
git

$git init

- git init 명령어를 실행하면 ".git" 숨김 폴더가 생성된다.
- .git 폴더가 있는 폴더내(프로젝트폴더)의 정보가 .git폴더에 저장된다.

$git status

- git status 명령어는 git이 관리하는 프로젝트 폴더내의 파일들의 변경상태를 알 수 있다.
- 항상 status명령어를 통해 폴더내의 변화를 항상 확인하는 것이 매우 중요하다.

$git add fileName

- 새로 생성한 파일을 추가하거나 수정된 파일에 대한 tracked 상태를 staged 상태로 만들어준다.
- $git add . ($git add *)은 하위 폴더 내 변경된 모든 파일들이 staged 된다.

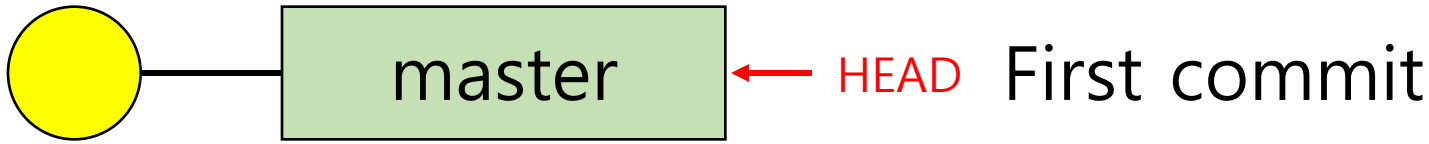
$git commit –m “message”

- Staged 상태에 있는 모든 파일을 git history에 저장한다.
- Message에는 어떤 내용을 수정하였는지에 대해 알 수 있는 내용으로 작성한다.
- Git은 head pointer가 현재의 commit을 가리키고 작업을 어떤 커밋에서 하는지 잘 확인해야 한다.
- Commit은 branch, ID, tag가 참조한다.
- ID는 커밋 고유의 정보를 저장하고 있기에 변동되지 않고 움직이지 않는다.
- Branch는 새로운 브랜치를 생성하여 작업을 할 때 커밋을 참조하고 필요없을 때에는 삭제가 가능하다.
- Tag는 필요 시 붙여줄 수 있다.

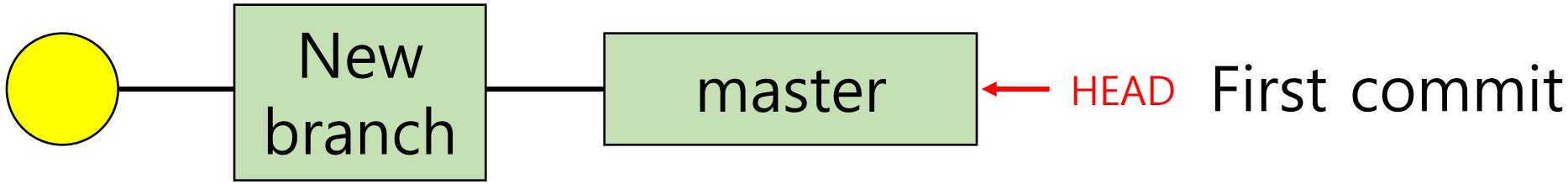
$gitk --all

- Commit history를 GUI환경으로 쉽고 빠르게 확인을 할 수 있다.

최초 커밋

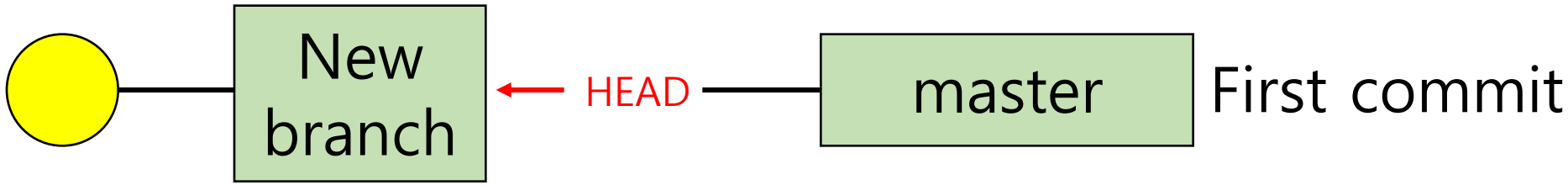


$git branch newBranch

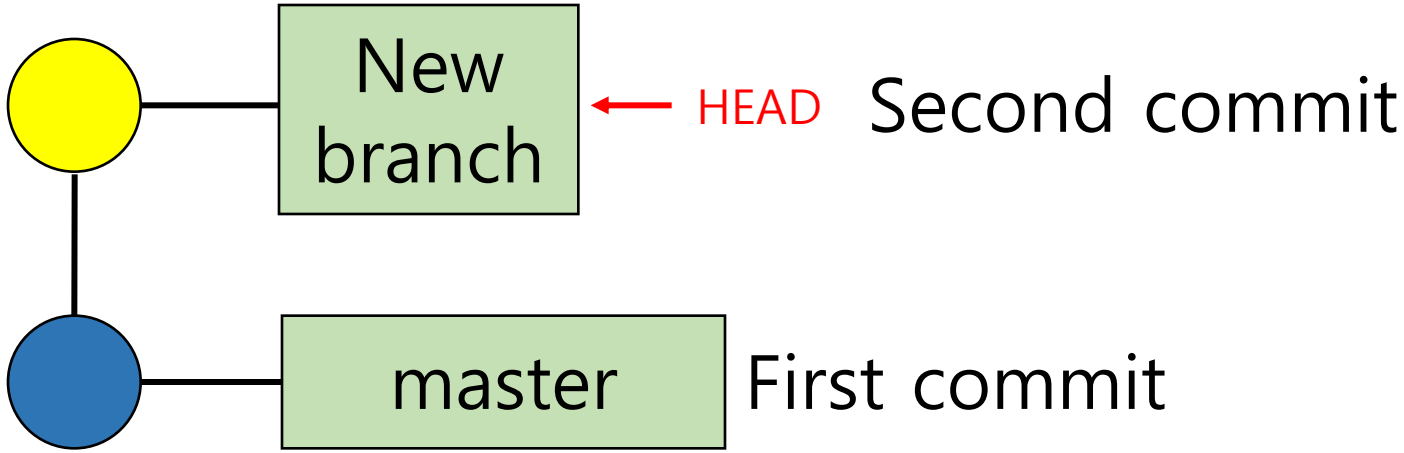


$git checkout newBranch

- HEAD pointe를 master에서 newBranch로 옮긴다.

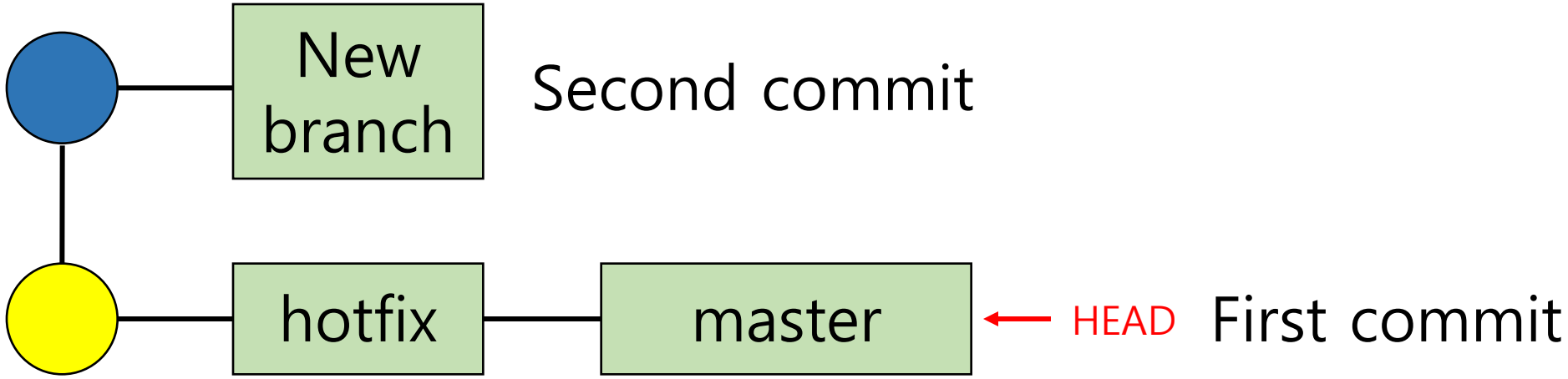


파일 수정 후 두 번째 커밋

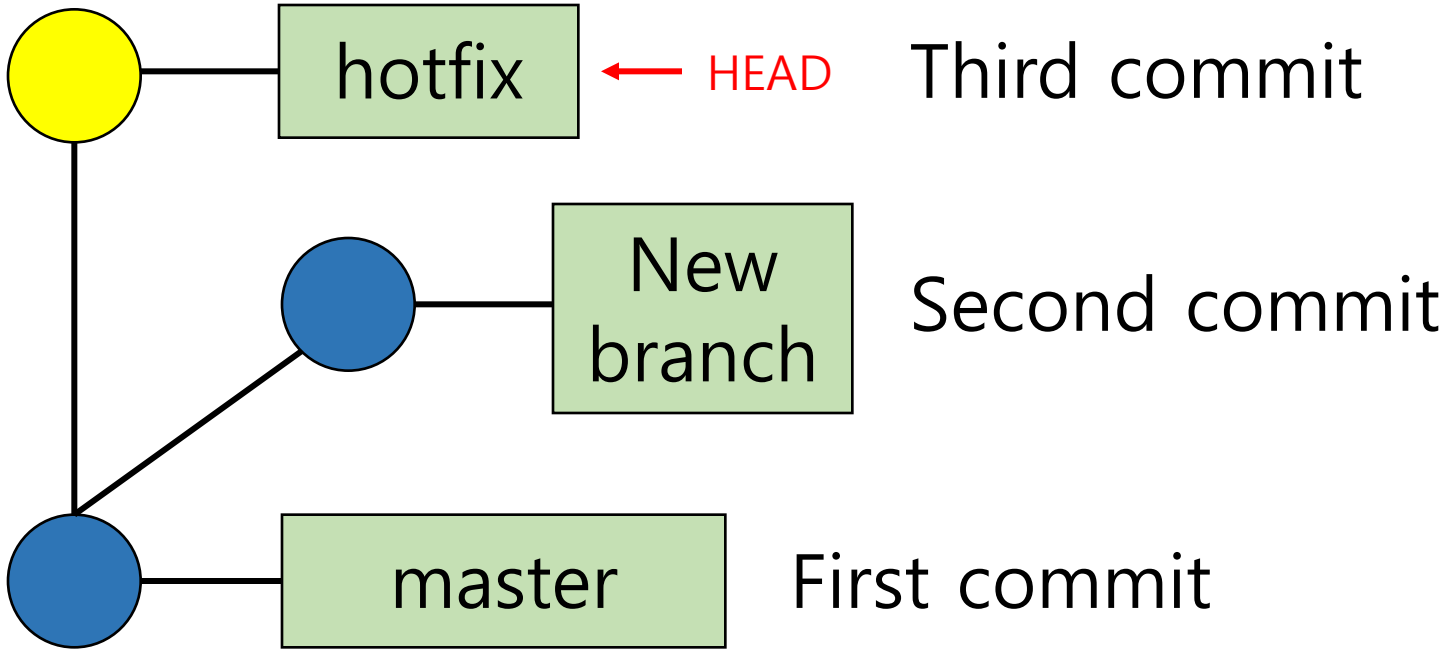


$git checkout master -> hotfix branch 생성

- HEAD pointe를 newBranch에서 master로 옮긴다.

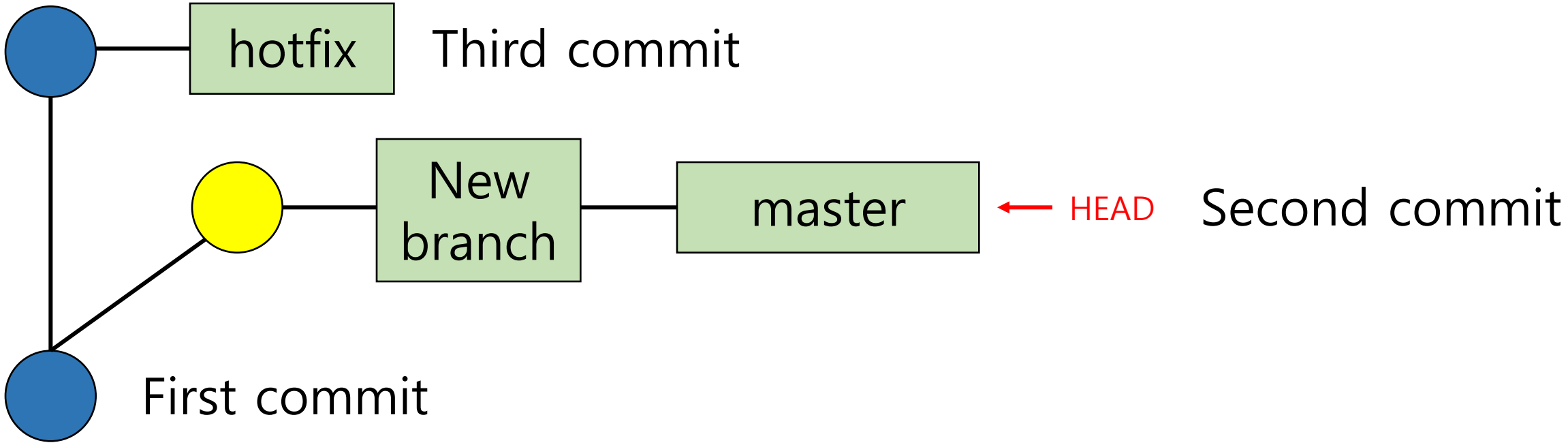


Hotfix 브랜치에서 파일 수정 후 세 번째 커밋



$git merge newBranch(fast-forward)

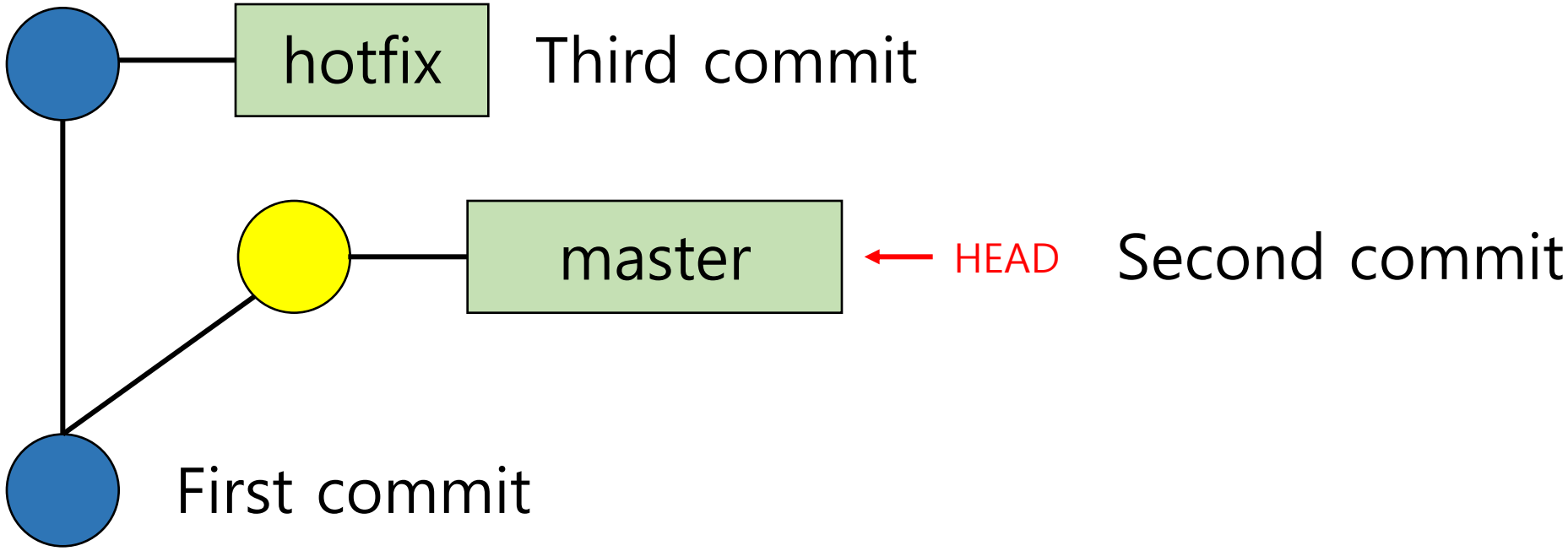
- ($git checkout master를 하여 현재의 head를 maste로 옮기고 maste를 기준으로 newBranch를 병합한다.)



- First commit은 참조하는 브랜치가 없기 때문에 고유의 ID로 pointer를 옮겨야한다.

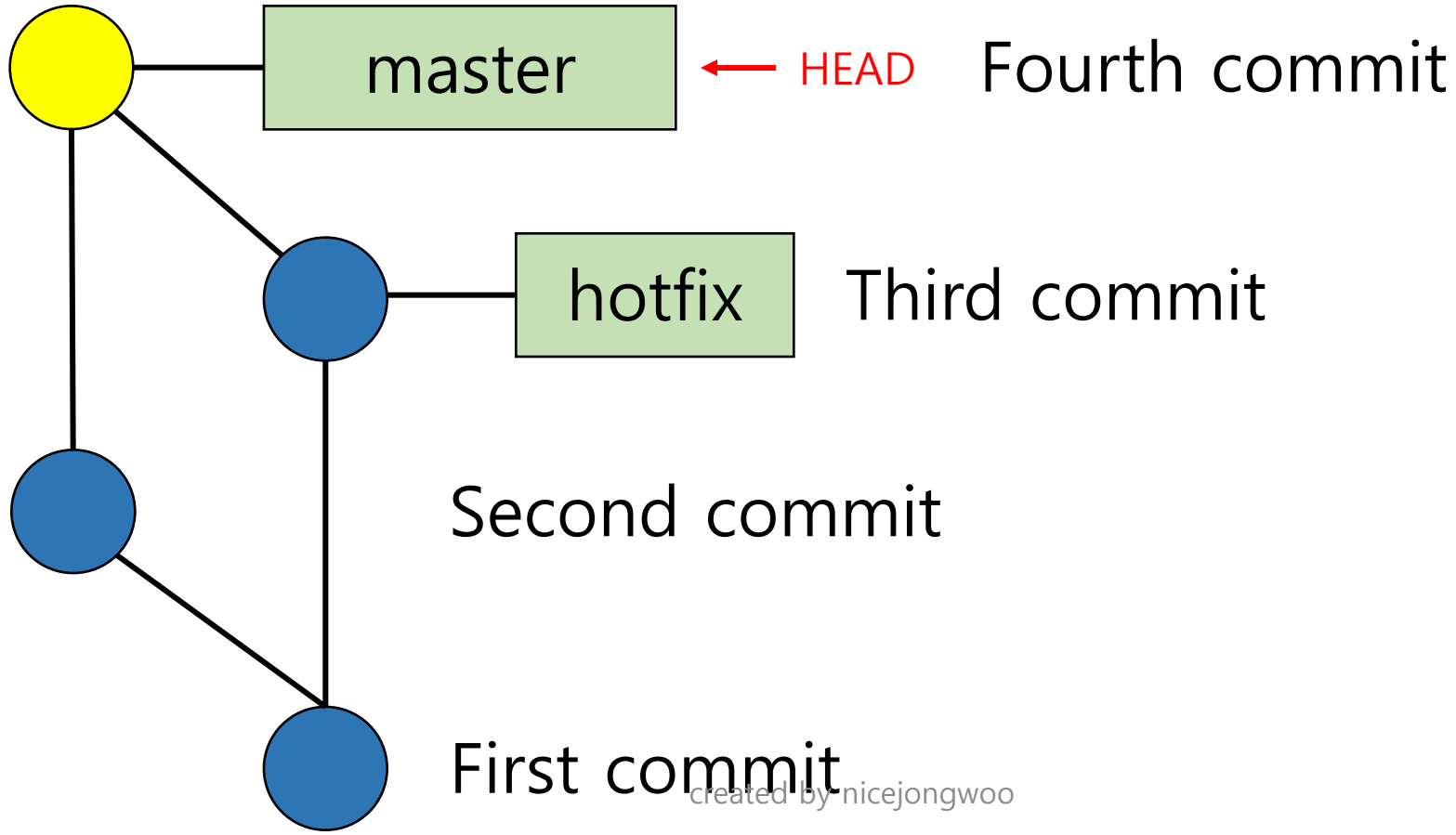
$git branch –d newBranch

- 기능구현이 끝난 브랜치를 병합하여 브랜치가 더 이상 불 필요하면 해당 브랜치를 삭제한다.



$git merge hotfix(3-way merge)

- 3개의 서로다른 커밋끼리 merge가 이루어지기에 새로운 커밋이 생기면서 현재의 브랜치가 참조하게된다.



Merge conflict

- 만약에 3-way merge를 할 때 같은 소스파일의 같은 라인이 다른 내용으로 수정 되었다면

  병합 충돌이 일어나서 해당 파일을 수정을 완료해 줘야 한다.

  Auto-merging 해당파일명

  CONFLICT (content): Merge conflict in 해당파일명

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

- 해당 파일을 수정하게 되면 git status 명령어를 실행하여 변화 상태를 확인하고 해당 파일을 staged 상태로 만들어 주기 위해 git add 명령어를 실행한 뒤 새로운 git commit을 실행한다.