**基本操作步骤：**

第一步：add，将文件纳入控制之中

第二步：commit，提交到本地缓存，每次修改了文件 在push之前也必须commit，否则无法push

第三步：push，推送到主分支

git add file 将制定文件提交暂存区

git add . 将所有修改过的工作文件提交暂存区

git commit -m message 带有message的提交

git push

git fetch origin # 抓取远程仓库

一、Git fetch origin master # 抓取远程仓库到本地

二、git diff origin master # 比较一下主分支与本地分支之间的区别

三、git merge origin/master # 合并远程主分支与本地分支

**在本地修改了代码，提交步骤：**

一：git commit -a -m "msg"

二：git push

备注：正常就提交成功了，失败了一般是冲突[远程文件在本地提交之前已经修改了]

在使用git bash在提交的时候，如果失败，一般都是冲突导致的:

冲突情况：

一：同一行冲突

二：非同一行冲

**冲突处理：**

先fetch把主分支的拉下来[注意：千万别直接pull，这样会很危险，因为pull是把主分支的拉下来，然后直接合并]，比较一下是哪些地方冲突了，然后合并[做一个pull的操作]，再在合并之后的文件上调一下顺序[合并之后代码的逻辑不能乱了]调好顺序之后，就可以提交了，步骤:

一：git fetch origin master # 抓取主分支的本地分支，为了下一步比较哪有有变化

二：git diff origin master # 比较主分支与本地分支的区别

三：git merge origin master，合并之后，需要调一下代码的顺序[看一下代码逻辑等等问题]

四：git commit -a -m msg

五：git push

**小技巧**：在本地编写代码可以直接pull[拉取服务器代码]，如果有冲突，就会有提醒解决冲突；如果没有就直接合并了，然后就可以直接commit

commit之后，用diff可以查看修改的文件与远程文件的区别

**比较暴力的删除一条提交记录【此方案不是很好】**

思路：如果在删除的记录处的后面已经有了很多次的提交记录，那么可以先此基础上创建一个分支，然后reset到需要删除的提交记录的前面一次的记录处 ，然后在分支上的删除的记录后面的提交merge过来

git reset --hard 8adeb3185d8f88702d1148260a3086171c8e8cef # reset到需要删除的提交记录的前面一次提交记录

git push origin HEAD --force

场景1：当改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令git checkout -- file。

场景2：当不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令git reset HEAD file，就回到了场景1，第二步按场景1操作。

场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考版本回退一节，不过前提是没有推送到远程库。

**//////////////////////// 回退 ///////////////////////////**

HEAD指向的版本就是当前版本，因此，Git允许我们在版本的历史之间穿梭，使用命令git reset --hard commit\_id。

穿梭前，用git log可以查看提交历史，以便确定要回退到哪个版本。

要重返未来，用git reflog查看命令历史，以便确定要回到未来的哪个版本。

**///////////////////本地库关联远程库//////////////**

方法一：在远程服务器上创建一个库，clone到本地

方法二：把一个已有的本地仓库与之关联

1、在本地的仓库下运行命令，关联远程库

git remote add origin 被关联的远程库，远程库的名字就是origin，这是Git默认的叫法，也可以改成别的

2、把本地库的所有内容推送到远程库上

git push -u origin master

**## push an existing repository from the command line**

1、git remote add origin https://github.com/zhouwei441300/test01.git

2、git push -u origin master

**本地创建一个仓库，然后同步到到远程**

# Command line instructions

Git global setup

git config --global user.name "xiaoge"

git config --global user.email "xiaoge@xiyou3g.com"

# Create demo new repository

git clone https://git.xiyoumobile.com/xiaoge/demo.git

cd demo

touch README.md

git add README.md

git commit -m "add README"

git push -u origin master

# Existing folder or Git repository

cd existing\_folder

git init

git remote add [origin 本地别名] https://git.xiyoumobile.com/xiaoge/a.git

git add .

git commit

git push -u origin master

**////////////////////////创建分支//////////////////////////**

1、创建dev1.1本地分支，然后切换到dev1.1分支

git checkout -b dev1.1 基于当前分支[A]创建一个dev1.1分支

本地新建一个分支后，必须要做远程分支关联。如果没有关联， git 会在下面的操作中提示你显示的添加关联。关联目的是如果在本地分支下操作： git pull, git push ，不需要指定在命令行指定远程的分支． 推送到远程分支后， 你只要没有显示指定， git pull 的时候，就会提示你。

**解决方法：**

使用命令git branch --set-upstream ;

实例如下，其中dev1.1为创建的分支

git branch --set-upstream dev1.1 origin/dev1.1

==相当于如下两条命令：

$ git branch dev1.1

$ git checkout dev1.1

2、在dev1.1分支上做完开发之后，想把dev1.1分支合并到A分支

git checkout [A]

git merge dev1.1

3、删除dev1.1分支

git branch -d dev1.1

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>

如果git pull提示“no tracking information”，则说明本地分支和远程分支的链接关系没有创建，用命令git branch --set-upstream branch-name origin/branch-name关联

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。

git log：

git log --pretty=oneline：格式化输出日志

git reset --hard HEAD^：回退到上一个版本

git reset --hard HEAD~100： 回退到上一百个版本

git reset HEAD readme.txt 把暂存区的修改撤销掉（unstage），重新放回工作区

git checkout -- readme.txt 把工作区的修改撤销掉

git reflog：显示操作命令记录

git log -p -2 #-p 选项展开显示每次提交的内容差异，用 -2 则仅显示最近的两次更新：