本地Git

刚安装Git需要做的

需要配置用户签名（用户名和邮箱），在提交的时候会连带这些信息一并提交，方便知道代码是由谁写的，以及追责。

git config --global user.name 用户名

git config --global user.email 邮箱

初始化本地库

创建文件夹，右键选择Git Bash Here，在文件夹打开Git命令行窗口

git init 初始化本地库

初始化之后可以使用

git status 查询本地库的状态

新建文件，添加到暂存区

新建文件之后，查询本地库状态时会提示有哪些文件未被追踪，只有被追踪的文件才会加入到"工作区->暂存区->本地库"这一过程当中，Git才会去检测文件是否被修改。

对于未被追踪的文件可以通过

git add **文件名** 来将文件添加到暂存区

此时文件已经到暂存区了，但是也可以通过以下命令从暂存区删除

git restore --staged **文件名**

git rm --cached **文件名**

提交到本地库

对于已经添加到暂存区的文件，可以使用以下命令

git commit -m "**修改日志**"

git commit -a -m "**一次将所有修改过的文件提交本地库**"

修改文件

对于修改过的文件，查看本地库状态时会有标注

需要先使用git add命令将文件添加到暂存区

然后使用提交命令将修改过的文件提交到本地库

（对于已经提交到本体库过的文件，不git add直接git commit也没有问题，但是不推荐这么做）

查看日志

***git reflog* 查看简洁版的日志**

***git log* 查看详细的日志**

版本穿梭

在开发过程中，我们有时候可能会需要返回到之前版本

可以使用git reflog或者是git log查询版本号

```
M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Python/Learn (master)
$ git  reflog
3f21f0d (HEAD -> master) HEAD@{0}: commit: test no add to commit
0ef42af HEAD@{1}: commit: test 2 commit
e2d5c63 HEAD@{2}: commit: test commit
2ec44e3 HEAD@{3}: commit: this is test commit -all
7f6837a HEAD@{4}: commit: third commit
5b234e4 HEAD@{5}: reset: moving to 5b234e45fe3068f2d5d0eba7162eaf052ab241f9
9ae7ee6 HEAD@{6}: commit: second commit
5b234e4 HEAD@{7}: commit (initial): first commit
```

前面有七位字母和数字（简化版，实际版本号更长）组成的版本号，然后使用以下命令

***git reset --hard* 版本号**

来进行版本穿梭

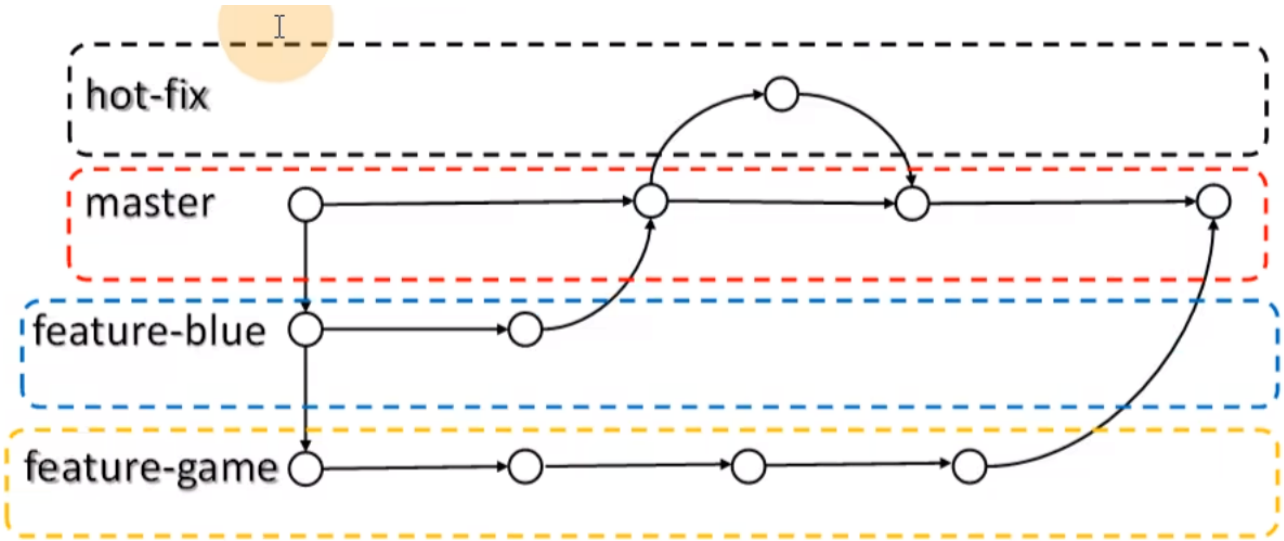
```
M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Python/Learn (master)
$ git reset --hard 3f21f0d
HEAD is now at 3f21f0d test no add to commit
```

```
M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Python/Learn (master)
$ git reflog
3f21f0d (HEAD -> master) HEAD@{0}: reset: moving to 3f21f0d
3f21f0d (HEAD -> master) HEAD@{1}: commit: test no add to commit
0ef42af HEAD@{2}: commit: test 2 commit
e2d5c63 HEAD@{3}: commit: test commit
2ec44e3 HEAD@{4}: commit: this is test commit -all
7f6837a HEAD@{5}: commit: third commit
5b234e4 HEAD@{6}: reset: moving to 5b234e45fe3068f2d5d0eba7162eaf052ab241f9
9ae7ee6 HEAD@{7}: commit: second commit
5b234e4 HEAD@{8}: commit (initial): first commit
```

分支

在实际工作中，会有许多分支，如客户使用的线上分支，测试和开发人员使用的开发分支。同时推进多个任务时为每个任务单独创建一个分支。

程序员在可以把工作从主线上分离开来，开发自己的分支不会影响主线分支的运行，也可以理解为副本。



好处： ①同时并行推进多个功能开发，提高开发效率；②某一个分支开发失败不会影响其它分支，这个分支推倒重来即可。

分支的操作

***git branch* 分支名**　创建分支

git branch -v　查看分支

***git checkout* 分支名**　切换分支

***git merge* 分支名**　把指定分支合并到当前分支

创建分支

```
M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Python/Learn (master)
$ git branch hot-fix
```

查看分支

```
M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Python/Learn (master)
$ git branch -v
  hot-fix 3f21f0d test no add to commit
* master  3f21f0d test no add to commit
```

切换分支

```
M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Python/Learn (master)
$ git checkout hot-fix
Switched to branch 'hot-fix'

M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Python/Learn (hot-fix)
$
```

分支合并（正常合并）

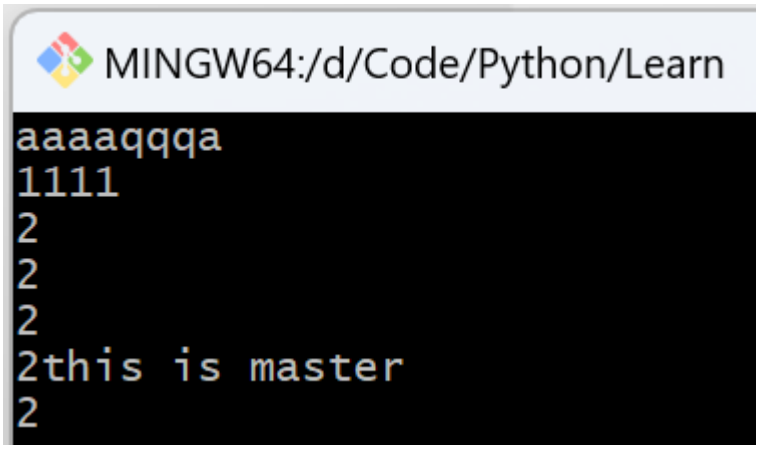
在master下输入以下代码，即将hot-fix合并到master

```
M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Python/Learn (master)
$ git merge hot-fix
Merge made by the 'ort' strategy.
 a.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 a.txt
```

分支合并（冲突合并）

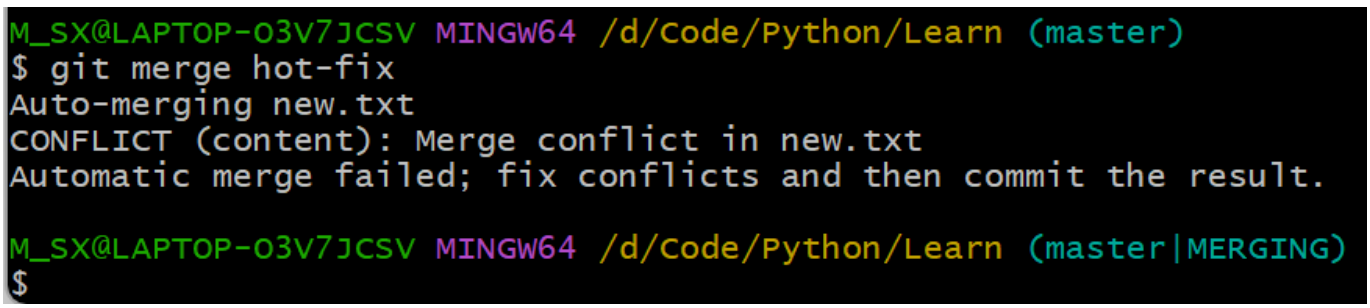
当两个分支在**同一个文件同一个位置**有两套完全不同的修改，Git无法决定使用哪一个，必须认为决定新代码内容

对new.txt在master分支之下作出修改，并提交

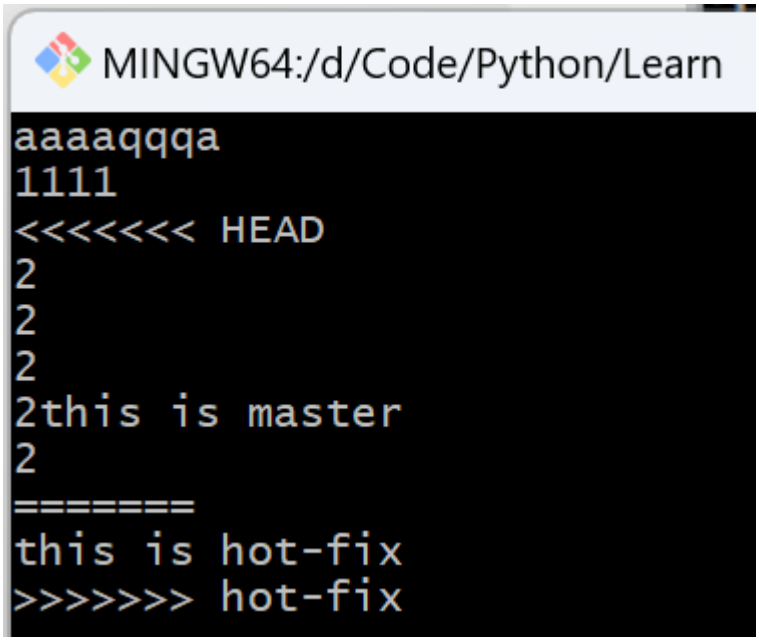


对new.txt在hot-fix分支之下作出修改，并提交

在master下合并hot-fix分支



如图，分支出现了冲突提示，master后面也有“MERGING（合并中）”的提示，需要手动修改new.txt



将new.txt提交到暂存区

```
M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Python/Learn (master|MERGING)
$ git add new.txt
```

将new.txt提交到本地库（**注意后面不需要跟文件名**）

```
M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Python/Learn (master|MERGING)
$ git commit -m "test"
[master e6ea8b7] test

M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Python/Learn (master)
$
```

GitHub的使用

团队内协作（简介）

推送: 本地库-----push----->远程库

克隆： 远程库-----clone----->本地库

拉取： 远程库-----pull----->本地库

git clone 用于新建一个本地库并将远程库的代码克隆到本地

git pull 已有本地库，用于将远程库的最新更改同步到本地库

git fetch 只是获取远程库上的更改信息，并不影响当前工作分支

跨团队协作（简介）

远程库1-----fork----->远程库2

远程库1<-----merge<-----审核<-----Pull request<-----远程库2

别名操作

git remote -v 查看所有远程地址的别名

```
M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Java/learn-test (master)
$ git remote -v
```

git remote add **别名 远程地址** 为远程地址设置好别名

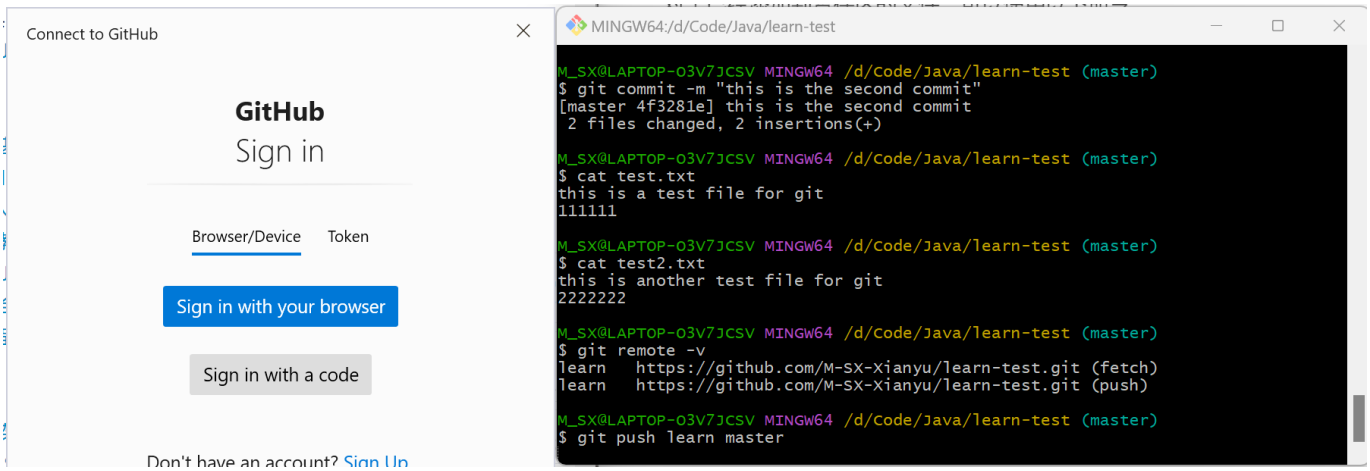
```
M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Java/learn-test (master)
$ git remote add learn https://github.com/M-SX-Xianyu/learn-test.git

M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Java/learn-test (master)
$ git remote -v
learn   https://github.com/M-SX-Xianyu/learn-test.git (fetch)
learn   https://github.com/M-SX-Xianyu/learn-test.git (push)
```

推送到远程库

git push **别名/URL 分支** 把指定分支推送到远程库

第一次push可能会需要登录GitHub账号




```
M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Java/learn-test (master)
$ git push learn master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (1/1), done.
Writing objects: 100% (2/2), 231 bytes | 231.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/M-SX-Xianyu/learn-test.git
   4f3281e..0155638  master -> master
```

从远程库拉取

git pull **别名/URL 分支** 把指定分支从远程库拉取过来

```
M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Java/learn-test (master)
$ git pull learn master
From https://github.com/M-SX-Xianyu/learn-test
 * branch            master     -> FETCH_HEAD
Updating 4f3281e..13a8b6f
Fast-forward
 new.txt   | 1 +
 test.txt  | 2 --
 test2.txt | 2 --
 3 files changed, 1 insertion(+), 4 deletions(-)
 create mode 100644 new.txt
 delete mode 100644 test.txt
 delete mode 100644 test2.txt
```

从远程库克隆

git clone URL 从远程库克隆到本地

```
M_SX@LAPTOP-O3V7JCSV MINGW64 /d/Code/Java/git-clone-test
$ git clone https://github.com/M-SX-Xianyu/learn-test.git
Cloning into 'learn-test'...
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 21 (delta 1), reused 15 (delta 0), pack-reused 0
Receiving objects: 100% (21/21), done.
Resolving deltas: 100% (1/1), done.
```