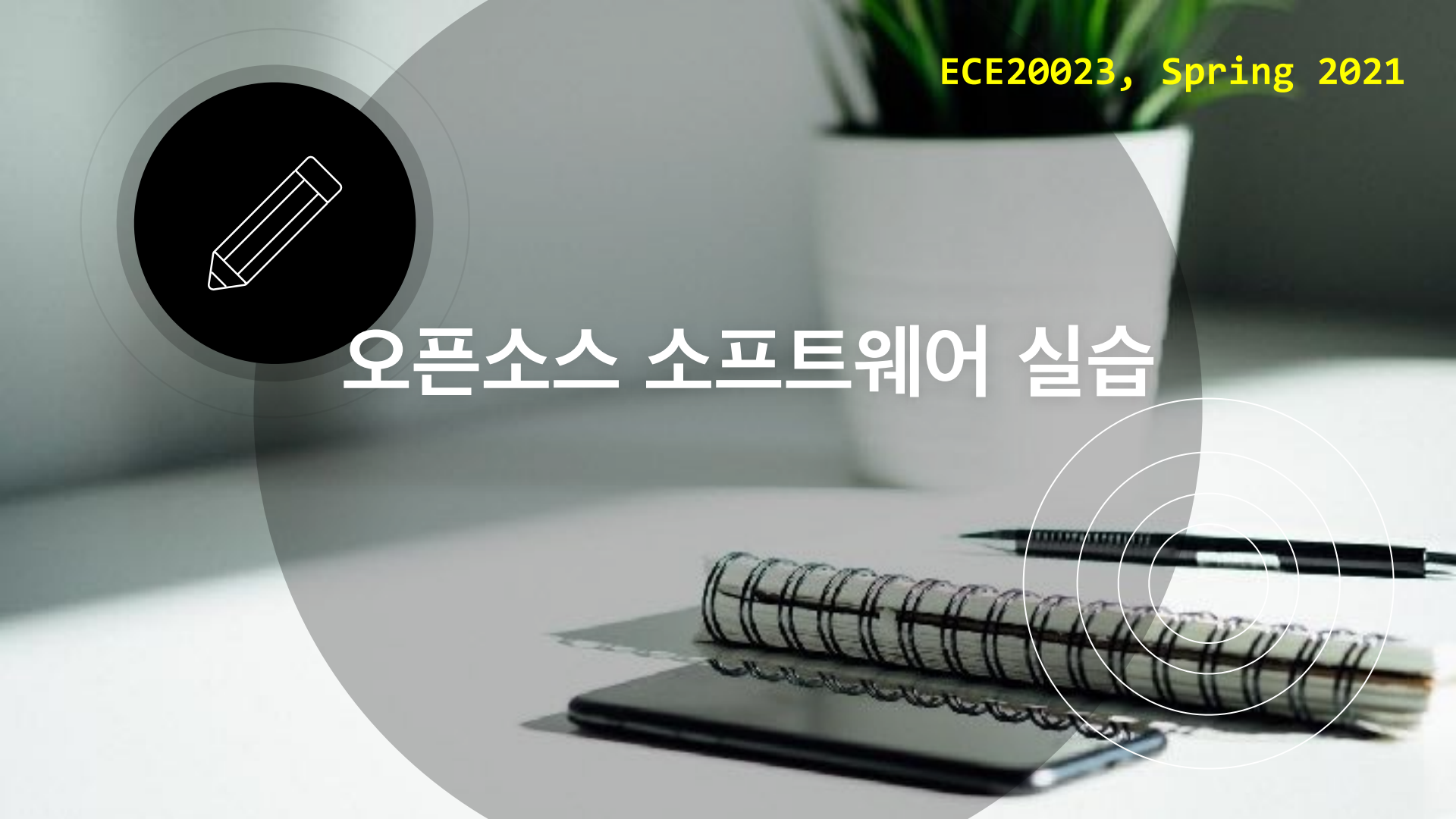
ECE20023, Spring 2021



오픈소스 소프트웨어 실습

13

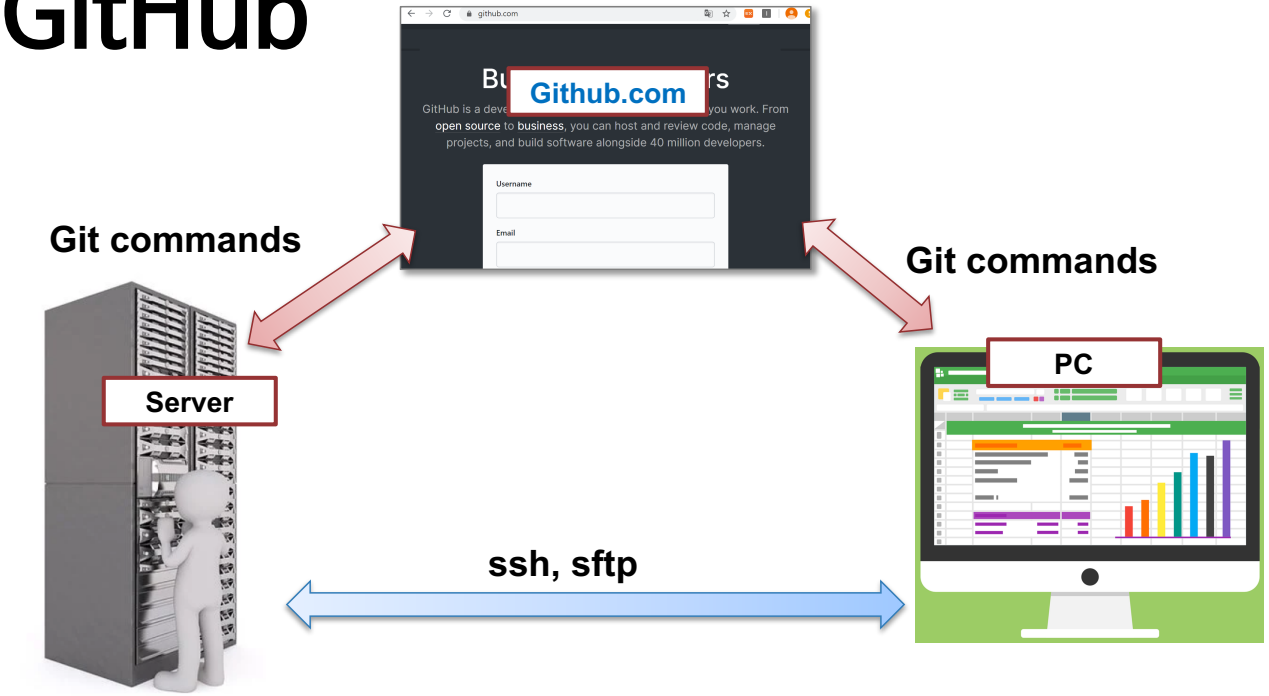
Git

VCS (Version Control System)

참고 :

https://backlog.com/git-tutorial/kr/

Git & GitHub



git stash

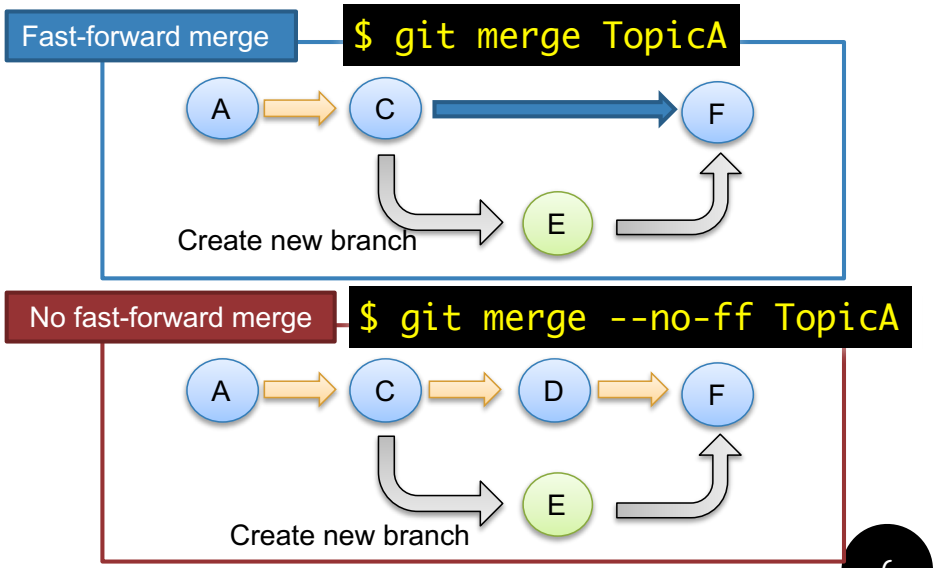
- 현재 작업 내용을 임시 저장하여 보관하는 장소
- WD(Working Directory)에서 수정한 파일만 저장
  - Modified면서 Tracked 상태인 파일
  - Staging Area에 있는 파일

```
$ git stash
$ git stash save
$ git stash list
$ git stash apply [stash이름]
$ git stash pop     // apply + drop
$ git stash drop
$ git stash clear
```

```
$ git commit -m "create fruit.txt"
$ vim fruit.txt
$ git stash
$ git stash
$ git stash list
$ cat fruit.txt
$ git stash pop
$ cat fruit.txt
```

git merge

- 브랜치와 브래치(master)에 병합하는 작업
- Fast forward(--ff) default
  - Merge commit 생성 X
- No Fast Forward(--no-ff)
  - Merge commit 생성 O
- --squash
  - Merge 후 파일 상태로 변경
  - Merge commit 생성 X
  - 별도 commit을 해야 함



git merge --ff

```
$ mkdir git2
$ cd git2
$ git init
$ touch fruit.txt
$ git add fruit.txt
$ git commit -m "C1"
$ vim fruit.txt
$ git commit -am "C2"
$ git checkout -b TopicA
$ vim fruit.txt
$ git commit -am "C3"
$ git checkout -
$ git merge TopicA
$ git log --oneline –graph
```

```
walab-HGU:~/lab7/git2:> git log --oneline --graph
* dbb65d1 (HEAD -> master, TopicA) C3
* 825c2a1 C2
* 0b2b6d3 C1
```

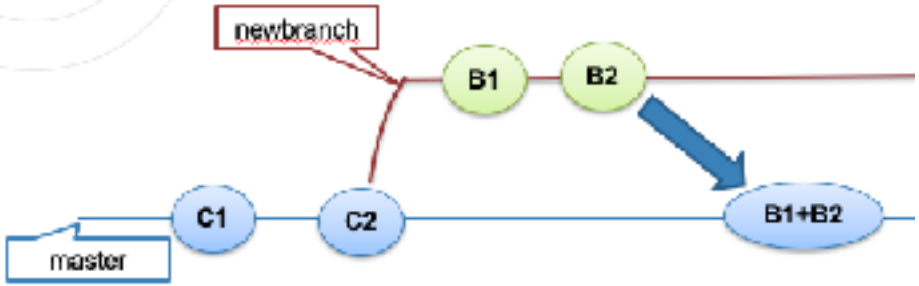
git merge --no-ff

```
$ vim fruit.txt
$ git commit -am "C4"
$ git checkout -b TopicB
$ git branch
$ vim fruit.txt
$ git commit -am "C5"
$ git checkout -
$ git merge --no-ff TopicB
$ git log --oneline --graph
```

```
walab-HGU:~/lab7/git2:> git log --oneline --graph
*   c8997d2 (HEAD -> master) Merge branch 'TopicB'
|\
| * 69c0440 (TopicB) C5
|/
* 7f9daf0 C4
* dbb65d1 (TopicA) C3
* 825c2a1 C2
* 0b2b6d3 C1
```

git merge --squash

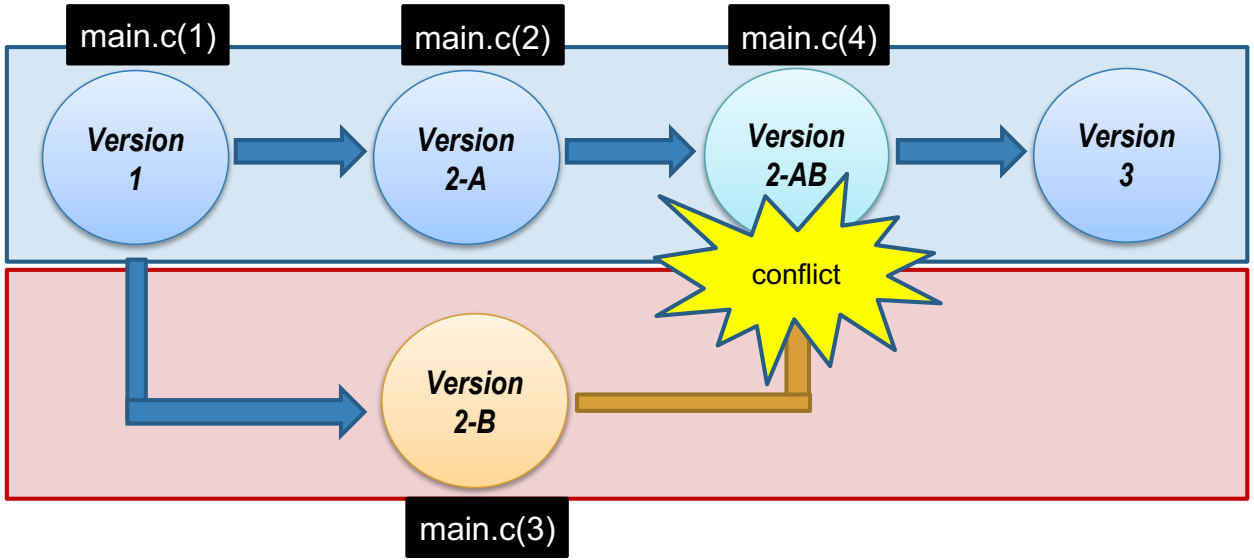
- 다른 브랜치에 있는 여러 커밋을 하나로 병합
- Merge commit이 만들어지지 않음
- 별도로 Commit을 해야 함




```
$ mkdir git4
$ cd git4
$ ls
$ git init
$ touch fruit.txt
$ git add fruit.txt
$ git commit -m "C1"
$ vim fruit.txt
$ git commit -am "C2"
$ git log --oneline
$ git checkout -b TopicA
$ vi fruit.txt
$ git commit -am "B1"
$ git log --oneline
$ vi fruit.txt
$ git commit -am "B2"
$ git checkout -
$ git merge --squash TopicA
$ git log --oneline
$ git commit -am "B1+B2"
$ cat fruit.txt
```

merge conflict

- 두 브랜치에서 같은 파일을 수정하여 각각 커밋한 후, 병합하는 경우 충돌 발생



merge conflict

- 충돌 메세지 발생

```
walab-HGU:~/lab7/git3:> git merge TopicA
Auto-merging main.c
CONFLICT (content): Merge conflict in main.c
Automatic merge failed; fix conflicts and then commit the result.
```

- 충돌일 발생한 파일

```
<<<<<<< HEAD
V1 : This is some content for merging
V2-A: content to append
=======
V2 : totally different content to merge later
>>>>>>> new_branch
```

merge conflict

- 충돌 해결방법
  - 파일의 내용을 적절히 수정
  - git add
  - git commit

```
#include <stdio.h>
<<<<<<< HEAD
int minus(int n1, int n2){
        return n1-n2;
=======

int plus(int n1, int n2){
        return n1+n2;
>>>>>>> TopicA
}
int main(){
        int num1, num2;
        printf("두 수를 입력:");
        scanf("%d %d", &num1, &num2);
        return 0;
}
```

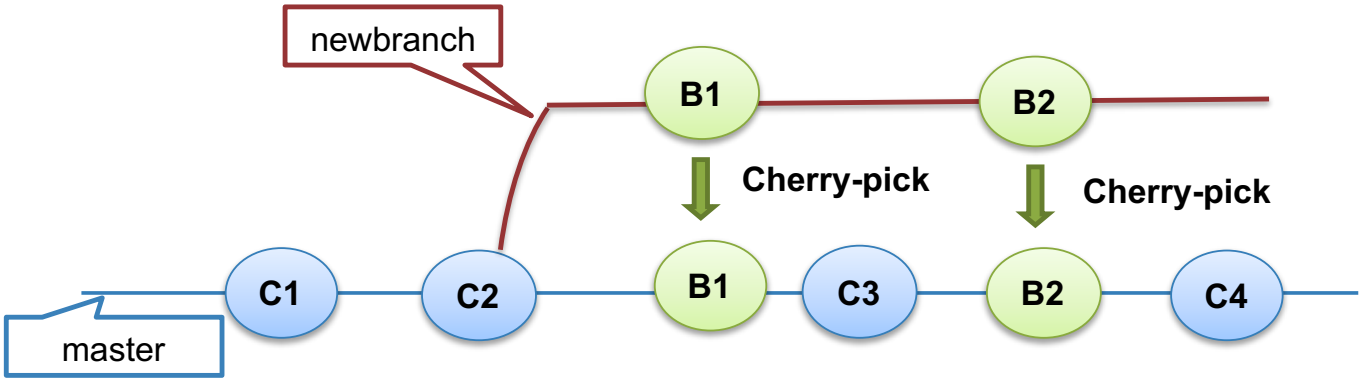
merge conflict

Master branch	TopicA
main.c 파일 생성	
두 수 입력 추가	
	브랜치 생성
뺄셈함수 구현	덧셈함수구현
TopicA 병합	
충돌	
충돌해결	

```
$ mkdir git3
$ cd git3
$ git init
$ touch main.c
$ git add main.c
$ git commit -m "create main.c"
$ vim main.c
$ git commit -am "add 두수입력"
$ git log --oneline
$ git checkout -b TopicA
$ vi main.c
$ git commit -am "add 덧셈함수"
$ git log --oneline
$ git checkout -
$ vi main.c
$ git commit -am "add minus func."
$ git log --oneline
$ git merge TopicA
$ git status
$ vim main.c
$ git commit -am "add 커밋해결"
$ git log --oneline --graph
```

Cherry-pick

- 다른 브랜치의 특정 커밋을 선택하여 현재 브랜치에 적용하는 명령어
- Git merge를 통한 충돌을 피할 수 있음




```
$ git cherry-pick [commit_checksum]
```

Reflog

- 모든 참조 기록을 확인

```
walab-HGU:~/lab7/git4:> git reflog
0890f98 (HEAD -> master) HEAD@{11}: commit: B1+B2
e7b1a91 HEAD@{12}: checkout: moving from TopicA to master
6015707 HEAD@{13}: commit: B2
9efdfb9 HEAD@{14}: commit: B1
e7b1a91 HEAD@{15}: checkout: moving from master to TopicA
e7b1a91 HEAD@{16}: commit: C2
e5da715 HEAD@{17}: commit (initial): C1
```

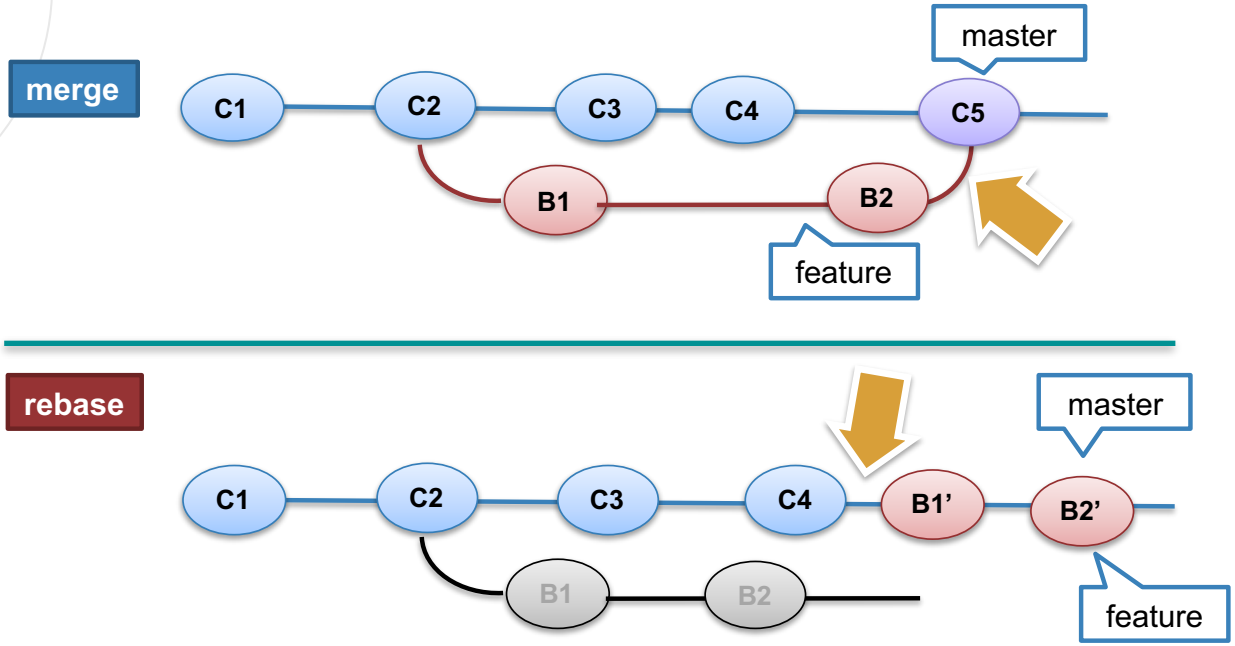
- 원하는 커밋으로 이동(취소)

```
walab-HGU:~/lab7/git4:> git reflog
walab-HGU:~/lab7/git4:> git reset --hard [commit id]
```

Rebase

- merge와는 다른 형태의 병합
- 새 브랜치의 Base를 master의 마지막 commit으로 rebase
  - [새브랜치] 로 checkout
  - [master] 로 rebase
  - [master] 로 checkout
  - [master] 브랜치에서 [새브랜치] 를 병합 (fast-forward merge)

Merge vs Rebase



Rebase -i

- commit 히스토리를 편집할 때 사용

```
walab-HGU:~/lab7/git4:> git rebase -i [수정커밋의 이전커밋]
```

- 수정커밋 ~ 현재커밋(HEAD) 범위에 있는 모든 commit 리스트 출력
- 명령어

```
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
# d, drop = remove commit
```

Rebase -i

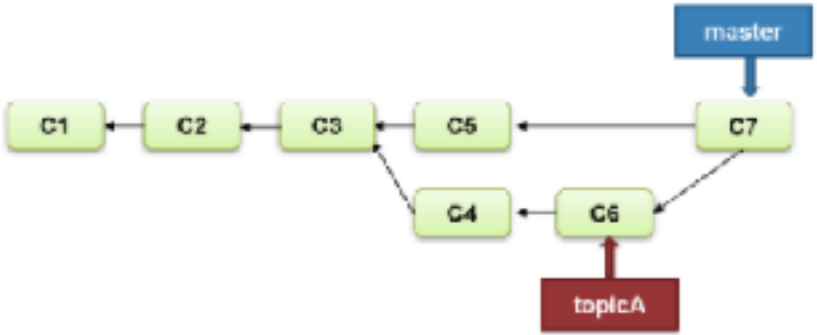
- Commit history 변경하기

```
walab-HGU:~/lab7/git5:> git log --oneline
1d7b0d6 (HEAD -> master) C4
a3e7438 C3
353b47c C2
ac22061 C1
```

```
walab-HGU:~/lab7/git5:> git log --oneline
35d02e1 (HEAD -> master) C3
dfaec5e C2(Reword)
ac22061 C1
```

과제설명

3. 다음 그림을 참고하여 아래와 같은 commit 이력 C1~C7까지 표현될 수 있도록 Local Repository를 새로 생성하여 자신이 시나리오를 만들어서 작성하세요.
   - commit message는 "C1" 로 작성
   - 파일의 내용은 자유
   - 브랜치 이름은 TopicA
   - 충돌이 일어나는 경우를 설명하고, 충돌해결하는 방법에 대해 작성할 때



제출내역

- 아래 그림과 같은 commit history를 만들기 위해 필요한 명령어 리스트(**text**)
- git log --oneline --graph (**화면캡처**)
- 충돌이 예상되는 commit 번호는? 해결방법 제시(**text**)

중간고사 :

- 자세한 사항은 4/19 월요일에 히즈넷에 공지예정
- Quiz 공지
  - 일시 : 4/20 (8주차 화요일 수업시간), HD LMS 퀴즈에서 시험
    - 1분반 : 10시 ~11:15 (2교시)
    - 2분반 : 11:30 ~ 12:45 (3교시)

- 실기시험(동영상)
  - 기한 : ~ 4/25(일) 23:50 까지
  - 실습과제를 동영상으로 제작 제출