* **Branch의 기능**

같은 뿌리에서 나왔지만 서로 다른 역사를 쓰는 것. 요구사항은 고객마다 서로 다르기 때문에 그에 맞춰 특정 버전에서 요구사항을 만족시킨다.

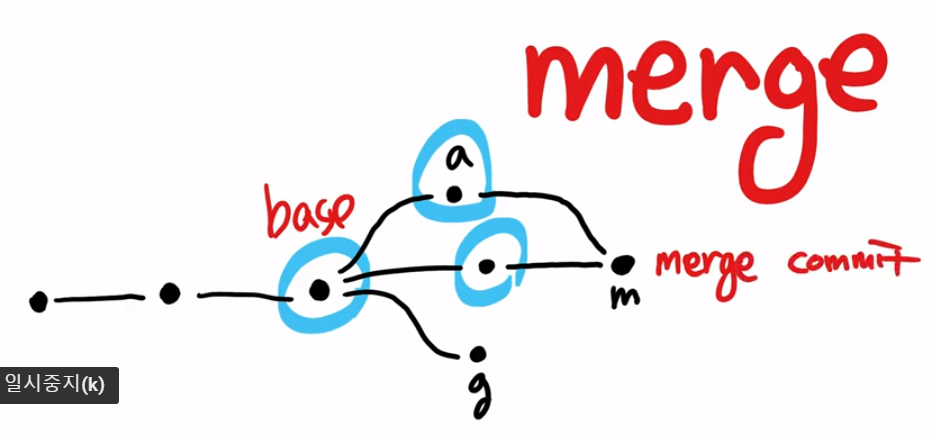
* **Conflict의 기능**

하나의 브랜치에 있는 텍스트파일 a와 다른 브랜치에 있는 텍스트파일 b를 합치고싶을 때 깃에서 자동으로 합쳐준다. 근데 같은파일을 합치려고 할 때 충돌(conflict)이 일어나기 때문에 합병을 하는 사람에게 경고를 날린다.

* **Branch의 사용법**

Checkout 으로 브랜치 이동한 후, 그것에 대해 버전을 최신화 하면 됨.

* **브랜치 병합**



헤드는 마스터를 가리키고 있어야 하고, merge 명령어로 나의 현재 브랜치와 특정 브랜치를 병합 시킨다.

* Log –p 하면 해당 브랜치의 줄기에 대해서만 로그가 보인다.

파일의 명이 다르면 브랜치에 병합될 뿐이고, 파일의 명이 같으면 다른 브랜치이고 내용이 다를지라도 내용을 병합해준다.

그러나, 같은 이름의 파일의 같은 위치를 다르게 수정한다면 충돌이 일어난다.

Git status 로 어떤 파일들이 충돌이 일어났는지 알 수 있고, 파일 내용을 열어서 어떤 부분에서 충돌이 일어났는지 알 수 있다.

# title

content

<<<<<<< HEAD 여기부터

master

======= 🡪 구분자 여기까지 master branch의 내용

o2

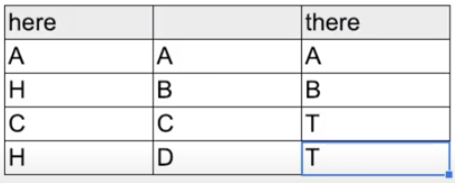
>>>>>>> o2 여기부터

# title

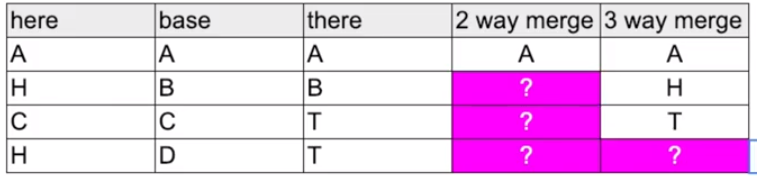
Content 여기까지 o2 branch의 내용

실제로 이런식으로 저장이 되는데 알아서 수정해서 다시 저장하면 된다.

* **3 way merge**



이런 3개의 브랜치가 병합된다면 어떻게 될까?



이렇게 베이스가 되는 버전과 3개를 비교했을 때, 수정된 부분만 채택하면 되기 때문에 저런 식으로 작업이 이루어 진다.

* **외부도구를 이용하여 병합하는 방법**

P4mergetool 설치 후

Git config –global merge.tool p4mergetool 입력.

이후 cat ~/.gitconfig 입력해서 툴이 적용됐는지 확인할 수 있음. 이후 다른 설정을 해야하니 구글링.

* **Cherry pick**

베이스로 시작된 두 브랜치 A, B가 있다고 가정. A의 진행 중 한 버전의 내용을 그대로 가져오고 싶다면 체리 픽을 이용한다.

* **Rebase**

다른 브랜치를 베이스로 새 시작하는.. 타임라인이 깔끔해 보이는 그런 효과가 있다고 한다.

\*\*명령어

Git branch = 브랜치의 목록을 보여줌.

Git branch 이름 = 이름의 브랜치를 만듬.

Git log –all –graph –oneline = 모든 브랜치를 시각적으로 한줄로 나타냄.

Git checkout 브랜치이름 = 해당 브랜치가 마지막에 커밋되었던 시점으로 워킹카피가 옮겨진다.

Git commit –m “메세지” 특정파일이름 = 특정 파일 하나만 커밋한다.

Git commit –amend = 커밋 메시지를 수정할 수 있다.

\*\*에러