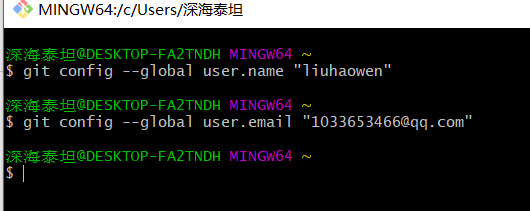
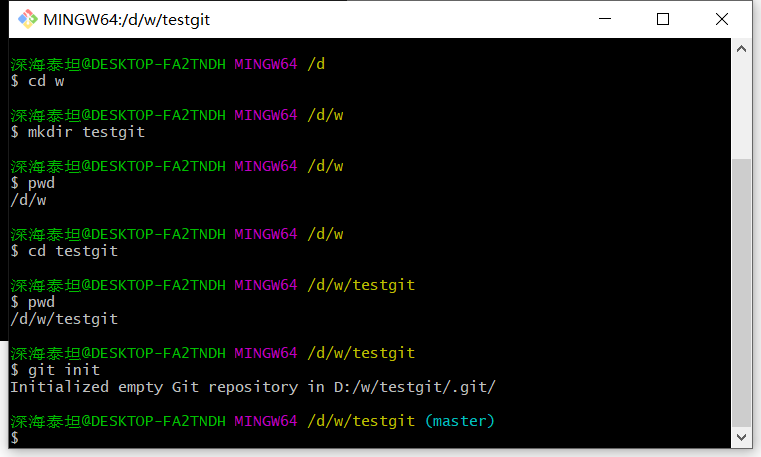
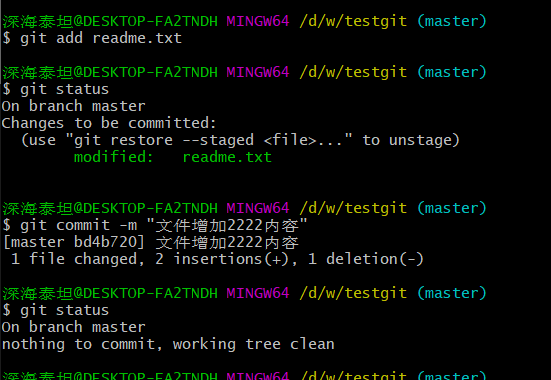
在Git分布式版本控制系统中填写用户名和邮箱



建立Git可以管理的仓库



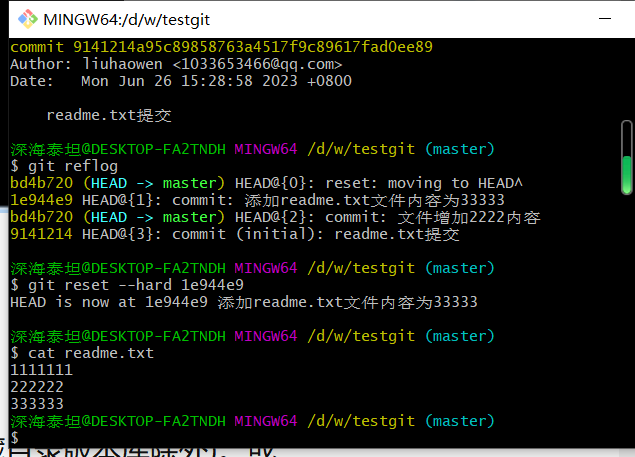
修改和提交文件



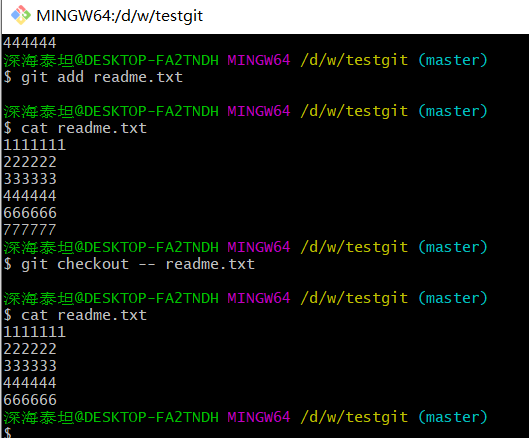
查找历史记录



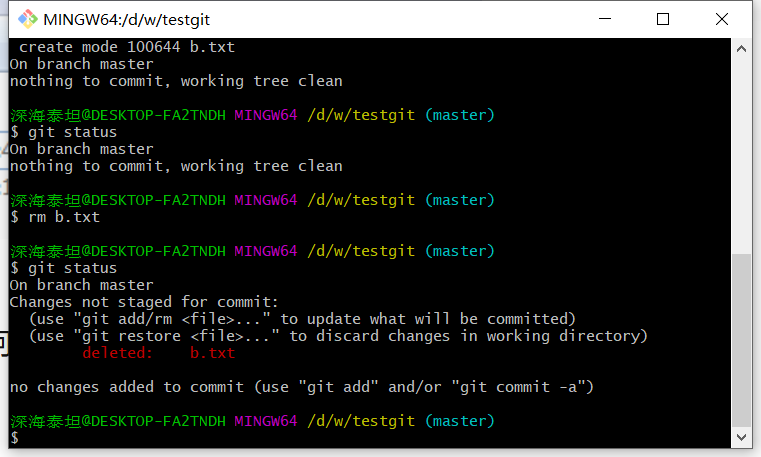
删除和恢复版本



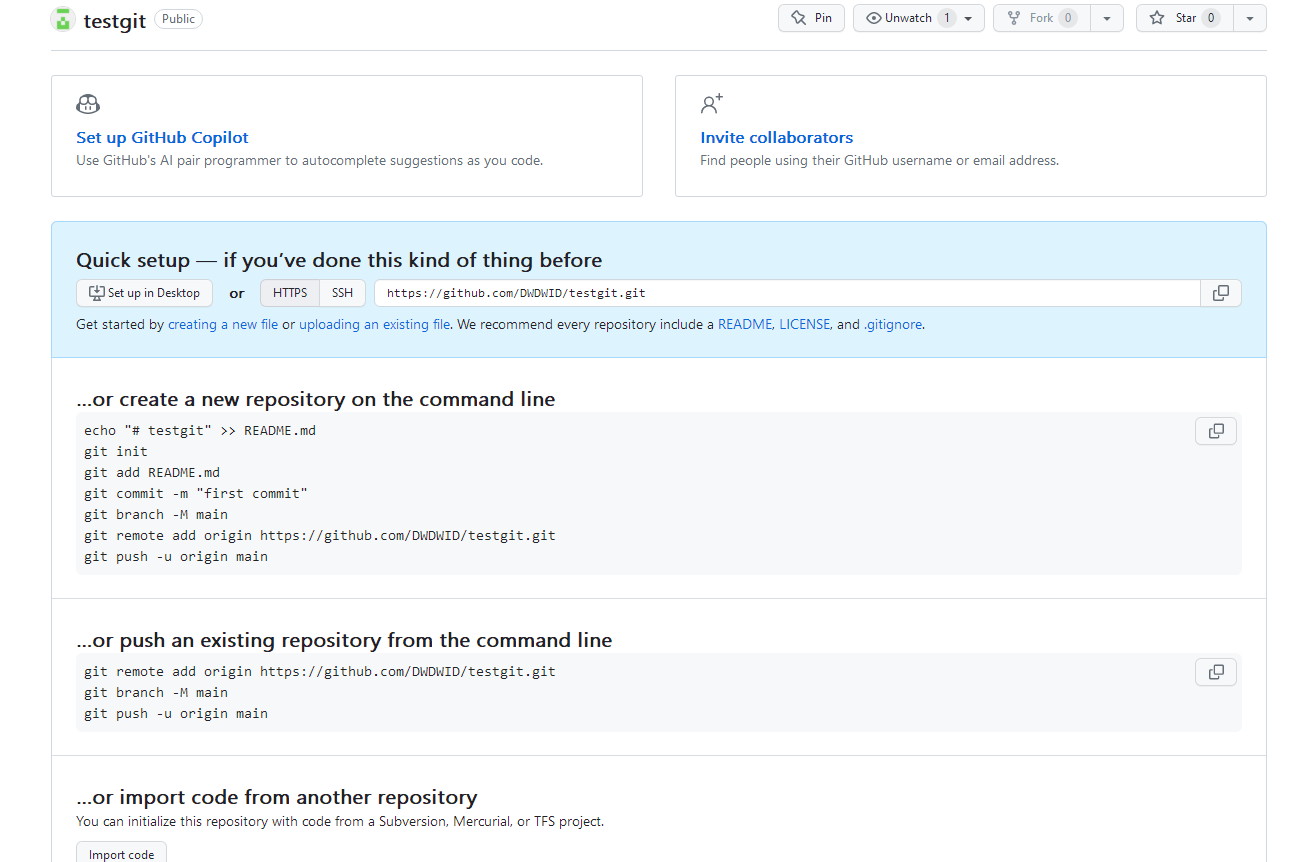
撤销修改文件



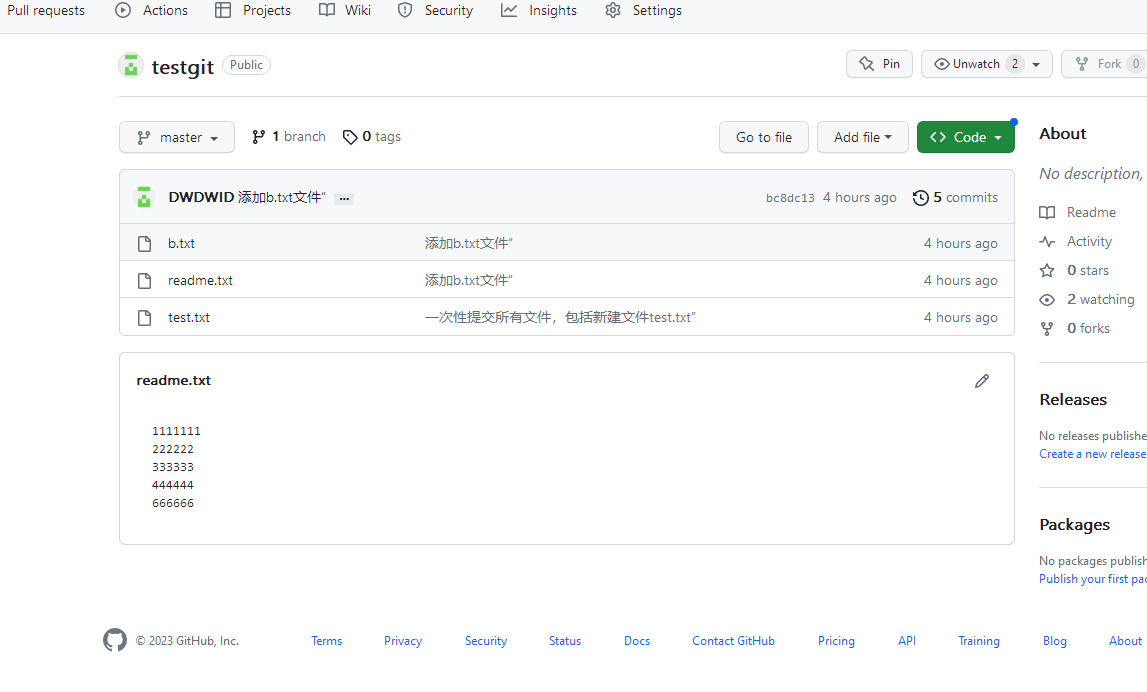
删除文件



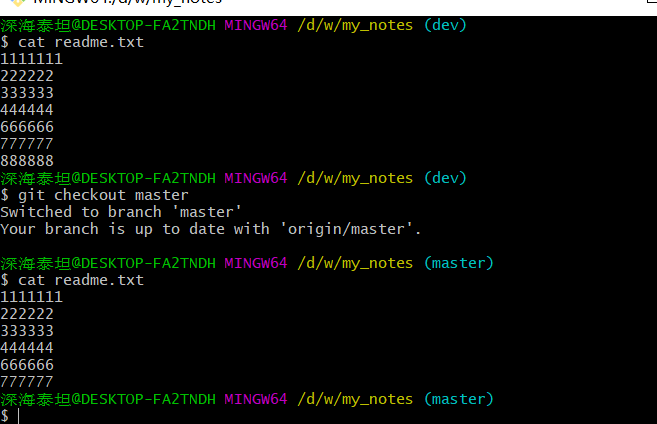
在github上创建一个新的git仓库



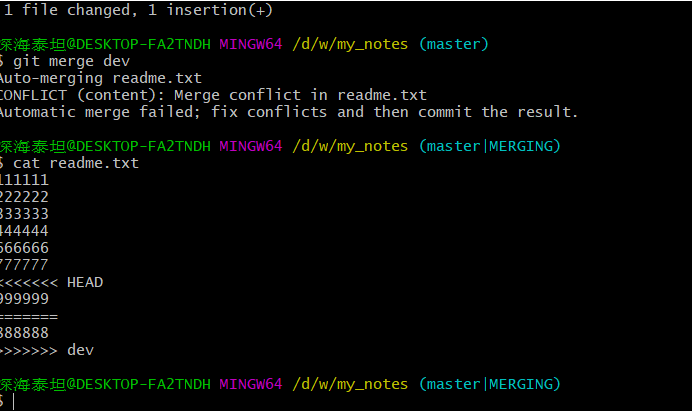
将github仓库和本地仓库相关联，并且将本地仓库的内容推送到github仓库中



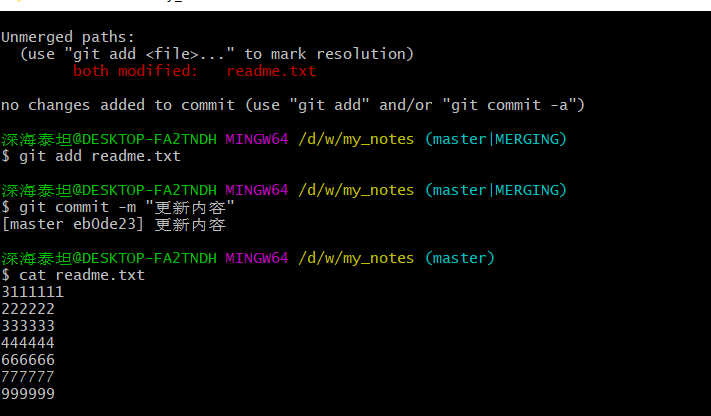
在分支上操作的结果



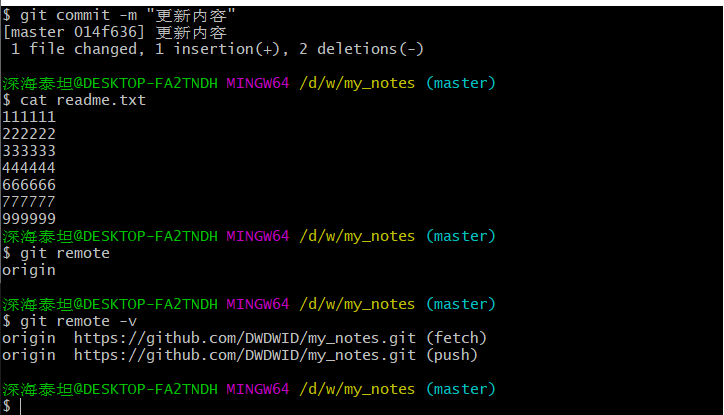
在分支处理中与主干发生冲突



解决冲突后

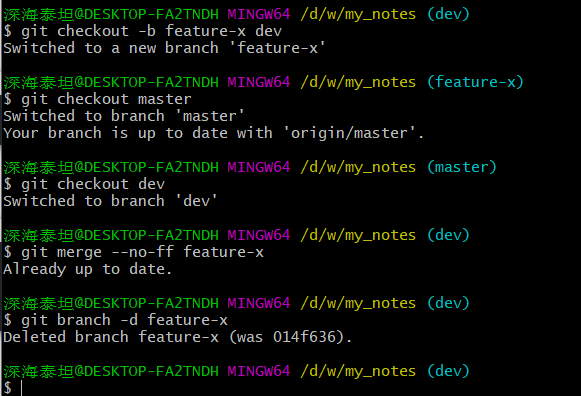


查看远程库的信息以及详细信息



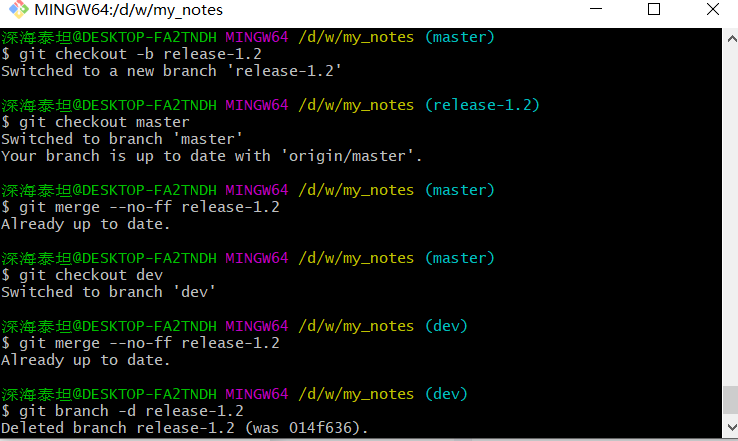
在日常开发的时候主分支只用来发布重大版本，日常开发应该在另一条分支上完成。我们把开发用的分支，叫做Develop。这个分支可以用来生成代码的最新隔夜版本（nightly）。如果想正式对外发布，就在Master分支上，对Develop分支进行"合并"（merge）。并且在合并的时候最好要添加--no-ff参数，只为保证版本演进的清晰，并且还有三个临时分支用于应对特定目的的版本开发。

建立功能分支



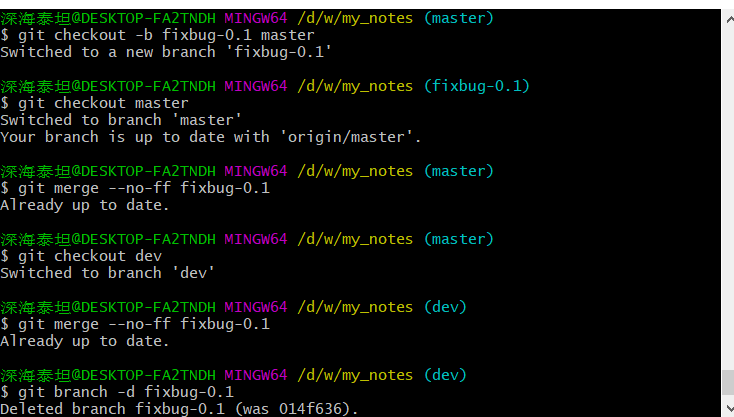
实现预发布分支

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。



修补bug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。



Github仓库快速导入Gitee及同步更新

