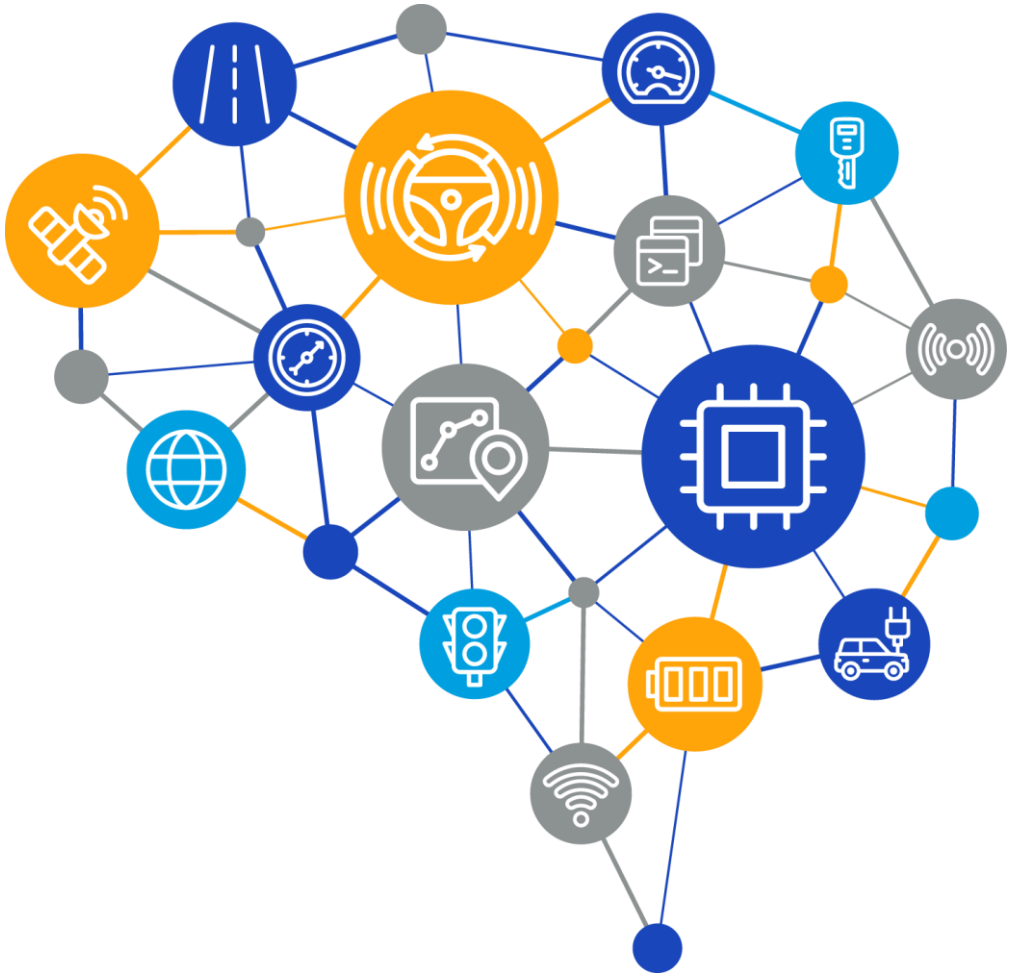
브랜치 병합과 충돌해결



Git & Github
for pull request

강환수 교수

DMUAi 동양미래대학교
인공지능소프트웨어학과

AI Experts Who Lead
The Future

학습목표

- **브랜치의 병합(merge)을 이해하고 수행할 수 있다.**
  - 빨리 감기(fast-forward) 병합을 수행할 수 있다.
  - 3-way 상태를 이해하고 3-way 병합(merge)을 수행할 수 있다.
  - 기본이 fast-forward 병합이 수행되는 일렬 상태에서 3-way 병합(merge)을 수행할 수 있다.

AI Experts
Who Lead
The Future

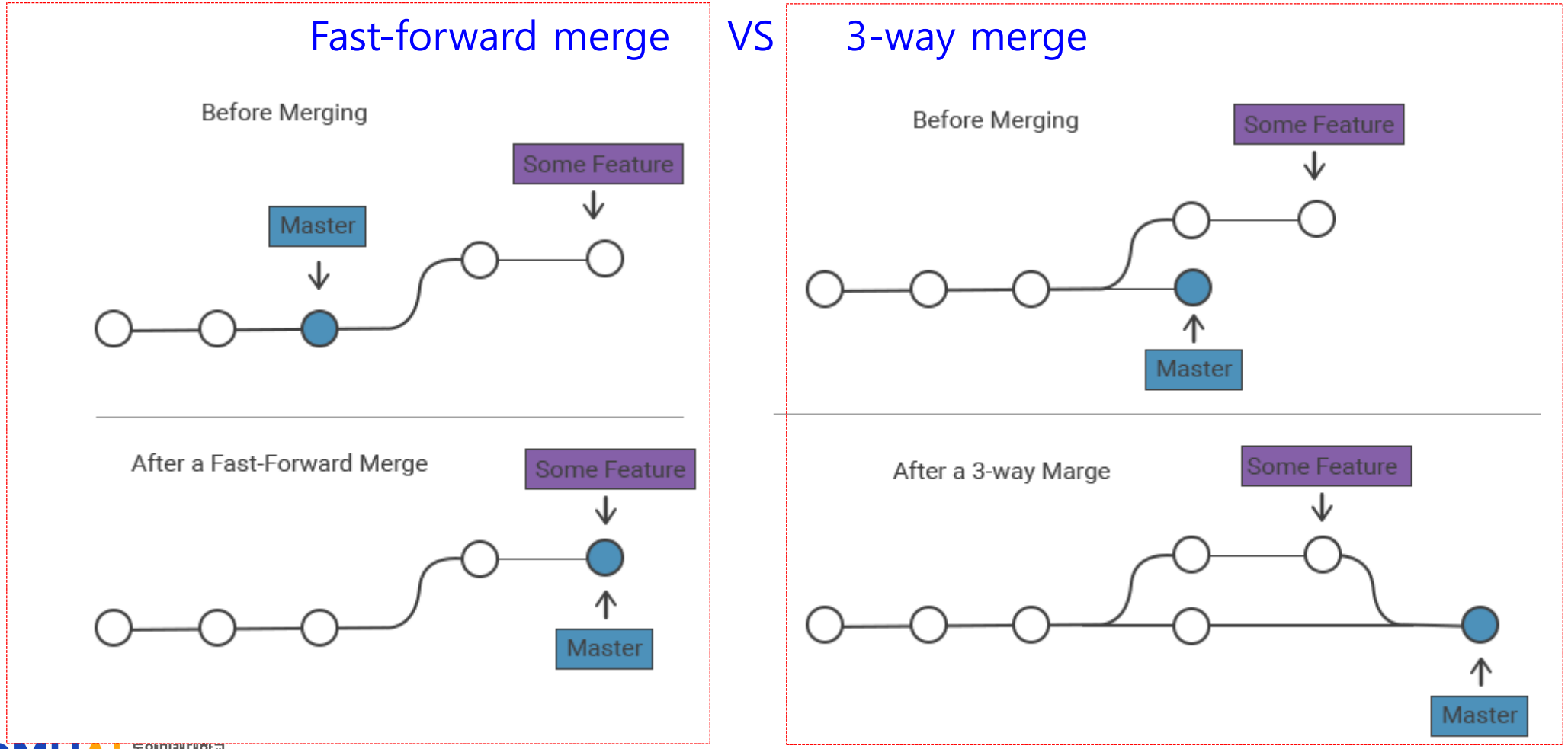
01

브랜치 merge

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

브랜치 통합하기: merge

- **merge**
  - 여러 개의 브랜치를 하나로 모음
    - fast-forward(빨리 감기) 병합
    - 3-awy 병합

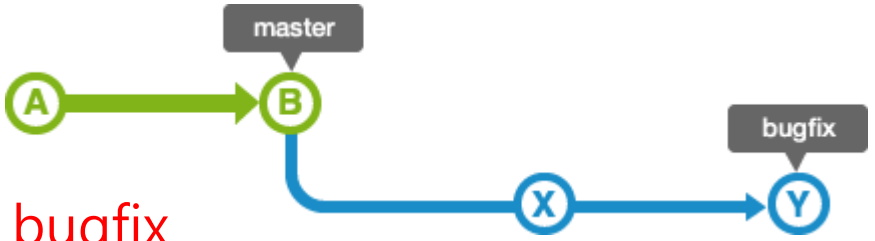


fast-forward(빨리 감기) 병합

- Merge할 대상이 현재 커밋의 직접적인 뿌리가 되는 경우, fast-forward Merge가 실행
  - 'bugfix' 브랜치의 이력이 'master' 브랜치의 이력을 모두 포함하는 경우
  - 'master' 브랜치는 단순히 이동
    - 이 때는 합칠 내용이 없음
      - 간단히 가리키는 지점이 대상 커밋이 되고 대상 커밋의 내용을 Checkout 함
    - 작업공간과 스테이징 영역이 이동되는 Y 상태로 됨

- 기준 브랜치 master
  - $ git merge bugfix

  - 병합 전

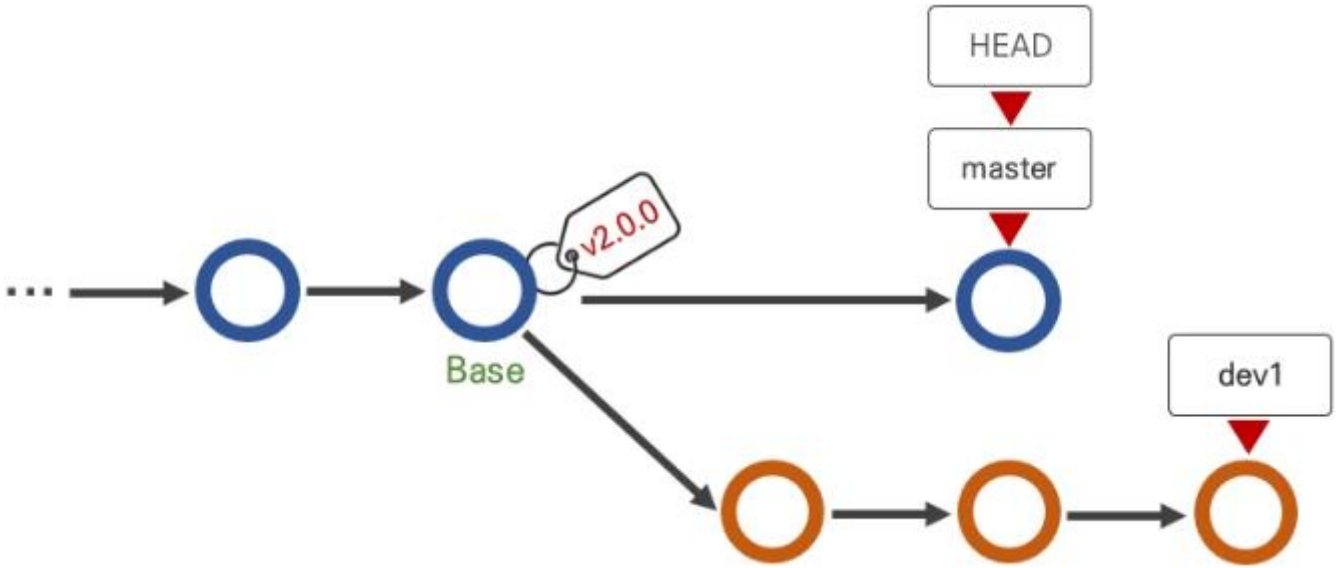


$ git merge bugfix

  - 병합 후

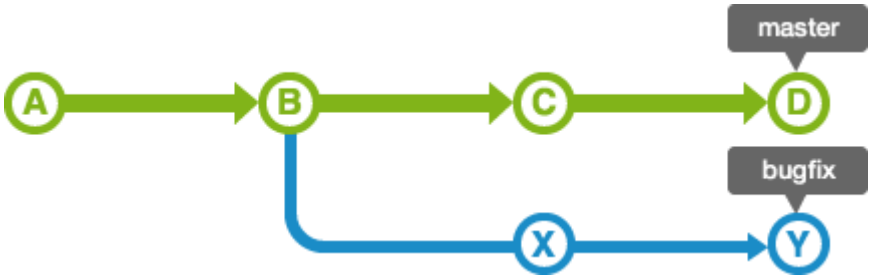
두 분기가 갈라진 상태: 3-way 상태

- **두 브랜치의 조상이 같은 경우**
  - 3-way 병합은 새로운 커밋을 사용하여 두 기록을 합침



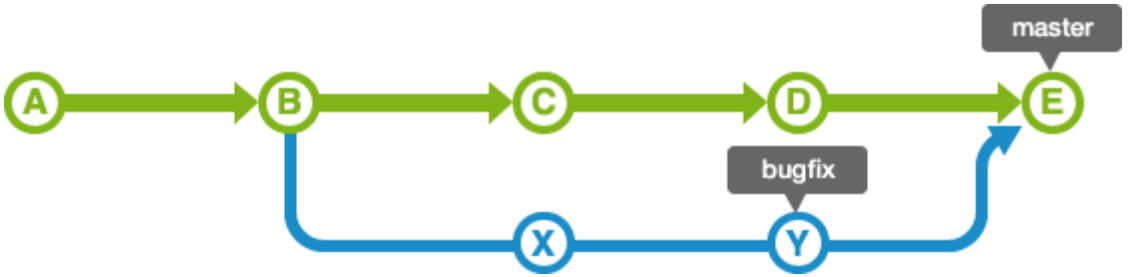
3-way merge

- **브랜치 분기 후 'master'에 변경 사항이 생긴 경우**
  - 'bugfix' 브랜치를 분기한 이후에 'master' 브랜치에 여러 가지 변경 사항이 적용
  - 'master' 브랜치 내의 변경내용과 'bugfix' 브랜치 내의 변경내용을 하나로 통합할 필요



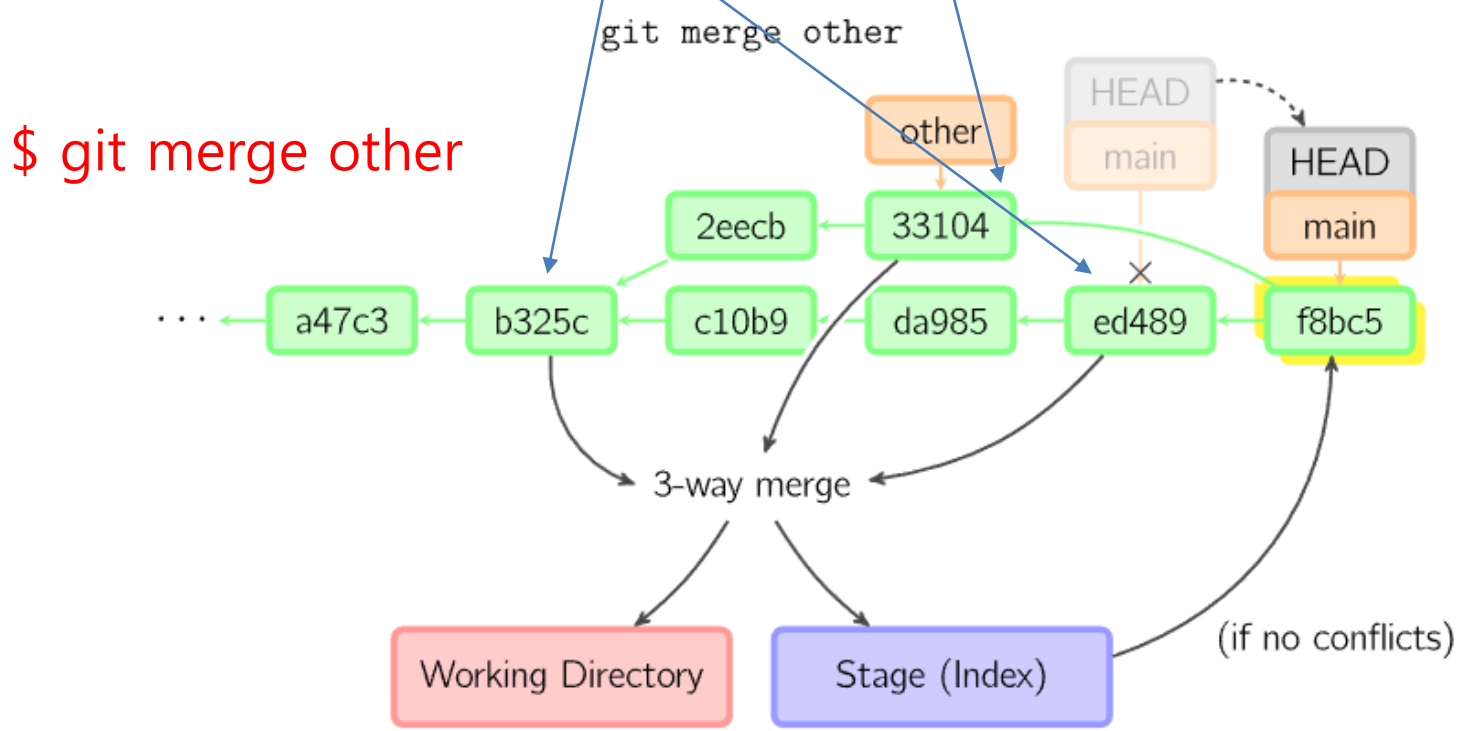
- **새로운 커밋이 생성**
  - 양쪽의 변경을 가져온 'merge commit(병합 커밋)'을 실행
  - 병합 완료 후, 통합 브랜치인 'master' 브랜치로 통합된 이력이 아래처럼 생성

```
$ git merge bugfix
```



3-way merge

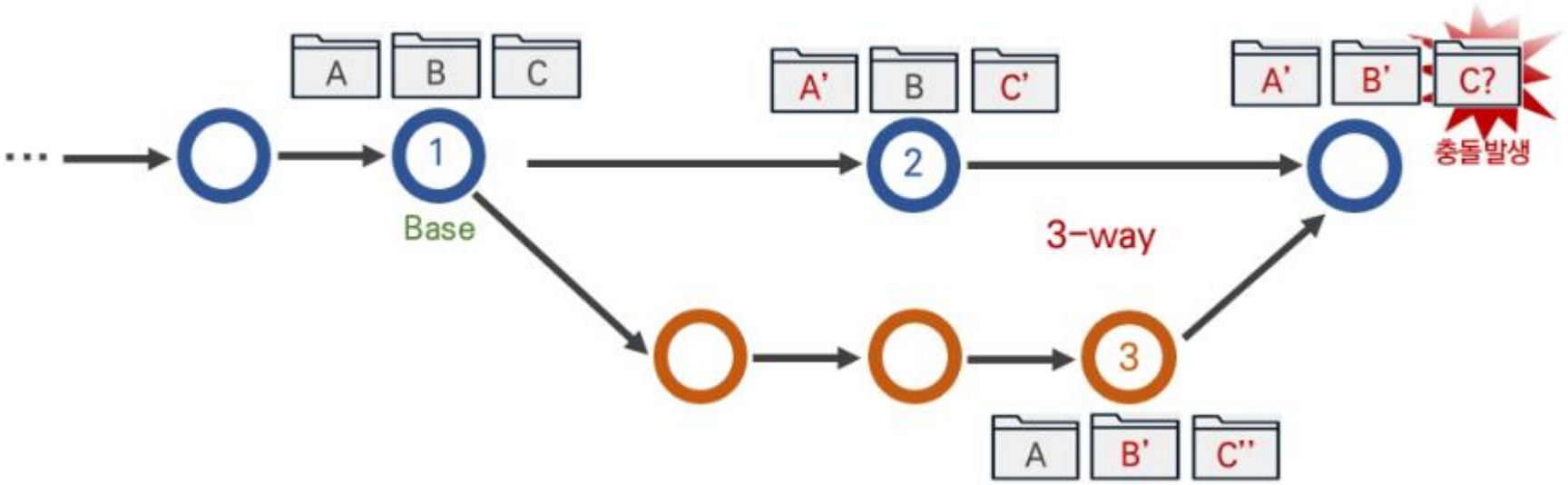
- **$ git merge other**
  - 3-way Merge를 수행
    - 현재 커밋(ed489), 대상이 되는 커밋(33104)
    - 공통의 뿌리가 되는 커밋(b325c)
  - Merge한 결과는 작업 디렉토리와 Stage 영역에 저장
    - 부모가 여럿(33104, ed489)인 새 커밋을 만듦



충돌과 해결

- 만약에 두 commit 모두에서 변경사항이 발생한 파일에 대해서는 충돌(conflict)이 발생
  - 충돌이 발생하면 해당 파일의 충돌을 해결 한 후 add, commit 진행

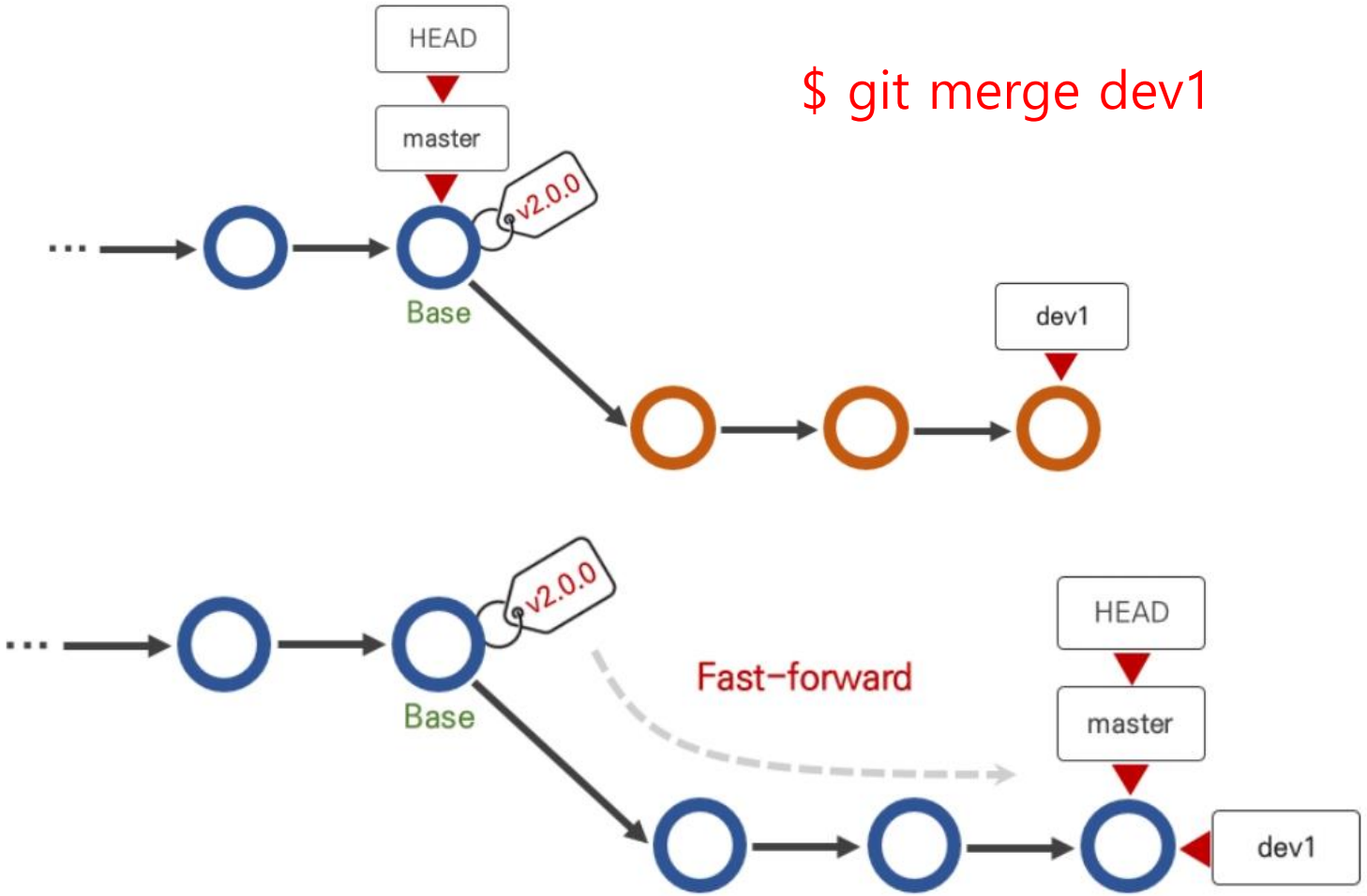
    - 파일 A, B는 문제 없으나 C는 충돌



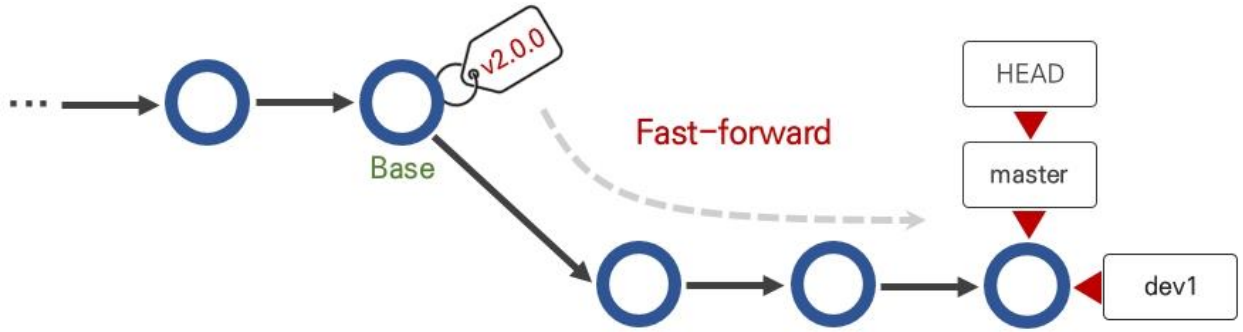
- 충돌 해결
  - 파일 내용을 수정 후 저장
  - Add 한 후 커밋
  - 필요하면 합병된 이전 브랜치 삭제

일렬 상태에서 기본이 Fast forward 병합

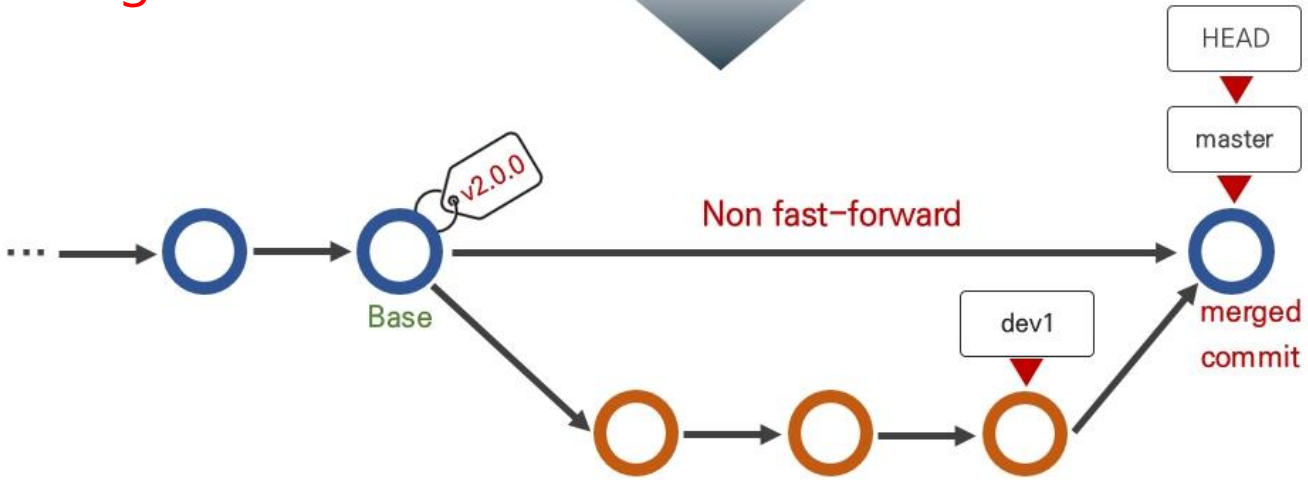
- 병합할 브랜치의 조상이 기준 브랜치인 경우(일렬 상태) 병합은 기본은 Fast forward 병합
  - Master에서 dev1을 merge
    - 기본: fast-forward 병합



- Master에서 dev1을 non Fast forward로 merge(3-way merge)
  - 옵션 --no-ff
    - $ git merge dev1 --no-ff

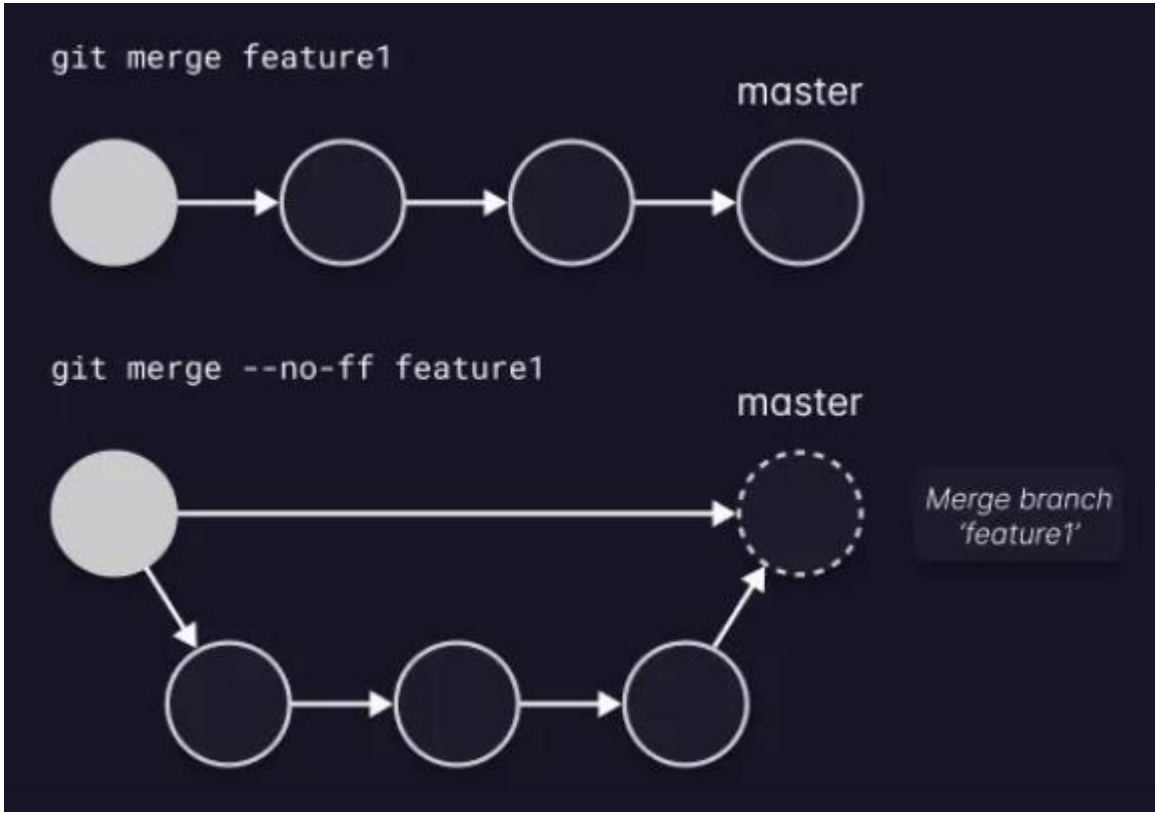


$ git merge dev1 --no-ff



두 브랜치의 일렬 상태에서

- **두 가지 병합 방법**
  - 옵션 없는 경우
    - 빠른 감기(fast-forward) 병합

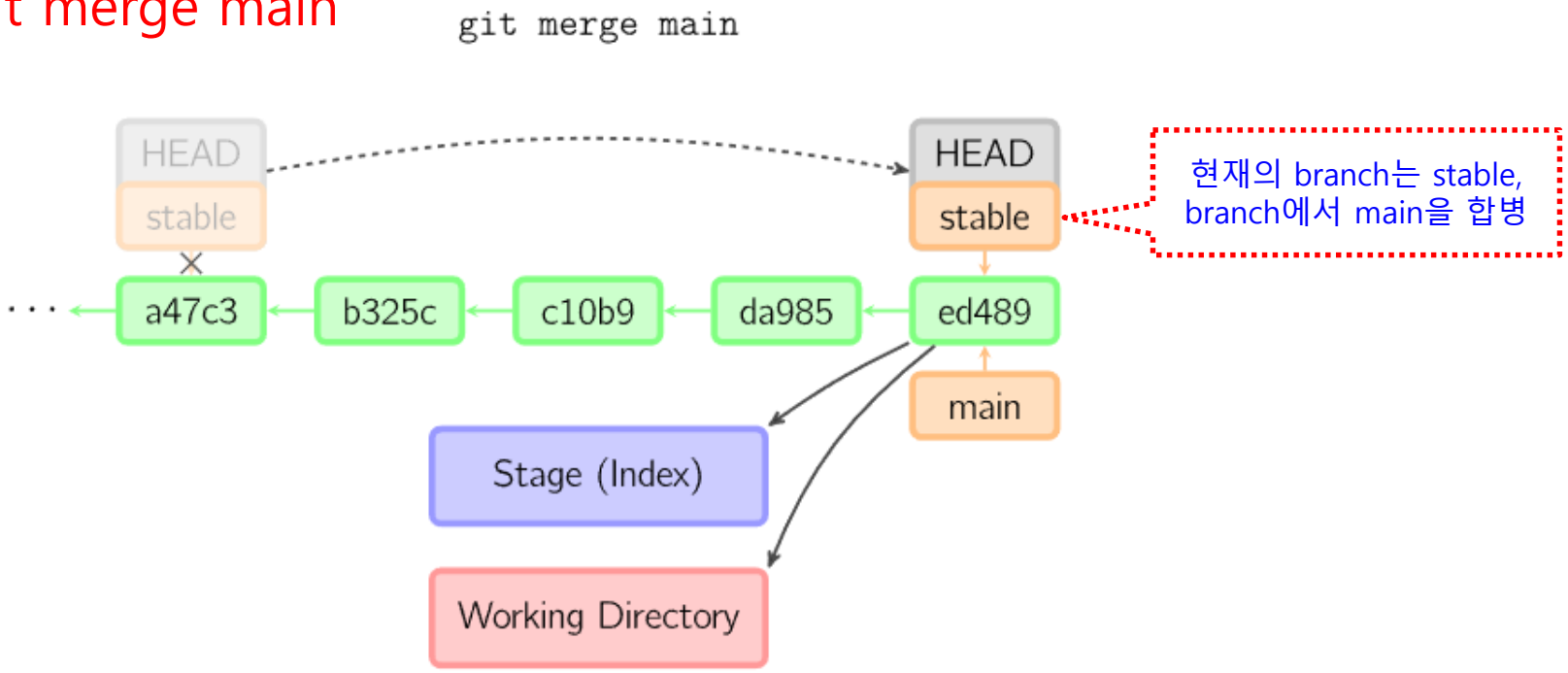


  - 옵션 --no-ff
    - 3-way merge

병합 시 주의점

- **Merge 명령을 실행하기 전에**
  - 작업과 Stage 영역에 작업중인 파일이 없는지 꼭 확인

$ git merge main



실습 후 우리는 ~~~할 수 있다.

- 빨리 감기(fast-forward) 병합을 수행 (merge)할 수 있다.

- 3-way 상태를 이해하고 3-way 병합(merge)을 수행할 수 있다.
  - 충돌이 발생 시 이를 해결하고 커밋, 병합할 수 있다.

- 기본이 fast-forward 병합이 수행되는 일렬 상태에서
  - 3-way 병합(merge)을 수행할 수 있다.

기준 브랜치로 이동해서 머지

- $ git checkout main
- $ git merge feat/list

```
PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (feat/list)
$ git lo
56361e0 (HEAD -> feat/list) Add print list of range, feat/list
5be70c1 Add print list, feat/list
834e72e (main) Add print literals, main
fab1006 Create basic.py, main

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (feat/list)
$ git checkout main
Switched to branch 'main'

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main)
$ git lo
834e72e (HEAD -> main) Add print literals, main
fab1006 Create basic.py, main

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main)
$ git merge feat/list
Updating 834e72e..56361e0
Fast-forward
 basic.py | 2 ++
 1 file changed, 2 insertions(+)
```

머지 후 확인

```
$ git lo
56361e0 (HEAD -> feat/list) Add print list of range, feat/list
5be70c1 Add print list, feat/list
834e72e (main) Add print literals, main
fab1006 Create basic.py, main
```

```
PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main)
$ git lo
56361e0 (HEAD -> main, feat/list) Add print list of range, feat
/list
5be70c1 Add print list, feat/list
834e72e Add print literals, main
fab1006 Create basic.py, main
```

3-way 병합 준비 1/2

- 먼저 main에서 한 번 커밋(# 5번째 커밋)

- $ git branch
  - 확인
  -
- 파일 basic.py 수정 후 커밋
  - print('branch basic')
  - print(1, 2, 3)
  - print([1, 2, 3])
  - print(list(range(1, 11))) # 4번째 커밋
  - a, b = divmod(20, 3) # 5번째 커밋

- $ git commit –am ‘add divmod, main’

```
PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main)
$ git st
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   basic.py

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

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main)
$ git commit -am 'add divmod, main'
[main 2ecae57] add divmod, main
 1 file changed, 3 insertions(+), 1 deletion(-)

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main)
$ git lo
2ecae57 (HEAD -> main) add divmod, main
56361e0 (feat/list) Add print list of range, feat/list
5be70c1 Add print list, feat/list
834e72e Add print literals, main
fab1006 Create basic.py, main
```

3-way 병합 준비 2/2

- 먼저 feat/list에서 한 번 커밋(# 5번째 커밋)
-
- $ git switch feat/list
- 브랜치와 파일 확인
- 파일 basic.py 수정 후 커밋
  - print('branch basic')
  - print(1, 2, 3)
  - print([1, 2, 3])
  - print(list(range(1, 11))) # 4번째 커밋
  - print(tuple(range(1, 11, 2))) # 5번째 커밋
- $ git commit –am ‘add print of tuple, feat/list’

브랜치 이동 후 파일 수정

```
PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main)
$ git lo
2ecae57 (HEAD -> main) add divmod, main
56361e0 (feat/list) Add print list of range, feat/list
5be70c1 Add print list, feat/list
834e72e Add print literals, main
fab1006 Create basic.py, main

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main)
$ git switch feat/list
Switched to branch 'feat/list'

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (feat/list)
$ git branch
* feat/list
  main

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (feat/list)
$ cat basic.py
print('branch basic')
print(1, 2, 3)
print([1, 2, 3])
print(list(range(1, 11)))     # 4번째 커밋

print(tuple(range(1, 11, 2)))   # 5번째 커밋
```

커밋

```
PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main)
$ git switch feat/list
Switched to branch 'feat/list'

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (feat/list)
$ git branch
* feat/list
  main

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (feat/list)
$ cat basic.py
print('branch basic')
print(1, 2, 3)
print([1, 2, 3])
print(list(range(1, 11)))     # 4번째 커밋

print(tuple(range(1, 11, 2)))   # 5번째 커밋

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (feat/list)
$ git commit -am 'add print of tuple, feat/list'
[feat/list 3c4835f] add print of tuple, feat/list
 1 file changed, 2 insertions(+)

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (feat/list)
$ git lo
3c4835f (HEAD -> feat/list) add print of tuple, feat/list
56361e0 Add print list of range, feat/list
5be70c1 Add print list, feat/list
834e72e Add print literals, main
fab1006 Create basic.py, main
```

각각의 브랜치에서 커밋 확인

```
PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (feat/list)
$ git branch
* feat/list
  main

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (feat/list)
$ git lo
3c4835f (HEAD -> feat/list) add print of tuple, feat/list
56361e0 Add print list of range, feat/list
5be70c1 Add print list, feat/list
834e72e Add print literals, main
fab1006 Create basic.py, main

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (feat/list)
$ git switch main
Switched to branch 'main'

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main)
$ git lo
2ecae57 (HEAD -> main) add divmod, main
56361e0 Add print list of range, feat/list
5be70c1 Add print list, feat/list
834e72e Add print literals, main
fab1006 Create basic.py, main
```

3-way 머지 실습

- 브랜치 main으로 이동해서
- $ git switch main

- $ git merge feat/list

```
1  print('branch basic')
2  print(1, 2, 3)
3  print([1, 2, 3])
   Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
4  <<<<<<< HEAD (Current Change)
5  print(list(range(1, 11)))   # 4번째 커밋
6
7  a, b = divmod(20, 3)        # 5번째 커밋
8  =======
9  print(list(range(1, 11)))     # 4번째 커밋
10
11 print(tuple(range(1, 11, 2)))  # 5번째 커밋
12 >>>>>>> feat/list (Incoming Change)
13
```

병합에서 충돌 발생

```
PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main)
$ git merge feat/list
Auto-merging basic.py
CONFLICT (content): Merge conflict in basic.py
Automatic merge failed; fix conflicts and then commit the result.

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main|MERGING)
$ git lo
2ecae57 (HEAD -> main) add divmod, main
56361e0 Add print list of range, feat/list
5be70c1 Add print list, feat/list
834e72e Add print literals, main
fab1006 Create basic.py, main

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main|MERGING)
$ git st
On branch main
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

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

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

충돌 발생 의미

충돌 발생 이해

- 동일 소스의 같은 부분을 수정

```
print('branch basic')
print(1, 2, 3)
print([1, 2, 3])
<<<<<<< HEAD
print(list(range(1, 11)))    # 4번째 커밋

a, b = divmod(20, 3)         # 5번째 커밋
=======
print(list(range(1, 11)))      # 4번째 커밋

print(tuple(range(1, 11, 2))) # 5번째 커밋
>>>>>>> feat/list
```

병합 취소 후 다시 병합

- $ git merge --abort
- $ git merge feat/list

```
PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main|MERGING)
$ git merge --abort

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main)
$ git merge feat/list
Auto-merging basic.py
CONFLICT (content): Merge conflict in basic.py
Automatic merge failed; fix conflicts and then commit the result.
```

충돌 해결

- **직접 수정 후 저장**
  - 충돌 표시 모두 제거

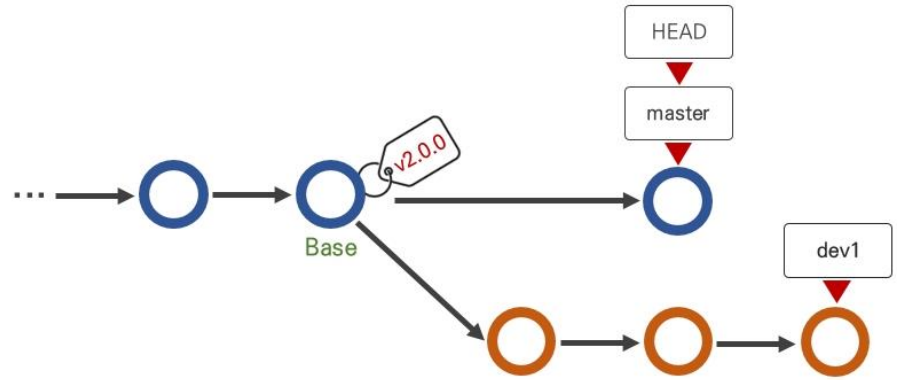
```
print('branch basic')
print(1, 2, 3)
print([1, 2, 3])
print(list(range(1, 11)))       # 4번째 커밋

a, b = divmod(20, 3)            # 5번째 커밋
print(a, b)                     # 5번째 커밋
print(tuple(range(1, 11, 2)))   # 충돌 해결
```

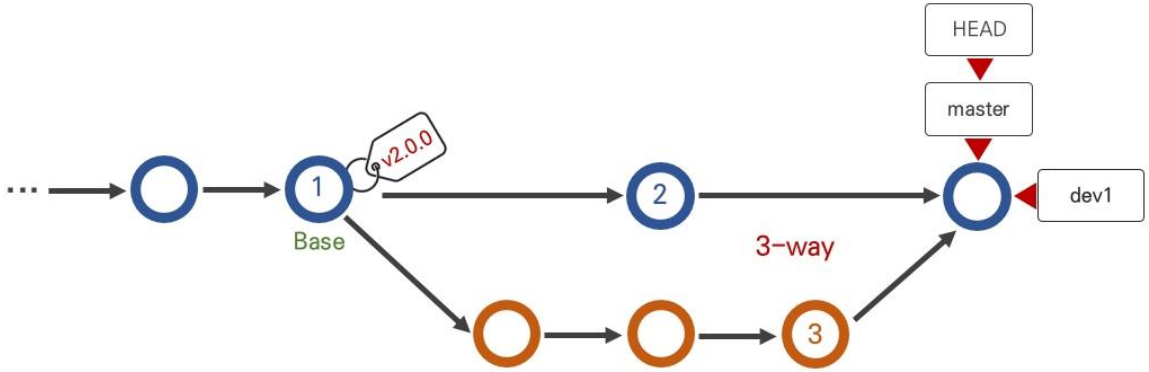
- **커밋 다시**
- **S git commit –am 'Resolve conflict, main'**

```
PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main|MERGING)
$ git st
On branch main
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

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

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

PC@DESKTOP-482NOAB MINGW64 /c/[git tutorial]/branch-lab (main|MERGING)
$ git commit -am 'Resolve conflict, main'
[main c1d4ca6] Resolve conflict, main
```

병합 후 커밋 그래프

```
$ git logg
*   commit c1d4ca63ec9b6d23320b2704251e608c3ff03881 (HEAD -> main)
|\  Merge: 2ecae57 3c4835f
| | Author: edu4py <edu4py@outlook.kr>
| | Date:   Sun May 29 11:55:00 2022 +0900
| |
| |     Resolve conflict, main
| |
| * commit 3c4835fba9b97405253449ac4ad7b6f311d9671a (feat/list)
| | Author: edu4py <edu4py@outlook.kr>
| | Date:   Sun May 29 11:31:23 2022 +0900
| |
| |     add print of tuple, feat/list
| |
* | commit 2ecae572f644eed3d67204b3ce902e36cabc29c8
|/  Author: edu4py <edu4py@outlook.kr>
|   Date:   Sun May 29 11:19:47 2022 +0900
|
|       add divmod, main
|
* commit 56361e06d63dfe1f850aacd913b041cb5e2f5ae4
| Author: edu4py <edu4py@outlook.kr>
| Date:   Sun May 29 10:26:38 2022 +0900
|
|     Add print list of range, feat/list
|
* commit 5be70c13d396a4ea3ccee0ca516dfff26829096c
| Author: edu4py <edu4py@outlook.kr>
| Date:   Sun May 29 10:15:50 2022 +0900
|
|     Add print list, feat/list
|
* commit 834e72e6ba2c21f921f345e4b2a6e15c93792171
| Author: edu4py <edu4py@outlook.kr>
| Date:   Sun May 29 09:53:34 2022 +0900
|
|     Add print literals, main
|
* commit fab10065687594c7e835b44c38d1a40694778545
  Author: edu4py <edu4py@outlook.kr>
  Date:   Sun May 29 09:50:24 2022 +0900

      Create basic.py, main
```





AI Experts
Who Lead
The Future

02

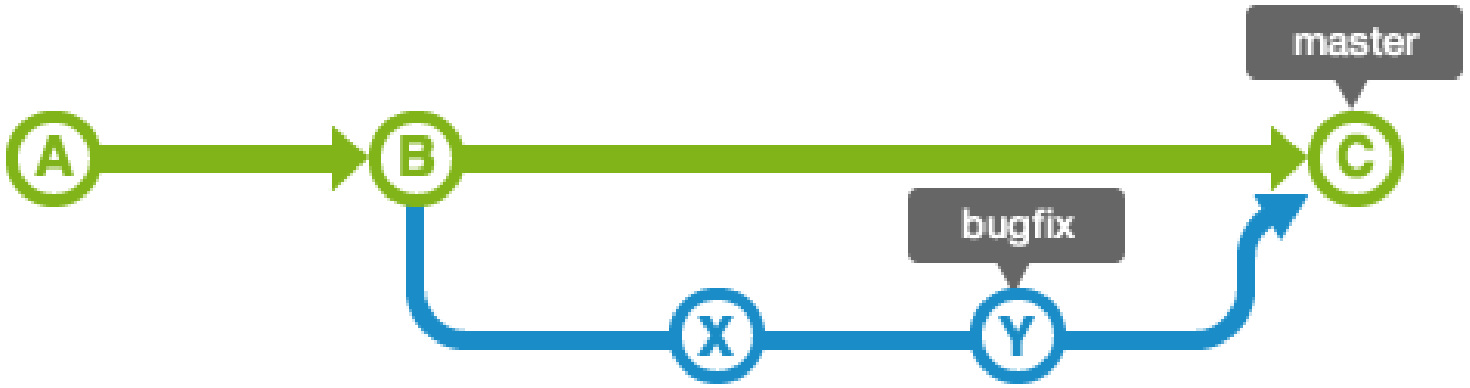
Merge의 다양한 옵션

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

'non fast-forward 병합' 옵션을 지정

- **non fast-forward 병합**
  - 병합 실행 시에 'fast-forward 병합'이 가능한 경우라도
  - 브랜치가 그대로 남기 때문에
    - 그 브랜치로 실행한 작업 확인 및 브랜치 관리 면에서 더 유용

- **$ git merge --no-ff {병합할 브랜치 명}**
  - 1. 보통의 병합 (--ff)
  - 2. 이기적 병합 (--no-ff)
  - 3. 소심한 병합 (--ff-only)
  - 4. 강압적 병합 (--squash)



병합(merge)의 다양한 옵션 종류

- **$ git merge {병합할 브랜치 명}**
  - 보통의 병합, 융통성 있는 병합
  - 현 브랜치와 병합할 브랜치가 Fast-Forward 관계 O
    - 병할할 브랜치(커밋)을 따라감, without Merge 커밋
  - 현 브랜치와 병합할 브랜치가 Fast-Forward 관계 X
    - 병합할 브랜치와 병합됨. with Merge 커밋

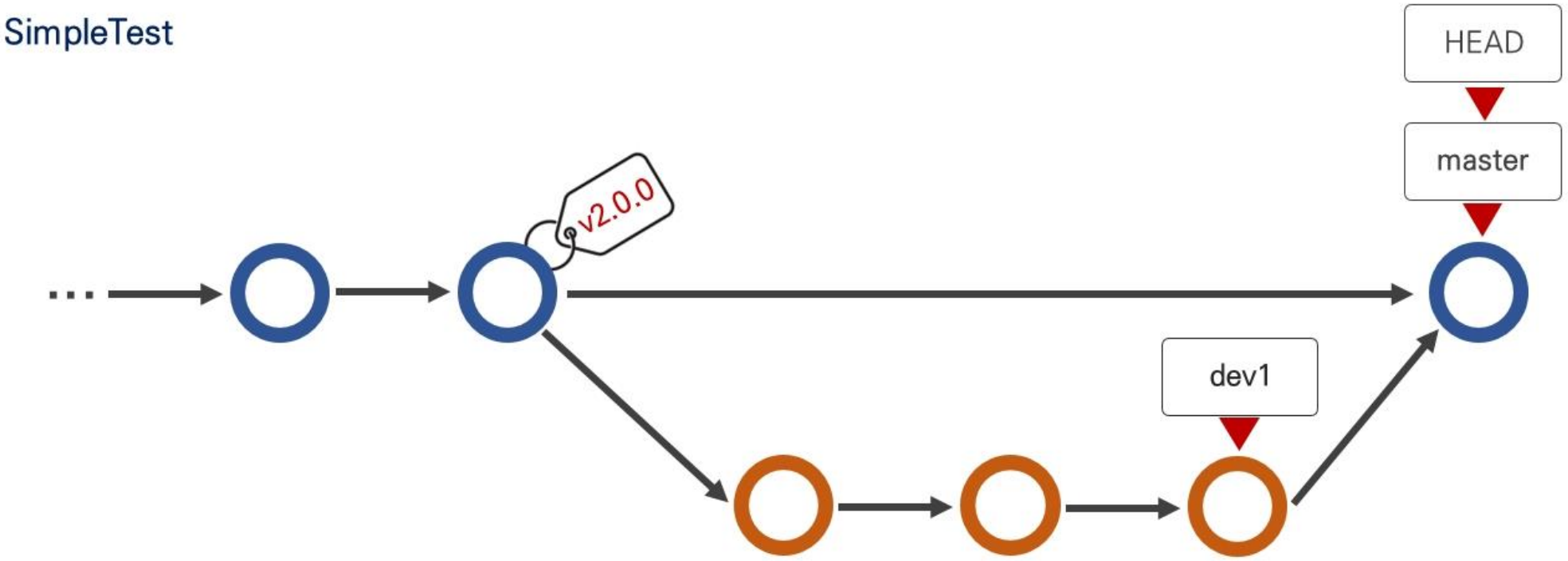
- **$ git merge --no-ff {병합할 브랜치 명}**
  - 무조건 Merge 커밋과 같이 병합되는 옵션

- **$ git merge --ff-only {병합할 브랜치 명}**
  - 현재 브랜치와 병합 대상의 관계가 Fast-Forward인 경우에만 병합 진행
    - Merge 커밋 생성되지 않음

- **$ git merge --squash {병합할 브랜치 명}**
  - 현재 브랜치에 병합 대상과의 차이나는 commit을 하나로 합쳐서 커밋함

--no-ff 옵션 merge

- 브랜치 관계에 상관없이 필요한 commit만 가져올 수 있다.
- 어떤 브랜치에서 merge를 했는지 기록을 남길 수 있다. (Merge branch 'dev1')
  - 무조건 3-way merge



-squash 옵션

- **git merge --squash**

- **강압적인(?) 병합**
  - 사전적 의미(짓 뭉개다)에서 알 수 있듯이
    - commit이력과 merge된 브랜치 이력도 남기지 않음
    - 새로운 commit에 상대 브랜치의 내용을 모두 뭉쳐 놓은 것으로 보임