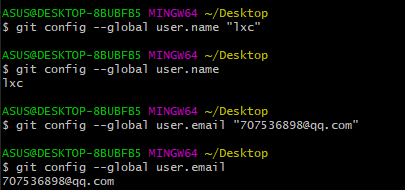
# 配置

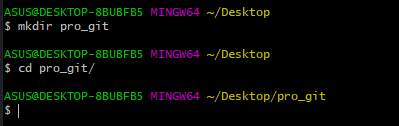
1. 首次使用需进行全局配置

git config --global user.name “用户名”

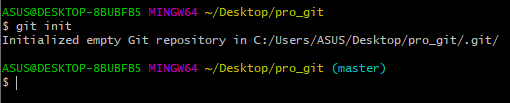
git config --global user.email “邮箱地址”



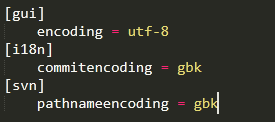
1. 创建仓库



1. git仓库初始化（让git管理该目录）



乱码：



# git常用指令操作

1. 查看当前状态：git status
2. 添加到缓存区：
3. git add 文件名
4. git add 文件名1 文件名2 文件名3
5. git add. (添加当前目录至缓冲区)
6. 提交至版本库：git commit -m “注释内容”



# 版本回退

1. 查看版本，确定需要回到的时刻点

指令：

git log

git log --pretty=oneline

1. 回退操作

指令：

Git reset --hard 版本号

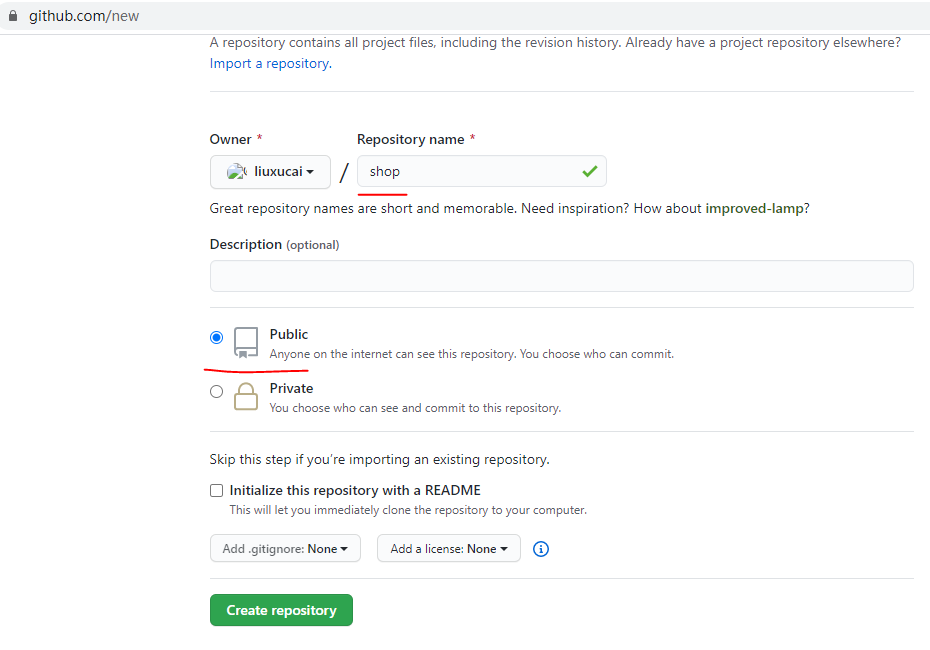
1. 想回到之前最新的版本，需使用指令查看历史操作，以得到最新的版本号

指令：

Git reflog

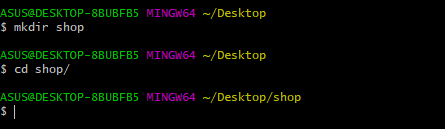
1. 版本号可不必写全，但至少需要4位

# 创建线上仓库



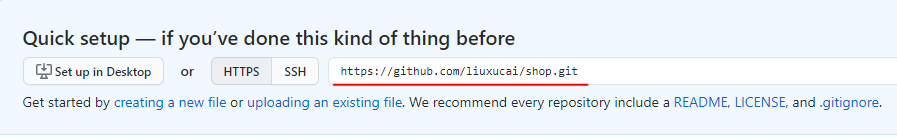
# 基于http协议

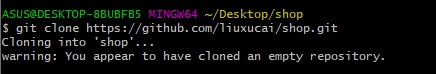
1. 创建空目录，名称为shop



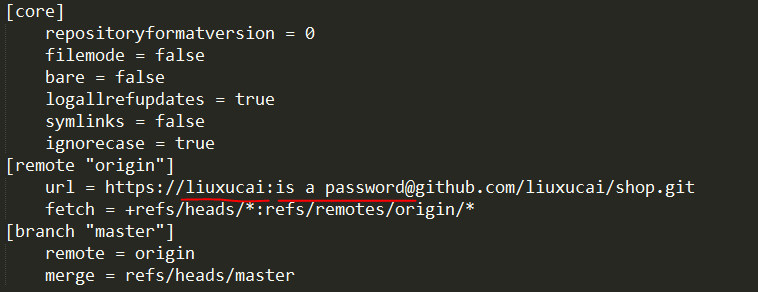
1. 使用clone指令克隆

语法：git clone 线上仓库地址

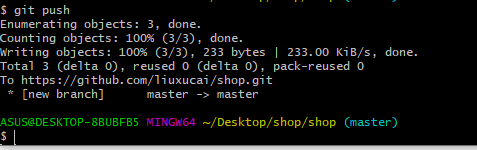




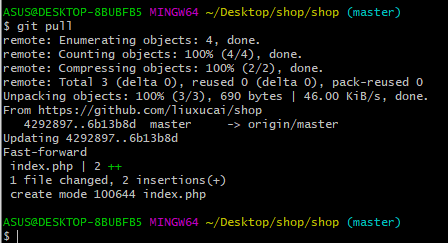
1. 首次往线上仓库提交内容前需配置.git/config文件



1. 使用指令git push 即可提交到线上仓库

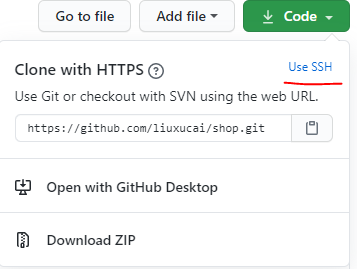


1. 使用指令git pull 拉取线上仓库



# 基于ssh

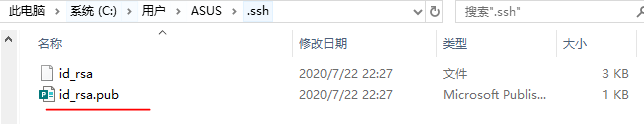
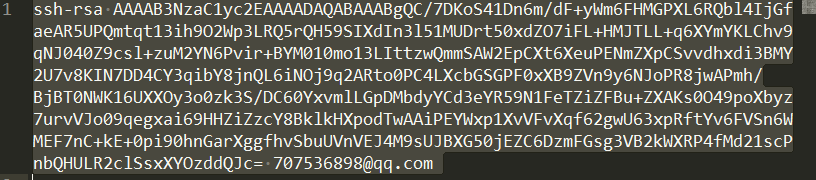
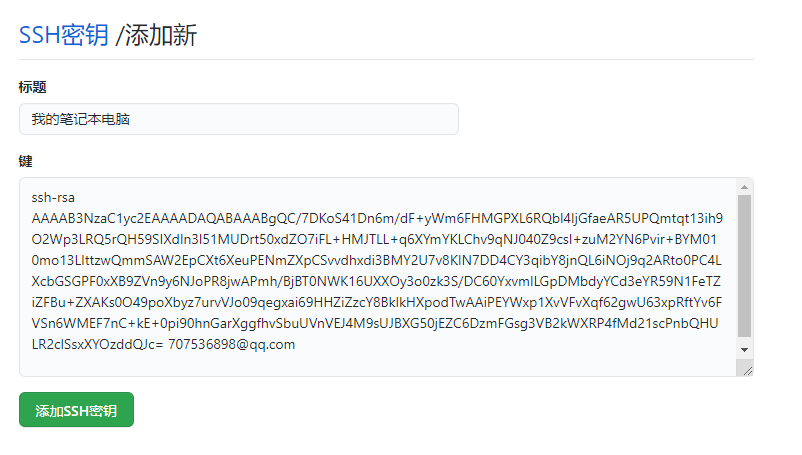
1. 打开提示



1. 创建公私钥对文件



1. 打开对应文件，复制并上传公钥文件内容

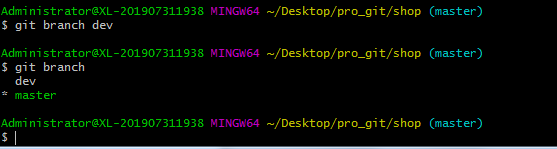
  

1. 使用指令 git clone 仓库地址 即可克隆线上仓库

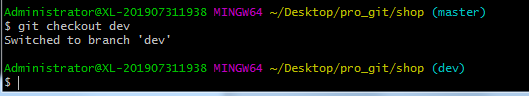


# 分支

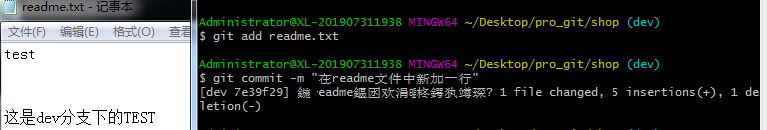
1. 查看分支： git branch
2. 创建分支： git branch 分支名

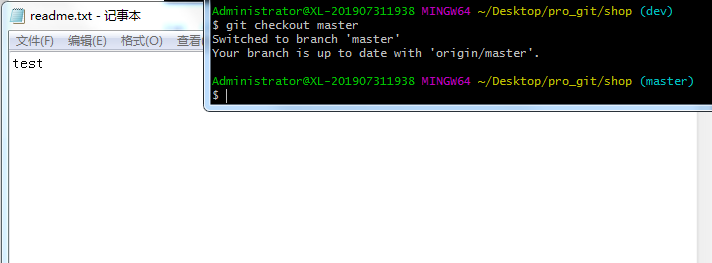


1. 切换分支： git checkout 分支名

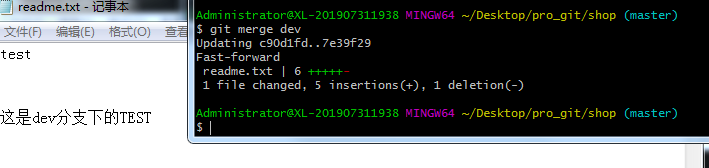


1. 合并分支： git merge 被合并的分支名

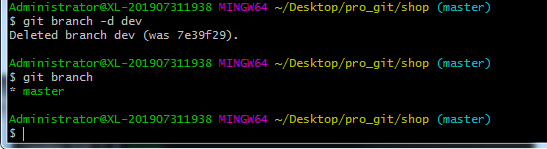




在dev分支中插入的文字已消失



1. 删除分支： git branch -d 分支名



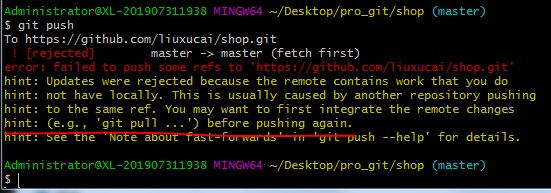
注意：在删除分支时，一定要退出要删除的分支，然后才能删除

1. 创建并切换分支： git checkout -b 分支名

# 冲突的产生于解决

案例：模拟产生冲突

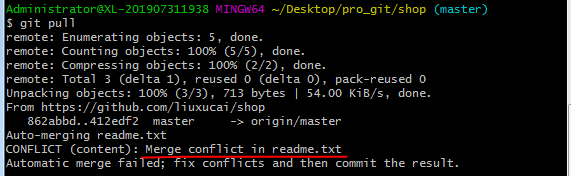
1. 同事在下班之后修改了线上仓库的代码，此时我本地仓库的内容与线上不一致
2. 第二天上班时，我没有git pull （拉取线上内容），而是直接修改了本地对应文件的内容
3. 在下班时，执行git push（提交本地内容至线上）



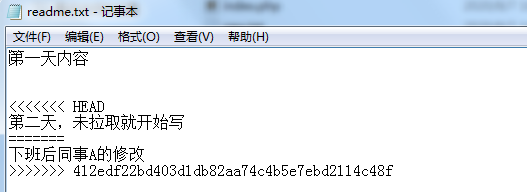
提升要在push之前先pull

解决冲突：

1. 先git pull



提示已合并冲突



1. 打开冲突文件，解决冲突（与同事商量）后，重新提交。

# 图形化工具

1. Github for Desktop
2. Source tree
3. TortoiseGit



# 忽略文件

某些文件夹或文件(如images、js、css)千年不变，或不想提交到线上，即可忽略。

1. 新建一个名为.gitgore的文件，用于声明忽略文件或不忽略文件

(因该文件没有后缀，需用命令行创建)

1. 常见规则写法：

/mtk/ 过滤整个文件夹

\*.zip 过滤所有.zip文件

/mtk/do.c 过滤具体某个文件

!Index.php 不过滤具体某个文件

在文件#开头的都是注释

