## Git的安装

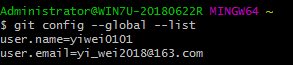
## Git的最小设置

配置user.name 和 user.email

$git config --global user.name ‘your\_name’;

$git config --global user.email [‘your\_email@domain.com’](mailto:‘your_email@domain.com’);

**查看当前配置:**



**Config 的三个作用域**

缺省等同于local

$git config --local （local只对某个仓库有效）

$git config --global （global对当前用户所有仓库有效）

$git config --system （system对系统所有登陆的用户有效）

显示config的配置,加 --list

$git config --list --local

$git config --list --global

$git config --list --system

## 建Git仓库

**两种场景:**

1. 把已有的项目代码纳入Git管理

$cd 项目代码所在的文件夹

$git init

1. 新建的项目直接用Git管理

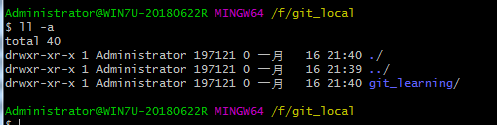
$cd 某个文件夹

$git init your\_project #会在当前路径下创建和项目名称同名的文件夹

$cd your\_project

进入cd /f/git\_loca

$git init git\_learning

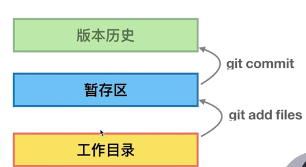


## 通过几次commit来认识工作区和暂存区

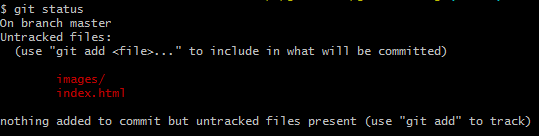
**往仓库里添加文件:**

4次提交,一个像模像样的静态页面生成

1. 加入index.html 和git-logo
2. 加入style.css
3. 加入script.js
4. 修改index.html 和 style.css



查看暂存区

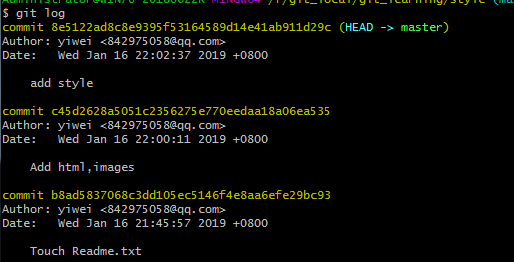


$git add index.html images/ #加入到暂存区

$git commit -m’add index images’ #提交暂存区的内容到版本历史

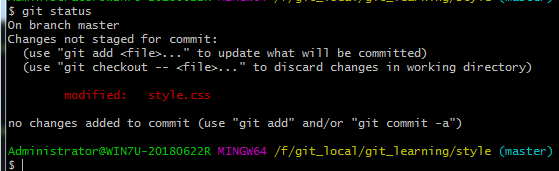
查看当前log

$git log

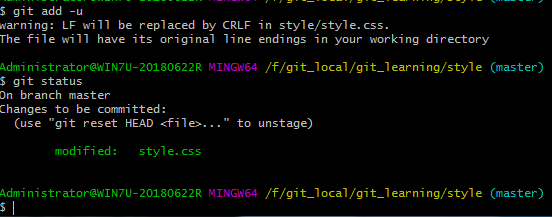


当我修改了文件内容：

$git status #修改信息将被显示

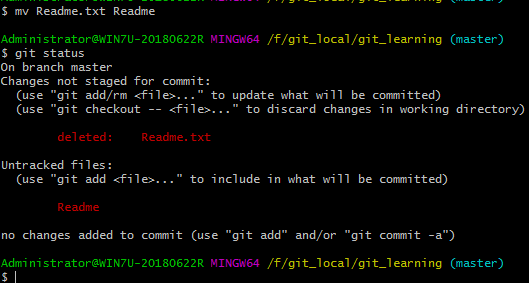


$git add -u #把当前所有修改的文件都加入到暂存区

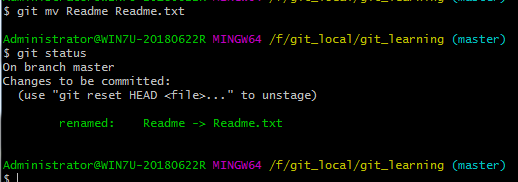


## 给文件重命名简便方法

当我修改一个文件名的时候,git将会做两步操作

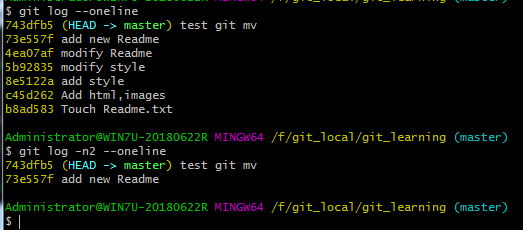


我们可以直接用git mv Readme Readme.txt #git 将会直接在暂存区中帮我们修改



## 通过git log查看版本演变历史

$git log --oneline



创建新的分支:

$git checkout -b temp 8e5122a



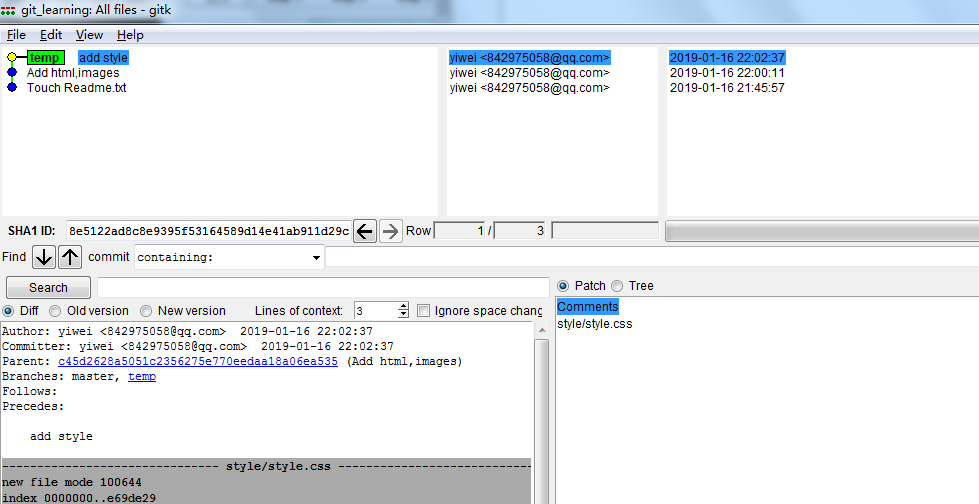
$git log #只能查看当前的分支

$git log -all 查看所有情况

$git log --oneline --all --graph

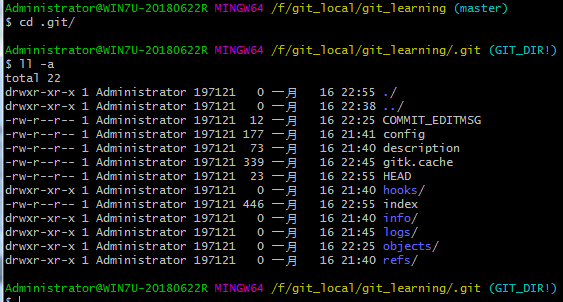
## Gitk:通过图形界面工具来查看版本历史

$gitk



## 探秘.git目录

**.git相关目录:**



**HEAD目录内容:**



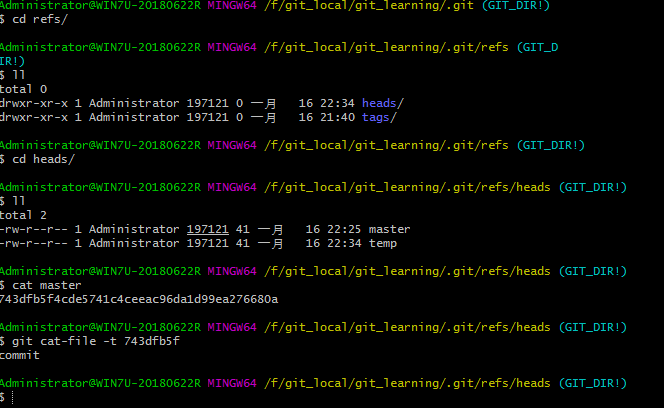
**Config 文件内容** #可以修改这个文件中的内容将影响相关内容



**Refs文件夹里面的内容:**

**heads #代表分支**

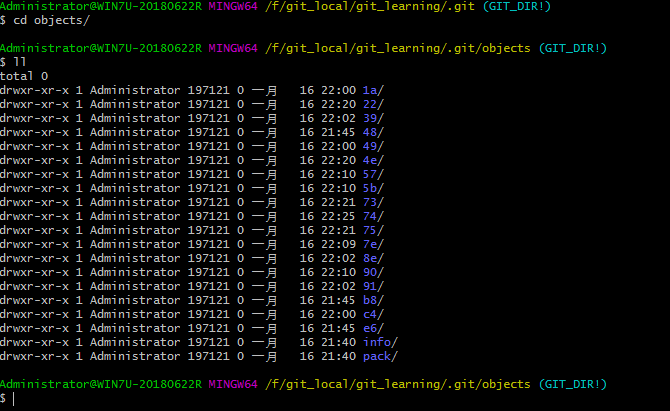
**tags #代表开发里程碑**



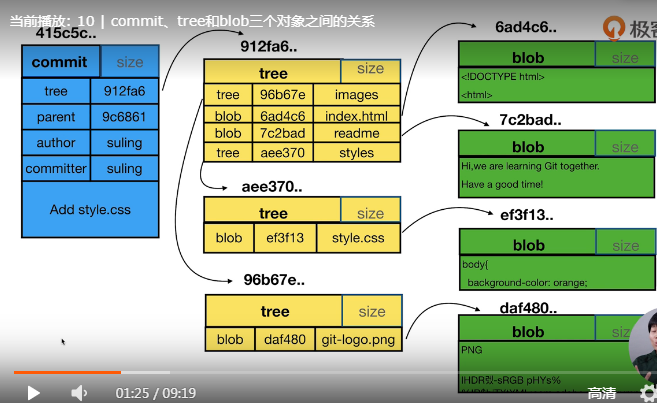
$git cat-file -t xxxx #查看类型

$git cat-file -p xxxx#查看内容

**Objects**/ #里面存放的是我们要用的内容



## Commit,tree,和blob三个对象之间的关系



测试:



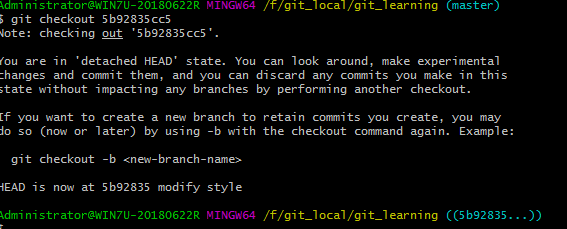


## 分离头指针情况下的注意事项

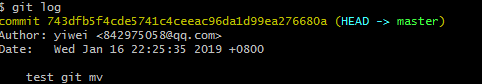
什么叫分离头指针？例如:

当我直接用git checkout 5b92835cc5这样的命令时,将会产生一个没有根节点的分支

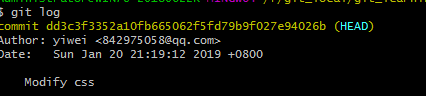
（如果我们在这个时候切换到原来的master分支,我们当前的修改将会被git丢弃）



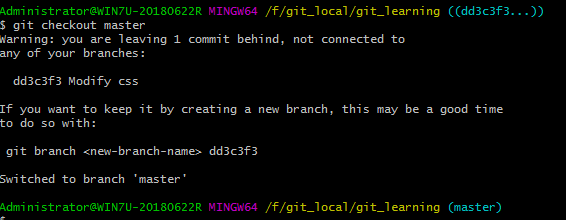
下图为我们没有分离头指针时,HEAD->master



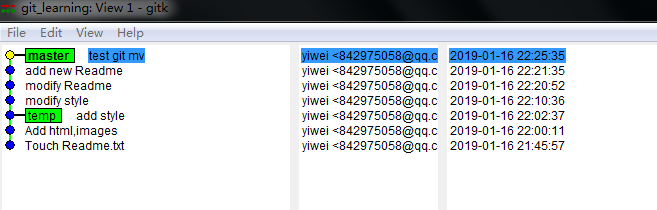
下图为我们分离头指针的时候,HEAD不指向任何分支



当切换回master分支的时候我们将有一下提示消息

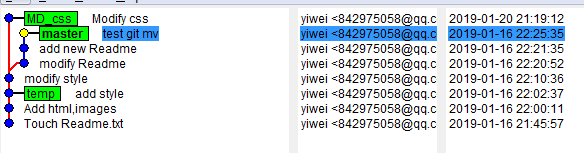


Gitk对比:



$git branch MD\_css dd3c3f

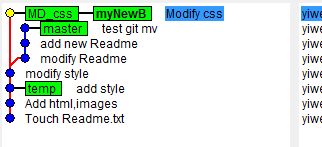
$gitk



## 进一步理解HEAD和branch

$git checkout -b fix\_readme fix\_css

根据fix\_css创建一个新的分支fix\_readme



这是我的HEAD指向我当前分支(黄色节点)

git diff HEAD 0c23c8 ======== git diff 03c54 0c23c8

git diff HEAD HEAD^ #当前分支和其父亲作对比

git diff HEAD HEAD^^ #当前分支和其父亲的父亲作对比

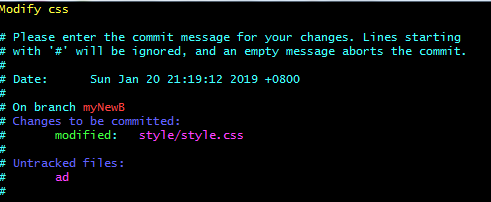
## 怎么删除不需要的分支

$git branch -D 9c681f3a

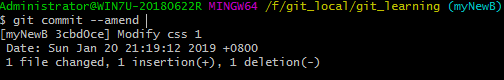
$git branch -D fix\_name

## 怎么修改最新commit的message?

1. git commit --amend #将弹出修改框如下

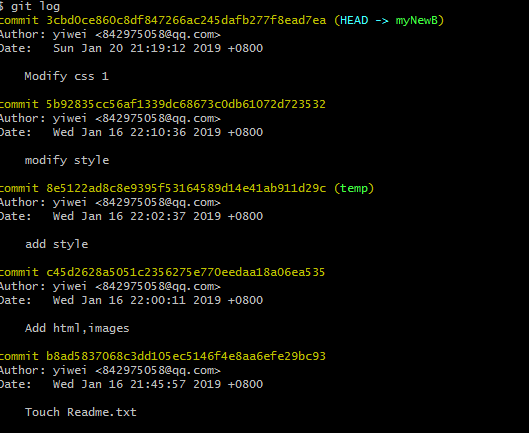


修改结束:

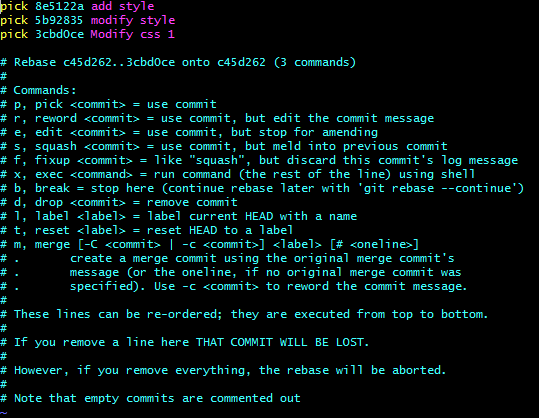


## 怎么修改老旧的commit的message?

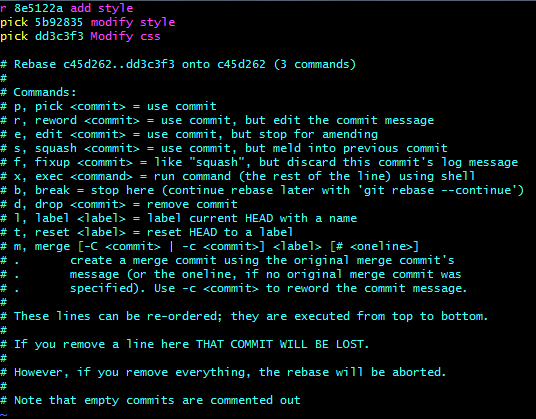
1.挑选要修改的分支信息



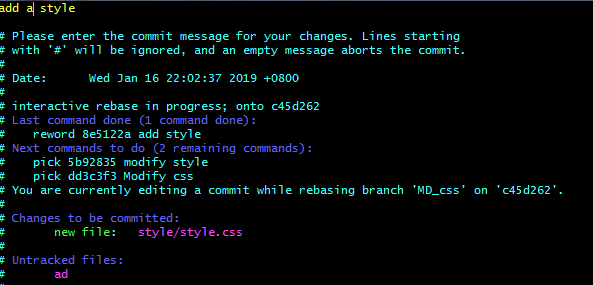
1. git rebase -i 8e5122ad8c



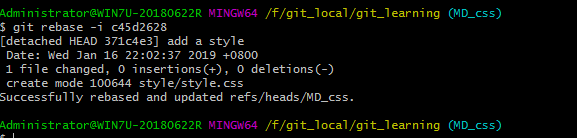
3.选择对应的commit用reword命令



保存上面信息,接着会弹出下面将要修改的信息(保存并退出):

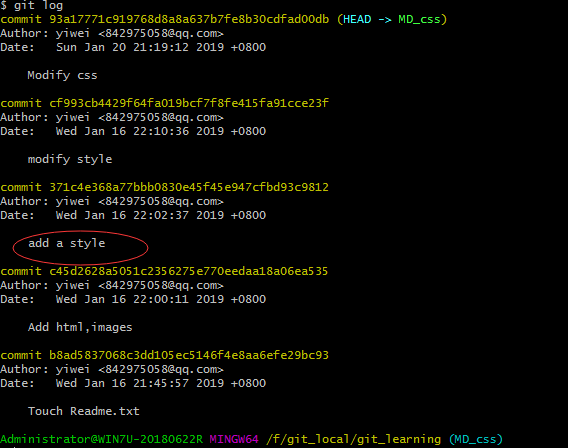


完成:



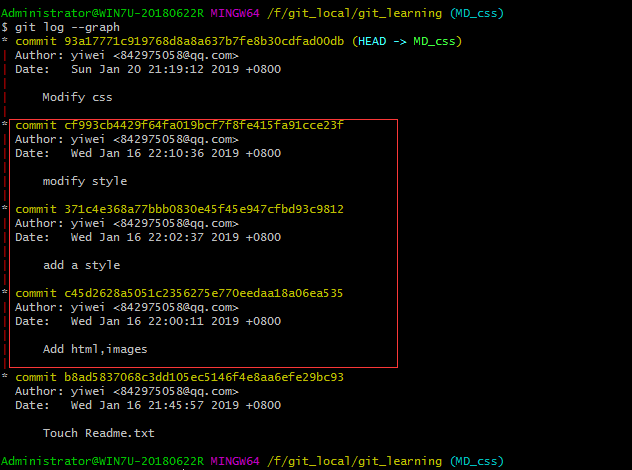
查看修改结果:1.commit信息被修改

2.相关自己与子commit ID均被修改



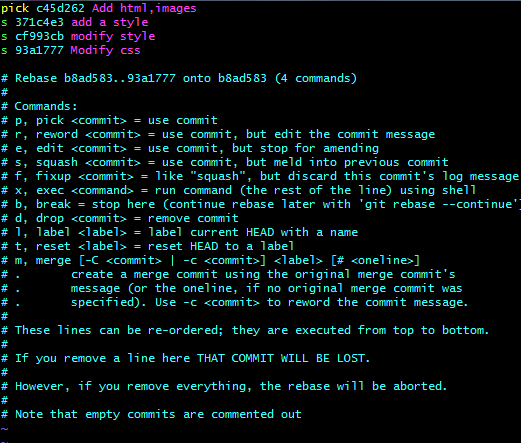
## 怎么把连续的多个commit整理成1个?

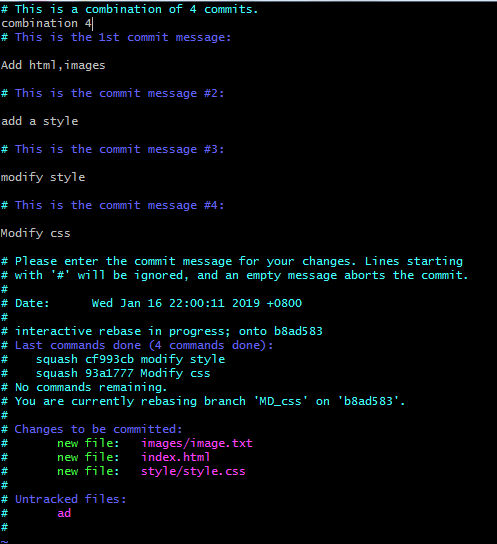
1. 选取要合并的分支:



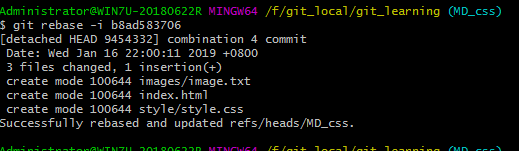
1. git rebase -i b8ad583706

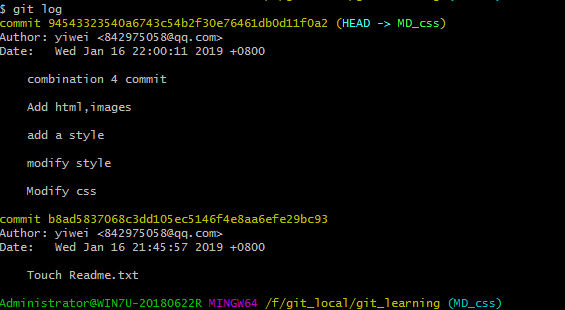
用到squash命令





保存(操作成功)查看结果





## 18.怎么把间隔的几个commit整理成1个?

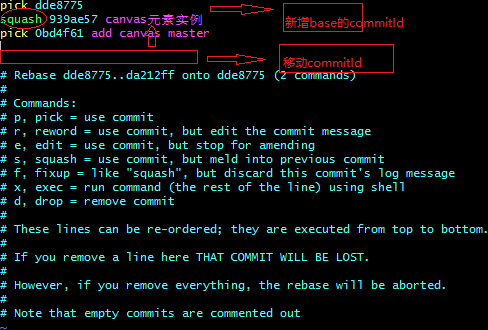
1.选取两个将要融合的commit



2.$git rebase -i dde8775

《1》由于base是最下层的 commit,所以需要新添加pick dde8775

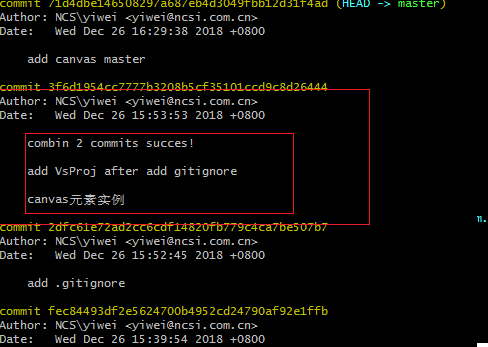
《2》把第三条commit 939ae57移动至第二条并且修改为pick=>squash



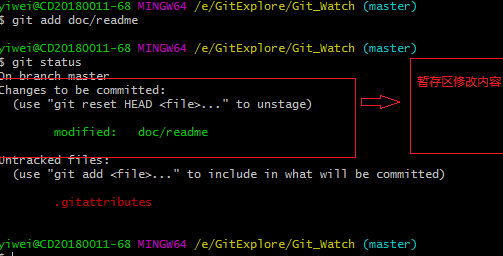
3.$git rebase –continue

添加相应message

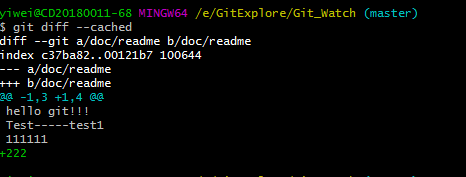
4.查看结果



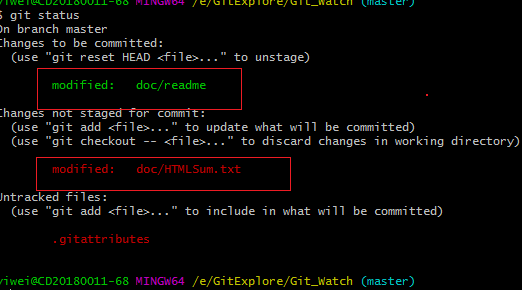
## 19.怎么比较暂存区和HEAD所含文件的差异?



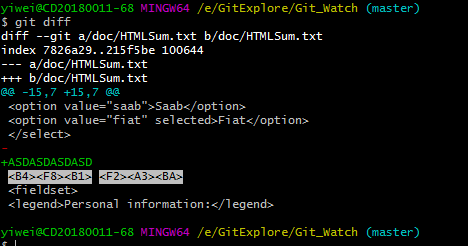
$git diff --cached



## 20.怎么比较工作区和暂存区所含文件的差异?

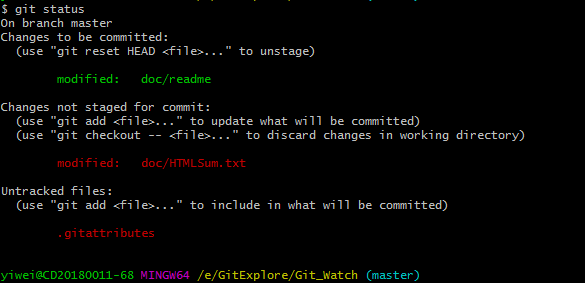


$git diff #比较所有工作区和暂存区的不同

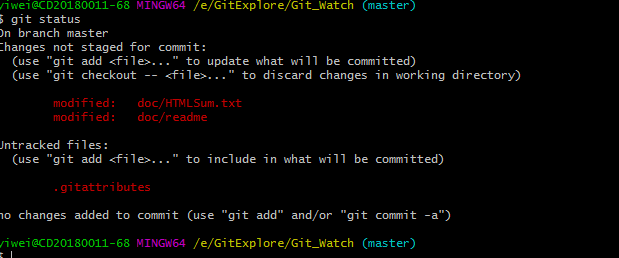


$git diff -- VsProjForHtml/ #只看单个文件

## 21.如何让暂存区恢复成和HEAD的一样?



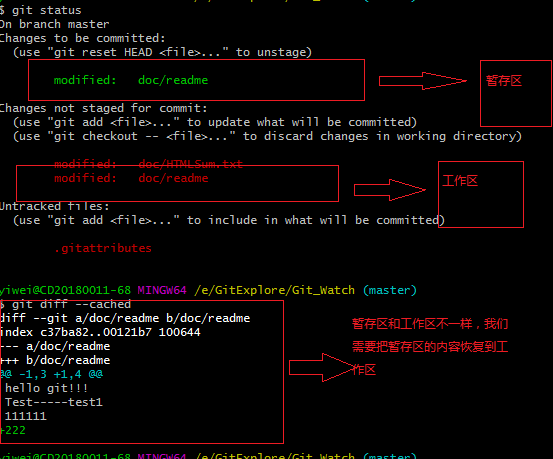
$git reset HEAD #取消暂存区里面的内容恢复成工作区



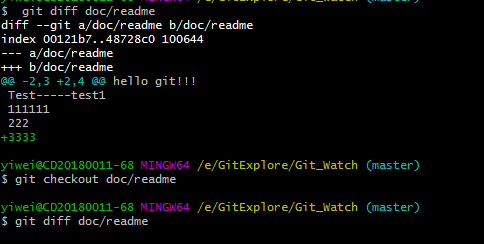
## 22.如何让工作区的内容恢复得和暂存区一样?

#一般我们希望把暂存区的东西恢复成工作区,我们一般用git reset xxx

#我们希望把工作区的东西恢复成暂存区,我们一般用git checkout xxx



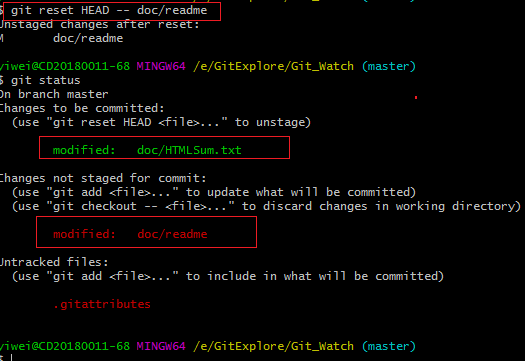
$ git checkout doc/readme



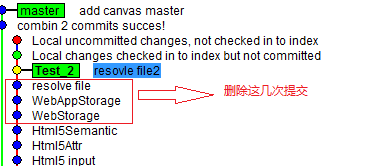
## 23.怎么取消暂存区部分文件的变更?

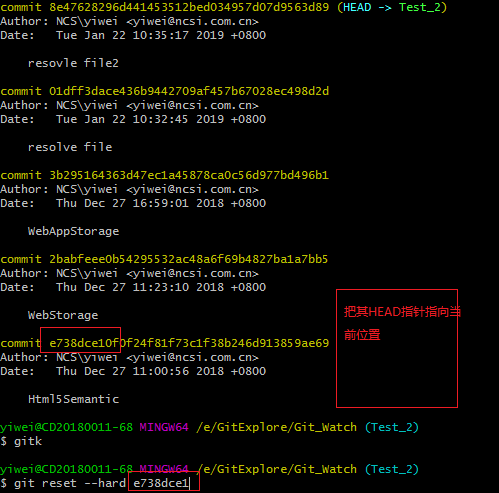


$git reset HEAD --doc/readme

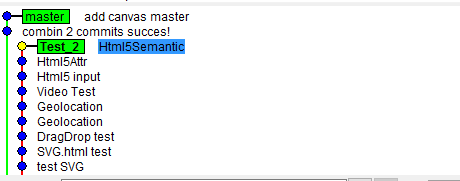


## 24.消除最近的几次提交?





操作结果如下:



## 25.看看不同提交的指定文件的差异