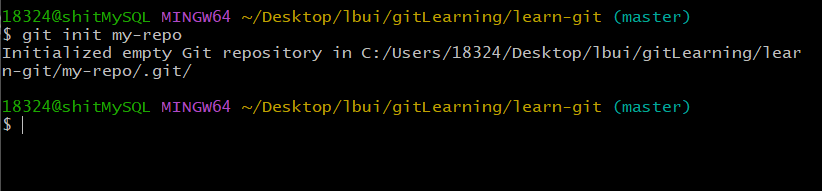
**Git学习**

P3新建仓库

git init my-repo创建名为my-repo的仓库

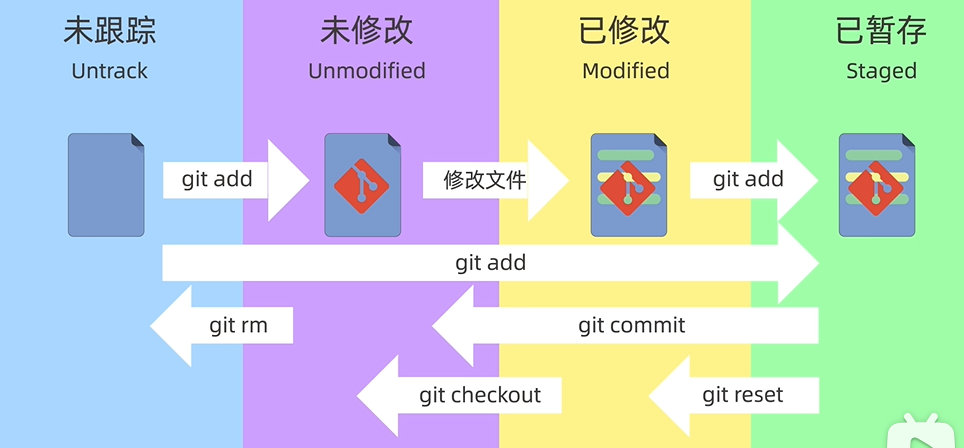


git clone 克隆远程仓库

文本

描述已自动生成

P4 工作区域和文件状态



P5 添加和提交文件

git status查看文件状态，红色表示文件未被跟踪

手机屏幕的截图

描述已自动生成

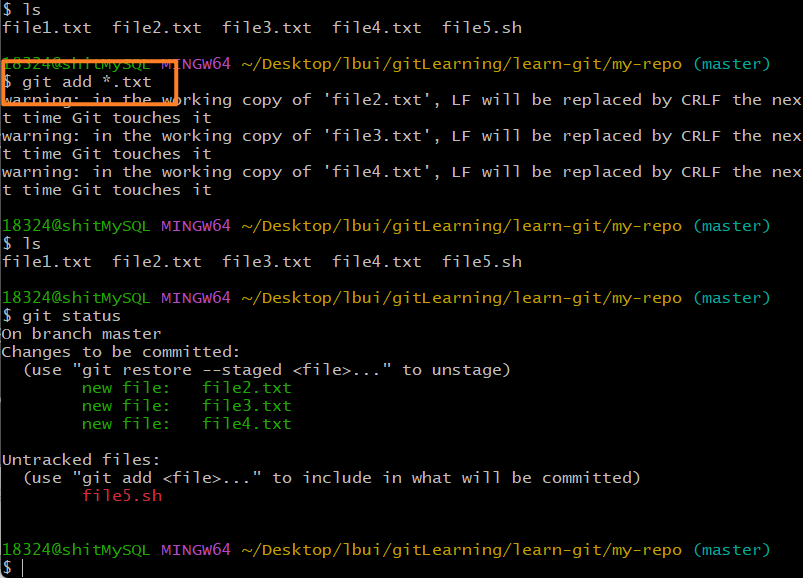
git add 将文件添加到暂存区，显示为绿色

文本

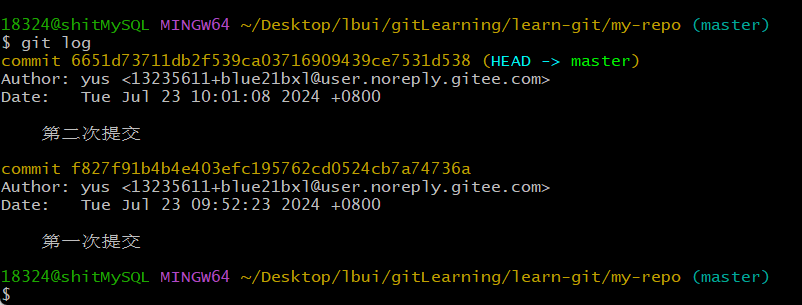
描述已自动生成

git commit -m “提交信息”，git commit只会提交暂存区中的文件，不输入-m提交信息的话会进入vim设置提交信息

git add \*.txt, 通过通配符选择提交内容（‘git add .’提交所有文件）



git log查看提交记录



git log –oneline为更简洁的提交记录

文本

描述已自动生成

P6 git reset 回退版本

git reset --soft 保留工作区和暂存区内容

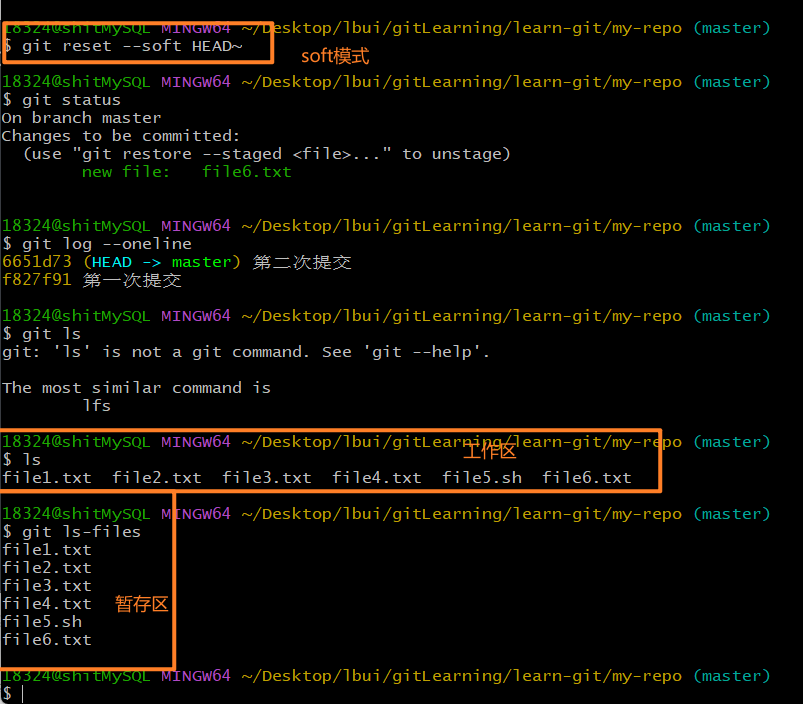
git reset --hard 不保留工作区和暂存区内容

git reset --mixed 保留工作区内容 不保留暂存区内容（默认）

git ls 查看工作区文件

git ls-files 查看暂存区文件

soft模式：



mix模式：

文本, 日程表

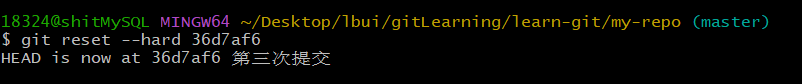
描述已自动生成

hard模式：

文本

描述已自动生成

如果误操作后可以通过git reflog查看版本号，然后通过git reset --hard <版本号>回退到指定版本



P7 使用git diff查看差异

git diff查看工作区和暂存区文件内容的差异

文本

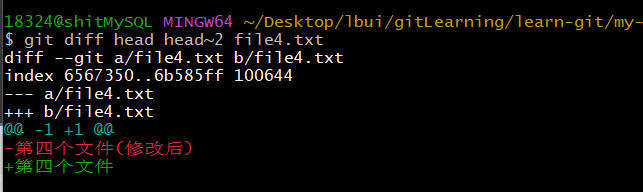
描述已自动生成

也可以通过版本号来查看版本之间的差异

文本

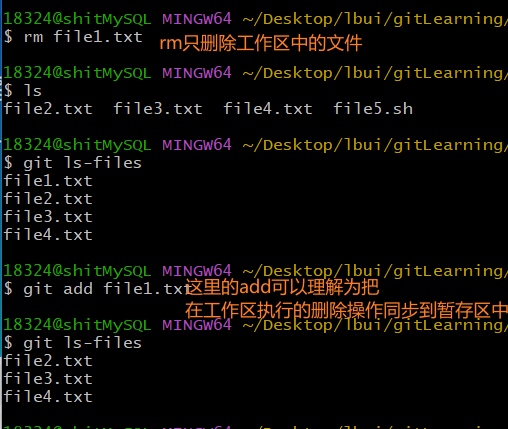
描述已自动生成

通过head、head~2以及文件名指定版本和特定文件的差异



P8 使用git rm删除文件

rm删除工作区文件，add同步删除操作到暂存区



git rm <文件名>,同时删除工作区和暂存区的文件

文本

描述已自动生成

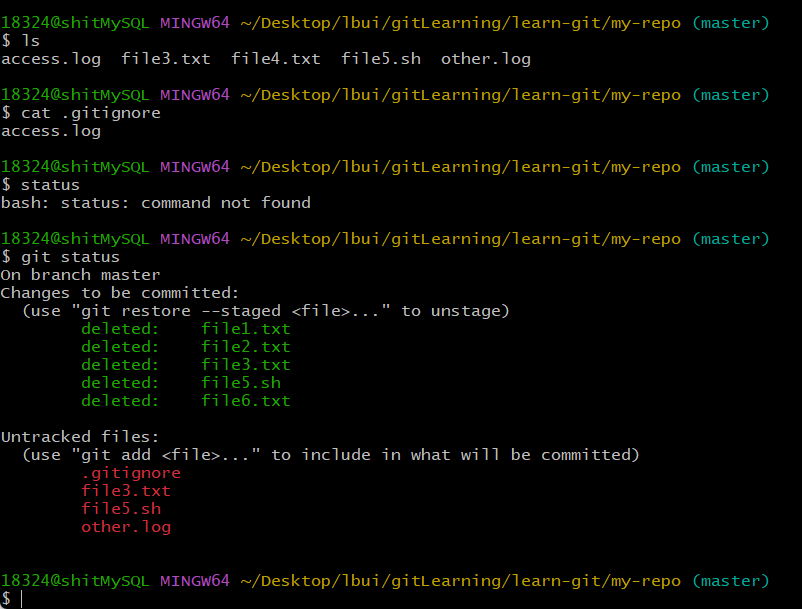
git rm –cache <文件名>,只删除暂存区文件

文本

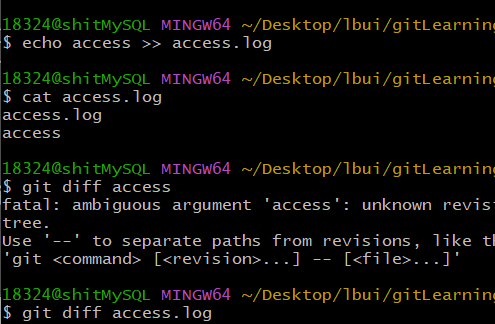
描述已自动生成

P9 gitignore忽略文件

把文件名添加到 .gitignore中之后git status查看不到加入的文件，且无法add到暂存区中，（注：.gitignore对已经添加到版本库中的文件不生效）

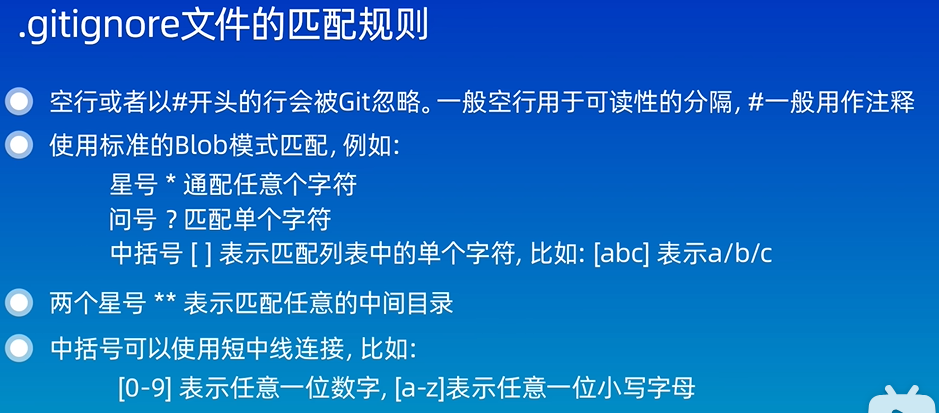


git diff对忽略的文件不生效



echo 111 >> access.log 在文件后追加111

.gitignore文件匹配规则



文本

描述已自动生成

P11 SSH配置和克隆仓库

配制SSH

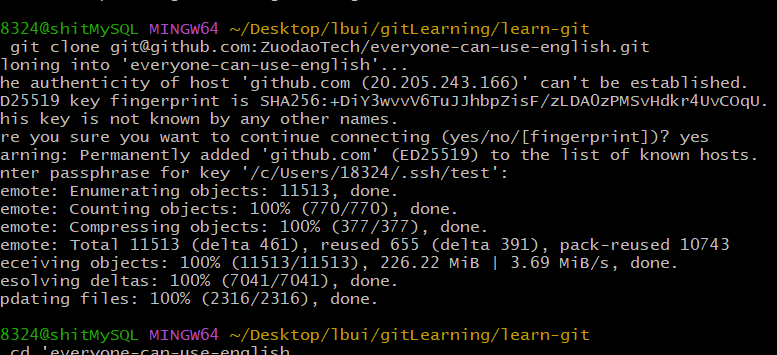
文本

描述已自动生成

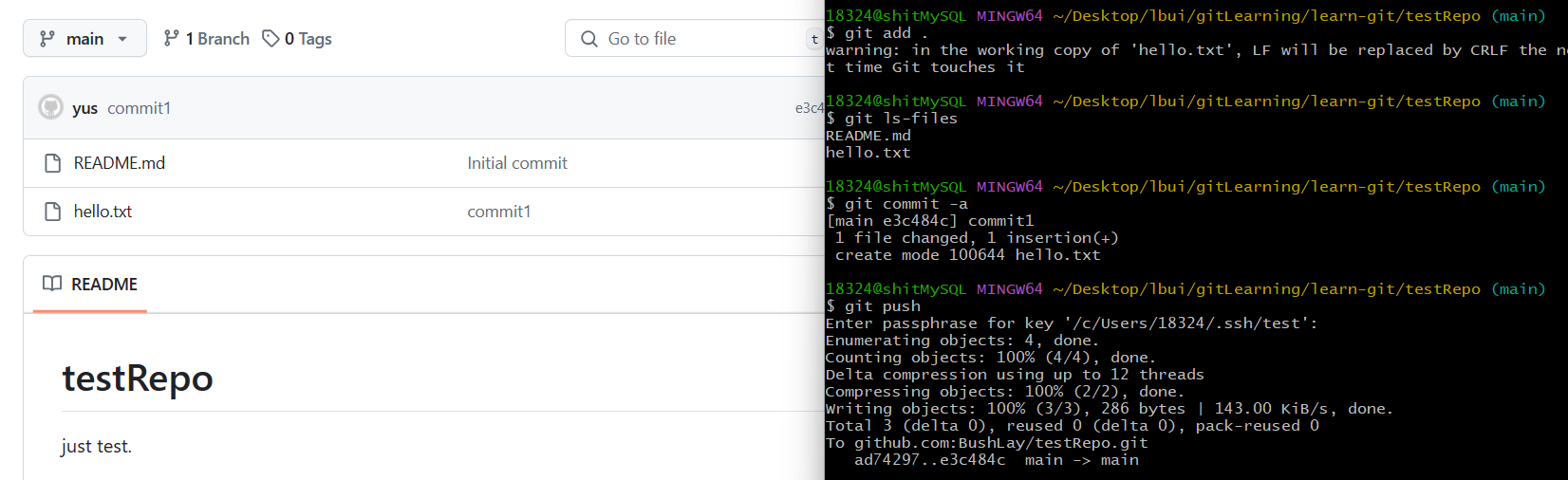
文本

描述已自动生成

通过SSH克隆远程仓库

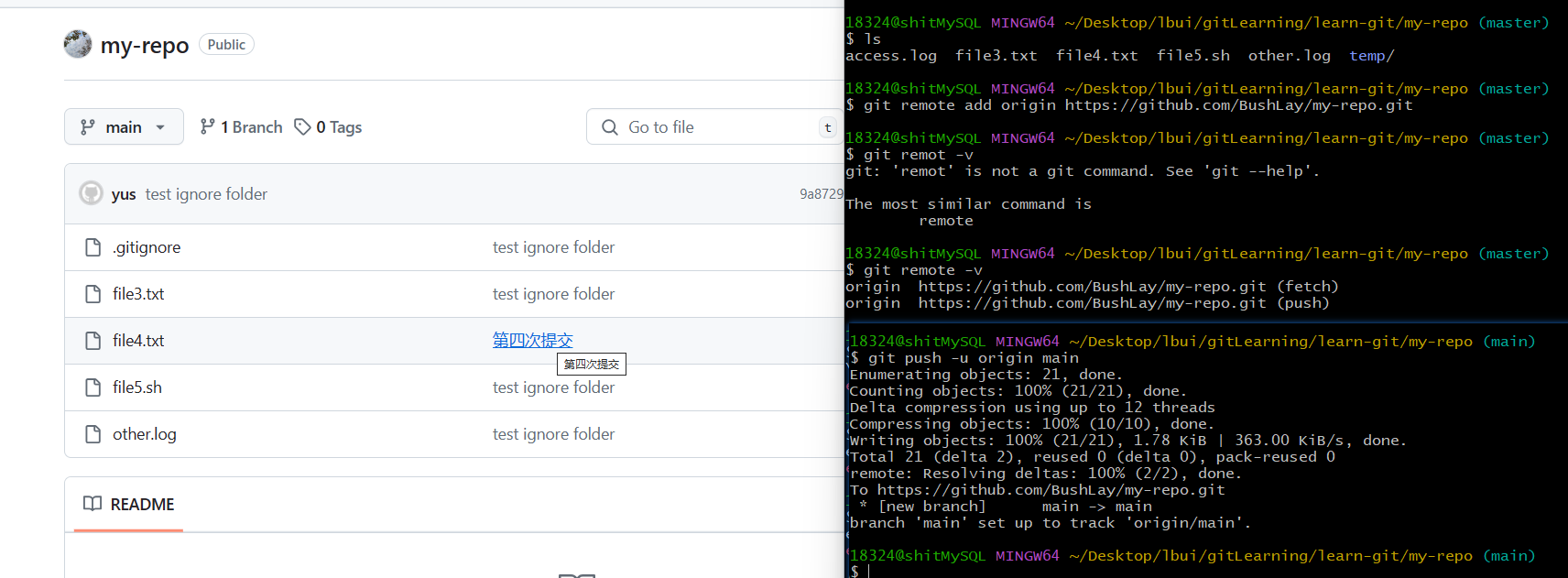


将本地仓库的内容推送到远程仓库中

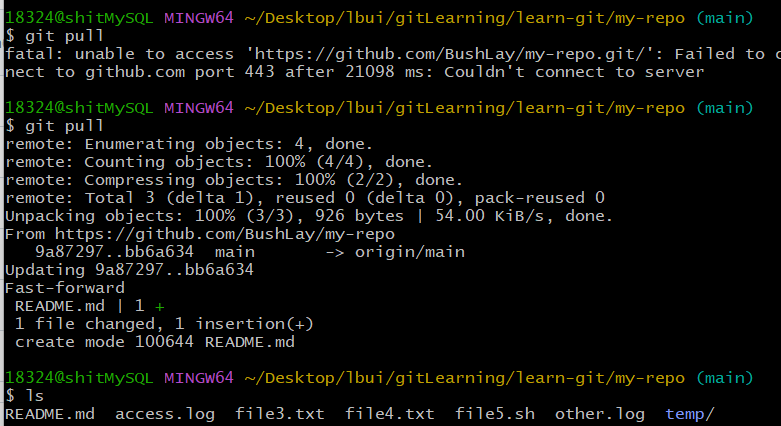


P12 关联本地仓库和远程仓库

将本地仓库部署到远程仓库



使用pull将远程仓库的文件拉取到本地



P15 在VSCode中使用Git

常见标识



图形用户界面, 应用程序

描述已自动生成

P16 分支简介和基本操作

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

git branch dev 新建名为dev的分支

git switch dev 选择dev分支

git merge dev 将dev分支合并到当前所在分支

图形用户界面, 文本, 应用程序

描述已自动生成

git branch -d dev 删除dev分支（注：如果dev分支没有被合并的话，不可以用-d删除，而必须使用-D强制删除）

图形用户界面, 文本, 应用程序

描述已自动生成

P17 解决合并冲突

产生冲突：

在两条分支修改同一行内容，并提交

合并分支时会报错

解决冲突：

报错后vim文本时发现文本内容多出来

<<<<<<< HEAD

=======

>>>>>>> new\_branch

在这个文件内修改，修改的内容会被当作最终合并后内容

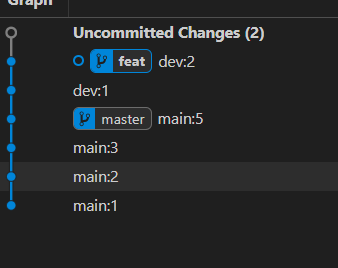
P18 回退和rebase

git reset --hard a6e35b43 回退到之前的版本

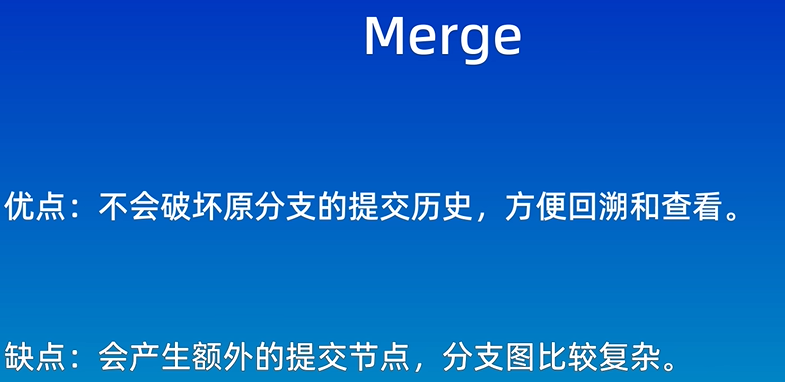
git switch feat

git rebase master

将feat分支变基到master（即将分叉之后的feat分支接到master分支之后）



git merge和git rebase的优劣

图形用户界面, 文本

描述已自动生成

P19 分支管理和工作流模型

分支分为主要分支（如main分支和develop分支）和其他分支,主要分支只能通过合并其他分支来修改

分支命名规范：

