오픈소스 소프트웨어 개발 – 14

(Online) Git – branch 2

광운대학교 이윤상
2017년 2학기

이번 시간에 할 것

- Branch 병합(merge) - git merge
- 병합 시 충돌 (merge conflict) 해결

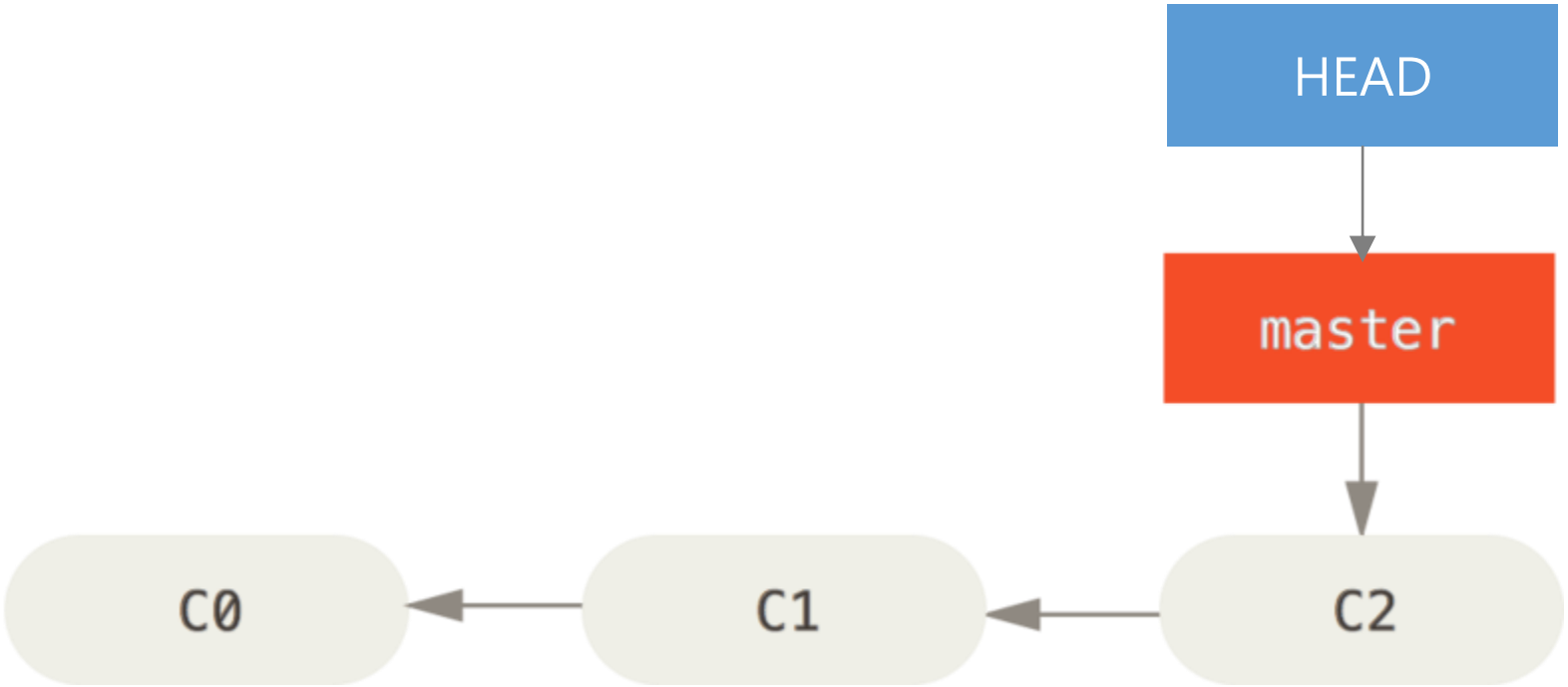
Branch 병합(Merge)

- 별도 branch에서 작업하던 실험적인 기능 (혹은 특정 이슈 해결 작업)의 구현과 테스트가 완료되어, 이제 main branch로 작업 내용을 합치려고 한다.

- -> Branch merge

Branch 병합의 예

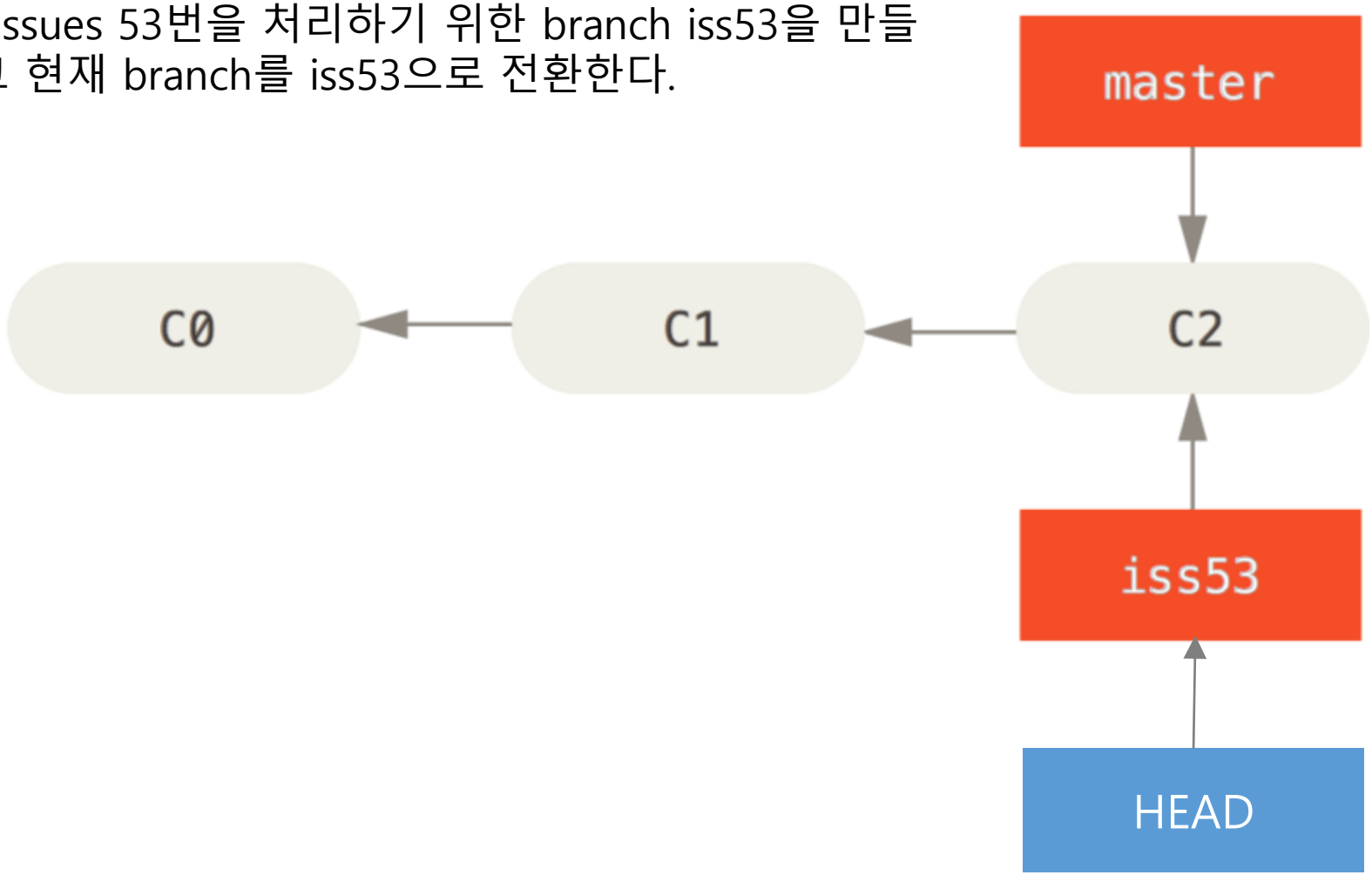
- Commit을 몇 번 한 웹사이트 프로젝트가 있다.
  - master branch에 실제로 서비스되는 코드가 있다고 가정



(Shell)

```
git checkout -b iss53
```

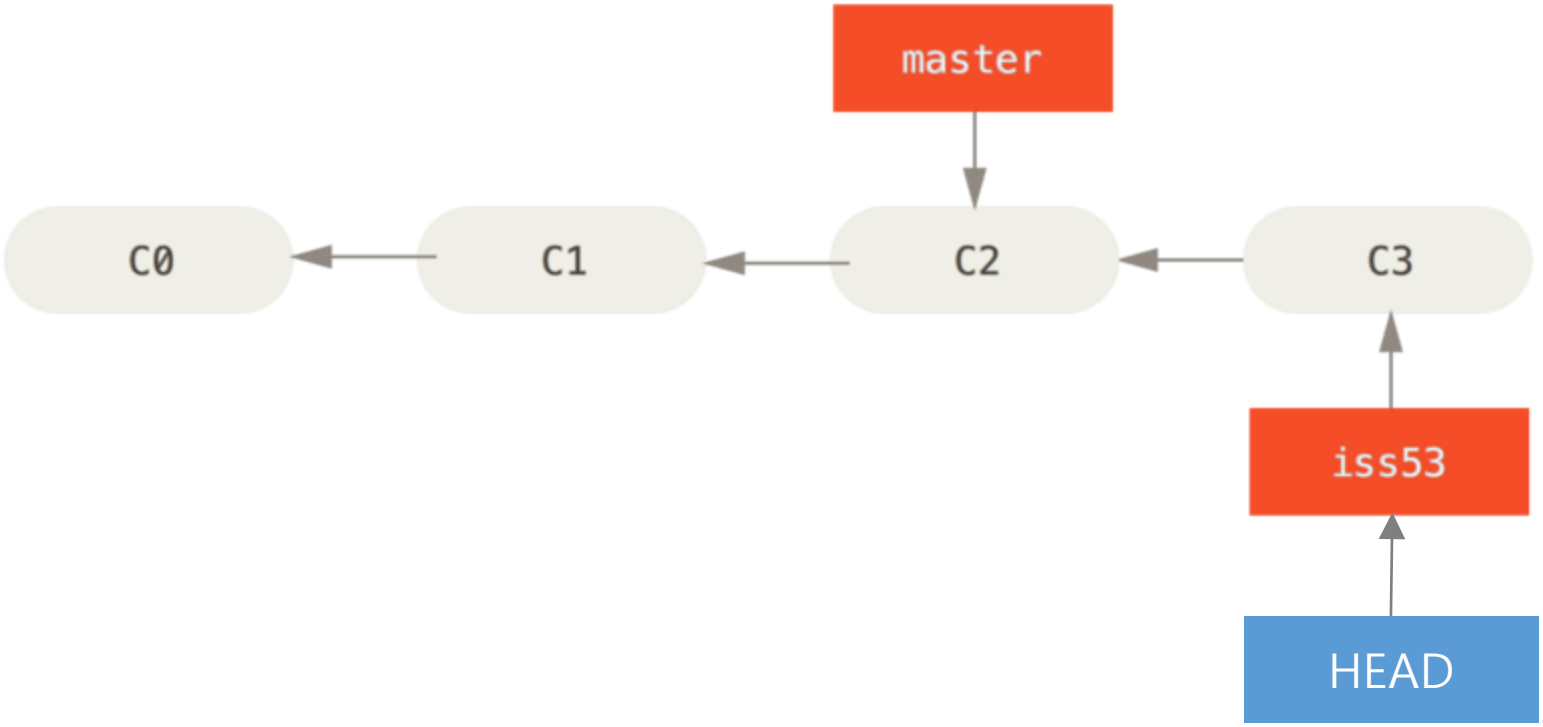
: issues 53번을 처리하기 위한 branch iss53을 만들고 현재 branch를 iss53으로 전환한다.



(Shell)

```
vi index.html
git add .
git commit
```

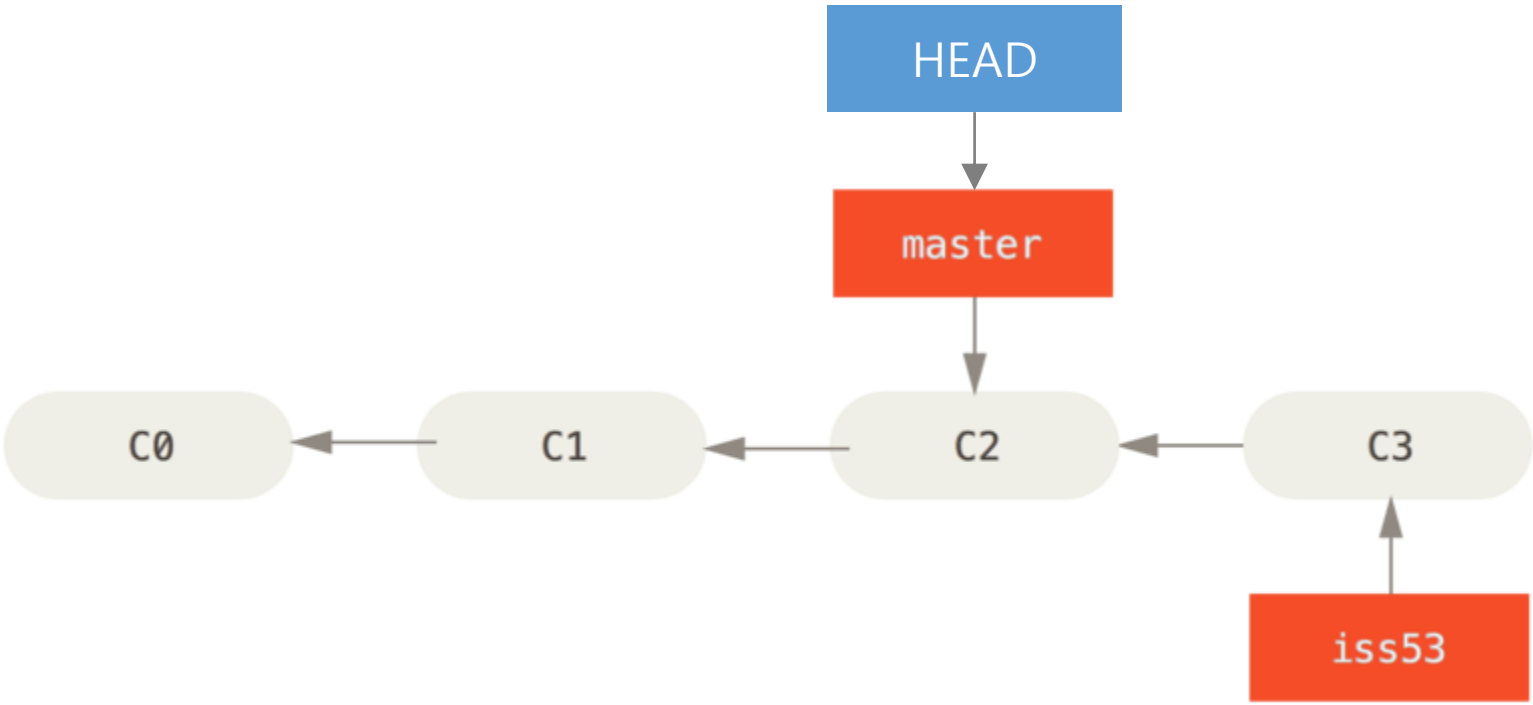
: iss53에 commit을 추가한다.



(Shell)

```
git checkout master
```

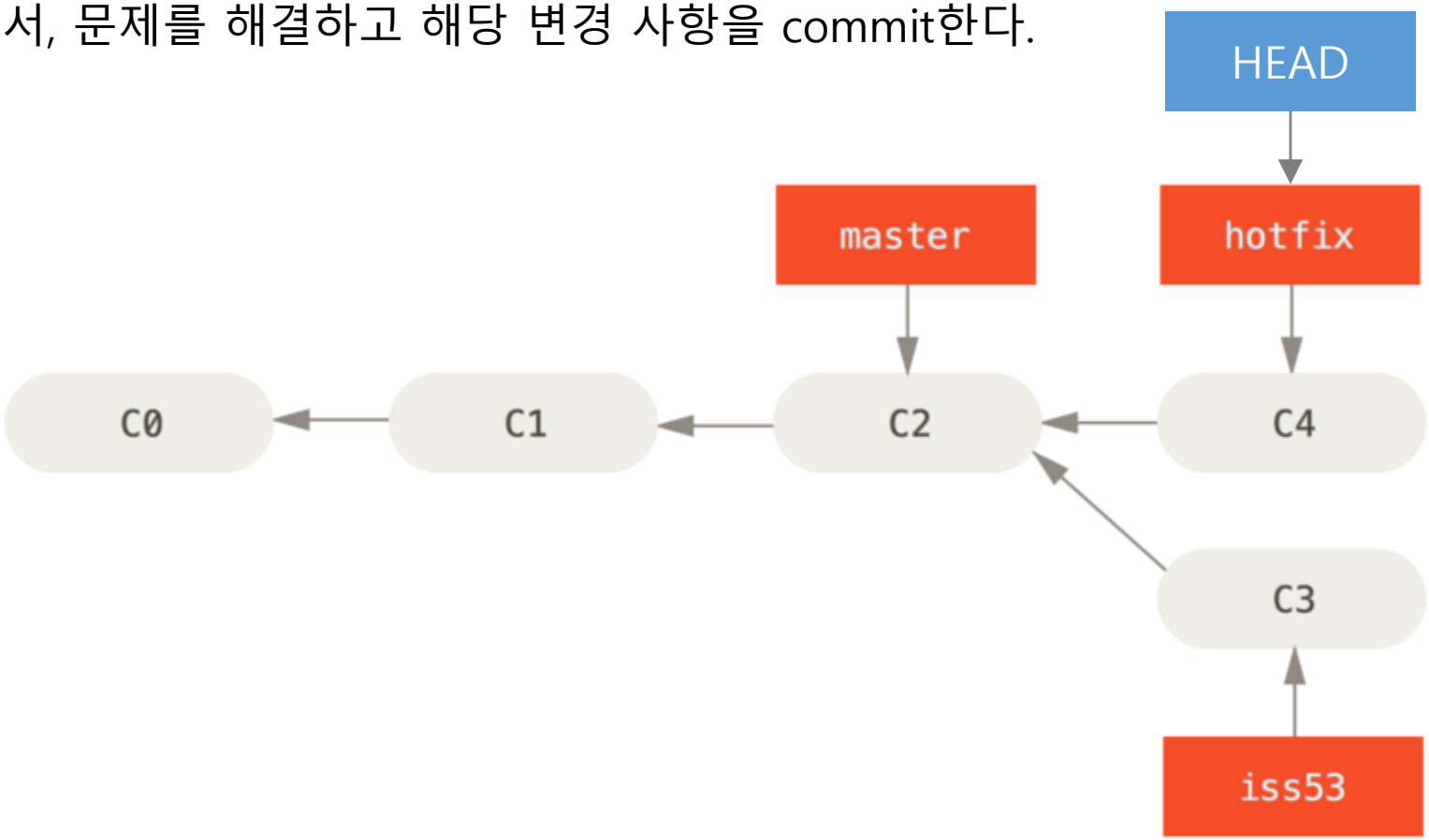
: 만들고 있는 사이트에 급한 문제가 생겨서 즉시 고쳐야 한다.
issue 53 작업 코드가 아닌 현재 서비스되고 있는 코드에서 고쳐야 하므로, 일단 master branch로 다시 돌아간다.



(Shell)

```
git checkout -b hotfix
vi index.html
git add .
git commit
```

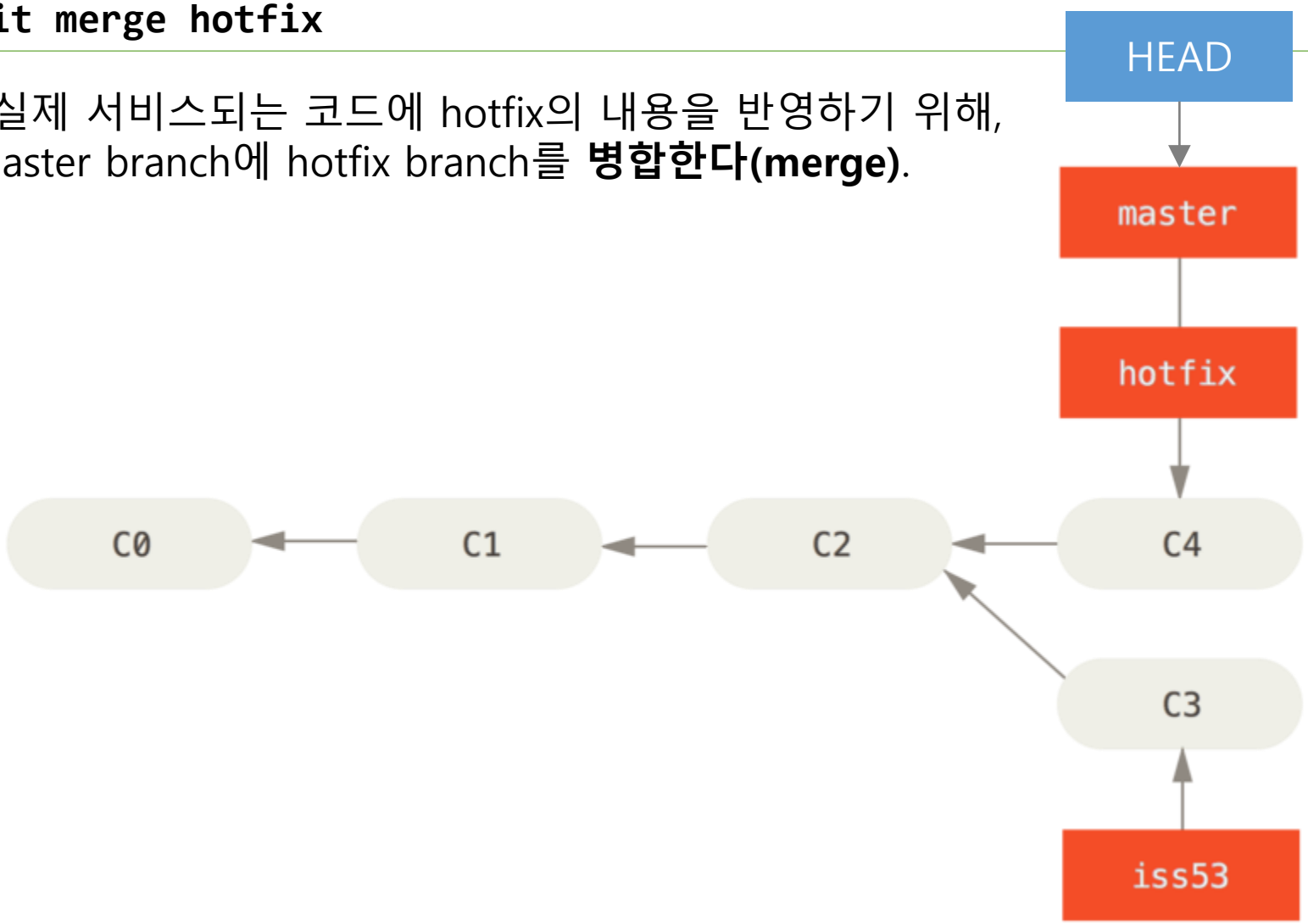
: 급한 문제를 해결하기 위한 hotfix branch를 만들고 전환해서, 문제를 해결하고 해당 변경 사항을 commit한다.



(Shell)

```
git checkout master
git merge hotfix
```

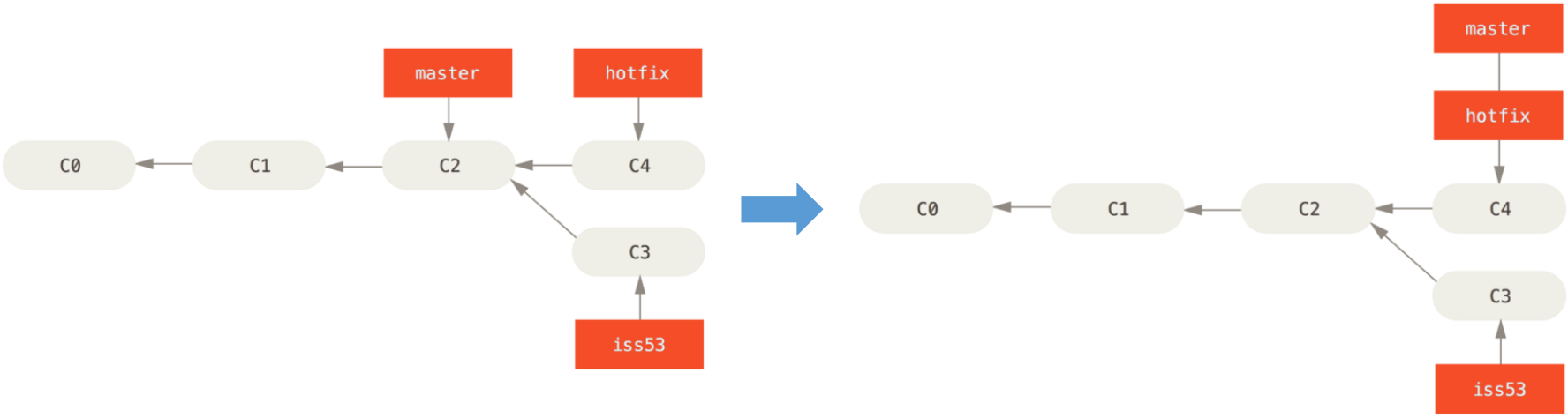
: 실제 서비스되는 코드에 hotfix의 내용을 반영하기 위해, master branch에 hotfix branch를 **병합한다(merge)**.



(Shell)

```
# 출력 메시지
Updating f42c576..3a0874c
Fast-forward
 index.html | 2 ++
 1 file changed, 2 insertions(+)
```

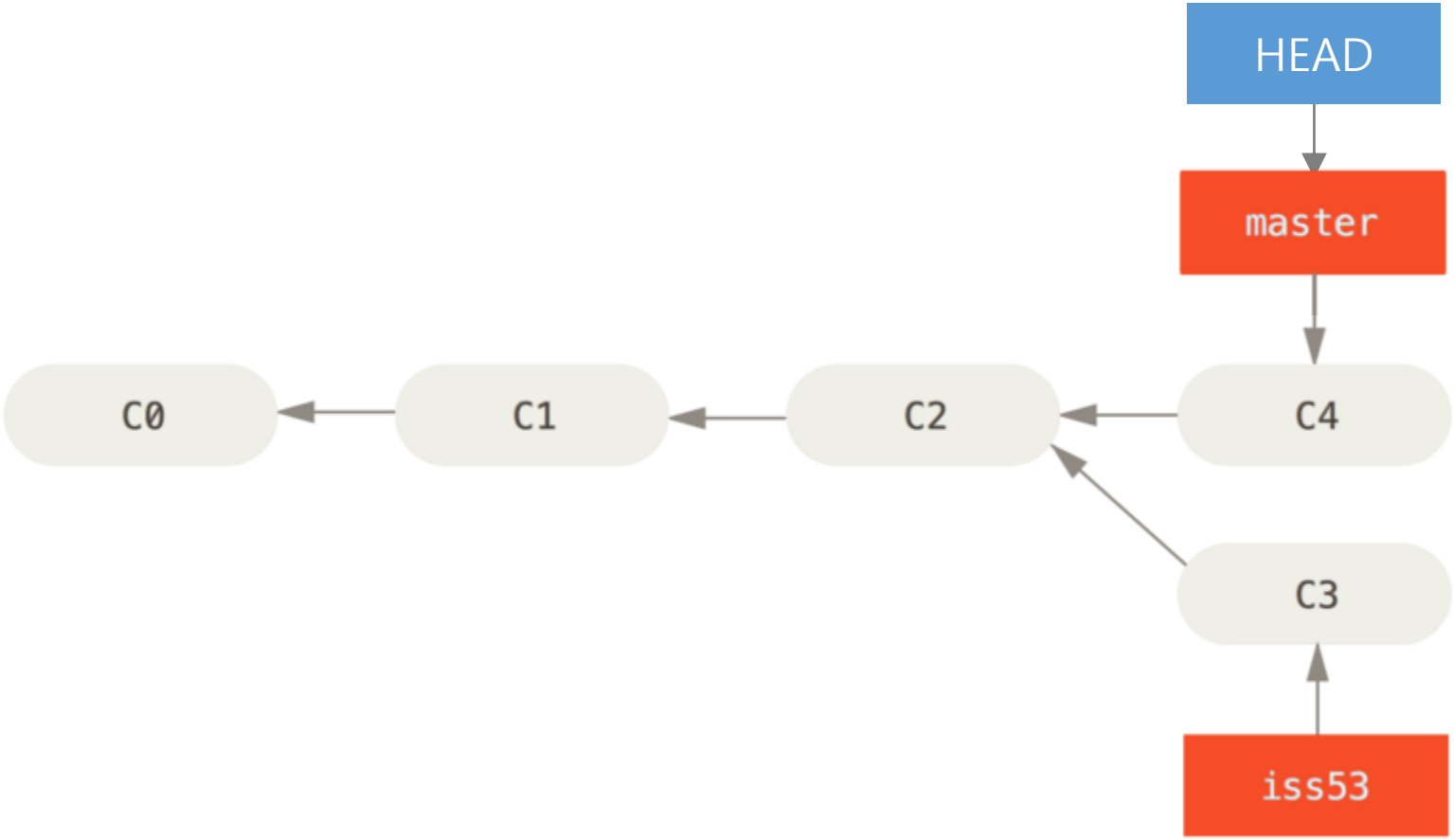
: hotfix가 master 이후의 commit을 가리키고 있기 때문에, 단순히 master가 hotfix와 동일한 commit을 가리키도록 이동하면 된다.
이러한 merge를 fast-forward merge라고 한다.



(Shell)

```
git branch -d hotfix
```

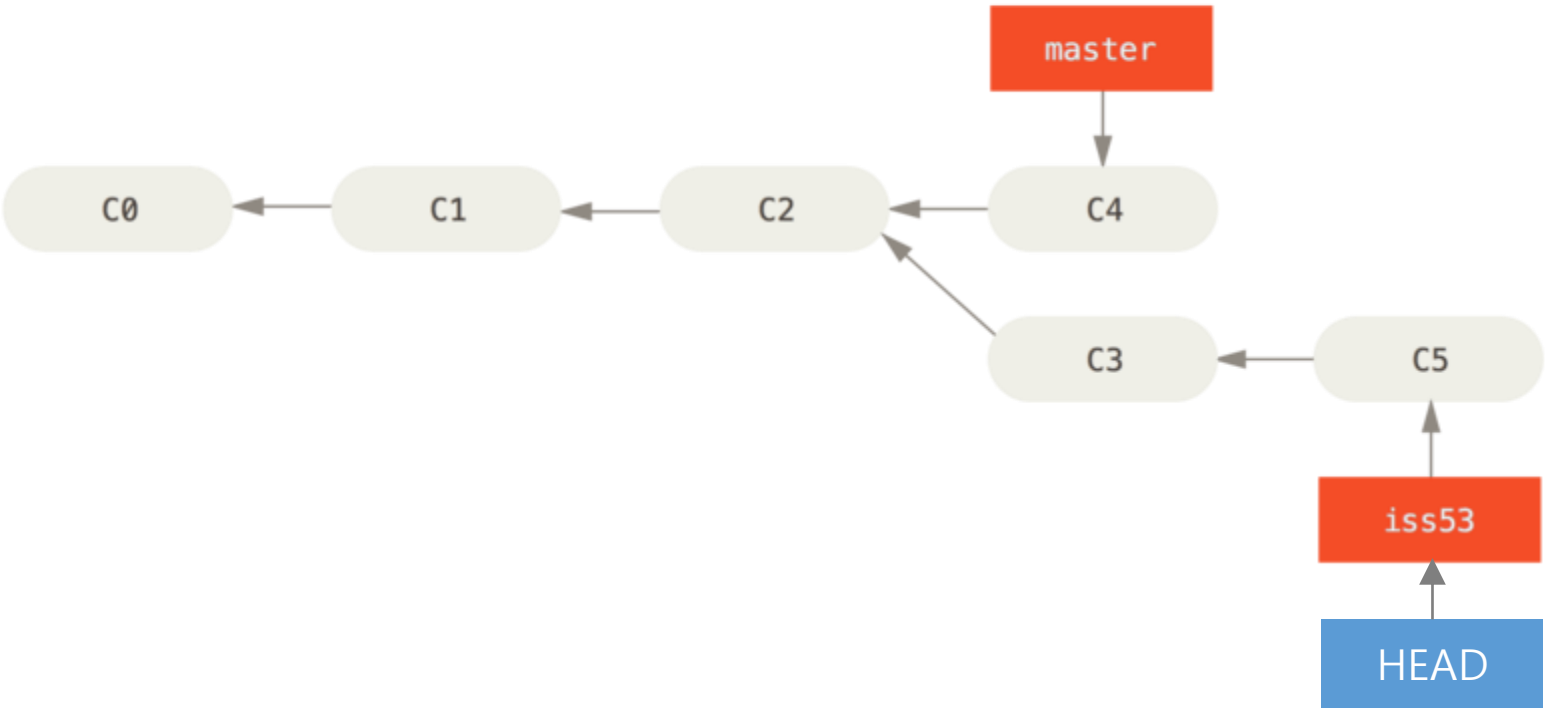
: 더 이상 필요 없는 hotfix branch를 삭제



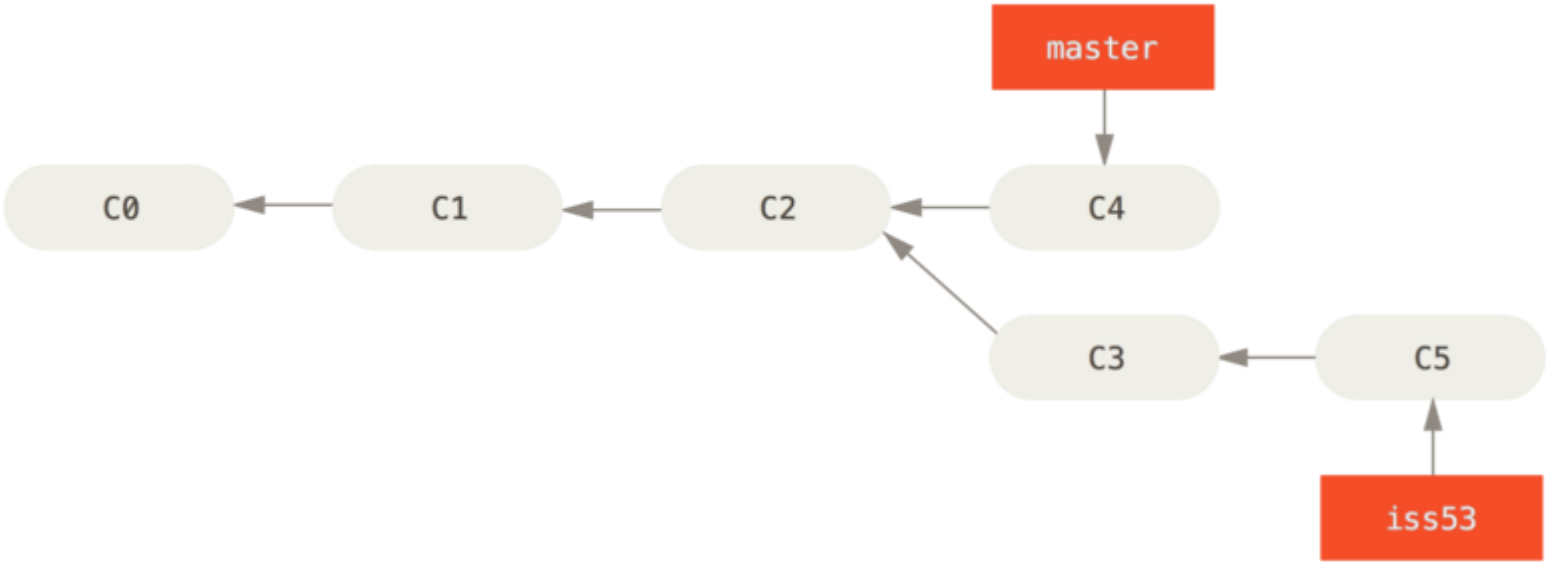
(Shell)

```
git checkout iss53
vi index.html
git add .
git commit
```

: iss53 branch로 돌아와서 하던 일을 계속한다.
앞서 작업한 hotfix는 iss53 branch에 반영되지 않은 상태이다.



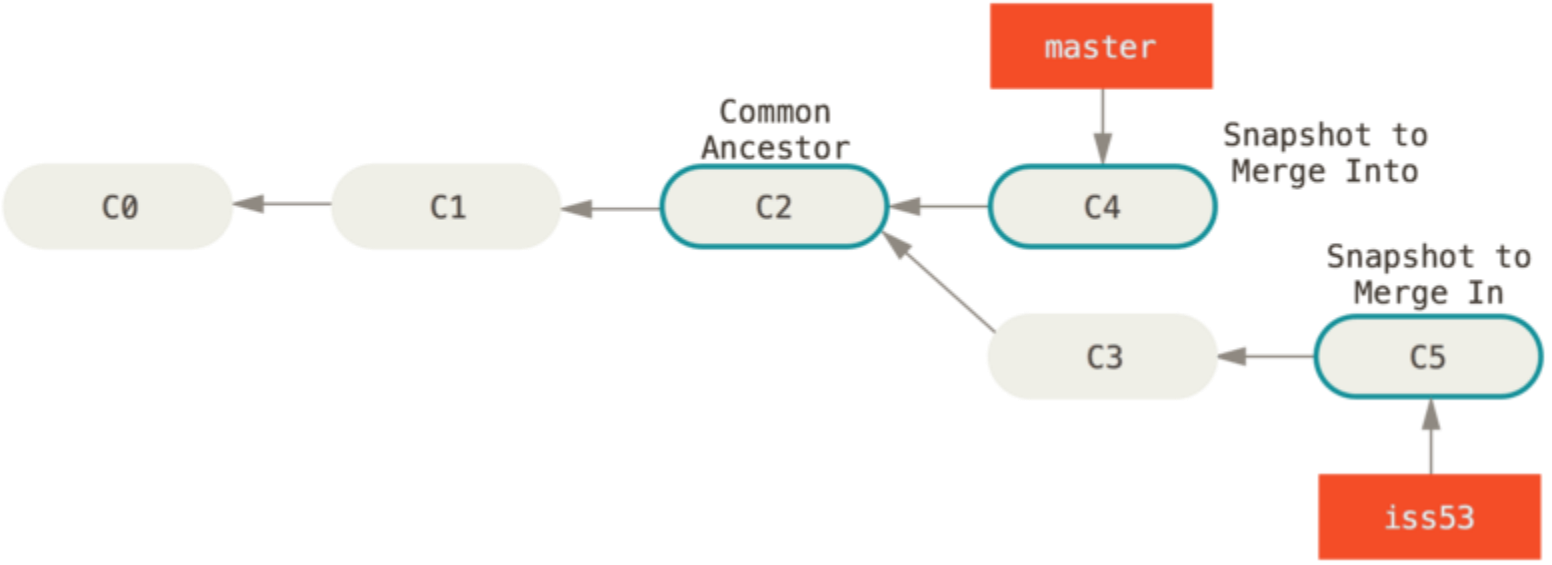
iss53의 구현을 완료했기 때문에, iss53을 master에 merge하려고 한다. iss53이 master 이후의 commit을 가리키지 않기 때문에, 앞의 fast-forward merge를 적용할 수 없다.



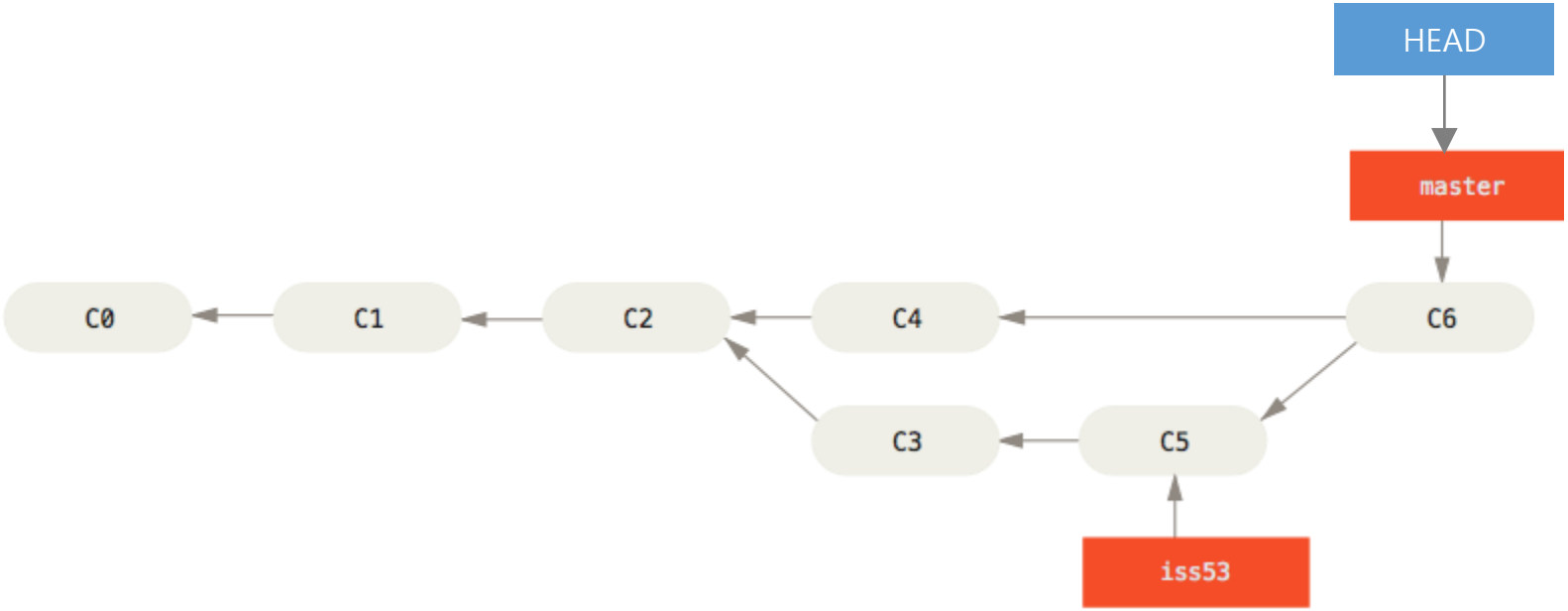
(Shell)

```
git checkout master
git merge iss53

# 출력 메시지
Merge made by the 'recursive' strategy.
README |    1 +
1 file changed, 1 insertion(+)
```

: 이런 경우 Git은 각 branch가 가리키는 commit 두 개와 두 commit의 공통 조상인 commit 하나를 사용하는 3-way merge를 한다.



3-way merge 결과를 별도의 commit으로 만들고, master branch가 해당 commit을 가리키도록 한다.
이와 같이 부모 commit이 여러 개인 commit을 **merge commit**이라고 한다.



[실습] Branch 병합 실습 준비

- 12-(Online)Git-basic2 강의에서 사용했던 VectorTest 코드를 새로운 디렉터리에 새롭게 작성하여 실습을 시작하자.

[실습] Branch 병합 실습 준비

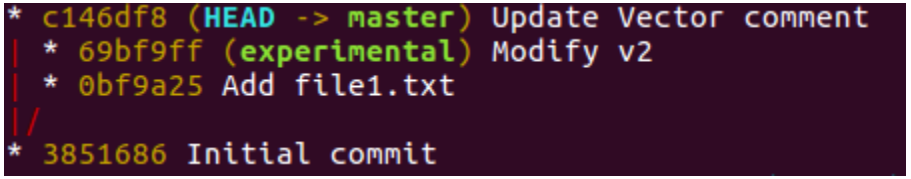
(Shell)

```
git init
git add .
git commit -m "Initial commit"

git checkout -b experimental
vi file1.txt  # hello
git add file1.txt
git commit -m "Add file1.txt"
vi main.cpp  # Vector v2(1.2, 5.2);
git commit -a -m "Modify v2"

git checkout master
vi vector.h  # // Vector class
git commit -a -m "Update Vector comment"
```

[실습] Branch 병합 실습 준비

(Shell)

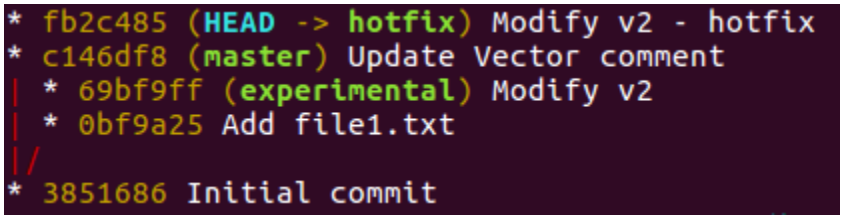
```
git log --branches --decorate --graph --oneline
```



[실습] Fast-Forward Merge 준비

(Shell)

```
git checkout -b hotfix
vi main.cpp  # Vector v2(10.2, 5.2);
git commit -a -m "Modify v2 - hotfix"

git log --branches --decorate --graph --oneline
```



[실습] Fast-Forward Merge

(Shell)

```
git checkout master
git merge hotfix

git log --branches --decorate --graph --oneline
```

```
* fb2c485 (HEAD -> master, hotfix) Modify v2 - hotfix
* c146df8 Update Vector comment
| * 69bf9ff (experimental) Modify v2
| * 0bf9a25 Add file1.txt
|/
* 3851686 Initial commit
```

[실습] 3-Way Merge

(Shell)

```
git merge experimental

# 출력 메시지
Auto-merging main.cpp
CONFLICT (content): Merge conflict in main.cpp
Auto-merging file1.txt
Automatic merge failed; fix conflicts and then commit the
result.
```

Merge 전에 어떤 상황이었는지 살펴보면..

master	공통 조상 commit	experimental
		file1.txt `hello`

vector.h
`class Vector //`
`Vector class`
`...`
	vector.h
	`class Vector`
	`...`
		vector.h
		`class Vector`
		`...`

main.cpp
`...`
`Vector v2(10.2, 5.2);`
`...`
	main.cpp
	`...`
	`Vector v2(4.2, 5.2);`
	`...`
		main.cpp
		`...`
		`Vector v2(1.2, 5.2);`
		`...`

Merge 전에 어떤 상황이었는지 살펴보면..

master	공통 조상 commit	experimental
		file1.txt `hello`

vector.h
`class Vector //`
`Vector class`
`...`
	vector.h
	`class Vector`
	`...`
		vector.h
		`class Vector`
		`...`

main.cpp
`...`
`Vector v2(10.2, 5.2);`
`...`
	main.cpp
	`...`
	`Vector v2(4.2, 5.2);`
	`...`
		main.cpp
		`...`
		`Vector v2(1.2, 5.2);`
		`...`

3-Way Merge를 하면..

master	공통 조상 commit	experimental	merge 결과
		file1.txt hello	file1.txt hello
vector.h class Vector // Vector class ...	vector.h class Vector ...	vector.h class Vector ...	vector.h class Vector // Vector class ...
main.cpp ... Vector v2(10.2,5.2); ...	main.cpp ... Vector v2(4.2, 5.2); ...	main.cpp ... Vector v2(1.2, 5.2); ...	main.cpp ... ??? ...

충돌 (Conflict)

(Shell)

```
git status

# 출력 메시지
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

Changes to be committed:

      new file:   file1.txt

Unmerged paths:
  (use "git add <file>..." to mark resolution)

      both modified:   main.cpp
```

[실습] 충돌 해결 (Resolving Conflict)

main.cpp

```
#include <iostream>
#include "vector.h"

int main()
{
    Vector v1(1.1, 2.2);
<<<<<<< HEAD
    Vector v2(10.2, 5.2); -> 이 부분이 현재 branch (master),
=======
    Vector v2(1.2, 5.2);  -> 이 부분이 experimental branch 내용이라는 뜻
>>>>>>> experimental

    v1.print();
    v2.print();
    (v1+v2).print();

    return 0;
}
```

Git이 자동으로 merge할 수 없으니 프로그래머가 직접 충돌을 해결하라는 뜻

[실습] 충돌 해결 (Resolving Conflict)

- main.cpp에서 >>>>, <<<<, ==== 등을 지우고 제대로 고쳐서 저장한 후 add & commit하면 된다.
- Merge branch 'experimental' 와 같은 commit message가 미리 표시되어 있을 텐데, 그대로 commit하면 된다.

(Shell)

```
vi main.cpp
git add .
git commit

git log --branches --decorate --graph --oneline
```

```
*   64ed2cf (HEAD -> master) Merge branch 'experimental'
|\
| * 69bf9ff (experimental) Modify v2
| * 0bf9a25 Add file1.txt
* | fb2c485 (hotfix) Modify v2 - hotfix
* | c146df8 Update Vector comment
|/
* 3851686 Initial commit
```

Conflict 발생 시 Merge 취소하는 방법

- 일단 지금은 취소하고 나중에 Merge 해야겠다고 생각한다면,
- git merge --abort 명령을 사용하면 됨.

(Shell)

```
git merge experimental

# 출력 메시지
Auto-merging main.cpp
CONFLICT (content): Merge conflict in main.cpp
Auto-merging file1.txt
Automatic merge failed; fix conflicts and then commit the
result.

git merge --abort
```

만일 3-Way Merge할 때 충돌이 일어나는 부분이 없다면?

- 만일 아래와 같았다면?

만일 3-Way Merge할 때 충돌이 일어나는 부분이 없다면?

- 알아서 Auto-merge가 된다.