Git

——使用Git进行版本管理

Section1. 前期准备

Step1.
进入需管理文件夹，并声明管理者ID与email

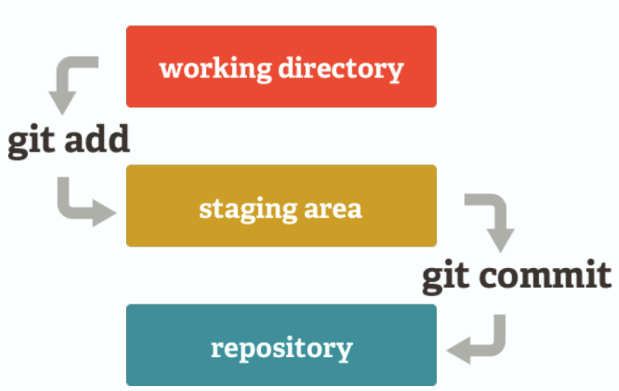
```
➜  Desktop cd CoreCodes
➜  CoreCodes ls
➜  CoreCodes git config --global user.name 'ZhengFang'
➜  CoreCodes git config --global user.email 'fz17@tsinghua.org.cn'
➜  CoreCodes git config user.name
ZhengFang
➜  CoreCodes git config user.email
fz17@tsinghua.org.cn
```

Step2.
在需管理的文件夹下，初始化
git init

```
➜  CoreCodes git init
Initialized empty Git repository in /Users/tsinghuafangzheng/Desktop/CoreCodes/.git/
➜  CoreCodes git:(master) ls -al
total 0
drwxr-xr-x   3 tsinghuafangzheng  staff  102  6 16 14:04 .
drwx------+ 12 tsinghuafangzheng  staff  408  6 16 13:54 ..
drwxr-xr-x  10 tsinghuafangzheng  staff  340  6 16 14:04 .git
➜  CoreCodes git:(master) cd .git
➜  .git git:(master) ls
HEAD        config      hooks       objects
branches    description info        refs
➜  .git git:(master) ls -al
total 24
drwxr-xr-x  10 tsinghuafangzheng  staff  340  6 16 14:05 .
drwxr-xr-x   3 tsinghuafangzheng  staff  102  6 16 14:04 ..
-rw-r--r--   1 tsinghuafangzheng  staff   23  6 16 14:04 HEAD
drwxr-xr-x   2 tsinghuafangzheng  staff   68  6 16 14:04 branches
-rw-r--r--   1 tsinghuafangzheng  staff  137  6 16 14:04 config
-rw-r--r--   1 tsinghuafangzheng  staff   73  6 16 14:04 description
drwxr-xr-x  11 tsinghuafangzheng  staff  374  6 16 14:04 hooks
drwxr-xr-x   3 tsinghuafangzheng  staff  102  6 16 14:04 info
drwxr-xr-x   4 tsinghuafangzheng  staff  136  6 16 14:04 objects
drwxr-xr-x   4 tsinghuafangzheng  staff  136  6 16 14:04 refs
```

Section2. 常规操作

使用git管理代码，分成3个状态：



需 git add，git commit 两步骤，记录一次新版本

SubSection1. 添加文件追踪信息
git add filename

SubSection2. 提交更新
git commit -m information

SubSection3. 查看版本日志
git log

SubSection4. 查看不同
git diff
git diff —cached（后有叙述，不算add的比较）

SubSection5. 查看状态
git status
git status -s（查看缩写版的状态，红色M表示有更改尚未add与commit，绿色M表示有更改已add但尚未commit）

e.g.
通过测试，可以看出，
修改需要通过add和commit两步，新版本才会被记录下来

```
➜ CoreCodes git:(master) ✗ vim ReadMe.txt
➜ CoreCodes git:(master) ✗ git add ./*
➜ CoreCodes git:(master) ✗ git commit -m 'creat ReadMe.txt'
[master (root-commit) 8b019f5] creat ReadMe.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ReadMe.txt
➜ CoreCodes git:(master) git log
➜ CoreCodes git:(master) git status
On branch master
nothing to commit, working tree clean
➜ CoreCodes git:(master) vim ReadMe.txt
➜ CoreCodes git:(master) ✗ git add ./*
➜ CoreCodes git:(master) ✗ vim ReadMe.txt
➜ CoreCodes git:(master) ✗ git diff
➜ CoreCodes git:(master) ✗ git commit -m 'add sentence1, not add sentence2'
[master 6dae8e3] add sentence1, not add sentence2
 1 file changed, 1 insertion(+)
➜ CoreCodes git:(master) ✗ git log
➜ CoreCodes git:(master) ✗ git diff
➜ CoreCodes git:(master) ✗ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   ReadMe.txt

no changes added to commit (use "git add" and/or "git commit -a")
➜ CoreCodes git:(master) ✗ git add ./*
➜ CoreCodes git:(master) ✗ git commit -m 'add again'
[master 7513bea] add again
 1 file changed, 1 insertion(+)
➜ CoreCodes git:(master)
```

提交第二次版本时，
Hello World! 被add过，在git diff时，前不显示+；
Git, fighting! 未被add过，在git diff时，前显示+。

```
diff --git a/ReadMe.txt b/ReadMe.txt
index 980a0d5..81b3528 100644
--- a/ReadMe.txt
+++ b/ReadMe.txt
@@ -1 +1,2 @@
 Hello World!
+Git, fighting!
(END)
```

想要看不算add，与前一次commit后的比对，可用git diff --cached

git log 显示版本日志
这是第二次git log时的截图

```
commit 6dae8e3c4cfb736aef87702ecfa05476cd227c7b
Author: ZhengFang <fz17@tsinghua.org.cn>
Date:   Sat Jun 16 23:55:54 2018 +0800

    add sentence1, not add sentence2

commit 8b019f55214c8397d6735c250f8a89806619c794
Author: ZhengFang <fz17@tsinghua.org.cn>
Date:   Sat Jun 16 23:52:02 2018 +0800

    creat ReadMe.txt
(END)
```

section3. 穿梭（版本粒度）

subsection1. 更改已经commit的版本
git commit —amend —no-edit

```
➜ CoreCodes git:(master) vim ReadMe.txt
➜ CoreCodes git:(master) ✗ git add ./*
➜ CoreCodes git:(master) ✗ git commit --amend --no-edit
[master a504156] add again
 Date: Sat Jun 16 23:57:58 2018 +0800
 1 file changed, 2 insertions(+)
➜ CoreCodes git:(master) git log --oneline
```

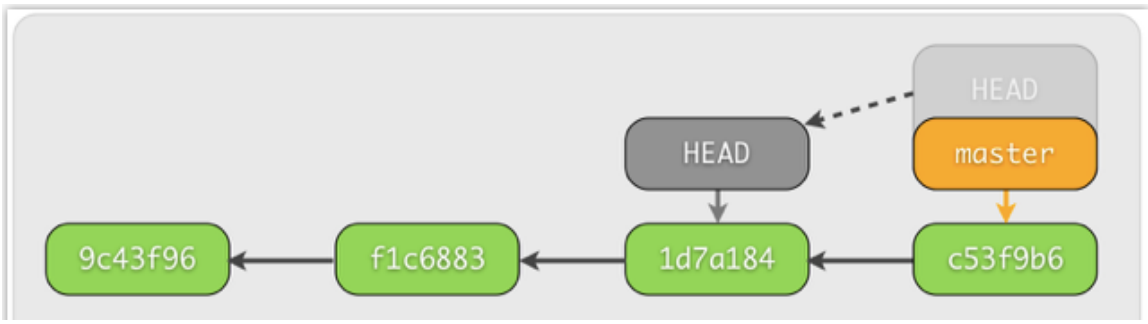
用git log —oneline，
显示得更加简练，
显示的是这样的（add again前面的id是会变的）：

```
a504156 add again
6dae8e3 add sentence1, not add sentence2
8b019f5 creat ReadMe.txt
(END)
```

subsection2. 版本回退
git reset —hard id

```
➜  CoreCodes git:(master) git log --oneline
➜  CoreCodes git:(master) git reset --hard 6dae8e3
HEAD is now at 6dae8e3 add sentence1, not add sentence2
➜  CoreCodes git:(master) vim ReadMe.txt
➜  CoreCodes git:(master) git log --oneline
➜  CoreCodes git:(master) git reflog
➜  CoreCodes git:(master) git reset --hard a504156
HEAD is now at a504156 add again
➜  CoreCodes git:(master) vim ReadMe.txt
```

原理是这样的：



以上例子，是这样的：
git reset —hard 6dae83e 后，
回退到仅有Hello World! 那个版本：

```
  1 Hello World!
~
~
~
```

但又想跳回有三句话的版本，于是通过
git reflog

```
6dae8e3 HEAD@{0}: reset: moving to 6dae8e3
a504156 HEAD@{1}: commit (amend): add again
7513bea HEAD@{2}: commit: add again
6dae8e3 HEAD@{3}: commit: add sentence1, not add sentence2
8b019f5 HEAD@{4}: commit (initial): creat ReadMe.txt
(END)
```

观察各版本对应id，
git reset —hard a504156 后，
回退到：

```
  1 Hello World!
  2 Git, fighting!
  3 Again!
~
```

section4. 穿梭（文件粒度）

subsection1. 文件回退
git checkout id — filename

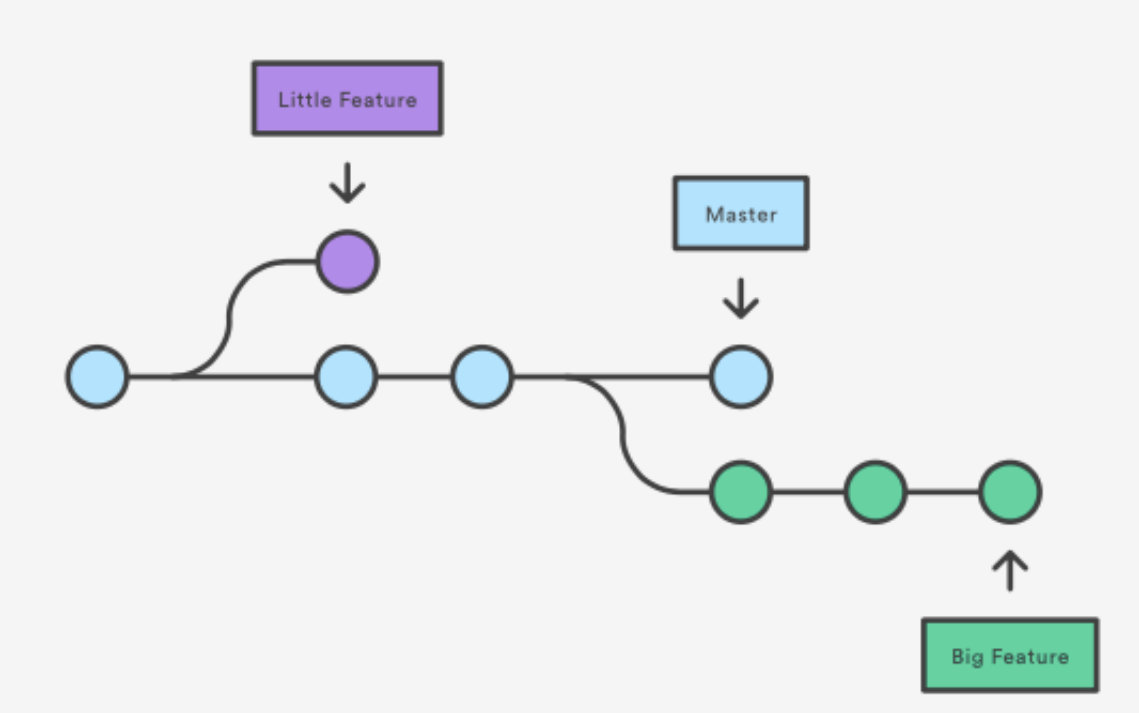
e.g.
若管理着多个文件，那么可通过
git checkout 6dae8e3 — ReadMe.txt
回退到该文件仅Hello World!的时候

```
➜  CoreCodes git:(master) vim ReadMeReference.txt
➜  CoreCodes git:(master) ✗ ls
ReadMe.txt          ReadMeReference.txt
➜  CoreCodes git:(master) ✗ git add ./ReadMeReference.txt
➜  CoreCodes git:(master) ✗ git commit -m 'add another txt'
[master 7349535] add another txt
 1 file changed, 1 insertion(+)
 create mode 100644 ReadMeReference.txt
➜  CoreCodes git:(master) git log
➜  CoreCodes git:(master) git checkout 6dae8e3 -- ReadMe.txt
➜  CoreCodes git:(master) ✗ vim ReadMe.txt
```

打开观察一下，
果然是这样，

```
  1 Hello World!
~
~
```

section5. 分支



subsection1. 创建分支
git branch name
git branch -d name（删除分支，此操作需要在位于其他分支时操作）

subsection2. 切换分支
git checkout name

subsection3. 添加追踪并提交更新
git commit -am information（若有新创建的文件，不能使用此命令）

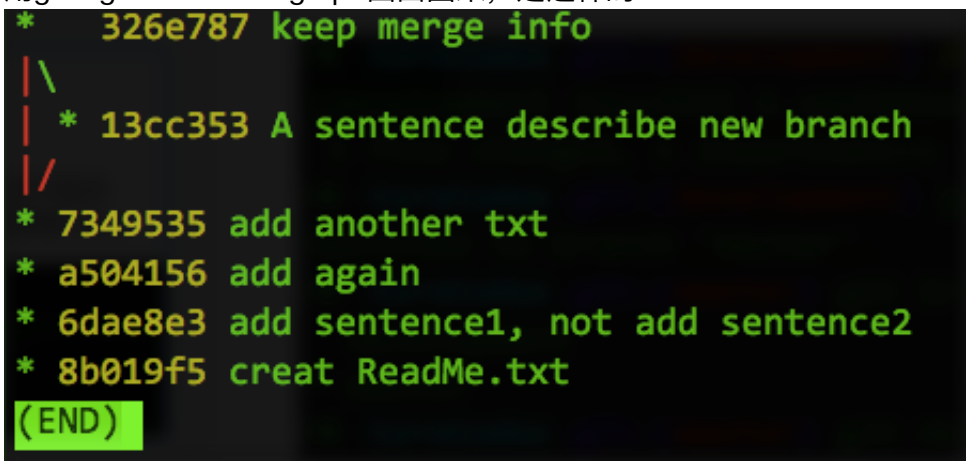
subsection4. 合并分支
git merge —no-ff -m information name（默认的fast forward模式不会记录信息，所以用--no-ff）

subsection5. 图形模式显示版本日志
git log —oneline —graph

e.g.
在开发版分支（development）中的ReadMe.txt文件上增加一行，
并且并入主分支（master）

```
➜ CoreCodes git:(master) git branch development
➜ CoreCodes git:(master) git branch
  development
* master
➜ CoreCodes git:(master) git checkout development
Switched to branch 'development'
➜ CoreCodes git:(development) vim ReadMe.txt
➜ CoreCodes git:(development) ✗ git commit -am 'A sentence describe new branch'
[development 13cc353] A sentence describe new branch
 1 file changed, 1 insertion(+)
➜ CoreCodes git:(development) git checkout master
Switched to branch 'master'
➜ CoreCodes git:(master) git branch
  development
* master
➜ CoreCodes git:(master) git merge --no-ff -m 'keep merge info' development
Merge made by the 'recursive' strategy.
 ReadMe.txt | 1 +
 1 file changed, 1 insertion(+)
```

用git log —oneline —graph画出图来，是这样的：

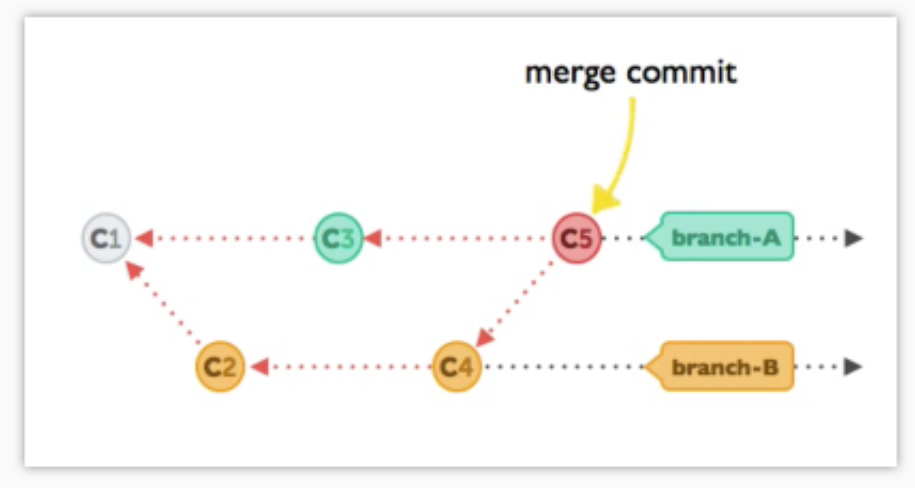


section6. 分支冲突

subsection1. merge分支冲突

两分支各自进行了一些操作，那么git应该相信谁呢?

（C5 = C4，C3 不同于 C5，C2 不同于 C4）



试想这样一种情况：
在master分支上，于ReadMeReference.txt中添加了一句，
又在development分支上，于ReadMeReference.txt中添加了另一句，

```
➜ CoreCodes git:(master) vim ReadMeReference.txt
➜ CoreCodes git:(master) ✗ git commit -am 'Branch master'
[master 19f02fb] Branch master
 1 file changed, 1 insertion(+)
➜ CoreCodes git:(master) git checkout development
Switched to branch 'development'
➜ CoreCodes git:(development) vim ReadMeReference.txt
➜ CoreCodes git:(development) ✗ git commit -am 'Branch development'
[development aa1a760] Branch development
 1 file changed, 1 insertion(+)
```

master分支：

```
  1 URL:https://morvanzhou.github.io/tutorials/others/git/
  2 This branch master
~
```

```
19f02fb Branch master
326e787 keep merge info
13cc353 A sentence describe new branch
7349535 add another txt
a504156 add again
6dae8e3 add sentence1, not add sentence2
8b019f5 creat ReadMe.txt
(END)
```

development分支：

```
  1 URL:https://morvanzhou.github.io/tutorials/others/git/
  2 This branch is development
~
```

```
aa1a760 Branch development
13cc353 A sentence describe new branch
7349535 add another txt
a504156 add again
6dae8e3 add sentence1, not add sentence2
8b019f5 creat ReadMe.txt
(END)
```

那么merge时，git就不知所措了（此时merge，从后面的图来看，—no-ff好像没起作用）

```
➜  CoreCodes git:(master) git merge --no-ff -m 'keep 2nd merge info' development

Auto-merging ReadMeReference.txt
CONFLICT (content): Merge conflict in ReadMeReference.txt
Automatic merge failed; fix conflicts and then commit the result.
➜  CoreCodes git:(master) ✗ vim ReadMeReference.txt
➜  CoreCodes git:(master) ✗ git commit -am 'Solve conflict'
[master fc1797a] Solve conflict
➜  CoreCodes git:(master) git log --oneline --graph
```

会报conflict，

打开ReadMeReference.txt：

```
  1 URL:https://morvanzhou.github.io/tutorials/others/git/
  2 <<<<<<< HEAD
  3 This branch master
  4 =======
  5 This branch is development
  6 >>>>>>> development
~
```

手动排除，成这样：

```
  1 URL:https://morvanzhou.github.io/tutorials/others/git/
  2 This branch is master and development
~
```

再git commit -am 一次，就好了，图是这样的：

结合上两张在各分支上的git log看此图，并不复杂，其实是：

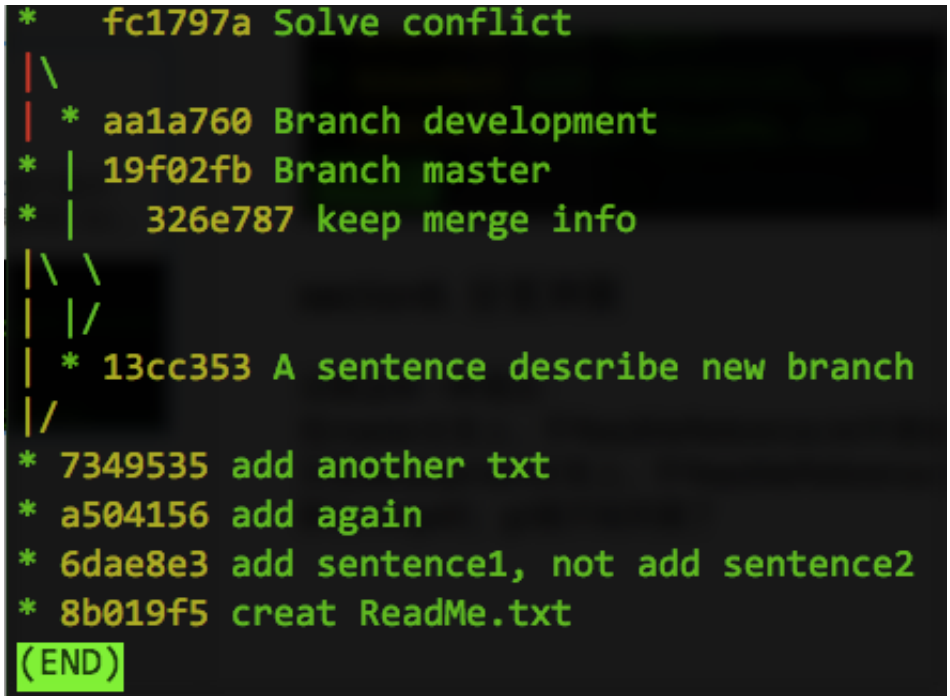
——keep merge info ——branch master

\

solve conflict

/

————————branch development



subsection2. rebase分支冲突
git rebase branchname

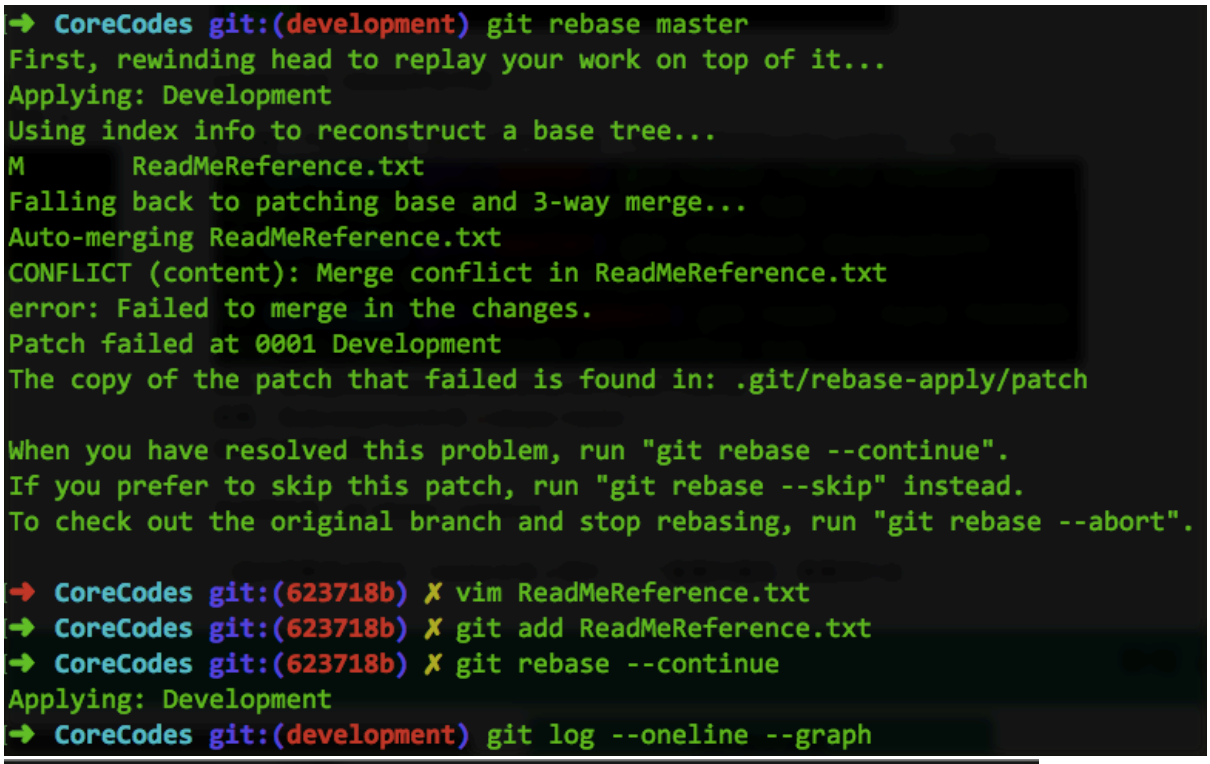
先准备一下实验环境，都reset到7349535（13cc353是在ReadMe.txt中加了一句话）

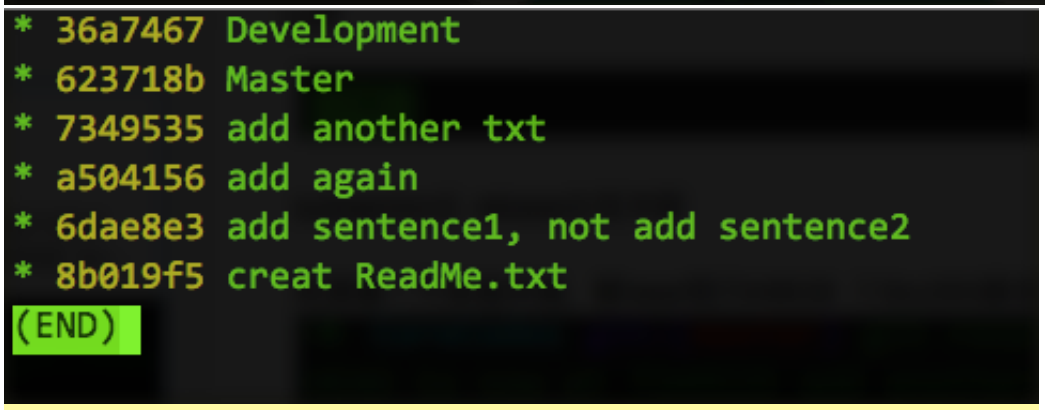
```
➜ CoreCodes git:(master) git reset --hard 7349535
HEAD is now at 7349535 add another txt
➜ CoreCodes git:(master) git checkout development
Switched to branch 'development'
➜ CoreCodes git:(development) git reset --hard 7349535
HEAD is now at 7349535 add another txt
```

然后，在两版本的ReadMeReference.txt中添加不同的话，
接着，在development分支上git rebase master
会出现这样的情况，

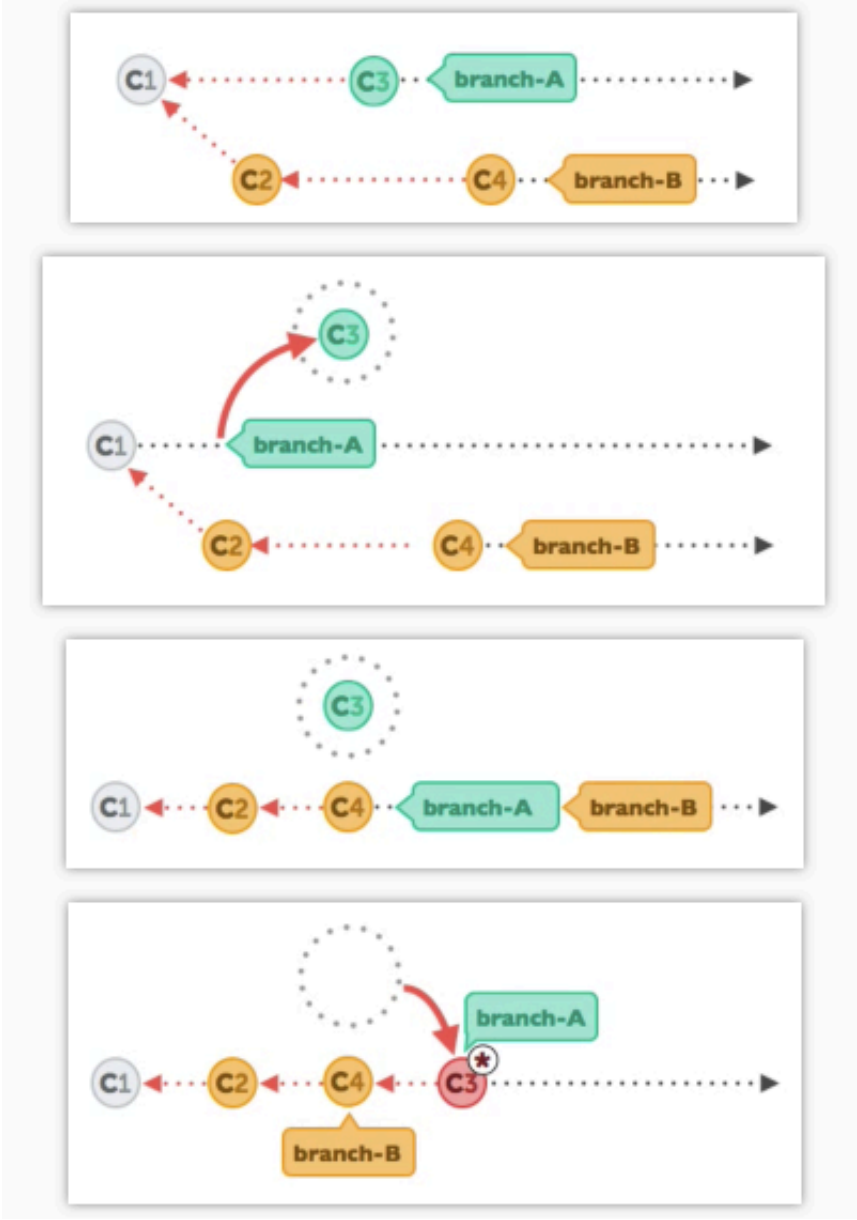
```
1 URL:https://morvanzhou.github.io/tutorials/others/git/
2 <<<<<<< 623718be89a51a3a7fdaf428b16295429e96ad19
3 Master
4 =======
5 Here development
6 >>>>>>> Development
~
```

手动修改后，git rebase —continue





一般的情景是这样的，
master分支上修补了一个重要的漏洞，需要我们同步到自己分支
（branch-A相当于development，但是下面的最后一张图，应该是C3*放在C1 C2间，
不过其实再一想，反正更改是手动完成的，更改信息并不没有顺序一说）



需要说明的是：
不建议使用rebase
更不建议在master分支上使用rebase，这会抹除别人的对版本贡献

section7. 临时挂起

试想这样一个场景：
你正愉快地写代码，
而此时，你需要处理一个紧急的任务，
这时候，git stash 就可以帮忙暂时挂起修改
挂起后，git status 就看不到了

```
➜ CoreCodes git:(development) vim ReadMe.txt
➜ CoreCodes git:(development) ✗ git status -s
 M ReadMe.txt
➜ CoreCodes git:(development) ✗ git stash
Saved working directory and index state WIP on development: 7349535 add another txt
HEAD is now at 7349535 add another txt
➜ CoreCodes git:(development) git status -s
➜ CoreCodes git:(development) git branch temp
➜ CoreCodes git:(development) git checkout temp
Switched to branch 'temp'
➜ CoreCodes git:(temp) vim ReadMe.txt
➜ CoreCodes git:(temp) ✗ git commit -am 'Solve urgent task'
[temp 032f773] Solve urgent task
 1 file changed, 1 insertion(+)
➜ CoreCodes git:(temp) git checkout master
Switched to branch 'master'
➜ CoreCodes git:(master) git merge --no-ff -m 'keep merge info' temp
Merge made by the 'recursive' strategy.
 ReadMe.txt | 1 +
 1 file changed, 1 insertion(+)
➜ CoreCodes git:(master) git log --oneline --graph
```

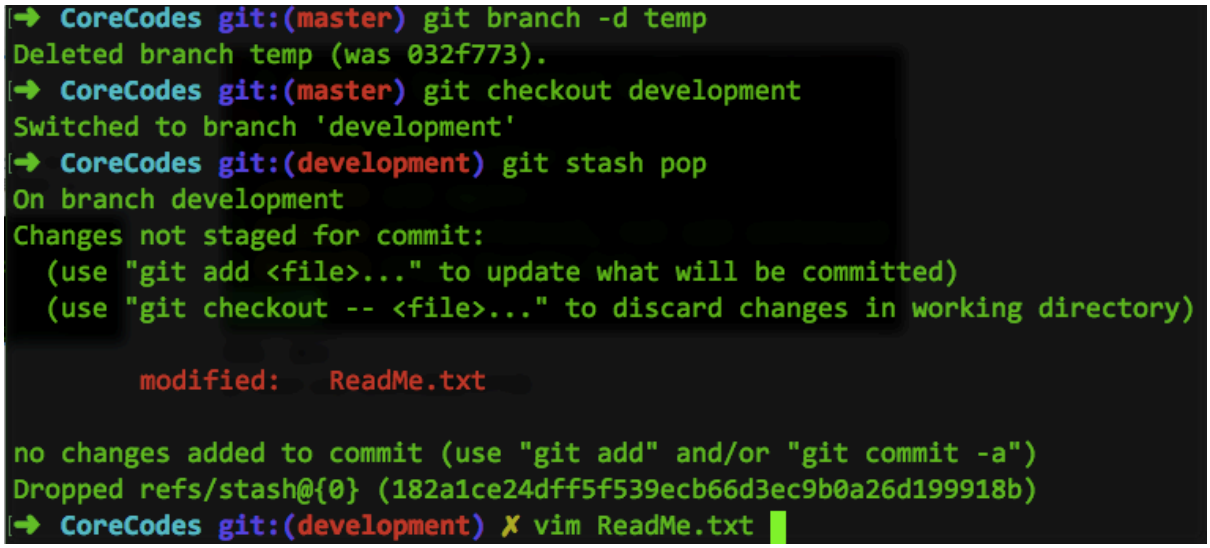
起一个分支temp，

处理完紧急任务后，git merge一下，日志是这样的：

```
*   473e709 keep merge info
|\
| * 032f773 Solve urgent task
|/
* 7349535 add another txt
* a504156 add again
* 6dae8e3 add sentence1, not add sentence2
* 8b019f5 creat ReadMe.txt
(END)
```

而后，通过

git stash pop 愉快地接着写代码


➜ CoreCodes git:(master) git branch -d temp
Deleted branch temp (was 032f773).
➜ CoreCodes git:(master) git checkout development
Switched to branch 'development'
➜ CoreCodes git:(development) git stash pop
On branch development
Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git checkout -- <file>..." to discard changes in working directory)
modified: ReadMe.txt
no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (182a1ce24dff5f539ecb66d3ec9b0a26d199918b)
➜ CoreCodes git:(development) ✗ vim ReadMe.txt