Git内容**（**所有的版本控制系统，只能跟踪文本文件的改动**）**

## 设置用户名和邮箱

git config --global参数

例如：git config --global user.name “lijiacheng”

git config --global user.email “1589740684@qq.com”

延伸：git config --list（查看）

git config user.name

## 创建版本库与修改

例如：在D盘的Git下创建一个testgit的版本库

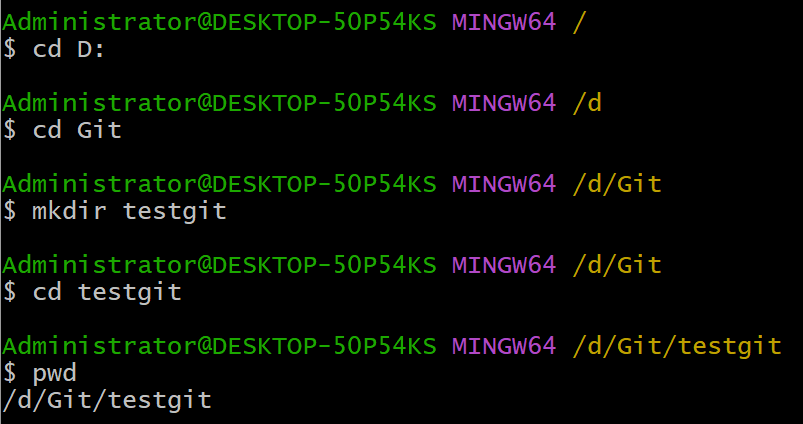
$ cd D：

$ cd Git

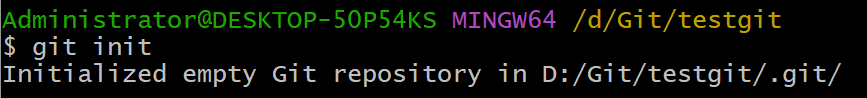
$ mkdir testgit

$ cd testgit

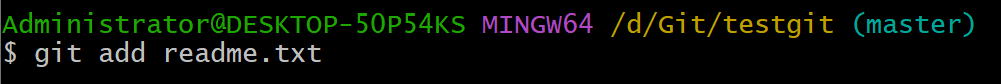
$ pwd(pwd命令是用于显示当前的目录)



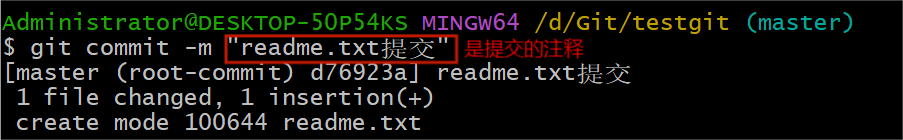
* git init命令：将该目录变成git可以管理的仓库



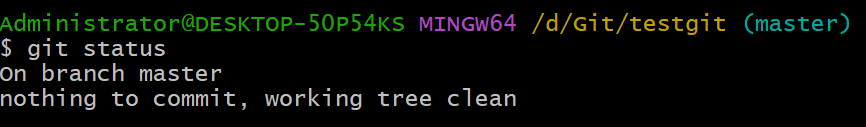
* git add +文件名 命令：把文件添加到暂存区中



* git commit -m “注释”命令：把文件提交到仓库

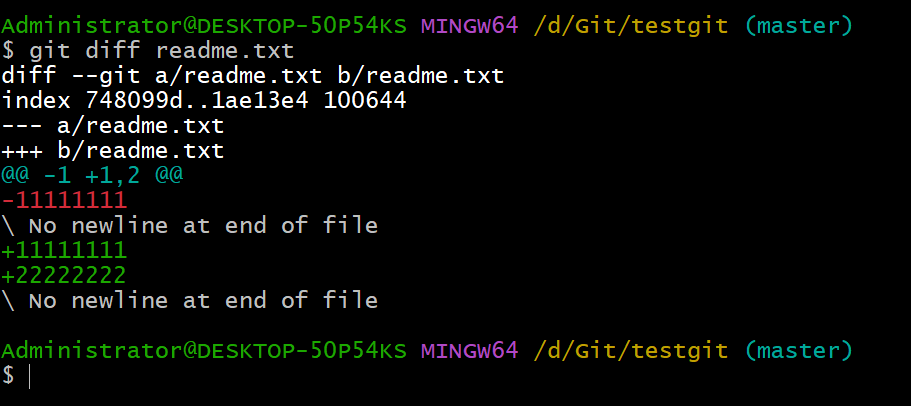


* git status命令：查看是否还有文件未提交



* git diff +文件名 命令：文件到底改了什么内容

（修改后需要git add和git commit这两步）



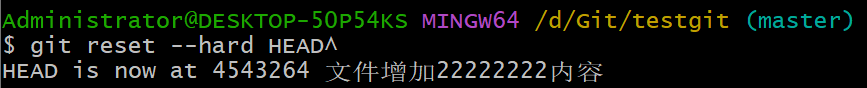
## 版本回退

* git log命令：查询历史记录（信息太多使用命令git log --pretty=oneline）

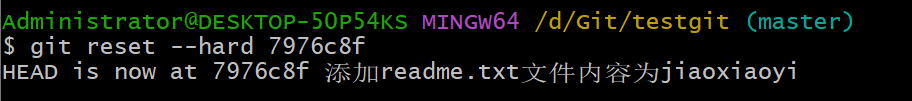




* git reset --hard HEAD^命令：回退到上一个版本



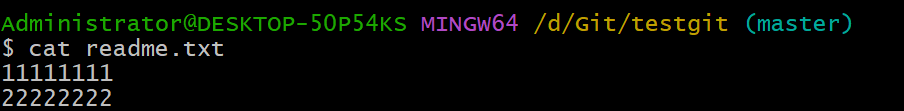
* git reset --hard HEAD^^命令：回退到上上个版本
* git reset --hard HEAD~100命令：回退到前100个版本
* git reset --hard 版本号 命令：回退到某一版本



* git reflog命令：查询版本号



* cat +文件名：查看文件



## 工作区与暂存区的区别

工作区：就是你在电脑上看到的目录，或者以后需要再新建的目录文件等等都属于工作区范畴。(.git隐藏目录版本库除外)  
版本库(Repository)：隐藏目录.git是版本库。其中版本库里最重要的就是stage(暂存区)，还有Git为我们自动创建了第一个分支master,以及指向master的一个指针HEAD。git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支上

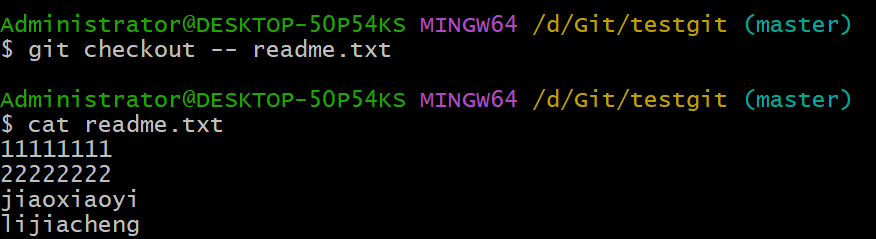
## Git撤销修改和删除文件操作

* 撤销：

方法一：在工作区直接修改然后add commit

方法二：版本回退

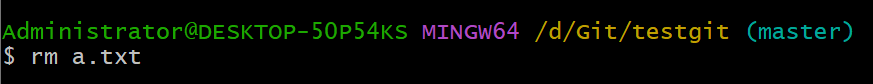
方法三：直接使用撤销命令——git checkout -- file可以丢弃工作区的修改（一种是还没有放到暂存区，使用撤销修改就回到和版本库一模一样的状态。另外一种是readme.txt已经放入暂存区了，接着又作了修改，撤销修改就回到添加暂存区后的状态。）



* 删除：

方法一：工作区直接删除

方法二：使用rm file命令（如想彻底删除commit，恢复用git checkout -- file）



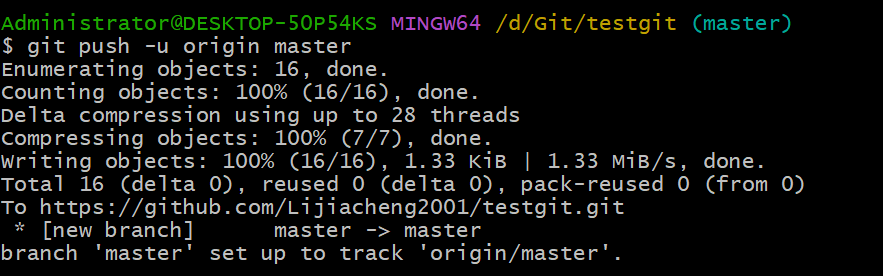
## 远程仓库

* 关联

git remote add origin <https://……>

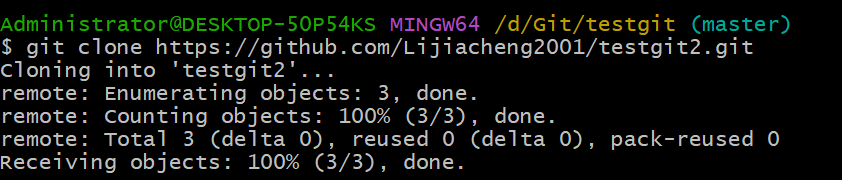


把本地库的内容推送到远程，使用 git push命令，由于远程库是空的，我们第一次推送master分支时，加上了 –u参数（以后本地做了提交就做以下命令git push origin master）



* 将远程库克隆到本地

创建远程库（勾选生成README.md文件）——git clone+网址



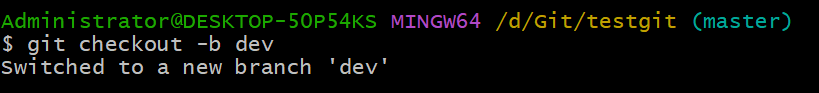
## 创建与合并分支

* 创建dev分支然后切换到dev分支上

git checkout -b dev表示创建并切换，相当于如下2条命令

git branch dev（创建）

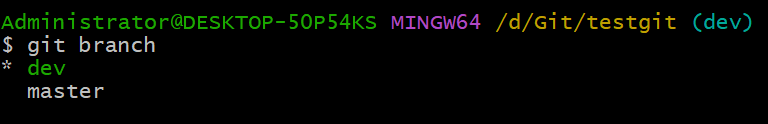
git checkout dev（切换）



* git branch命令查看本地分支（会列出所有的分支，当前分支前面会添加一个星号）

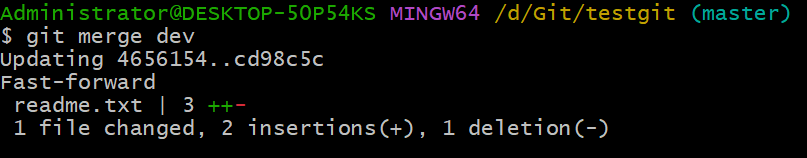
git branch -r  列出所有远程分支

git branch -a 列出所有本地分支和远程分支

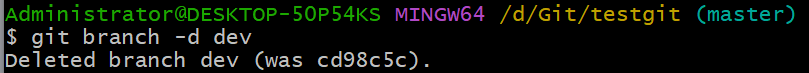


* 把dev分支上的内容合并到分支master上（在master分支上使用命令git merge dev）

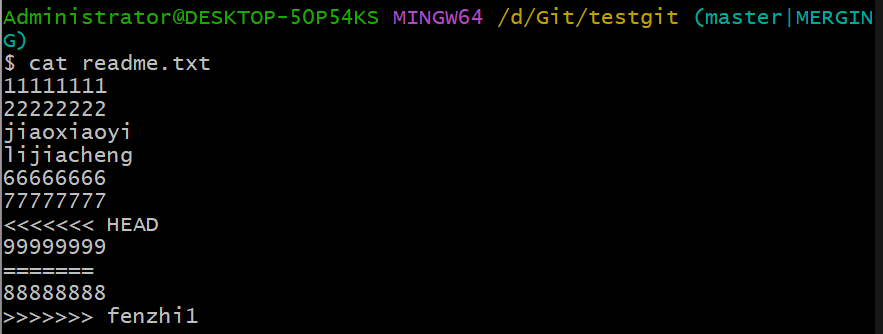
git merge name命令用于合并指定分支到当前分支上



* git branch -d name命令（删除分支）



* 冲突：当在分支上新增内容而且master分支上也新增内容合并的时候就会有冲突



后直接修改文件的内容

## 分支管理

合并分支时，git一般使用”Fast forward”模式，在这种模式下，删除分支后，会丢掉分支信息，现在我们来使用带参数 –no-ff来禁用”Fast forward”模式。

git merge --no-ff -m “注释” dev ：合并dev分支 --no-ff表示禁用fast forward

## Bug分支

* 在分支上修改修复完成合并分支，删除临时分支
* git stash命令：将当前的工作现场隐藏起来
* Git stash list命令：查看工作现场
* 1.git stash apply恢复，恢复后，stash内容并不删除，你需要使用命令git stash drop来删除。  
  2.另一种方式是使用git stash pop,恢复的同时把stash内容也删除了。

1. **多人协作**

* git remote：查看远程库的信息  
  git remote -v：查看远程库的详细信息



* 推送分支：git push origin master本地提交到远程库
* 抓取分支：发送方先将dev分支推送到远程

接收方克隆远程的库到本地，要在dev分支上做开发，就必须把远程的origin的dev分支到本地来，使用命令git checkout –b dev origin/dev 。修改后推送

注：推送发生冲突先用git pull把最新的提交从origin/dev抓下来，然后在本地合并

1. **临时性分支**

* 功能（feature）分支:为了开发某种特定功能，从Develop分支上面分出来的。开发完成后，要再并入Develop。

创建：Git checkout -b feature-x develop

合并：git checkout develop  
  git merge --no-ff feature-x

* 预发（release）布分支：发布正式版本之前（即合并到Master分支之前），我们可能需要有一个预发布的版本进行测试。预发布分支是从Develop分支上面分出来的，预发布结束以后，必须合并进Develop和Master分支。它的命名，可以采用release-\*的形式

创建：git checkout -b release-1.2 develop

合并：git checkout master  
   git merge --no-ff release-1.2  
  # 对合并生成的新节点，做一个标签  
   git tag -a 1.2

git checkout develop  
   git merge --no-ff release-1.2

* 修补bug（fixbug）分支：修补bug分支是从Master分支上面分出来的。修补结束以后，再合并进Master和Develop分支。它的命名，可以采用fixbug-\*的形式。

创建：git checkout -b fixbug-0.1 master

合并：git checkout master  
   git merge --no-ff fixbug-0.1  
   git tag -a 0.1.1

git checkout develop  
   git merge --no-ff fixbug-0.1

1. **Gitee 和 Github 同步更新**

**方法一：比较少分支的仓库**

1.首先通过 git remote -v 查看您要同步的仓库的远程库列表，如果在列表中没有您 Gitee 的远程库地址，您需要新增一个地址

git remote add 远程库名 远程库地址

eg: git remote add gitee git@github.com:xxx/xxx.git

如果在 add 的时候出现error: Could not remove config section 'remote.xxx'.一类的错误，通过把仓库下.git/config 文件里面的 [remote "xxx"] 删掉或者是用别的远程库名即可。

2、从 GitHub 上拉取最新代码到本地

git pull 远程库名 分支名

eg：git pull origin master

3、推送本地最新代码到 Gitee 上

git push 远程库名 分支名

eg：git push gitee master

**方法二：比较多分支的仓库**

