

Git을 사용한 버전 관리

Git 가지 치기



```
/* elice */
```

`/* elice */`

수강 목표

Git에서 **브랜치**를 만들 수 있습니다.

여러 작업을 **독립적**으로 진행할 수 있습니다.

브랜치의 내용을 **병합**할 수 있습니다.

목차

1. Git Branch
2. fast forward
3. Merge
4. conflict 해결

Git Branch

Git Branch?

독립적으로 어떤 작업을 진행하기 위한 개념

각각의 Branch는 다른 Branch의 영향을 받지 않음

Git Branch?



그림 git-scm.com

Git Branch 종류

메인 Branch

배포할 수 있는 수준의
안정적인 Branch

토픽 Branch

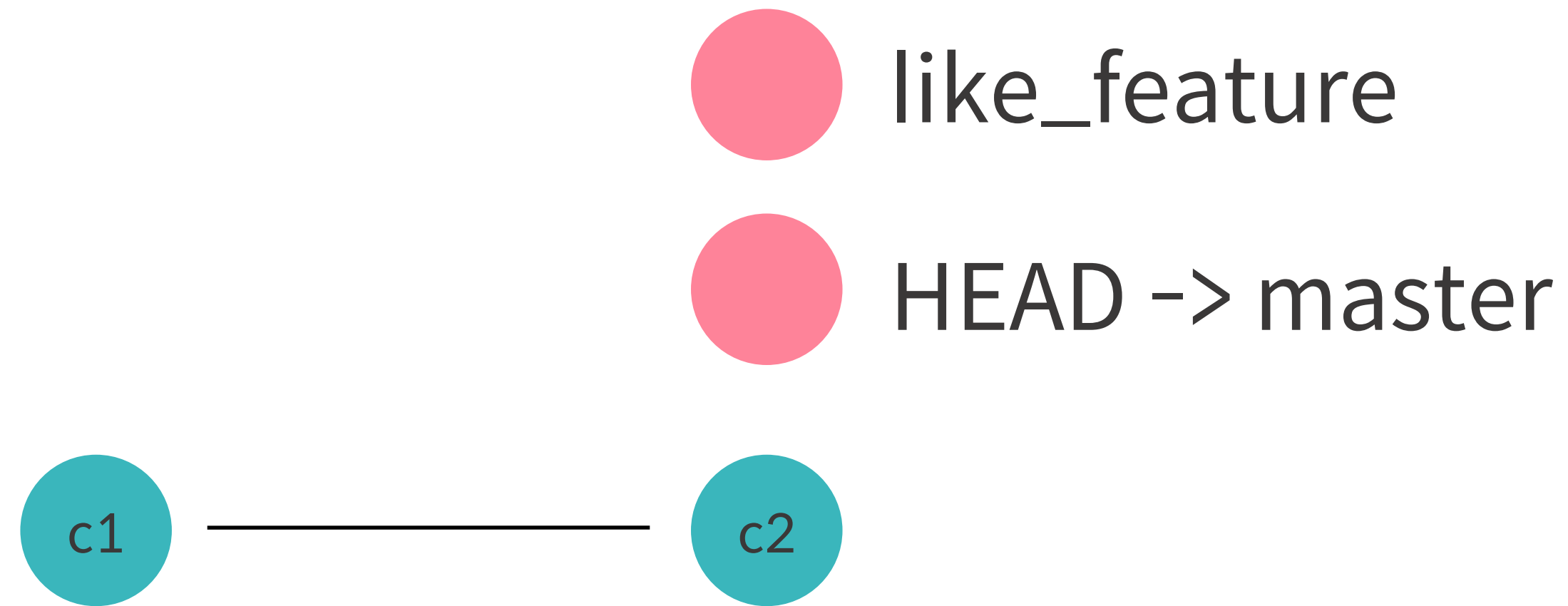
기능 추가나 버그 수정과 같은
단위 작업을 위한 Branch

Git Branch 생성

Branch는 아래의 명령어로 생성할 수 있습니다

```
$ git branch like_feature
```


Git Branch 생성



Git Branch 전환 (1)

현재의 Branch는 아래의 명령어를 통해 확인할 수 있습니다

```
$ git branch  
    like_feature  
* master
```

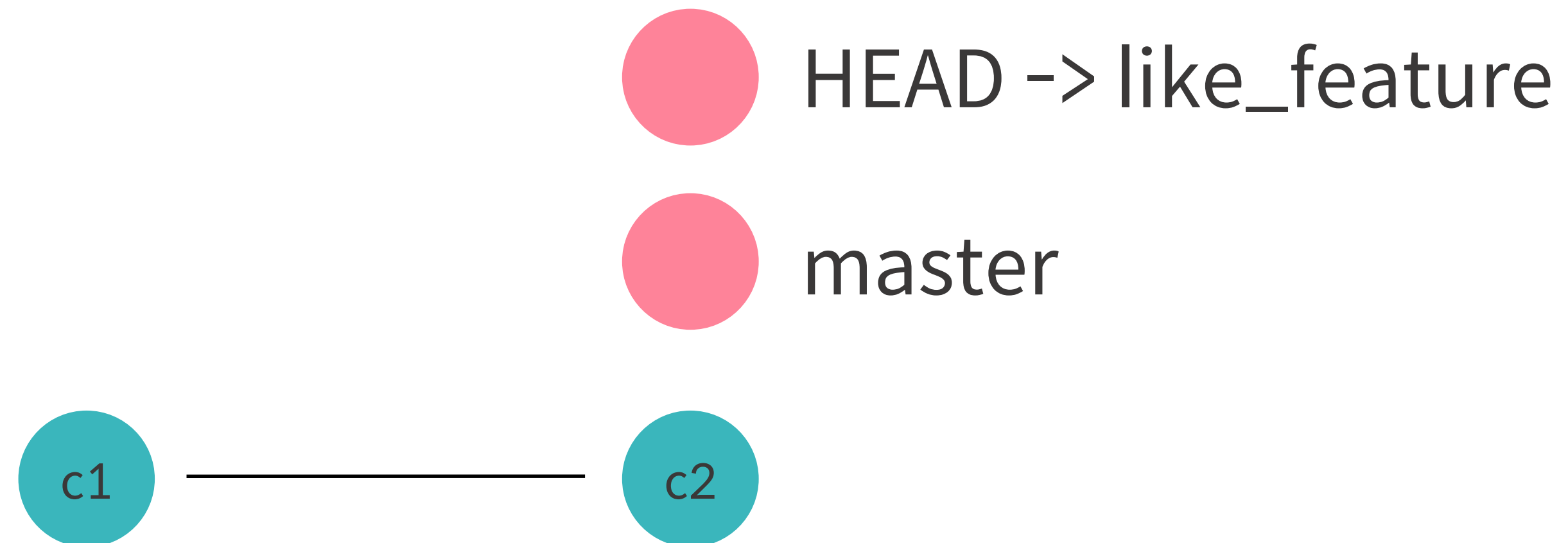
Git Branch 전환 (2)

Branch 전환은 아래의 명령어를 통해 할 수 있습니다.

```
$ git checkout like_feature
```

```
Switched to branch 'like_feature'
```

Git Branch 생성



Git Navigation

checkout은 branch를 전환하는데 사용할 수 도 있고
아래와 같이 `git log` 로 확인한 **snapshot**을 넘나들때도
사용이 가능합니다

```
git checkout <snapshot hash>
```

Git Navigation

```
$ git log --pretty=oneline  
e4abb6f... (HEAD -> master) this is master  
d97d387... another snapshot
```

```
$ git checkout d97d38
```

```
...  
HEAD is now at d97d38 another snapshot
```

```
$ git log --pretty=oneline  
e4abb6f... (master) this is master  
d97d387... (HEAD) another snapshot
```

Git Navigation

이렇게 snapshot의 hash값을 이용하여
과거의 파일 내용을 확인 할 수 있습니다.

fast – forward

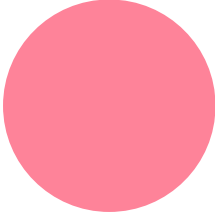
fast - forward

`like_feature` Branch의 working directory에서
새로운 정보를 넣어 commit해보겠습니다.

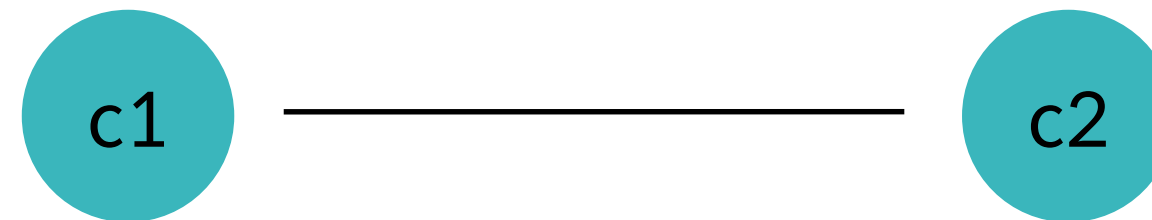
먼저 `like_feature` 로 위치를 이동시켜볼까요?

```
git checkout like_feature
```

fast - forward

 HEAD -> like_feature

 master



fast - forward



Git Merge

Git Merge

like_feature

Branch 에서의 작업을 끝나치고,

master

Branch 로 통합합니다

Git Merge

먼저 `master` Branch로 이동 하여

`like_feature` Branch를 병합합니다

```
$ git checkout master
```

```
Switched to branch 'master'
```

```
$ git merge like_feature
```

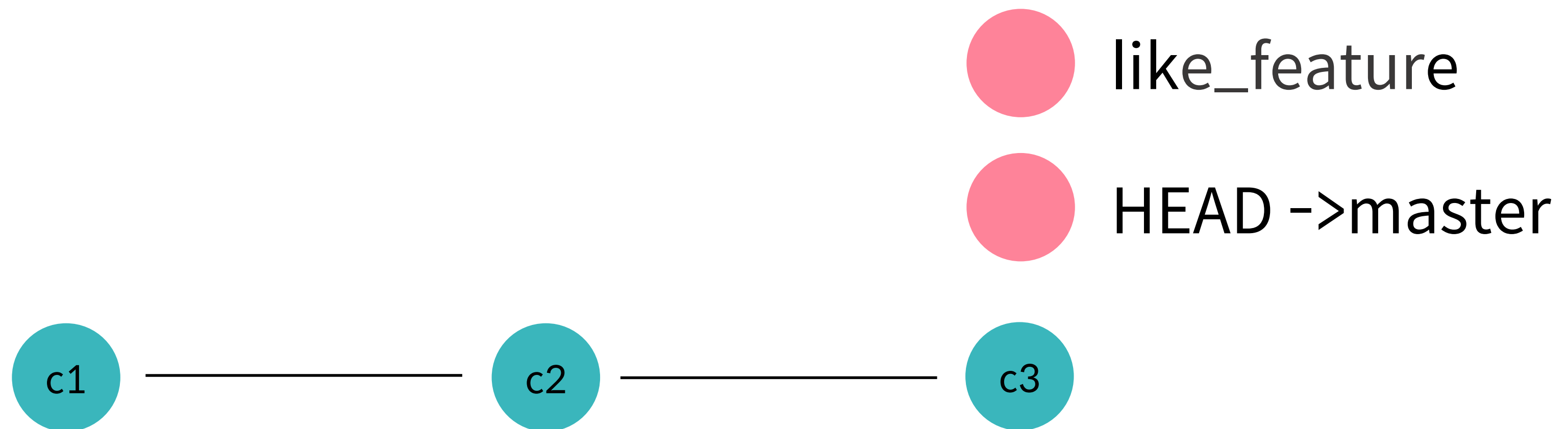
```
Updating d78ade4..a63hec2
```

```
Fast-forward
```

```
comment.js | 3 ++-
```

```
1 file changed, 2 insertions(+), 1 deletion(-)
```

fast - forward



fast - forward

like_feature Branch 의 내용이 master Branch에서
업데이트 된 내용이기 때문에 곧바로 merge가 되는 것을
확인 할 수 있습니다

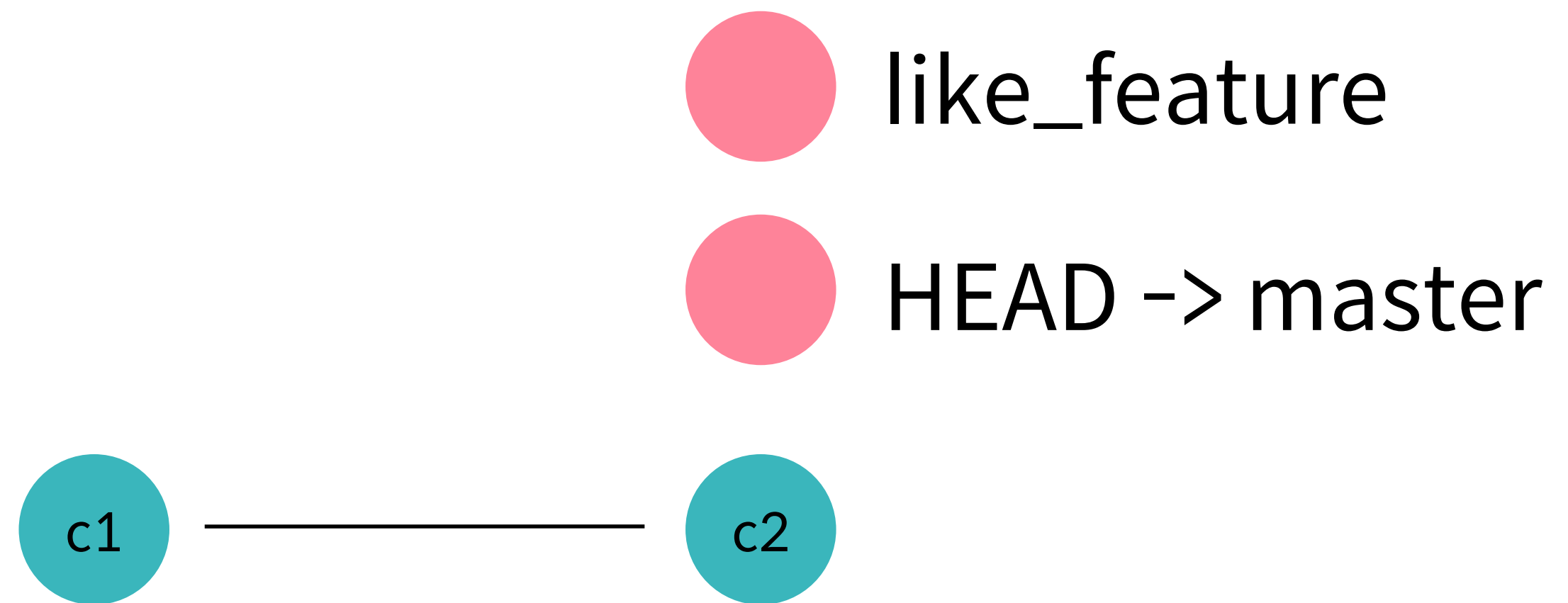
이렇게 merge가 이루어지는 것을 fast-forward 라고 부릅니다

갈라지는 branch

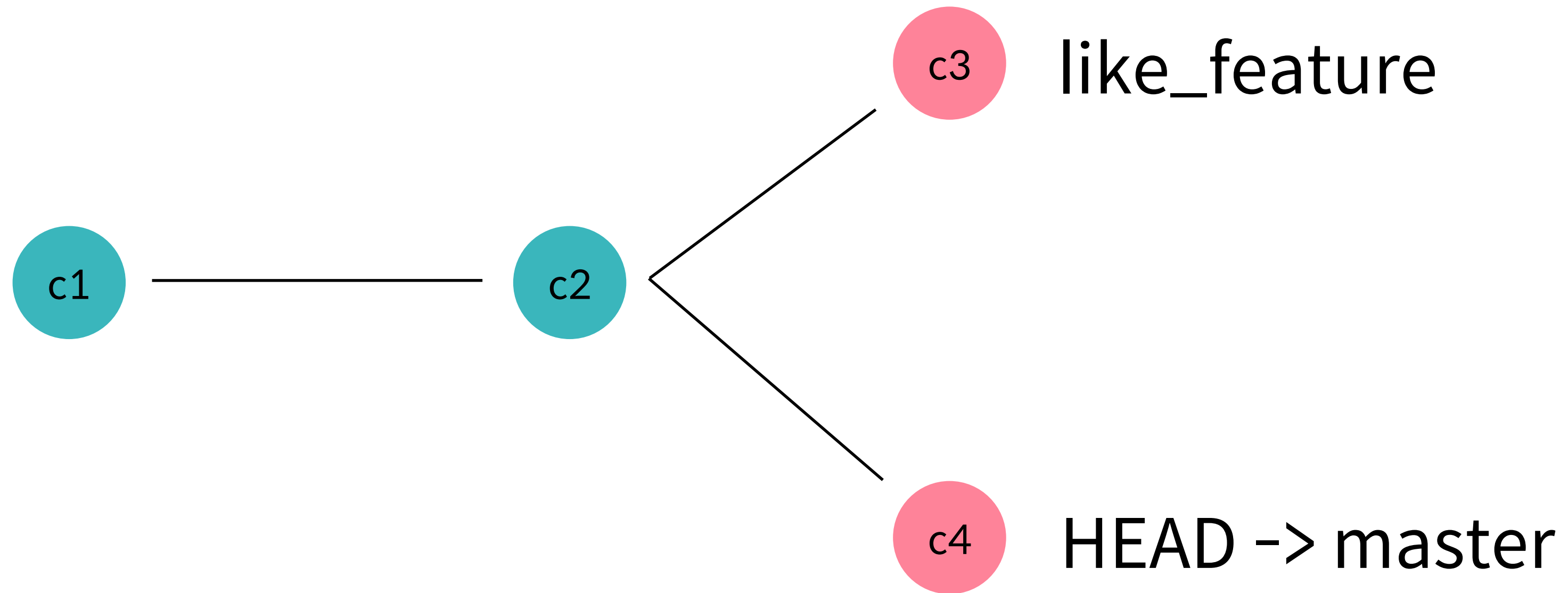
이번에는 각각의 Branch의 working directory에서
같은 파일의 내용을 다르게 수정해보겠습니다

명심하세요! 각각의 Branch는 다른 Branch의
영향을 받지 않기 때문에, 여러 작업을 동시에 진행 할 수 있습니다

갈라지는 branch



갈라지는 branch



갈라지는 branch

`git log --graph --all` 을 사용하면

commit graph를 확인 할 수 있습니다

추가로 옵션을 주어서 더 깔끔하게 볼 수도 있겠죠?

```
$ git log --pretty=oneline --graph --all
* c3360a...(HEAD->master) add master
| *782d90...(like_feature) add like_feature
| /
* 1c7881... init git
```

갈라지는 branch

`git checkout master` 을 이용하여 master로 checkout한 후

`git merge like_feature` 로 merge 해보겠습니다

갈라지는 branch

```
$ git checkout master
```

```
switched to branch 'master'
```

```
$ git merge like_feature
```

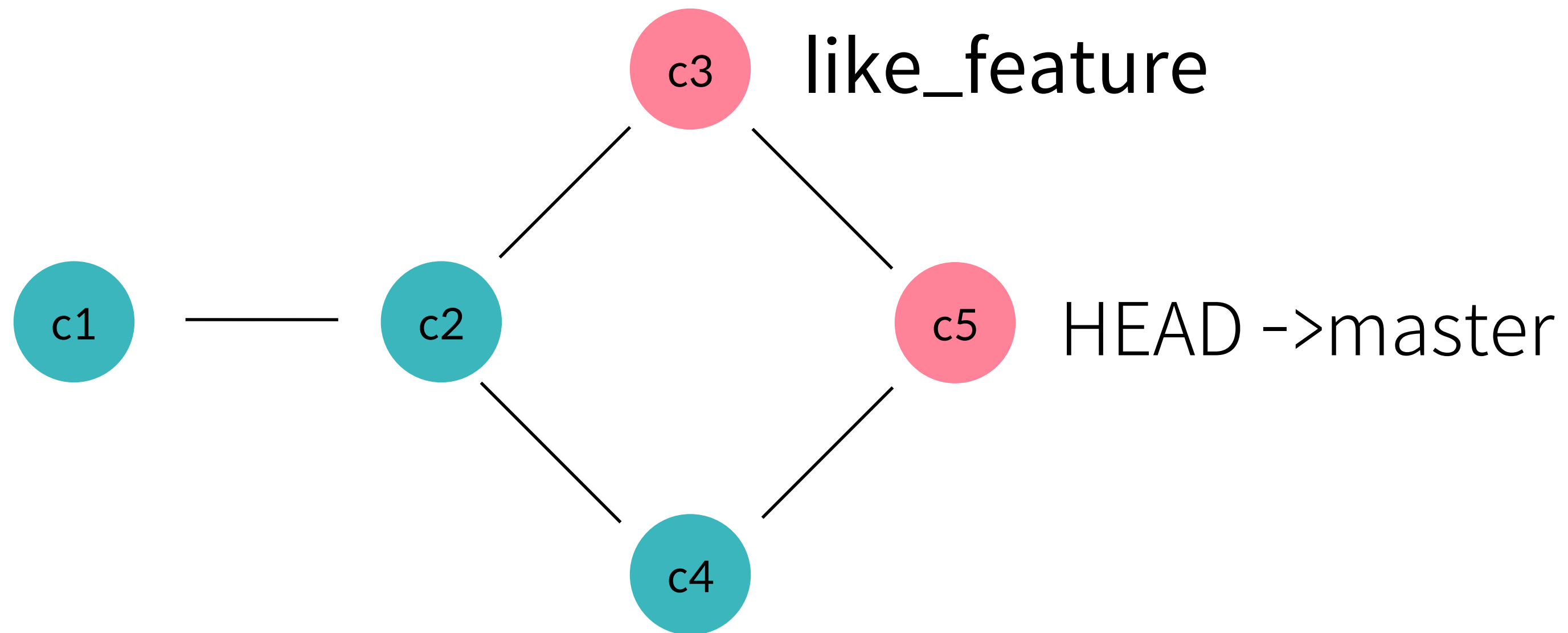
```
[main 2019-07-31T08:58:15.648Z] update#setState idle
```

```
Merge made by the 'recursive' strategy.
```

```
checkout.txt | 2 +-  
-  
+
```

```
1 file changed, 1 insertion(+), 1 deletion(-)
```

갈라지는 branch



Git Branch 삭제

아래의 명령어는 Merge된 Branch를 볼 수 있습니다

```
$ git branch --merged  
  like_feature  
* master
```


Git Branch 삭제

사용을 마친 branch는 `git branch -d <branch name>` 을
이용하여 삭제할 수 있습니다

```
$ git branch -d like_feature
Deleted branch like_feature (was 782d900).

$ git log --graph --pretty=oneline --all
*   3bf1a8... merging(HEAD -> master)
|\
| * 782d90... add like_feature
* | c3360a... add master
|/
* 1c7881 init git
```

Git Merge conflict

Merge conflict

Merge한 두 Branch에서
같은 파일을 변경했을 때 충돌이 발생합니다

Merge conflict

```
...  
$(".comment-good").text($(".comment-good").val()-1);  
...
```

```
...  
$(".comment-good").text($(".comment-good").val()-1 < 0 ? 0 :  
$(".comment-good").val()-1);  
...
```

Merge conflict

`git merge like_features` 명령을 수행했을 때

아래와 같이 충돌이 발생했습니다.

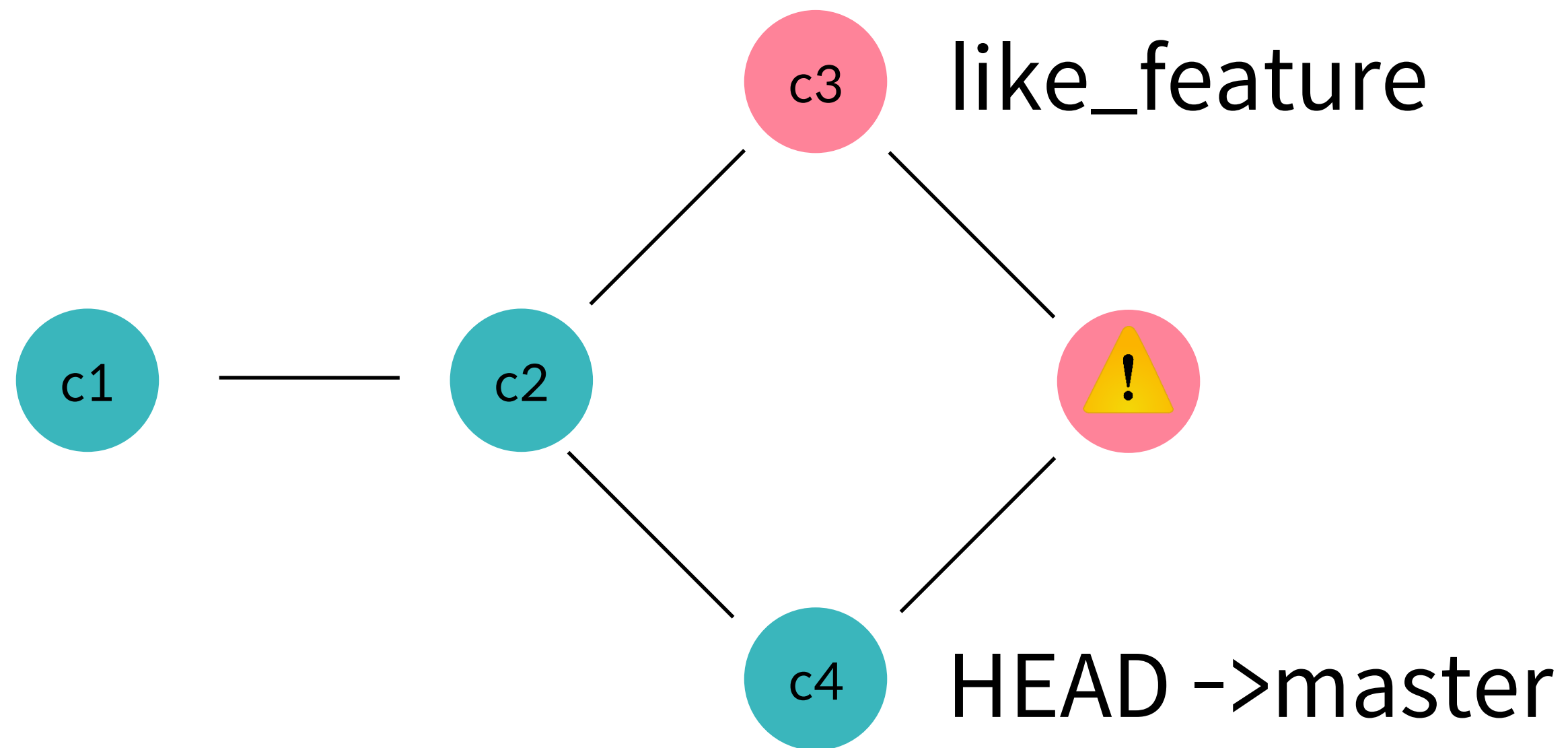
```
$ git merge like_feature
```

```
Auto-merging comment.js
```

```
CONFLICT (content): Merge conflict in comment.js
```

```
Automatic merge failed; fix conflicts and then commit the result.
```

Merge conflict



Merge conflict

`git status` 명령어로

어느 파일에서 충돌이 발생했는지 확인합니다.

```
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

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

        both modified:    comment.js

no changes added to commit (use "git add" and/or "git commit -a")
```

Git Merge 충돌 해결

충돌이 일어난 `comment.js` 파일을 열어봅니다.

```
<<<<<< HEAD
```

```
$(".comment-good").text($(".comment-good").val()-1);
```

```
=====
```

```
$(".comment-good").text($(".comment-good").val()-1 < 0 ? 0 :
```

```
$(".comment-good").val()-1);
```

```
>>>>>> like_feature
```


Git Merge 충돌 해결

수정 완료 후,

'<<<<<<<', '=====', '>>>>>>>'가 포함된 행을 삭제해줍니다.

```
$(".comment-good").text($(".comment-good").val()-1 < 0 ? 0 :  
$(".comment-good").val()-1);
```

Git Merge 충돌 해결

수정 완료 후 `git add`, `git commit` 과정을 거쳐
다시 Merge 해줍니다.

```
$(".comment-good").text($(".comment-good").val()-1 < 0 ? 0 :  
$(".comment-good").val()-1);
```

```
$ git merge like_feature  
[like_feature a63hec2] Merge branch 'master' into like_feature
```

Git Merge 충돌 방지

`master` Branch의 변화를
지속적으로 가져와서
충돌이 발생하는 부분을 제거



`/*elice*/`

contact@elice.io