**windows Git安装**：<https://gitforwindows.org/> 直接下载，默认选项安装

指定姓名邮箱：git config --global user.name "username"

git config --global user.email "username@example.com"

**备注：read.txt 文件名**

创建版本库，选择一个合适的地方创建文件夹 右击Git Bash

git status ：查看当前仓库的状态

git init：把目录变成Git可以管理的仓库

git add read.txt ：把文件添加到仓库。add后面可以跟多个不同的文件，可反复多次使用，添加多个文件

git commit -m "第一次提交" ：告诉git把文教提交到仓库，-m选项后面输入本次提交的说明。完成

git diff read.txt**：**查看文件被修改的内容

git log：查看提交的历史

git reset --hard HEAD^**：**回退到上一个版本，HEAD表示当前版本，HEAD^表示上一个版本，

HEAD^^表示上上一个版本，上100个版本HEAD~100。也可以指定版本ID号

git reset --hard 67b533：67b533最新版本的ID号**：**回到最新版本。退回上一个版本之后最新的版本已经看不到了

cat read.txt：查看read.txt的内容

git reflog：查看命令历史，以便确定要回到未来的哪个版本。

git add命令实际上就是把要提交的所有修改放到暂存区（Stage），然后，执行git commit就可以一次性把暂存区的所有修改提交到分支。

git checkout -- read.txt：撤销修改

一种是read.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；

一种是read.txt已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。

总之，就是让这个文件回到最近一次git commit或git add时的状态。

git reset HEAD read.txt**：**把暂存区的修改撤销掉（add后的文件），重新放回工作区

git rm read.txt：从版本库中删除文件，并且 **git commit -m "rm read.txt**"提交

git checkout -- read.txt：误删除从版本库恢复文件

注意从来没有被添加到版本库就被删除的文件，是无法恢复的

**添加远程仓库**：

1、创建SSH key： ssh-keygen -t rsa -C "邮箱地址"：在用户的主目录下有.ssh目录，下有id\_res（私钥不能给别人），id\_rsa.pub（公钥可以给）主目录：Ctrl+R输入.

2、登陆GitHub。打开 Settings，点击 SSH and GPC keys，点击 New SSH Key，添加标题和id\_rsa.pub,点击添加

3、在GitHub创建库

4、本地仓库关联远程仓库：git remote add origin git@github.com:Github的地址

5、关联后第一次推送到master分支的所有内容：git push -u origin master

如推送其他分支的全部内容就改成：git push origin dev

由于远程库是空的，我们第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令。

git push origin master**：**把本地库推送到远程是用命令，实际上是把当前分支master推送到远程。

6、取消本地仓库和远程仓库的关联：git remote remove origin

7、将远程库克隆到本地库：git clone git@github.com:Github地址

8、查看远程库信息，使用git remote -v；

本地新建的分支如果不推送到远程，对其他人就是不可见的；

从本地推送分支，使用git push origin branch-name，如果推送失败，先用git pull抓取远程的新提交；

在本地创建和远程分支对应的分支，使用git checkout -b branch-name origin/branch-name，本地和远程分支的名称最好一致；

建立本地分支和远程分支的关联，使用git branch --set-upstream branch-name origin/branch-name；

从远程抓取分支，使用git pull，如果有冲突，要先处理冲突。

**创建于合并分支：**

查看分支：git branch

创建分支：git branch <name>

切换分支：git checkout <name>

创建+切换分支：git checkout -b <name>

合并某分支到当前分支：git merge <name>

删除分支：git branch -d <name>

git merge --no-ff -m "merge with no-ff" dev：加上--no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并

**分支管理解决冲突：**

当Git无法自动合并分支时，就必须首先解决冲突。解决冲突后，再提交，合并完成。

解决冲突就是把Git合并失败的文件手动编辑为我们希望的内容，再提交。

查看分支合并图：git log --graph --pretty=oneline --abbrev-commit

查看分支合并图：git log --graph

**BUG分支：**

git stash：隐藏当前工作现场

git stash list：查看工作现场隐藏

一是用git stash apply恢复，但是恢复后，stash内容并不删除，你需要用git stash drop来删除

另一种方式是用git stash pop，恢复的同时把stash内容也删了

可以多次stash，恢复的时候，先用git stash list查看，然后恢复指定的stash，用命令：

git stash apply stash@{0}

**Feature分支：**

软件开发中，总有无穷无尽的新的功能要不断添加进来。

添加一个新功能时，你肯定不希望因为一些实验性质的代码，把主分支搞乱了，所以，每添加一个新功能，最好新建一个feature分支，在上面开发，完成后，合并，最后，删除该feature分支。

如果要丢弃一个没有被合并过的分支，可以通过git branch -D <name>强行删除