d.c, release 添加 e.c,合并结果:a.c b.c c.c e.c

远程仓库		
从远程仓库克隆本地仓库	git clone 10.0.0.10:/git_test_project/.git/ ./myproject	克隆远程仓库的***当前分支*** 到 myproject 目录下
查看远程仓库所有分支	git branch -r	查看***克隆源*** 仓库里的所有分支
从远程仓库获取分支	git fetch 10.0.0.10:/git_test_project/.git/ release:test	获取远程仓库的 release 分支成为本地的 test 分支