Git, github 스터디 02

2021-2 KCA

김지환

본 ppt의 자료는 git book, github을 참고하였음을 밝힙니다.

commit 간 이동 | git reset --{option} {commit_id}

- commit을 계속 해야 하는 이유?

⇒그 시점(버전)으로 이동할 수 있기 때문!!

```
PS F:\건국대\_동아리\KCA\2021.2학기 스터디\git> git log
commit 8cde8921f9049baebe94e4e01dee3ca394501662 (HEAD -> master)
Author: Turtle-hwan <kjhwan0802@naver.com>
Date:   Thu Sep 9 14:45:17 2021 +0900

    first
```

- 파랑 글씨 HEAD(포인터)가 가리키는 곳이 우리 눈에 소스 코드가 보이는 버전(커밋)

git reset --hard {해당 commit id} : HEAD를 해당 commit으로 이동

[옵션] git reset --{option} {commit_id}

- reset 옵션에 대하여..
- Hard, Mixed, Soft

- 공통점 : HEAD를 특정 commit을 가리키도록.

--hard
HEAD
-> <commit id>

Staging Area
-> <commit id>

Working Directory
-> <commit id>

--mixed
HEAD
-> <commit id>

Staging Area
-> <commit id>

Working Directory
-> <최근 커밋>

--soft
HEAD
-> <commit id>

Staging Area
-> <최근 커밋>

Working Directory
-> <최근 커밋>

[추가] git reflog

- 로컬 .git repository에서 일어났던 모든 .git history (!= commit history)를 보여준다.

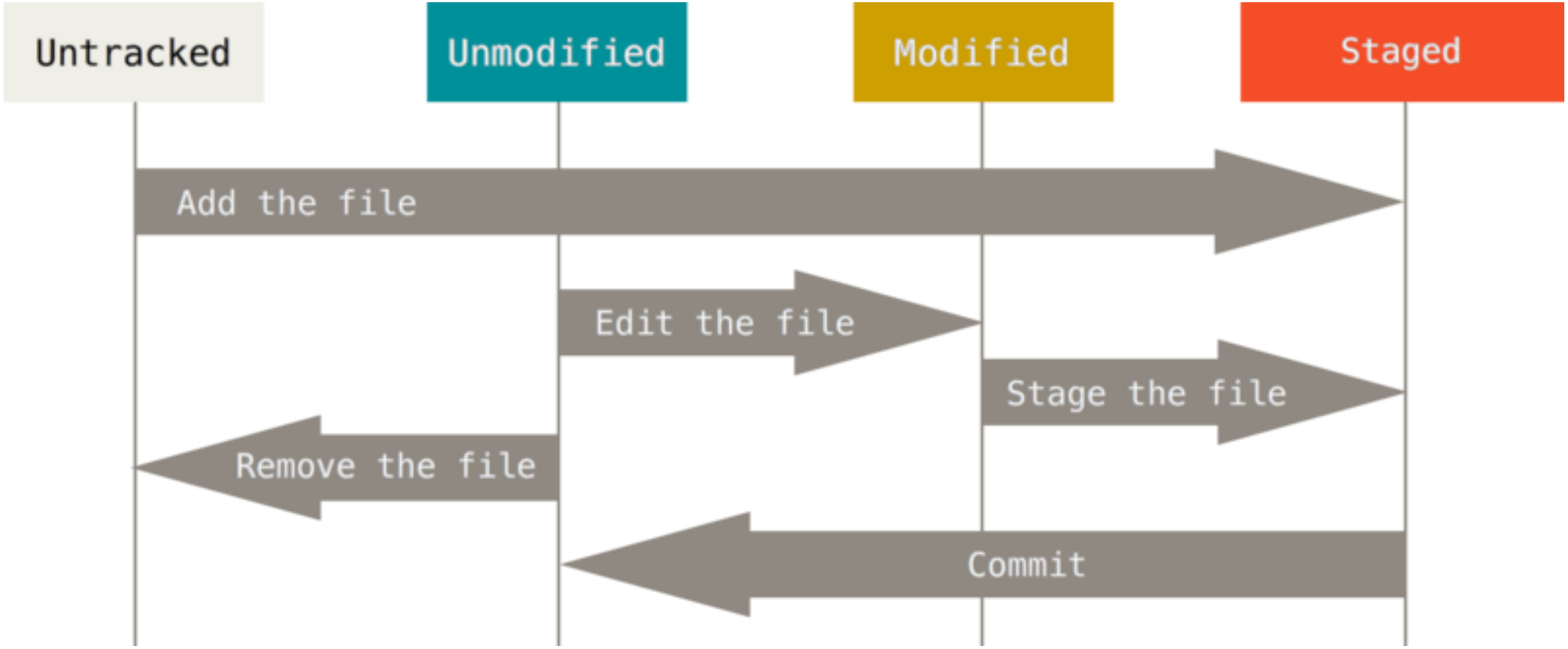
git update-ref 로 reflog 추가 가능하다.

- HEAD가 가리키는 커밋의 위치 변화를 주의해서 보자!

- HEAD@{숫자} == commit_id 로 대체해서 사용 가능하다.

- 여기서 이전 HEAD 위치로 reset 가능!

[추가] 파일의 상태 (Working Directory ~ Staging Area)



[실습] [github] git push

전 시간까지 git에서 commit을 어떻게 생성 / 읽기 / 수정 / 삭제 하는지 알아보았다!

이제 github에 내 commit 기록들을 올려보자!

git push <remote 저장소 이름(==주소)> <브랜치 이름>

-u 옵션을 추가하면, 이후엔 git push만 써도 됨. (뒤 슬라이드 참고)
보통 자신의 remote 저장소 이름은 origin, 기본 브랜치 이름은 master 또는 main으로 많이 쓴다.

참고 + 주의!) 강제로 github의 commits 기록을 덮어 쓸 때, git push -f (실제 개발에서는 쓰지 않음)

[참고] github push가 처음이라면!

[(commit이 없는) 새로운 repository를 만들 때]
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin <주소>
git push -u origin main

[(commit이 있는) 기존 repository를 연결할 때]
git remote add origin <주소>
git branch -M main
git push -u origin main

[추가] SSH, GPG 키 인증

- SSH : Secure Shell

HTTPS/SSH : .git repository를 서버에 올리고 내려받는 통신 옵션

- GPG : GNU Privacy Guard
  통신 암호화 + 개인 식별 서명

GPG : 특정 사용자가 인증되었고 믿을 수 있다는 것을 보증.

[참고]HTTPS 주소로 받은 remote repository

- HTTPS로 받은 저장소에 push, pull 할 때 자꾸 github 아이디와 비번을 묻는다면?

  git config --credential.helper cache : 일정 시간 동안 계정 정보 기억

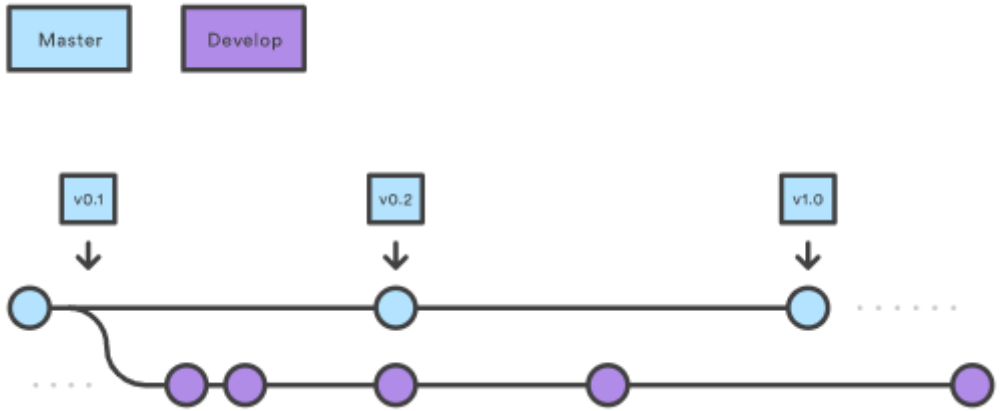
  제어판 - 자격 증명 관리자 확인

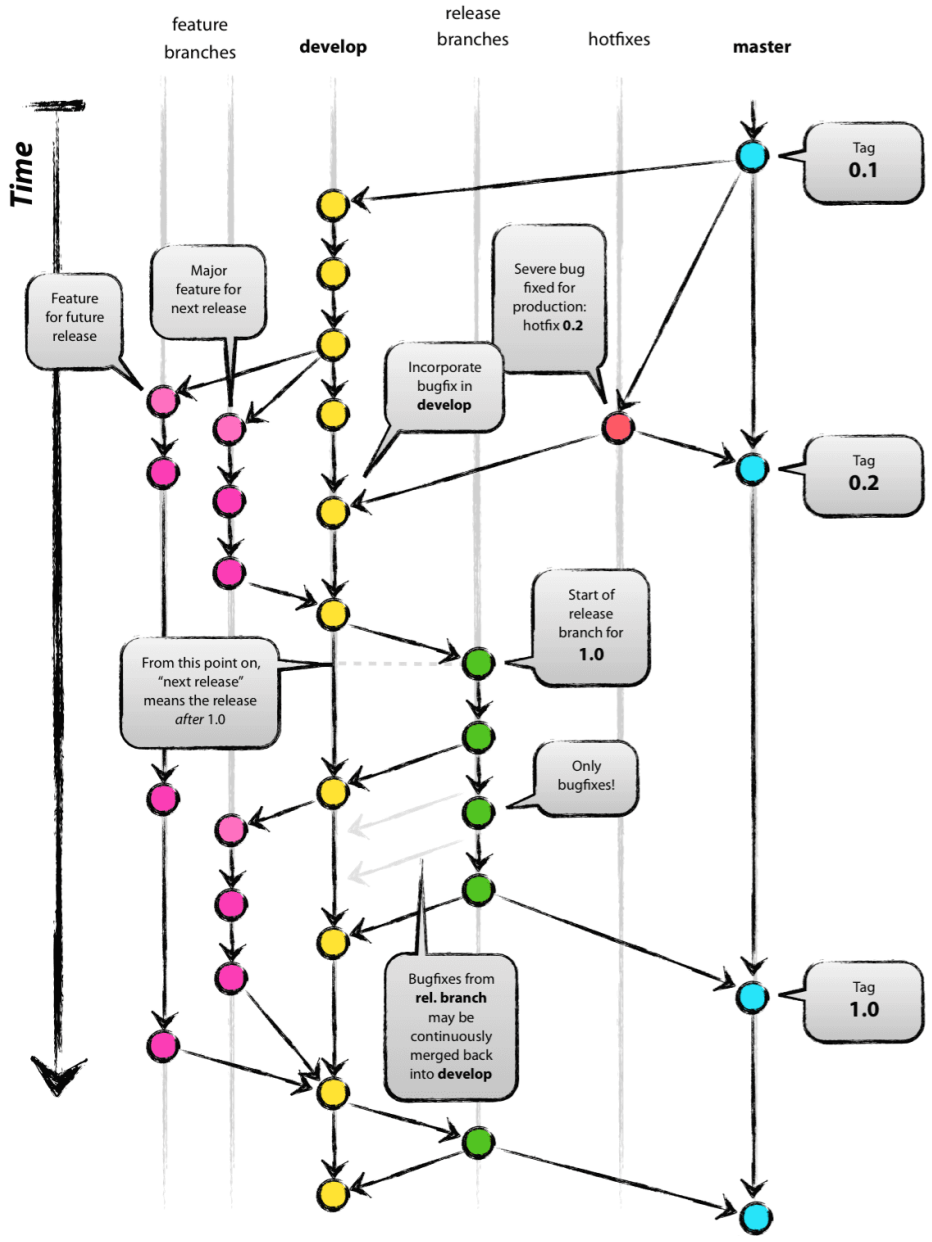
git pull (항상 pull을 먼저 하고, push한다)

- 원격 저장소에서 자료들을 끌어와서 로컬 저장소를 동기화 시켜준다.
- pull == fetch + merge
- fetch는 자료를 끌어오되 로컬 repository와 동기화 하지는 않음. (별도의 로컬 branch에 둔다.)
- merge는 뒤에 branch 얘기하면서 추가 설명!

Branch 개념

- Branch는 가지치기, 즉, 시점(버전, 커밋)의 분기





branch 생성과 확인

- git branch dev : dev 브랜치 생성
- git checkout {branch_name} : {branch_name}을 HEAD가 가리킴.
- git log => dev 생성된 것 및 HEAD 위치 확인

- git log --all(모든 branch 표시) --graph(그래프 형태로 표시) --oneline(각 커밋 상세 생략, 한 줄 표시)

branch merge

+) git rerere : 충돌 시마다 각 코드의 해결법을 기록해 두었다가 나중에 같은 충돌 시 자동 해결

- git merge {합쳐질 branch} : 명령을 실행한 branch에, {}가 합쳐진다.

- conflict 발생 시! : 발생 파일(git status로 확인)에 들어가 보면, 이런 내용이 뜬다!!

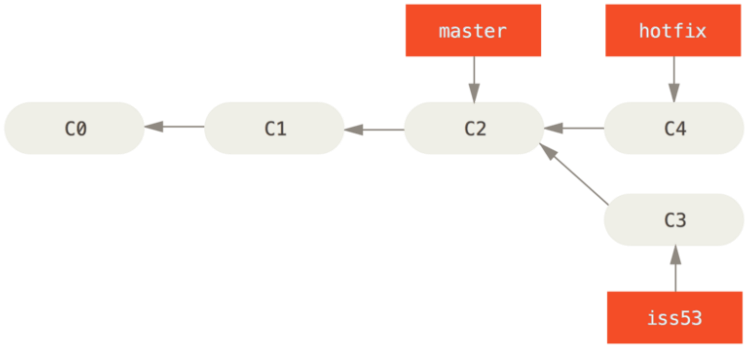
<<<<<<<< HEAD (current change)
코드코드
코드코드코드
========
변경코드
변경수정코드
>>>>>>>>> {합쳐질 branch_name} (incoming change)

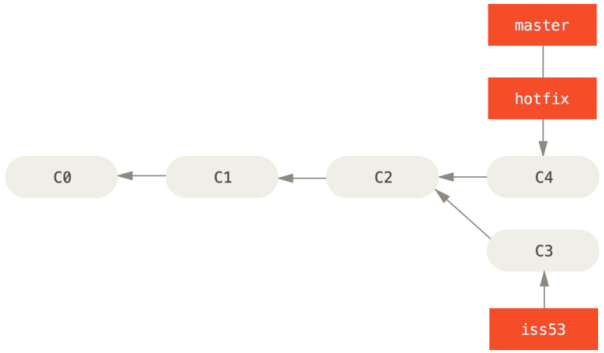
같은 부분에 다른 코드가 merge되어 자동으로 합칠 수가 없기 때문!! => 우리가 직접 수정해 주어야 한다!!

⇒ 우리가 필요한 최종 코드 모습으로 만들고, git add . / git commit -m "" 해주면 된다!

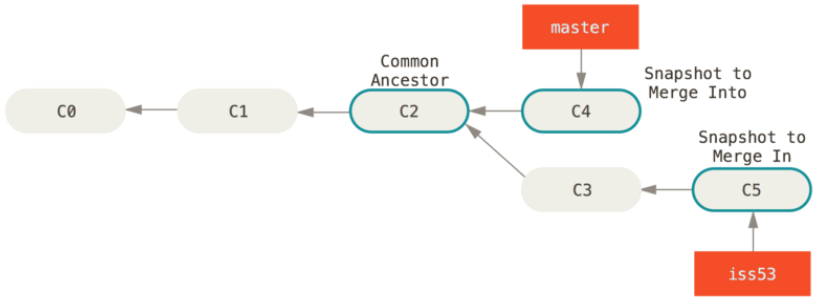
merge의 종류

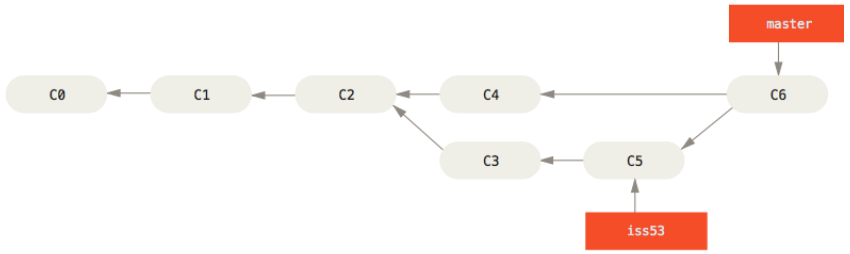
- fast-forward merge : 단순히 브랜치가 가리키는 commit 위치만 옮김.





- 3-way merge (non-fast-forward merge) : 새로운 merge commit을 만듦.





git push -u origin master 의미

- 로컬의 master 브랜치를 origin에 들어 있는 주소로 올린다.

- -u == --set-upstream : 로컬의 master 브랜치가 항상 github의 master 브랜치를 바라보도록.

  처음 -u 옵션을 주면, 이후에는 그냥 push, pull만 해도 자동으로 바라보는 곳에서 가져온다.

- push/pull은 branch 단위이다!!!

  다른 branch 를 올리려면, git push {주소} {branch_name} 입력해야 한다.

git branch 관련 명령어 정리 CRUD

- git branch {branch_name} : 새 branch 생성

- git branch == git branch --list : 로컬 branch list 나열
- git branch -r : 원격(리모트) branch 나열
- git branch -a : 모든 branch 나열

- git branch -m {name} : 현재 branch 이름을 바꿈

- git branch -d {branch_name} : 지정된 branch 안전 삭제(merge 안 되어 있으면 삭제 못함.)
  git branch -D {branch_name} : 지정 branch 강제 삭제

- git checkout {branch_name} : 현재 branch 변경