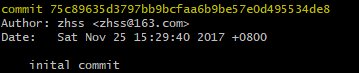
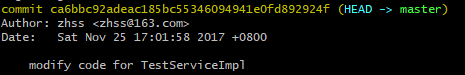
1、基于git log查看提交历史

在我们提交了很多次之后，可以查看提交历史，使用git log命令即可，可以看到每次commit的信息，包括了commit的SHA-1、作者、日期、提交说明。

包含了一个commit的SHA-1 hash值，40位的字符；作者，从之前安装好git之后做的设置来的；date；提交备注，inital commit，git commit -m ''



通过git log命令，我们可以看到到目前为止，一共是10次提交，尤其注意看一下最近一次提交



（HEAD -> master）是什么意思？

版本控制系统中，都有一个概念，叫做分支，就是同样的一份儿代码，可以拉取多个分支，每个分支的代码刚开始都是一样的，然后不同的人可以基于不同的分支去开发，然后开发完之后，再将多个分支的代码合并在一起

分支，就是一份代码拷贝，基于这份独立的代码拷贝，去写自己的代码

最后将多个分支合并在一起，就是将不同的人写的代码合并到一起

git里面，每次git init之后，默认初始就创建一个分支，叫做master分支

我们每次写代码，然后git commit提交，都会形成一次新的commit，然后默认就是在master分支对应的代码拷贝上，进行代码修改和提交的

每次提交之后，master分支就会对应着最新的一次提交

HEAD，指针，指向了我们当前所处的分支，因为当前我们默认就处于master分支上，所以HEAD指针就是指向master的

git log --patch -2，--patch可以显示每次提交之间的diff，同时-n可以指定显示最近几个commit。这个是很有用的，可以看最近两次commit之间的代码差异，进行code review是比较方便的。

用git log --stat，可以显示每次commit的统计信息，包括修改了几个文件，有多少行插入，多少行删除。

用git log --pretty=oneline，可以每个commit显示一行，就是一个commit SHA-1和一个提交说明。用git log --pretty=format:"%h - %an, %ar : %s"，可以显示短hash、作者、多长时间以前、提交说明。

用git log --oneline --abbrev-commit --graph，这是最有用的，可以看到整个commit树结构，包括如何合并的，就显示每个commit的SHA-1和提交说明，同时SHA-1显示短值。

--oneline：显示一行，不要显示多行那么多东西，一行里，就显示commit的标识符，SHA-1 hash值，40位的；提交备注；显示分支和HEAD指向哪个commit

--abbrev-commit：commit的标识符，每一次commit，都有一个唯一的标识符，就是一个SHA-1 hash值，40位，显示一个短值，默认显示前7位，就是说前7位就可以唯一定位这个commit了，不需要完整的40位

--graph：显示图形化的commit历史，这个大家后面学习到分支那里就知道了，如果有分支的话，commit历史会形成一棵树的形状，这个时候用--graph可以看清楚这颗commit树长什么样子，很有的

告诉大家，通过git log命令，就可以看到你多次往git仓库中提交不同代码版本的整个历史

2、深入图解git提交历史

就是你每次git add之后，git commit之后，会在git仓库里存储什么东西，存储的那些东西跟分支还有HEAD之间的关系是什么？？？

用一张图画清楚，工作区、版本库、暂存区、master分支以及HEAD指针的关系

工作区和版本库是两个分离的区域

在工作区修改代码之后，执行git add命令，会将代码放入版本库中的暂存区；接着执行git commit命令之后，会将暂存区中的代码提交到版本库中的master分支上，而HEAD指针就指向master分支的最新一次提交。

所以现在我们就很清楚了，git add，其实就是可以多次修改代码，多次git add，然后将每次修改的代码，都放入暂存区中；而git commit，就是一次性将暂存区中的代码，全部提交到master分支上，master分支会出现一个最新的commit，也就是一个最新的代码版本；而HEAD作为一个指针，永远指向master分支的最新一次commit的代码版本。

853769620