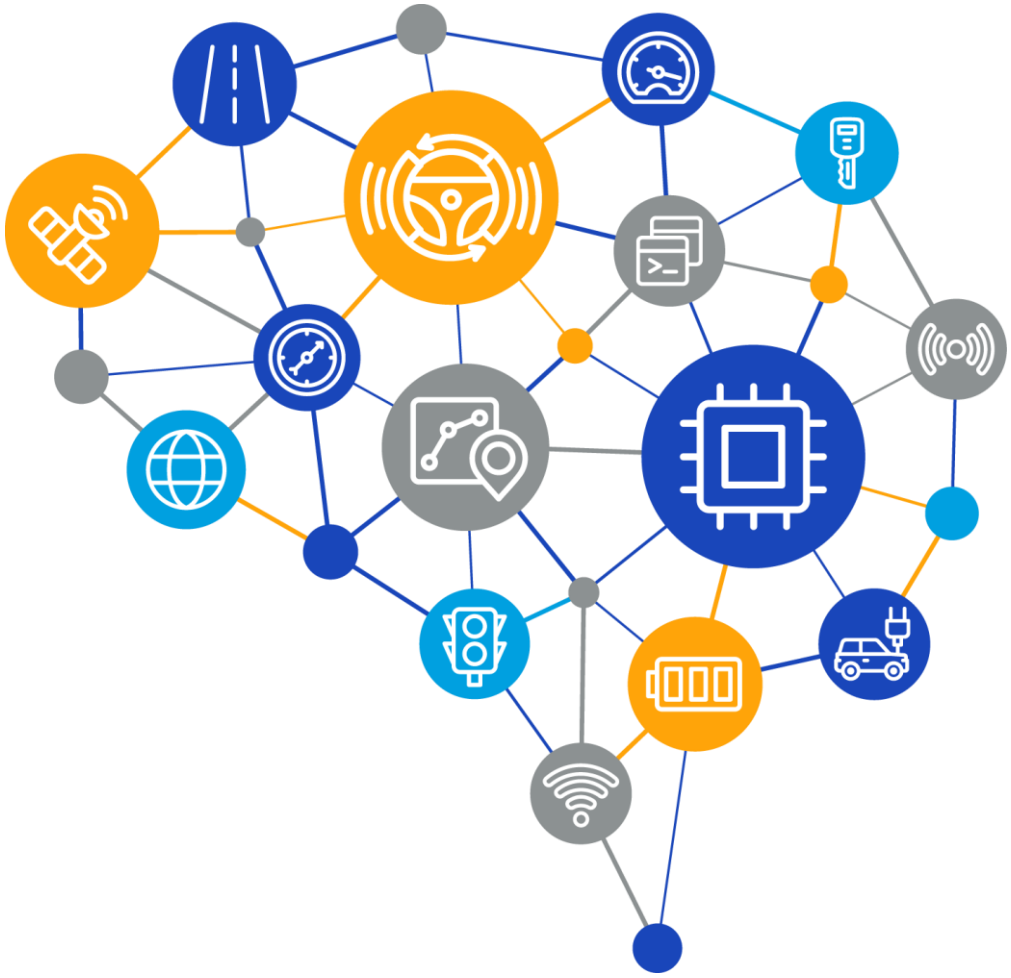
브랜치 rebase와 커밋이력 수정



Git & Github
for pull request

강환수 교수

DMUAi 동양미래대학교
인공지능소프트웨어학과

AI Experts Who Lead
The Future

학습목표

- **병합의 다른 방법인 rebase를 이해하고 수행할 수 있다.**
  - Base를 재배치하는 rebase 병합을 수행할 수 있다.
  - 3-way 병합과 rebase의 차이를 이해할 수 있다.
  - 커밋이력 수정을 위해 대화방식인 옵션 –i를 사용해 rebase -i로 수행할 수 있다.

AI Experts
Who Lead
The Future

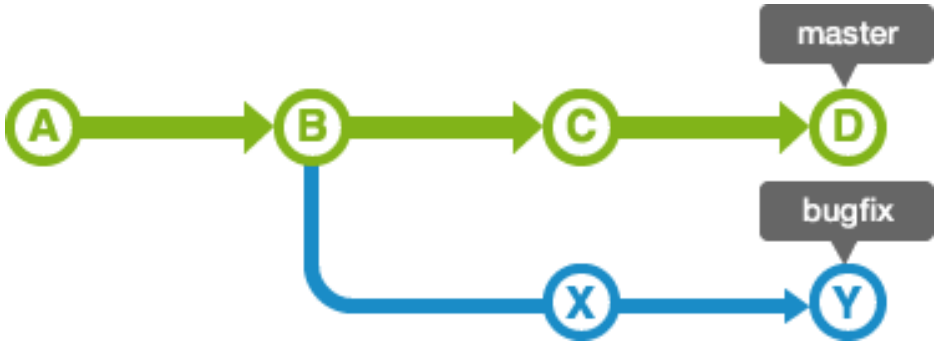
01

병합 rebase

https://meetup.toast.com/posts/116

선형적 통합 rebase 이해

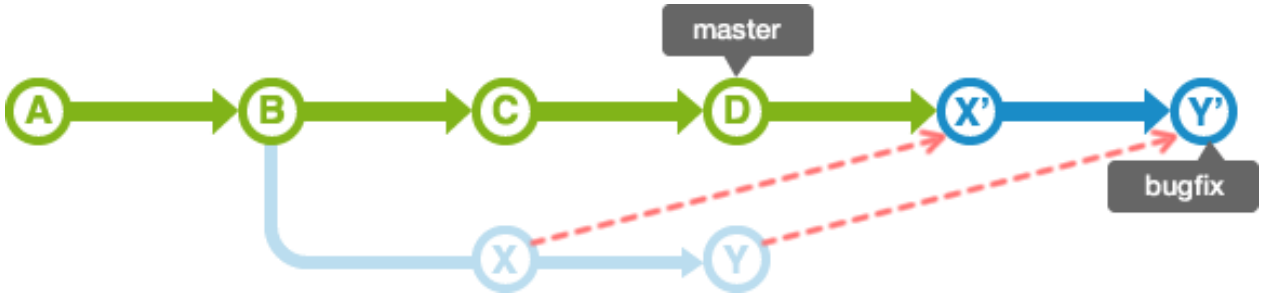
- 'master' 브랜치에서 분기하는 'bugfix' 브랜치가 있다고 가정



- Rebase 병합
  - 명령 rebase로 base를 수정하고
    - B에서 D로 수정, D 이후에 bugfix를 배치
  - 다시 'fast-forward 병합' 수행: 이 병합을 직접 다시 해야함
    - 'bugfix' 브랜치의 이력이 'master' 브랜치 뒤로 이동
      - 이력이 하나의 줄기로 이어짐

rebase를 이용한 브랜치 통합

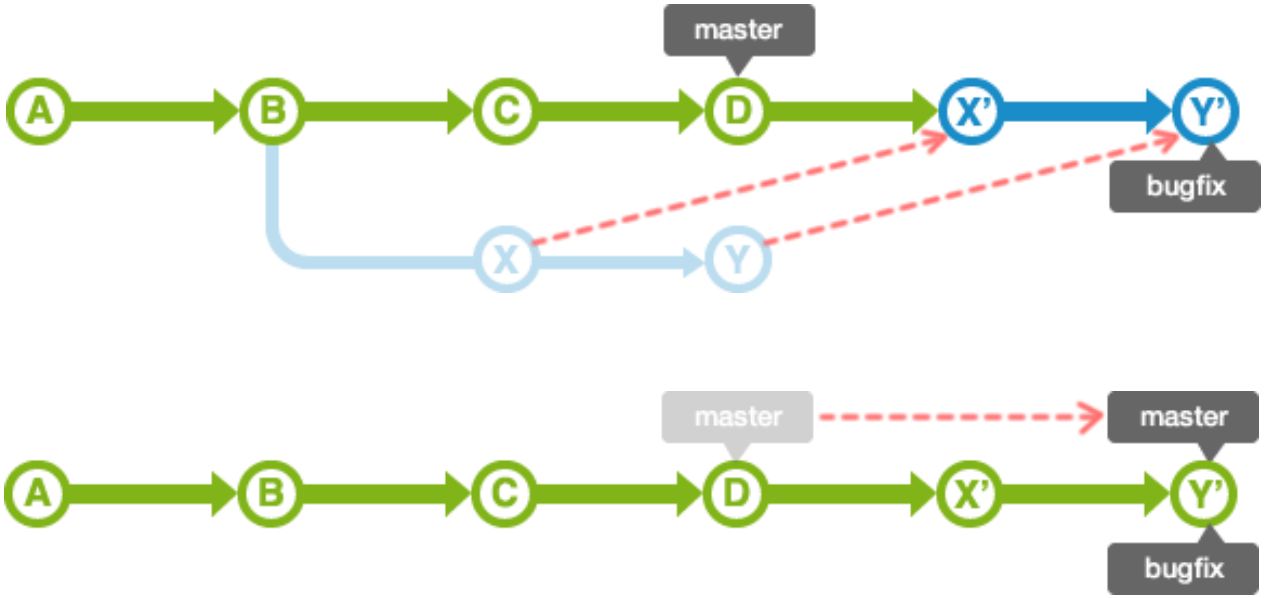
- 'fast-forward 병합' 방식
  - 'master' 브랜치 뒤로 'bugfix' 브랜치의 이력이 이동
    - 이력이 하나의 줄기로 이어짐
  - 충돌 발생 가능



- 'rebase'만 하면 다음 그림처럼 'master'의 위치는 그대로 유지
  - 'master' 브랜치의 위치를 변경하기 위해서는
    - 'master' 브랜치에서 'bugfix' 브랜치를 fast-foward(빨리감기) 병합 필요

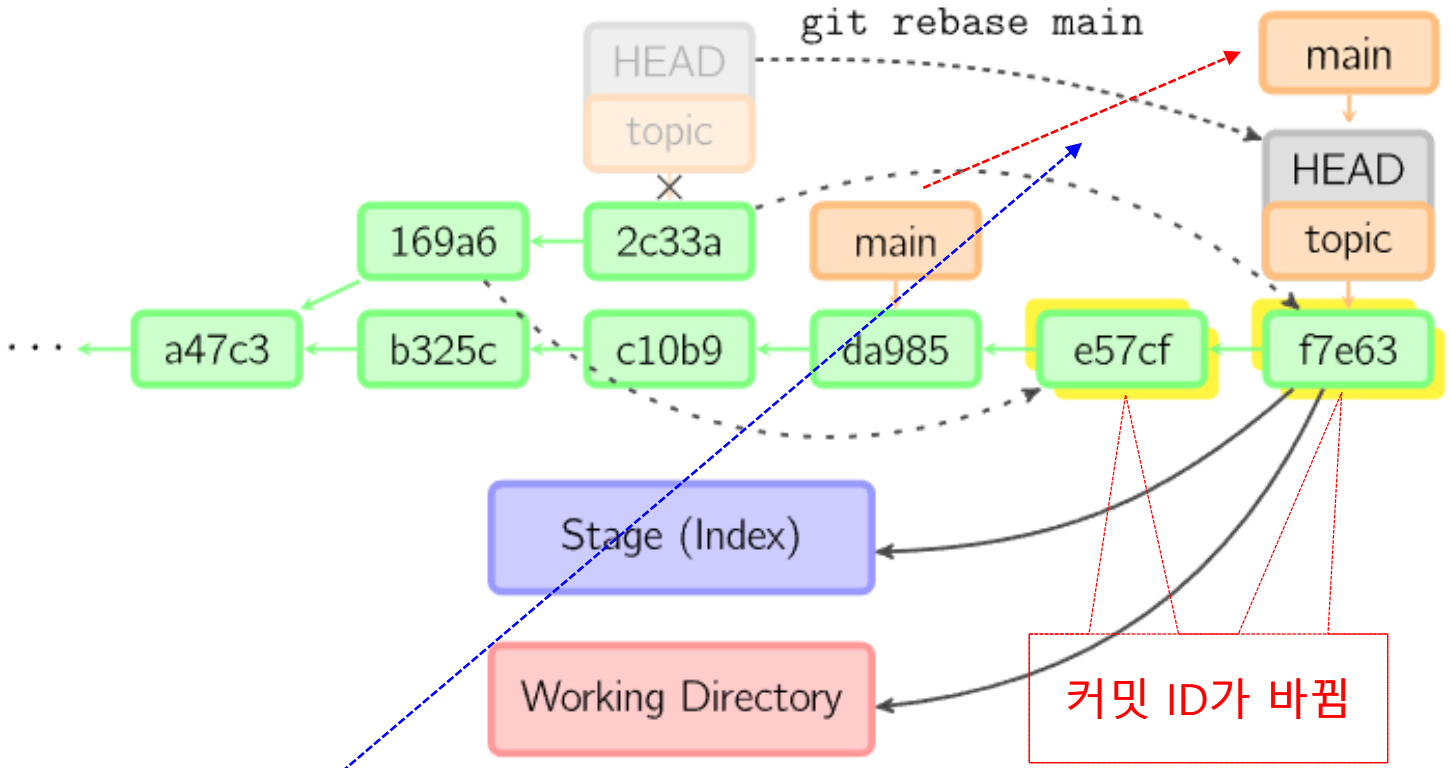
Rebase에서의 충돌

- 충돌 발생 가능
  - 이동되는 X와 Y의 내용이 'master'의 C, D 커밋들과 충돌하는 부분이 생길 수 있음
  - 각각의 커밋에서 발생한 충돌 내용을 수정 후, 추가, 계속 수행 필요
- 충돌 발생 후 해결 절차
  - 1. 파일 수정
  - 2. 파일 추가
    - $ git add <수정파일>
  - 3. rebase 계속 수행, 마지막 메시지 메시지 수정
    - $ git rebase --continue



명령 rebase <선행브랜치>

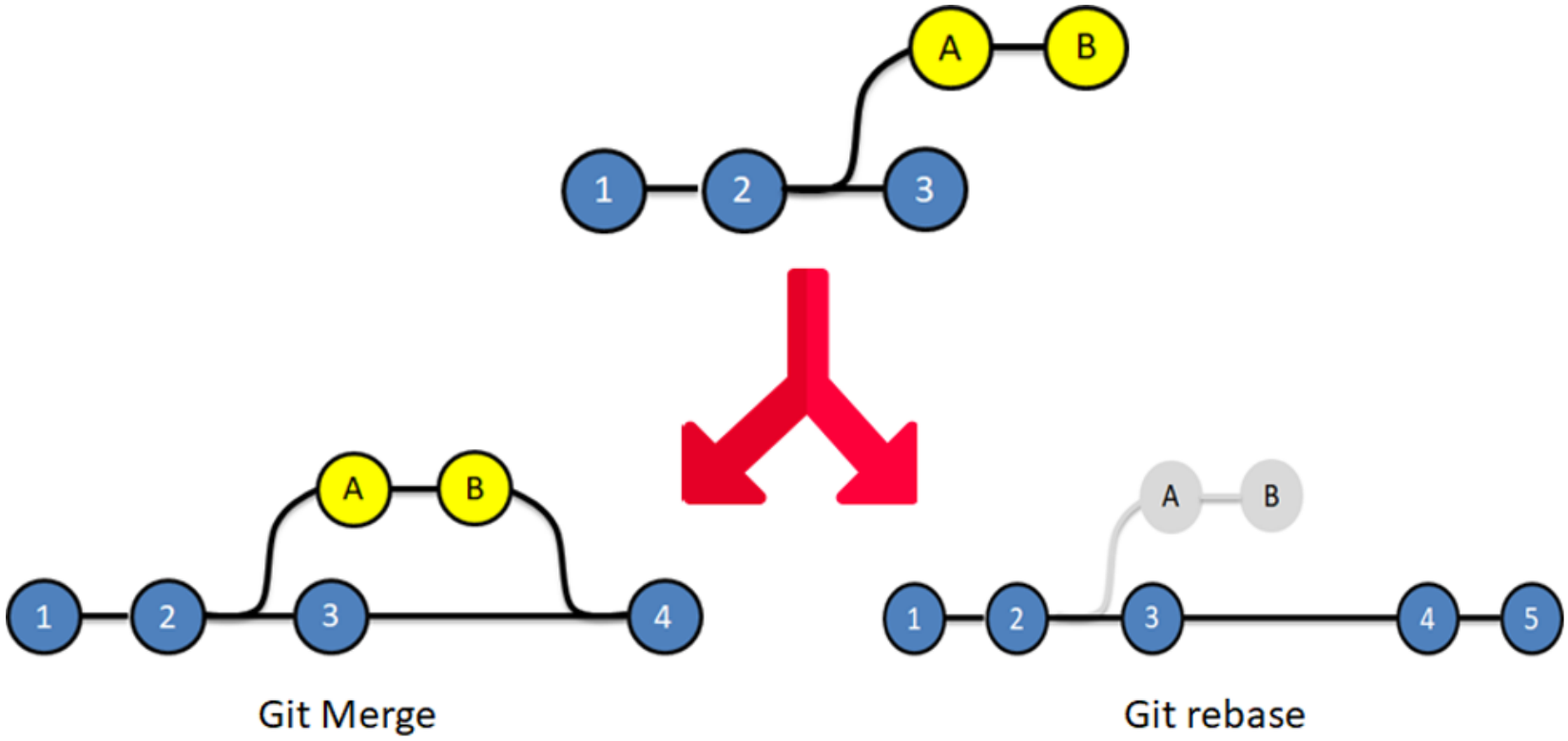
- **$ git rebase main**
  - topic 브랜치의 모든 커밋들(169a6와 2c33a)을 main 브랜치 이후에 추가



- **$ git merge topic**
  - Main으로 이동(checkout)해서 topic으로 fast-forward 병합이 필요

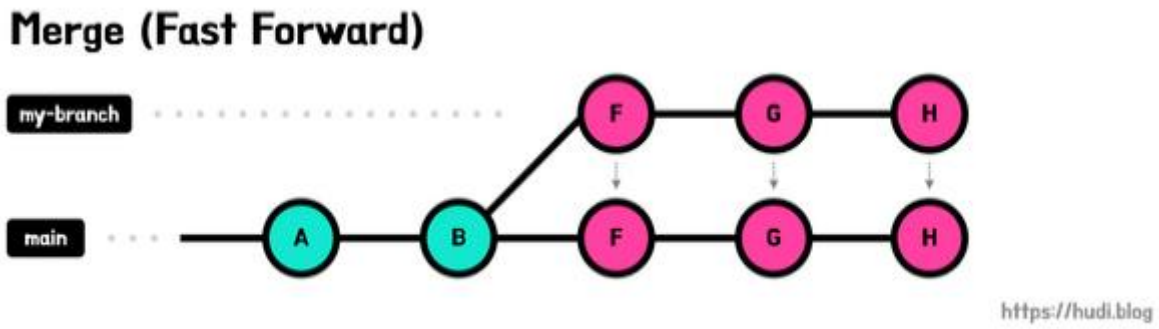
3-way merge와 rebase 비교

- **merge**
  - 변경 내용의 이력이 모두 그대로 남아 있기 때문에 이력이 복잡해짐
- **rebase**
  - 히스토리가 선형으로 단순해지고 좀 더 깨끗한 이력을 남김
  - 원래의 커밋 이력이 변경됨
    - 정확한 이력을 남겨야 할 필요가 있을 경우에는 사용하면 안됨

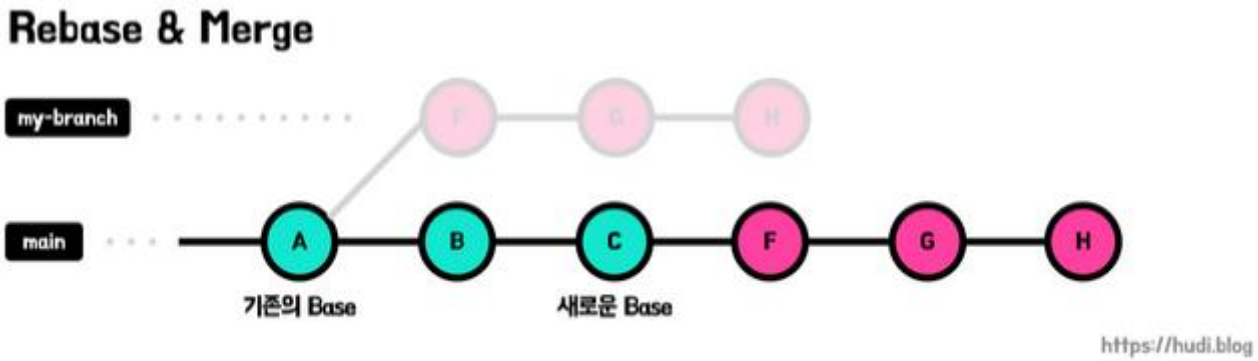


fast-forward merge와 rebase 비교

- **Fast-forward merge**
  - 조상에 위치한 브랜치에서 선행 브랜치로 이동하는 병합
- **rebase**
  - 두 갈래의 브랜치에서
    - 이전 베이스를 병합할 브랜치 마지막 커밋으로 베이스를 수정하는 병합



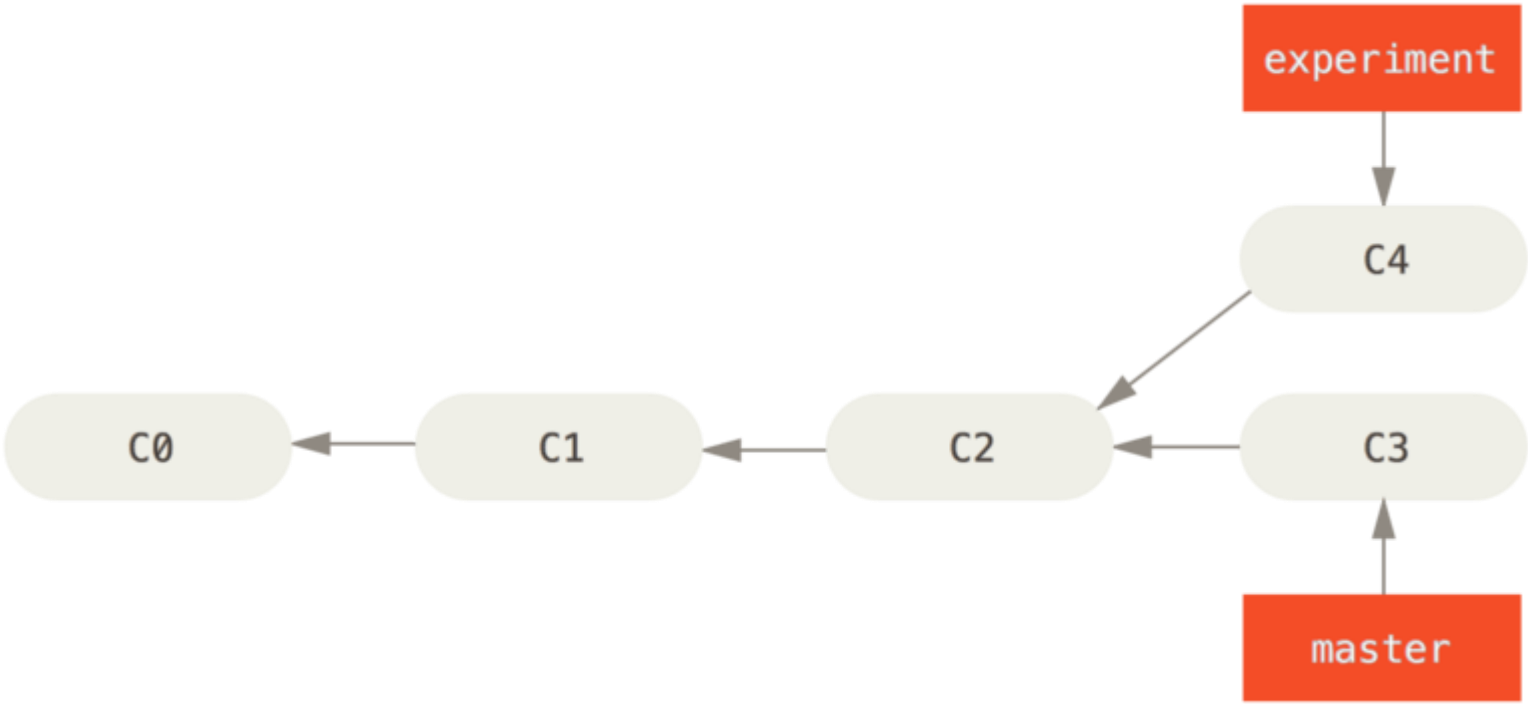
Fast-Forward Merge



Rebase & Merge

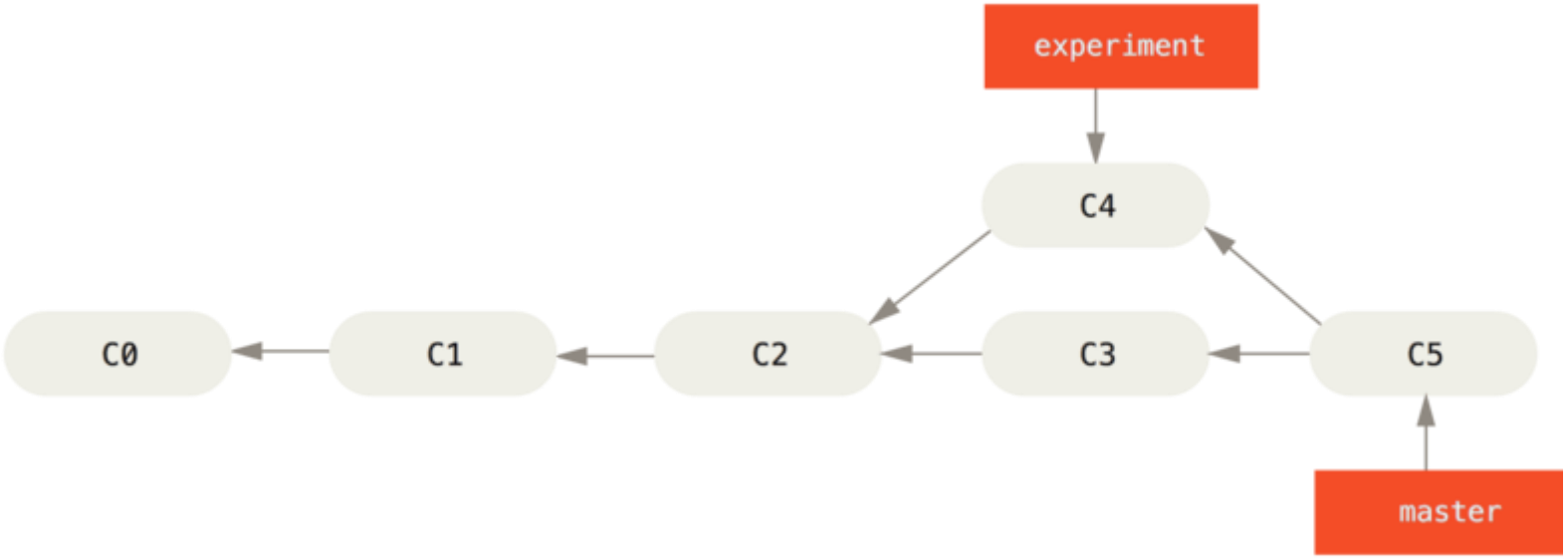
https://hudi.blog/git-merge-squash-rebase/

두 브랜치

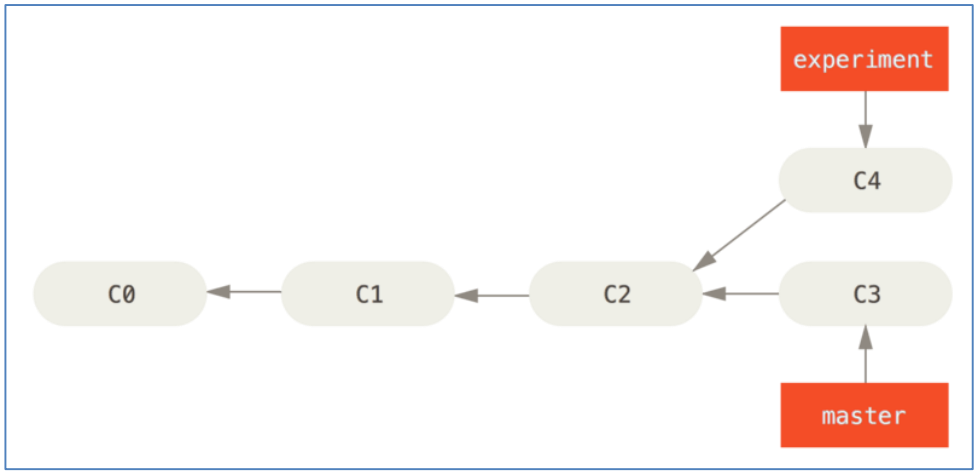


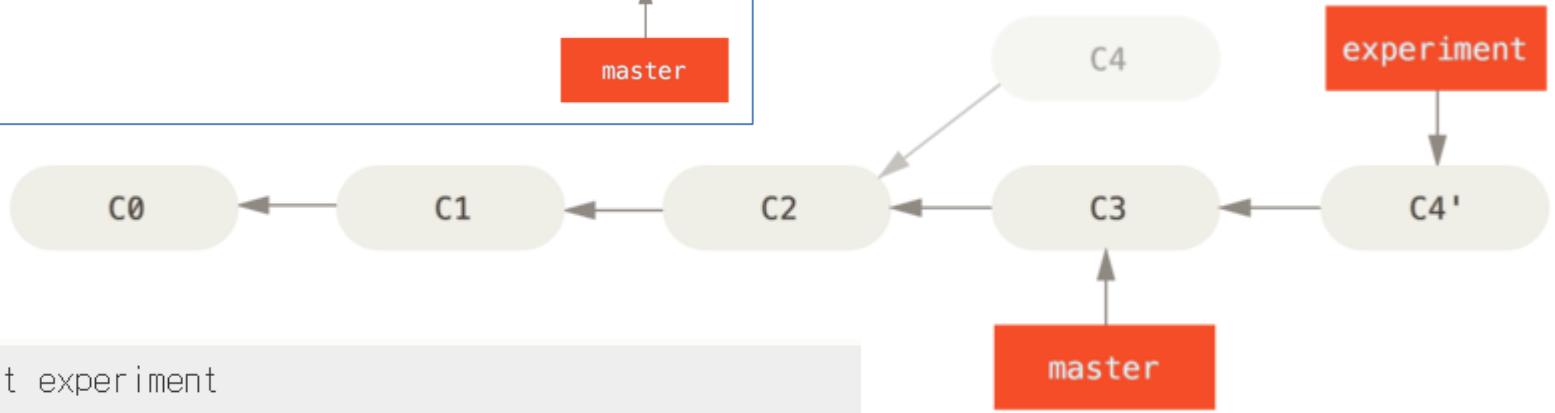
1. 3-way 병합

- $ git checkout mater
- $ git merge experiment

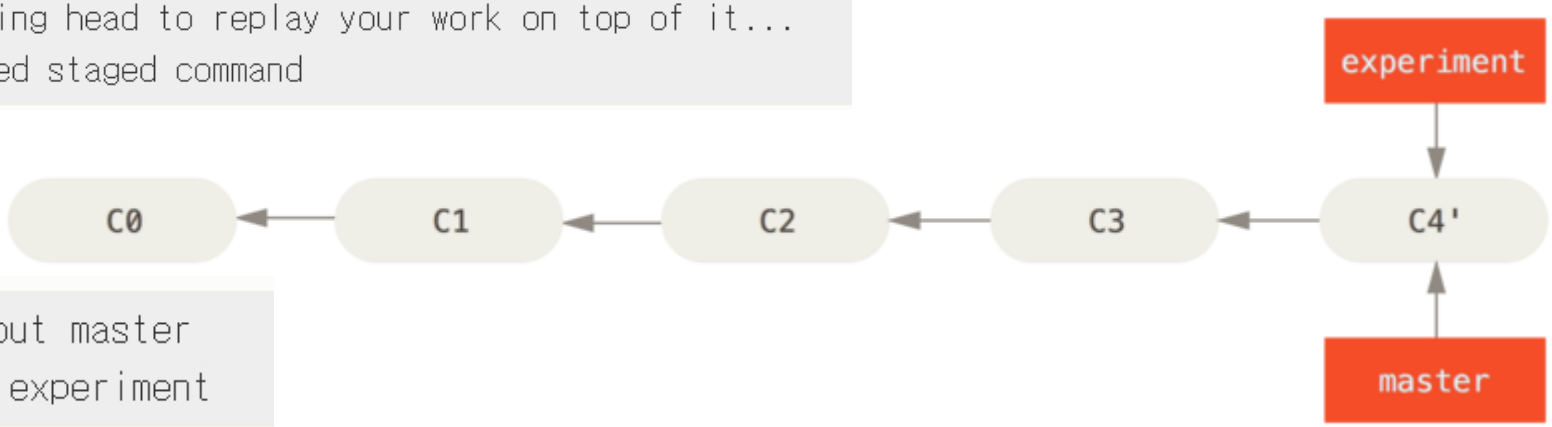


2. Rebase 병합






```
$ git checkout experiment
$ git rebase master
First, rewinding head to replay your work on top of it...
Applying: added staged command
```




```
$ git checkout master
$ git merge experiment
```

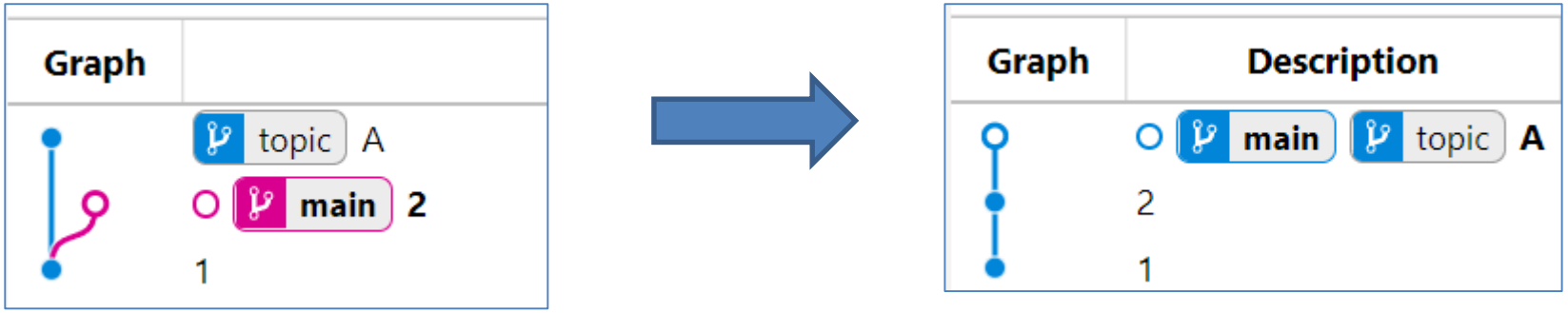
$ git rebase <newparent> <branch>

- **일반적 rebase 방법**
  - topic에서 main을 rebase 한 이후, 다시 main으로 이동 fast-forward 병합 수행
    - $ git checkout topic
    - $ git rebase main
    - $ git checkout main
    - $ git merge topic

- **다른 rebase 방법: 브랜치 어디에서든 main topic 순서로 재배치 방법**
  - $ git rebase main topic
  - $ git checkout main
  - $ git merge topic

$ git rebase <newparent> <branch>

<branch>에서 선행으로 <newparent> 브랜치의 최근 커밋 이후에 <branch>를 배치해 선형 병합



AI Experts
Who Lead
The Future

02

커밋이력 수정

https://meetup.toast.com/posts/116

커밋 메시지 수정

- 최근 커밋 메시지를 설정된 편집기로 수정하는 방법
  - $ git config --global core.editor ‘code --wait’
  - $ git commit --amend

- 최근 커밋 메시지를 직접 입력해 수정
  - $ git commit --amend -m "an updated commit message“
    - 새로운 커밋 ID로 수정됨

최근 커밋의 파일의 추가 등 수정

- 1회 커밋
  - # Edit hello.py and main.py
  - $ git add hello.py
  - $ git commit

- 이미 커밋한 이후에
- 파일 수정 등 후 파일 다시 추가
  - # Realize you forgot to add the changes from main.py
  - $ git add main.py # 잊은 파일을 더 추가 후 다시 커밋 수정

- 다시 커밋: 추가 커밋이 아닌 커밋 수정
  - $ git commit --amend --no-edit # 메시지 수정 없이 다시 커밋 수정
    - 새로운 커밋 ID로 수정됨

rebase의 --interactive를 사용

- 작업공간이 깨끗한 이후 이전 여러 개의 커밋을 수정
  - 이전 커밋을 다시 작성한 경우 새 ID가 부여

- rebase의 --interactive를 사용하여 커밋 시퀀스를 새로운 기본 커밋에 결합
  - HEAD~3: 수정할 커밋의 직전 커밋, 실제 HEAD~2부터 수정 가능
    - $ git rebase -i HEAD~3

- 주요 대화형 명령어
  - p(ick): 해당 커밋을 수정하지 않고 그냥 사용
  - r(eword): 개별 커밋 메시지를 다시 작성
  - s(quash): 계속된 이후 커밋을 이전 커밋에 결합(p-s 순서로)
  - d(rop): 커밋 자체를 삭제

학습목표와 실습

- **병합의 다른 방법인 rebase를 이해하고 수행할 수 있다.**
  - Base를 재배치하는 rebase 병합을 수행할 수 있다.
  - 3-way 병합과 rebase의 차이를 이해할 수 있다.
  - 커밋이력 수정을 위해 대화방식인 옵션 –i를 사용해 rebase -i로 수행할 수 있다.