# Git操作

## Git简介：

Git是什么？

Git是目前世界上最先进的分布式版本控制系统（没有之一）。

Git有什么特点？简单来说就是：高端大气上档次！

那什么是版本控制系统？

如果你用Microsoft Word写过长篇大论，那你一定有这样的经历：

想删除一个段落，又怕将来想恢复找不回来怎么办？有办法，先把当前文件“另存为……”一个新的Word文件，再接着改，改到一定程度，再“另存为……”一个新文件，这样一直改下去，最后你的Word文档变成了这样：

过了一周，你想找回被删除的文字，但是已经记不清删除前保存在哪个文件里了，只好一个一个文件去找，真麻烦。

看着一堆乱七八糟的文件，想保留最新的一个，然后把其他的删掉，又怕哪天会用上，还不敢删，真郁闷。

更要命的是，有些部分需要你的财务同事帮助填写，于是你把文件Copy到U盘里给她（也可能通过Email发送一份给她），然后，你继续修改Word文件。一天后，同事再把Word文件传给你，此时，你必须想想，发给她之后到你收到她的文件期间，你作了哪些改动，得把你的改动和她的部分合并，真困难。

于是你想，如果有一个软件，不但能自动帮我记录每次文件的改动，还可以让同事协作编辑，这样就不用自己管理一堆类似的文件了，也不需要把文件传来传去。如果想查看某次改动，只需要在软件里瞄一眼就可以，岂不是很方便？

这个软件用起来就应该像这个样子，能记录每次文件的改动：

## Git常规操作：

Git init 初始化仓库 版本库又名仓库，英文名repository

readme.txt 文件 一般仓库会有这个东西（可有可无）说明文档、自述文件而已

Git add readme.txt 将这个文件放到暂存区

多种操作方式：

Git add readme.txt ss.txt 提交多个文件

Git add . 提交所有改动的文件

（执行上面的命令，没有任何显示，这就对了，Unix的哲学是“没有消息就是好消息”，说明添加成功。）

Git commit -m “提交 readme.txt文件” 将文件放入本地仓库 ( -m后面输入的是本次提交的说明，可以输入任意内容，当然最好是有意义的，这样你就能从历史记录里方便地找到改动记录。) git commit命令执行成功后会告诉你，1 file changed：1个文件被改动（我们新添加的readme.txt文件）；2 insertions：插入了两行内容（readme.txt有两行内容）

Git status 查看仓库当前的状态

Git pull 拉取最新仓库代码

Git push 推送本地代码到远程仓库

git branch 查看本地所有分支

git branch -r 查看远程所有分支

git branch -a 查看本地和远程的所有分支

git branch <branchname> //新建分支

git branch -d <branchname> //删除本地分支

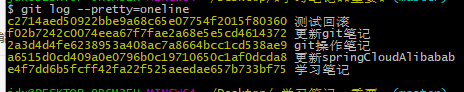
git branch -d -r <branchname> //删除远程分支，删除后还需推送到服务器

git push origin:<branchname> //删除后推送至服务器 git branch -m <oldbranch> <newbranch> //重命名本地分支

## Git进阶操作：

### 版本回退

Git log 查看提交日志 显示从最近到最远的提交日志，每一次提交都有对应的 commit id 和commit message （如果嫌弃输出的信息杂乱无章，那么加上 --pretty=oneline 参数试试吧！）



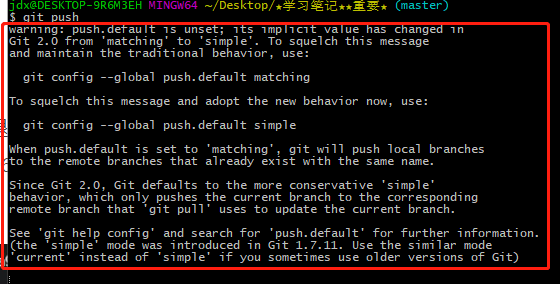
Git reset --hard id 根据id回退到指定的版本

### 2.压制拉取时的警告：

这一串英文是说你没有指定要推送的远端仓库分支

你只需要在第一次执行推送的时候输入 git push -u origin master 就可以了，

如果没执行也没关系，



### 3.Git保存仓库的账号密码：

# 保存本地仓库的账号

git config --local credential.helper store

# 保存git全局账号

git config --global credential.helper store

2、执行一般git命令，clone、pull等。输入账号密码，命令执行完成后，即保存了账号密码。下次执行git命令不会要求再次输入。

注意：如果第一次输入账号密码是错的，一定要在下次执行一般git命令前，先执行保存命令以重新接受新的账号密码。

### 拉取仓库最新代码解决本地冲突：

当有人提交代码报错后，本地无法解决冲突的情况下可以使用这个代码来恢复正常拉取代码

git fetch --all git fetch是将远程主机的最新内容拉到本地

git reset --hard origin/master 抛弃我所有的分阶段和未分阶段的更改，忘记我当前的本地分支上的一切，使它与origin / master完全相同。

你可能想在你运行命令之前问这个。通过使用与“硬复位”中相同的词暗示了破坏性本质。

### 5.创建分支

### 6.切换分支

分支的概念：

分支是用来标记特定代码的提交，每一个分支通过SHA1sum值来标识，所以对分支的操作是轻量级的，你改变的仅仅是SHA1sum值。

Git checkout dev 切换分支切换到dev 分支

### 7.合并分支

当前应该所在需要合并的分支中（master）

Git merge dev //将dev分支合并到当前分支(master)中