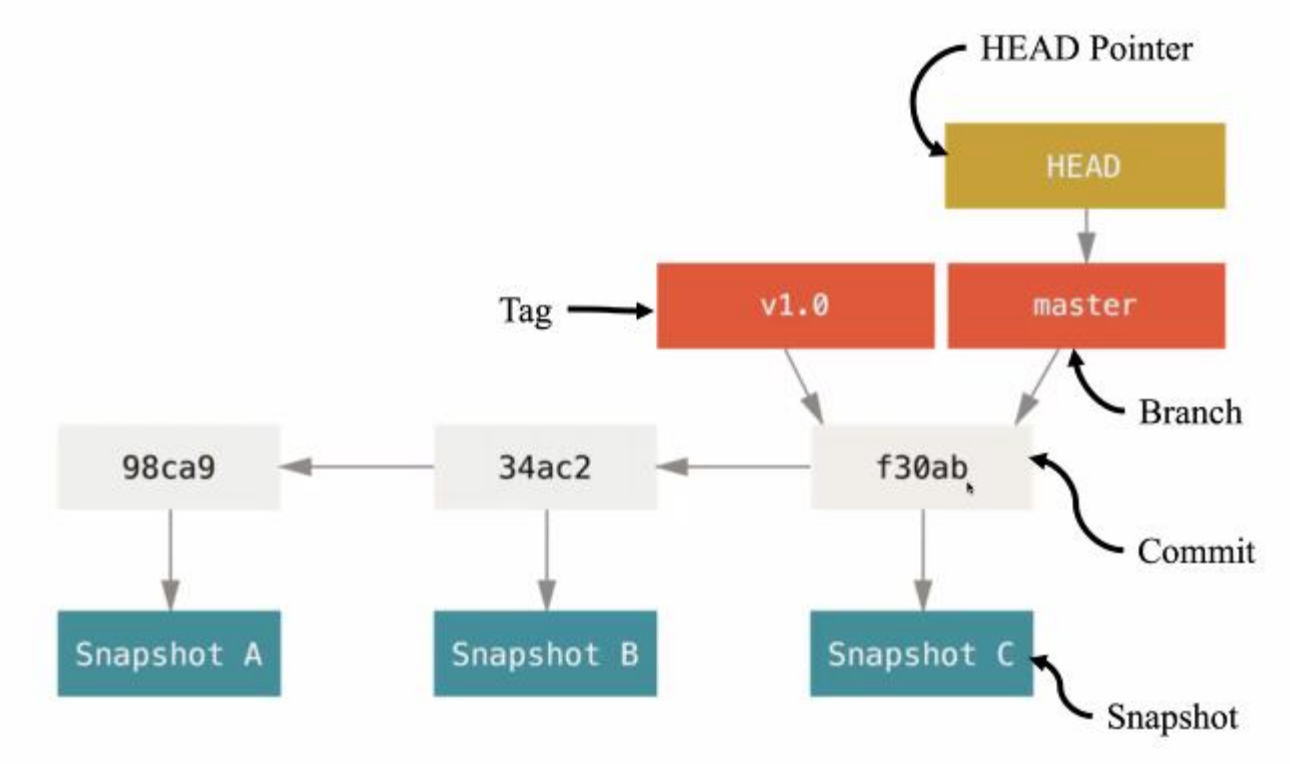
Git Branch

소프트웨어의 기능을 가지치면서 개발하기 위해 사용한다. 즉, **독립적인 개발**을 위해 사용한다.

1. Git 의 주요요소



- Snapshot: Commit 을 통한 변경 사항
- Tag 와 Branch: 특정한 commit 을 가리킴

1) SnapShot

- **Commit 을 할 때 작성한 변경 사항을 저장한다.**

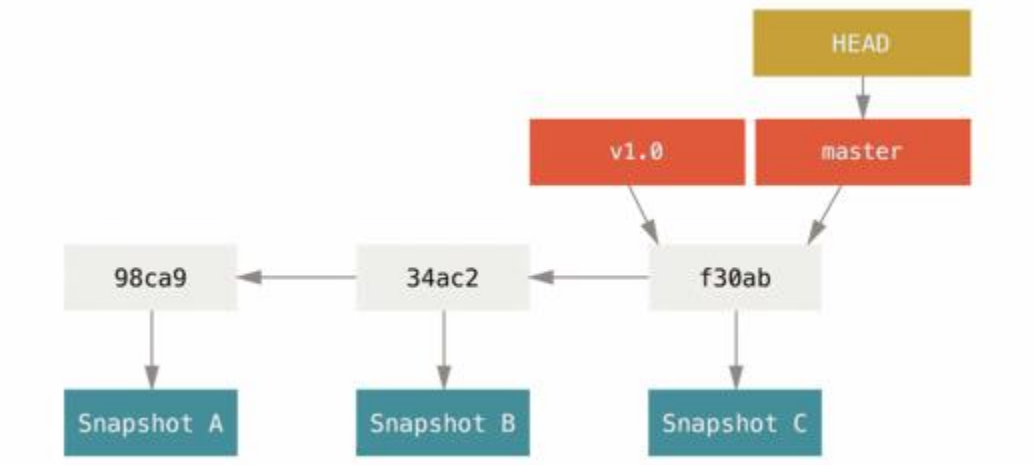
- 내부적으로 트리 형태로 저장되고 blob 이라는 형태로 관리된다.
- 누가 저장했고, commit 할 때 남긴 메세지도 저장한다.
- Commit 은 부모 자식관계를 가지지고 있어, 이전 Commit 을 가리킨다.
- 최초 Commit 을 제외한 나머지는 모두 부모를 가리킨다.

2) Branch

- 레포지터리 생성 시 디폴트 Branch 가 생성됨(master)
- git hub 에서는 main 으로 디폴트 Branch 를 가지는 것을 기본으로 가지고 있다.
- 독립적인 개발을 할 때 독립적인 branch 를 만들어 작업한다.
- 고객들의 요구사항이 다를 때도 각각의 기능에 대해 branch 를 만들어서 개발할 수 있다.
- 실험적인 기능을 개발할 때도 사용하기도 한다.
- 비선형 개발이 가능하다는 장점이 있다. (선형 개발을 하면 서로 개발 중인 것들끼리 영향을 미칠 수 있음)

3) HEAD

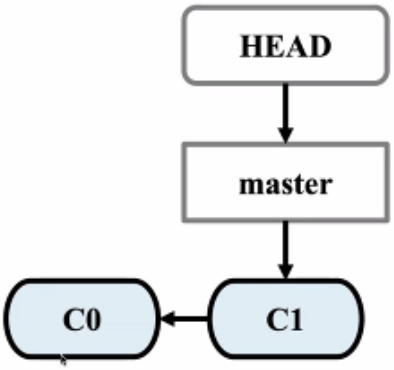
- 현재 작업 중인 branch 를 가리키는 것이다.



예제에서는 f30ab 가 현재 워킹디렉토리에서 작업 중

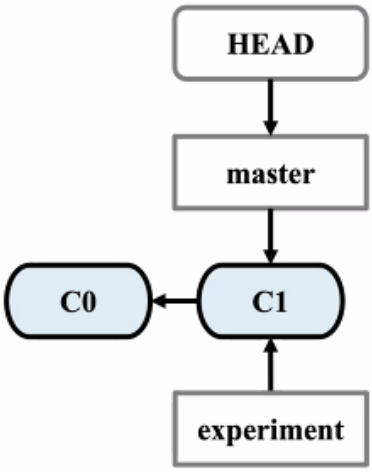
2. Git branch 순서

1) Git branch 순서



(1) 현재 상태

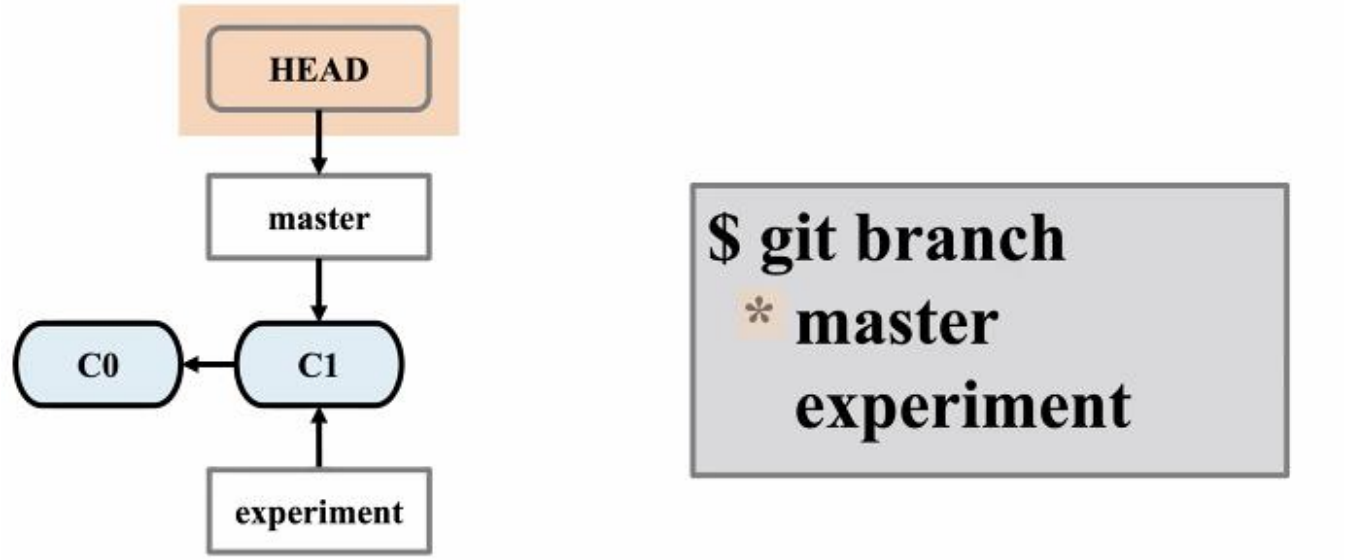
- C 는 commit 을 의미함
- master 라는 branch 는 C1 이라는 커밋을 가리킴
- HEAD 는 master 라는 branch 를 가리킴



git branch experiment

(2) experiment라는 branch를 만듦

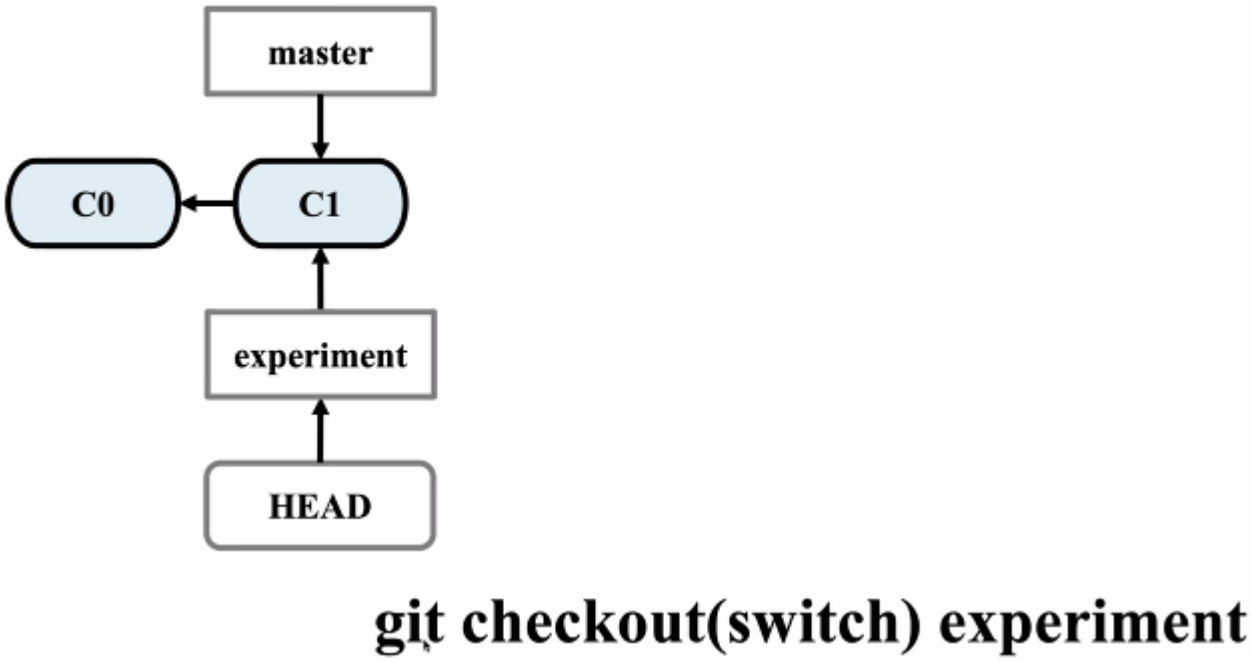
- git brabch experiment 라는 명령어로 experiment 라는 branch 를 만듦



(3)

branch와 사용중인 branch(*) 표시

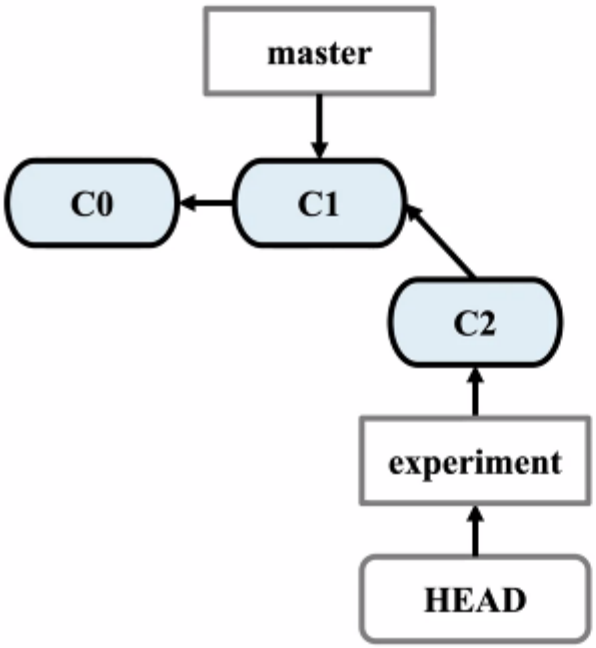
- git branch 를 통해 현재 디렉토리의 모든 branch 와 사용중인 branch(*)가 나옴



(4)

experiment branch 사용

- git checkout(또는 switch) experiment 라는 명령어로 experiment 로 사용 중인 branch 를 바꿈

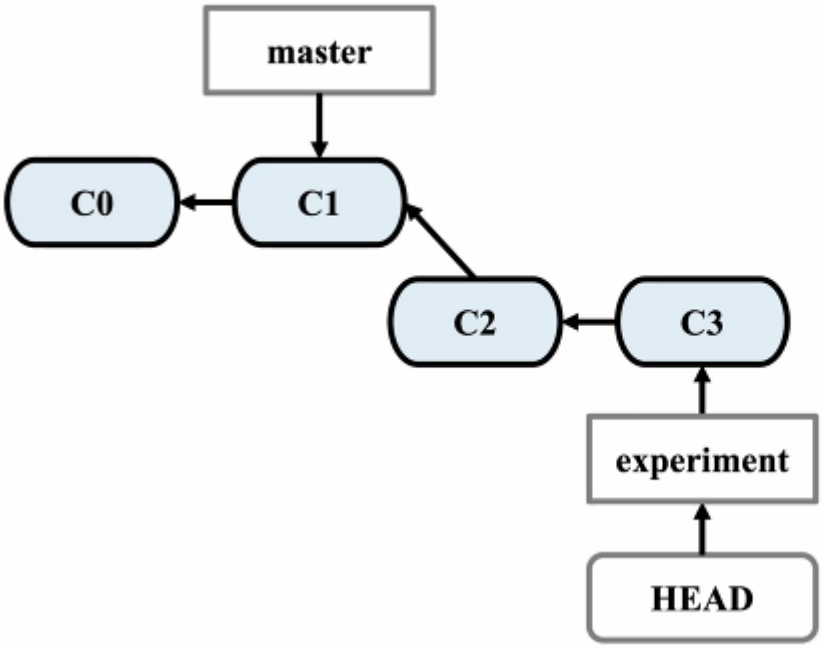


git commit

(5)

파일 수정 및 add, commit 진행

- C2 라는 commit 이 생기고 experiment 가 C2 를 가리킴
- 현재 사용중인 branch 도 commit 을 하는 것에 맞춰 이동함

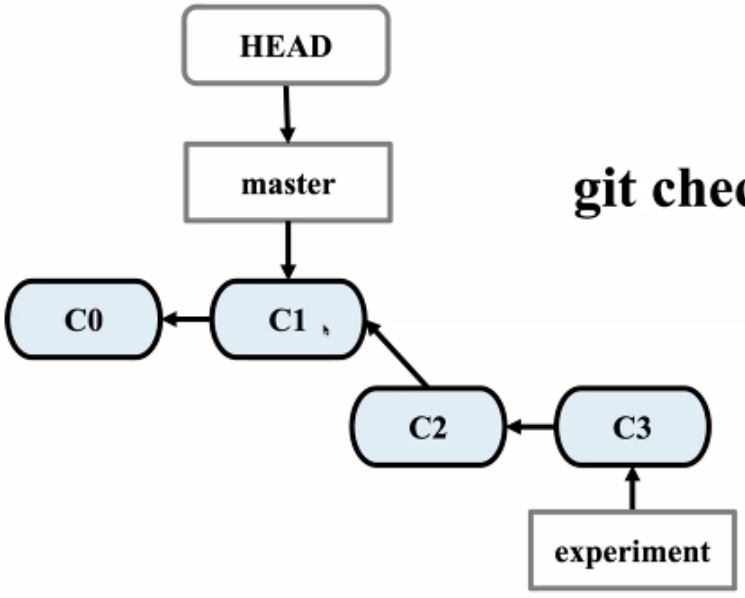


git commit

git commit

(6) 파일 수정 및 add, commit 진행 2

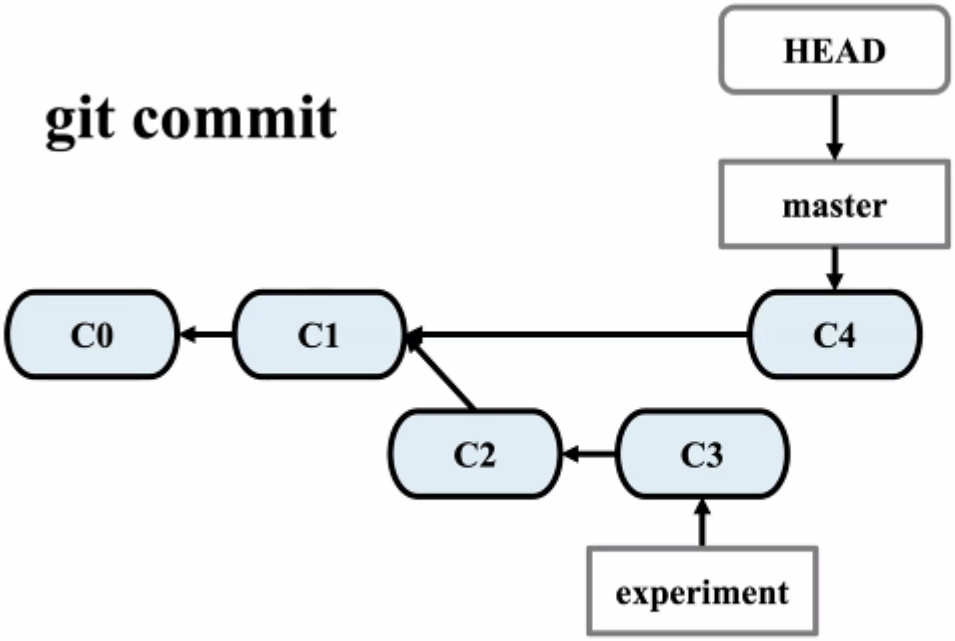
- experiment 라는 branch 가 C3 를 가리킴



git checkout(switch) master

(7) 작업중인 branch를 master로 바꿈

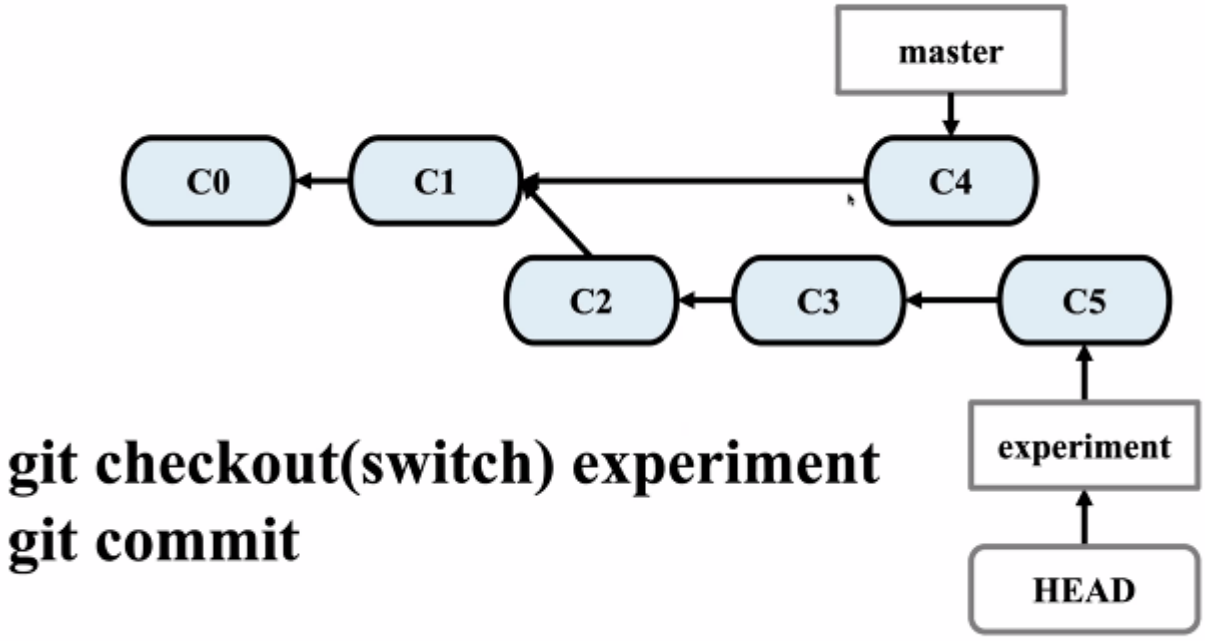
- 작업했던 branch 를 master 로 바꾸기 때문에 작업한 내용이 C1 으로 바뀜



(8)

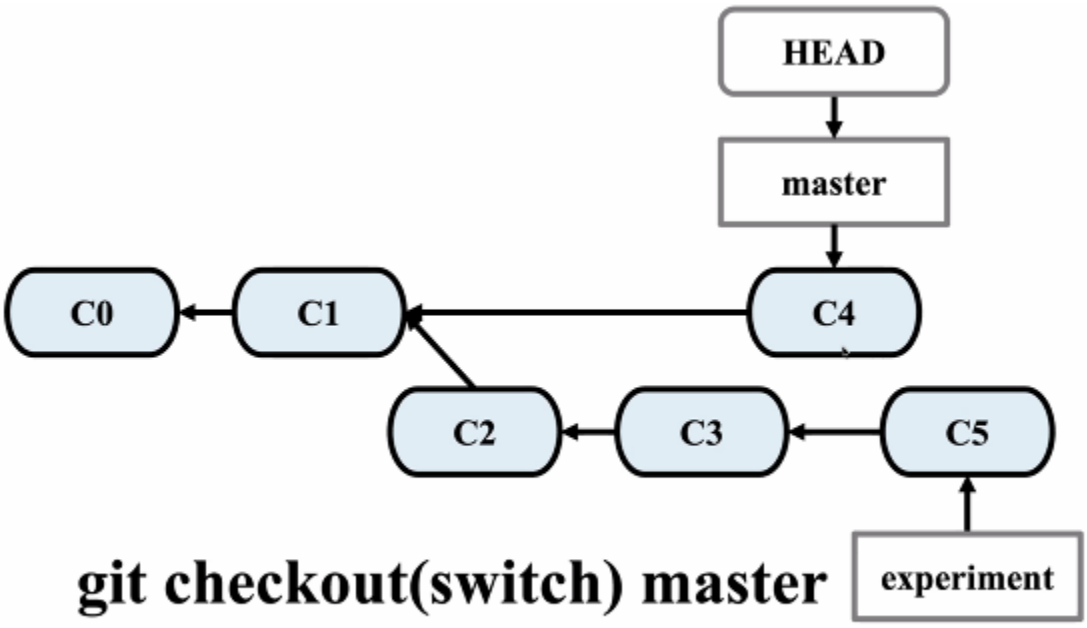
파일 수정 및 add, commit 진행 3

- 현재 작업중인 branch 가 master 이기에 C1 에서 C4 가 만들어지고 master 는 이를 가리킴
- C4 는 C1 을 가리킴



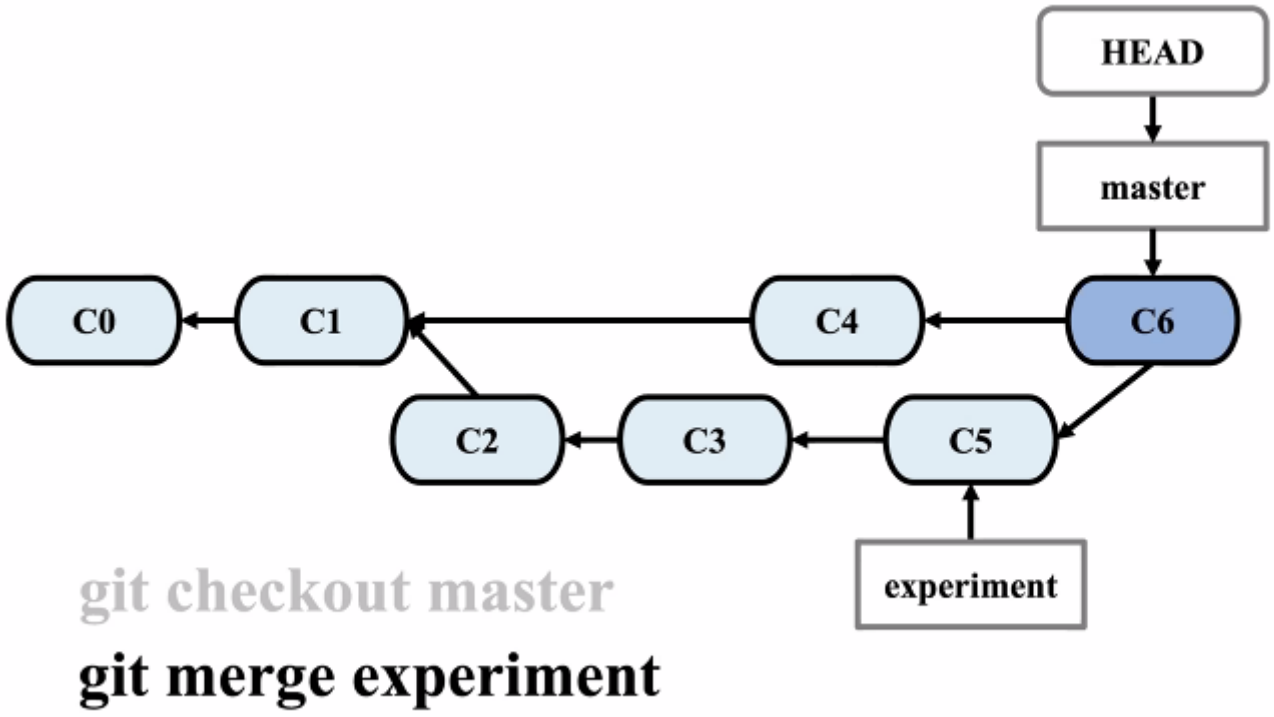
(9) 사용중인 branch를 experiment로 변경 후 commit

- 현재 작업 중인 branch 를 experiment 로 바꾸고 파일 수정, add, commit 을 진행함
- 이에 따라 C5 가 생성되고 experiment 는 이를 가리킴
- C5 는 C3 commit 을 가리킴



(10)

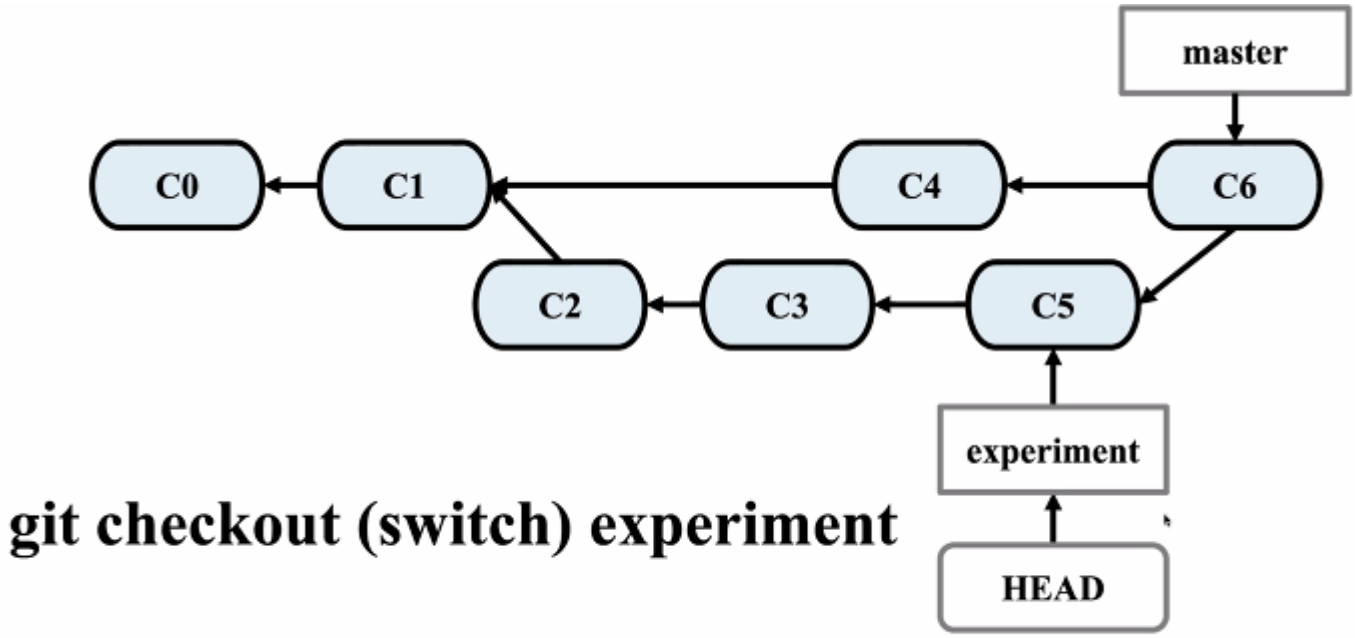
merge를 위한 기준 branch로 이동



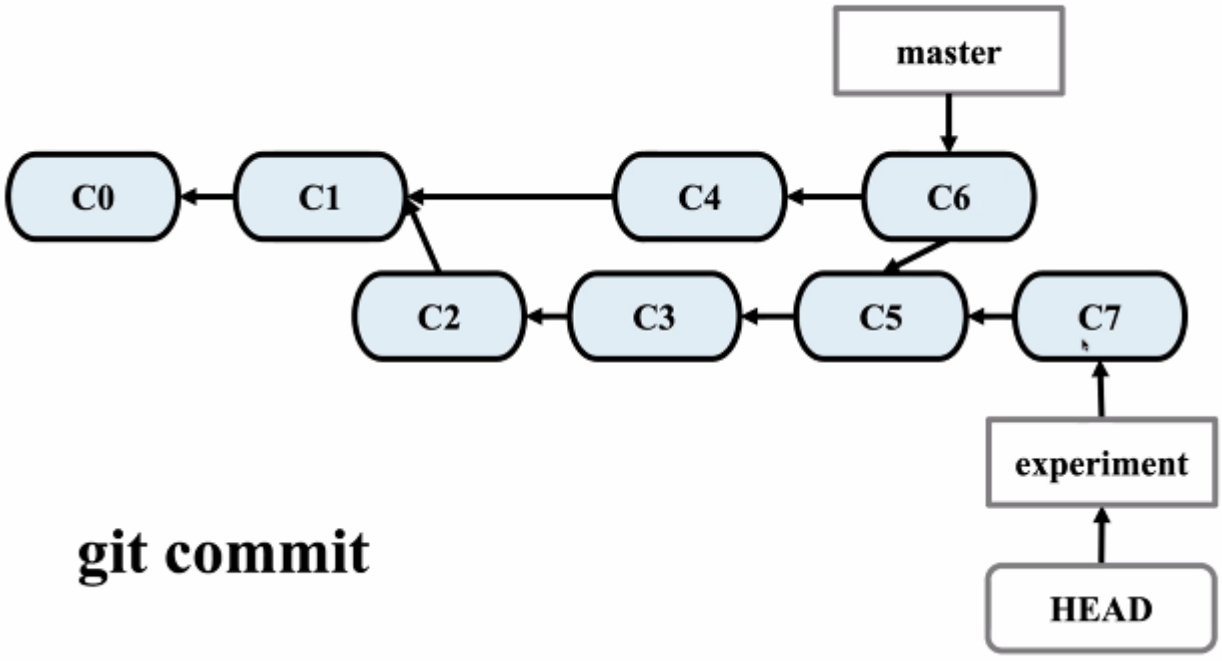
(11) master branch에서 merge 진행

- Merge: 두개의 branch 를 합친다.

- 합친 결과를 저장하고 싶은 branch 로 체크아웃을 한 뒤, 합치고 싶은 branch 를 지정하면 작업한 내역들이 사용한 branch 로 합쳐지고 C6 라는 commit 이 생김

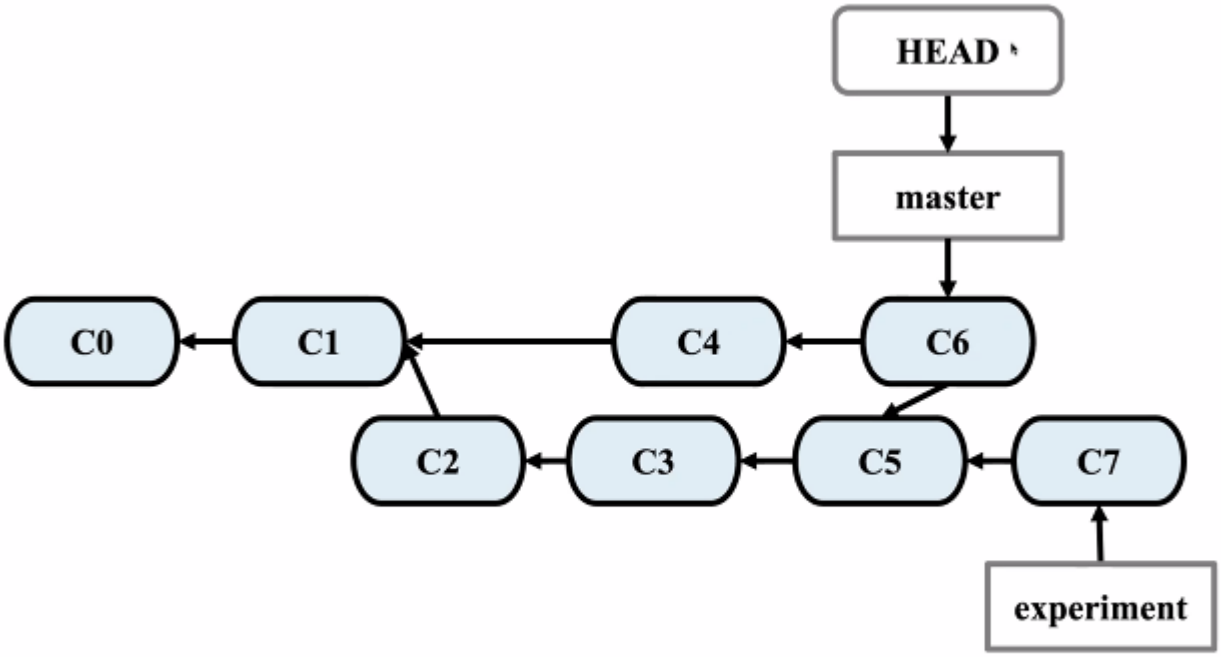


(12) 사용중인 branch를 experiment로 옮김



(13)

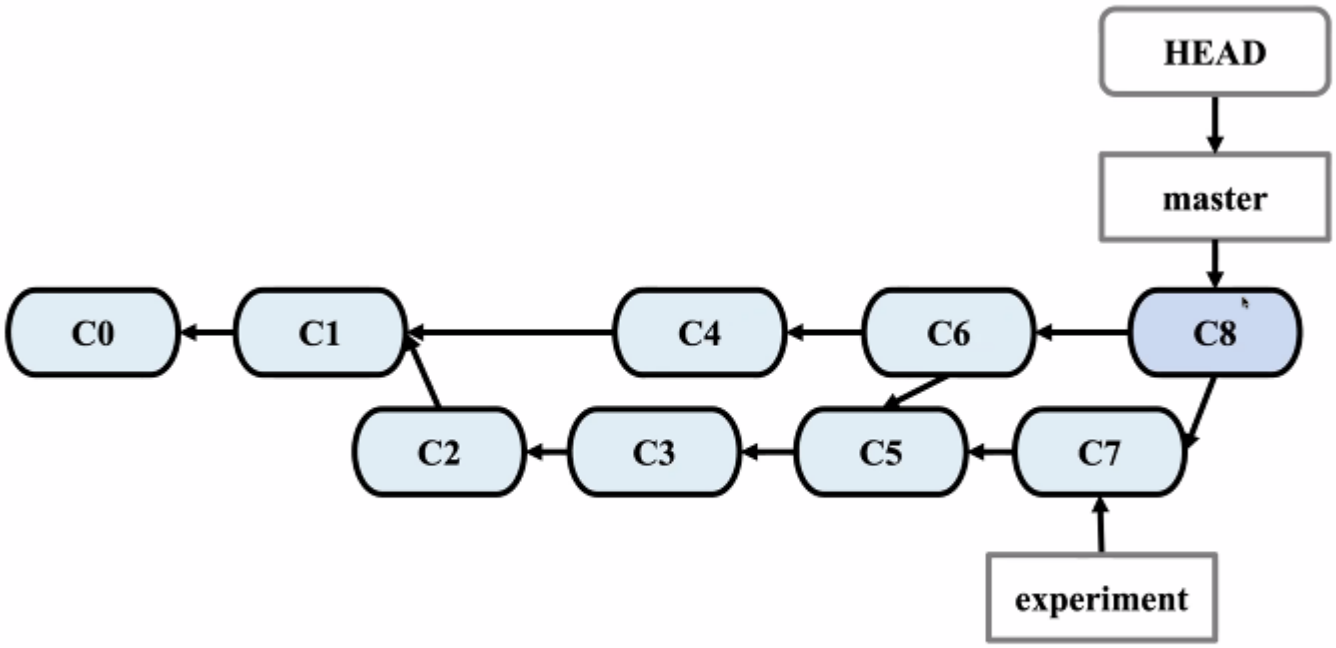
commit 진행



git checkout(switch) master

(14)

master로 사용 중인 branch를 옮김

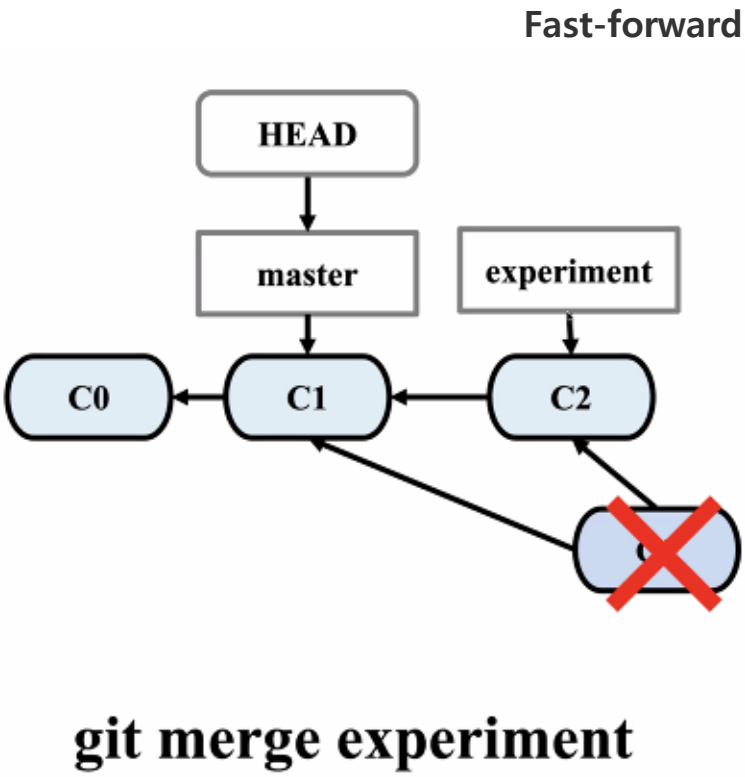


git merge experiment

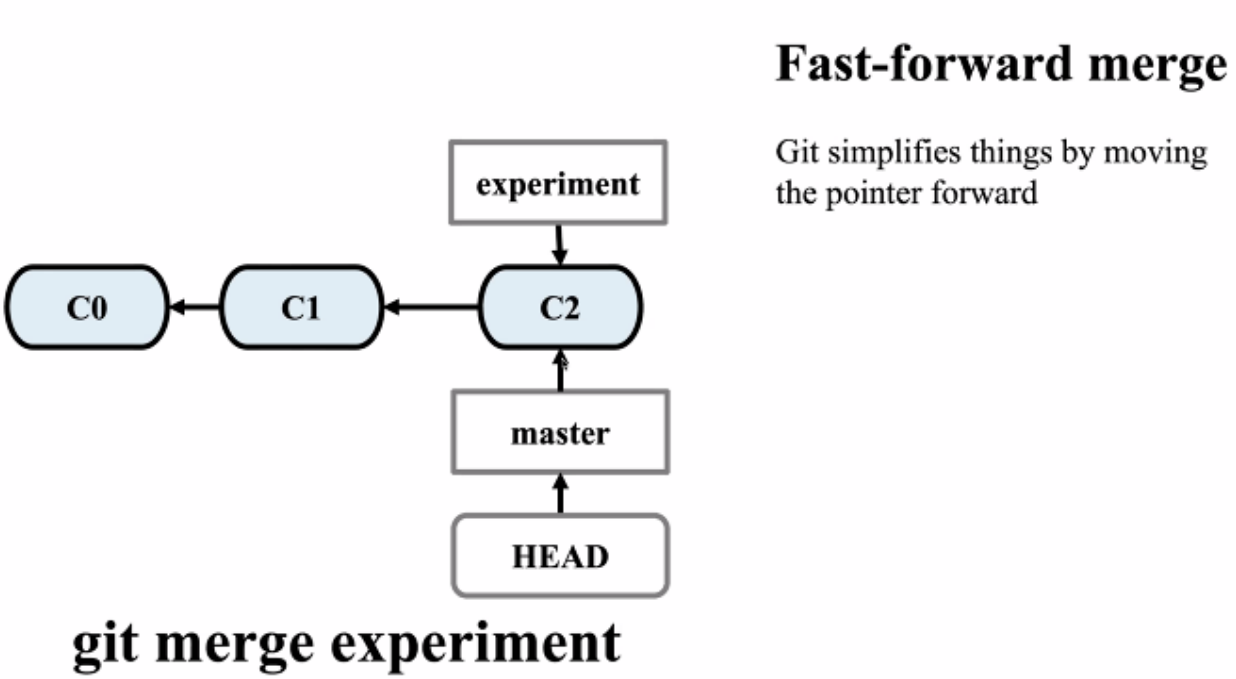
(15) 새로운 merge 진행

- C7 과 C6 를 합쳐서 새로운 commit 을 생성함

2) Git merge 사용 시 유의 사항



- C2 는 C1 에서 수정해서 만들었기 때문에 부모 자식관계인 경우에는 merge 를 할 때 새로운 commit 을 만들지 않는다.
- 따라서 아래와 같이 동작한다.

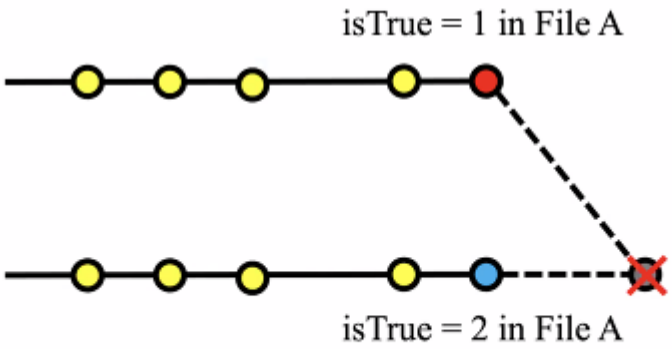


- 위의 경우를 **Fast-forward merge** 라고 하고 master 가 C2 를 가리킴

3) Git branch Conflicts

충돌이 발생하면 commit 중이되기에 다른 작업을 진행할 수 없기에 왜 충돌이 발생하는지와 해결 방법을 알아야한다.

- **Git automatically integrate new changes**
- **Conflicts Arise**
  - **Two people have changed the same line in a file**
  - **One people delete a file while another was modifying it**
  - **Git cannot automatically determine what is correct**



(1) 충돌은 다음과 같은 상황에서 발생한다.

i) 같은 라인에서 같은 부분을 수정해 충돌이 발생한 경우

ii) 한 파일에는 없지만 다른 파일에는 수정된 경우

iii) Git 이 자동적으로 수정할 수 없는 경우

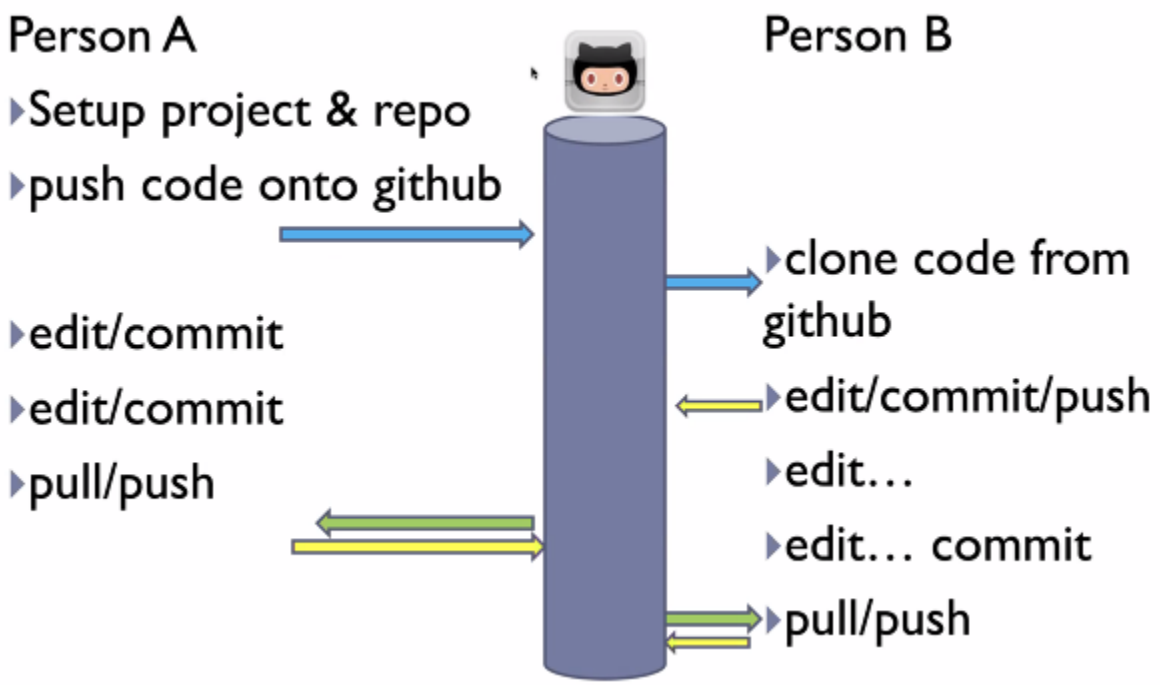
(2) 충돌 해결 방법

i) 충돌이 발생할 때는 직접 파일을 수정해주고 다시 merge 를 진행하는 방법이 있다.

ii) 해결할 수 없는 문제인 경우에는 git merge --abort 라고 작성해서 commit 중인 것을 취소할 수 있다.

3. Remote Repository

1) Remote Repository 와 관련된 개념



(1) workspace

- 워킹 디렉토리

(2) index

- stage area

(3) clone

- remote repository 에 저장되어있는 내용을 모두 복제한다.

(4) push

- tracking branch 로 지정되어있으면 upstream branch 에 push 하여 remote repository 로 전송되고 이를 반영한다.

(5) fetch

- remote repository 로부터 commit 을 받는다. 따라서 local 에 영향을 미치는 것은 없다.

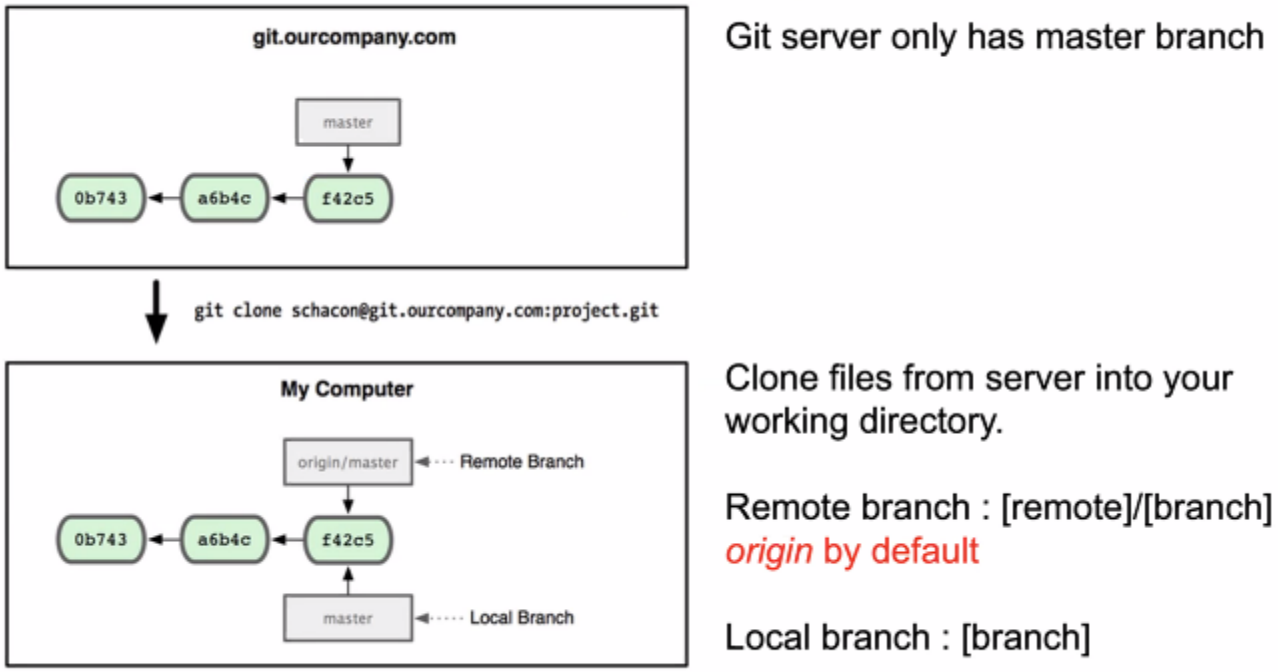
(6) pull

- 특정한 디렉터리를 지칭하여 가져오고 remote repository 에 저장하는 역할로 git fetch 와 git merge 와 동일한 작업을 한다. 이는 **remote 와 local repository 에 있는 내용을 모두 merge** 한다.

2) remote branch

- Remote Repository 가 가지고 있는 branch
- 서로 작업한 내용을 push 할 때 각각 원하는 branch 에 저장해야한다.

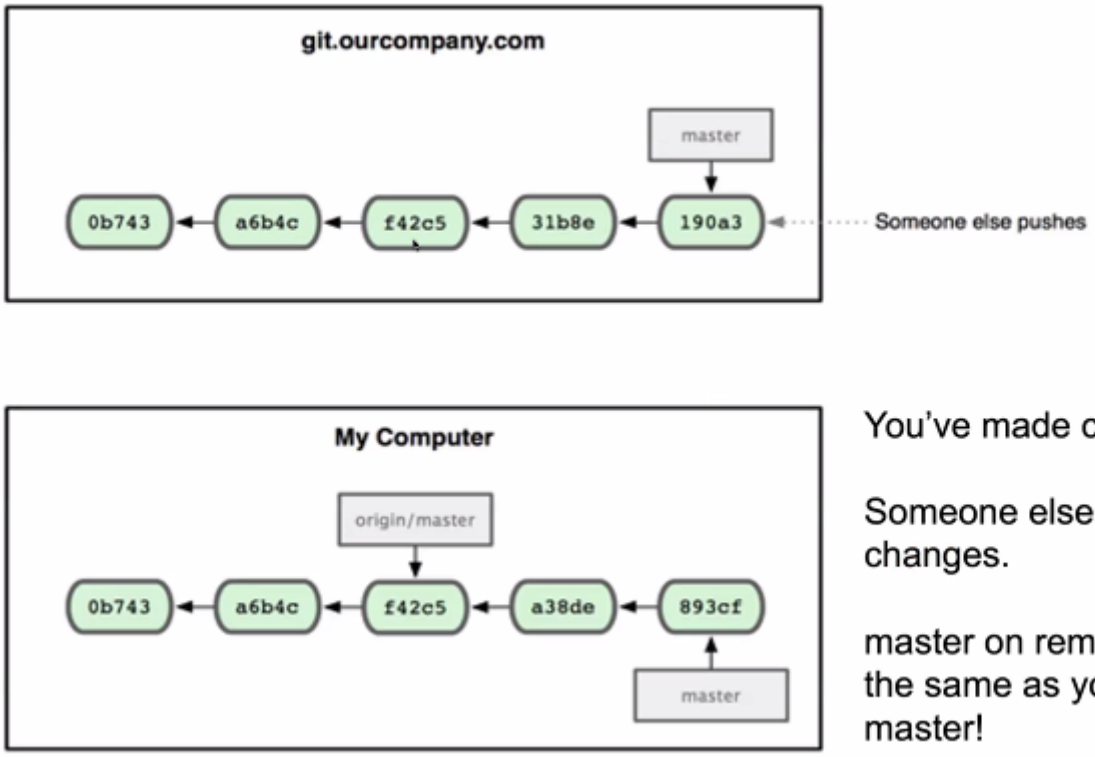
(1) remote branch 과정



(1)

clone

- remote repository 에 저장되어있는 것을 그대로 local repository 에 저장함
- 이때 local repository 에 있는 master branch 와 구분해주기 위해 origin/master 라는 이름으로 저장됨
- 이때는 [remote repository 의 별칭]/[branch]의 이름으로 branch 가 지정됨
- origin 은 remote repository 의 별칭을 정해주지 않았을 때 사용



(2)

remote repository와 local repository에서 각각 commit 진행

- origin/master 는 별도의 동작을 하지 않았기에 clone 을 했을 당시의 commit 을 가리킴
- local repository 에서는 따로 작업을 진행함
- **origin/master branch 는 upstream branch 을 의미한다.**
- **master branch 는 tracking branch 을 의미한다.**

2) tracking branch

- local branch 로부터 remote branch 를 추적하는 것을 의미한다.
- local branch 중에 특정 Remote Repository 에 있는 branch 를 바꿀 수 있는 것을 tracking branch 라고 한다.
- Remote Repository 에 있는 branch 로 Local Repository 의 branch 를 바꿀 수 있는 것을 upstream branch 라고 한다.
- 명시적으로 코드를 작성해 remote branch 를 생성할 수 있음