STEP

디지털 기초

Git 기본 퀵 가이드

4차시 : 실무에서 유용한 명령

중급 명령어 익히기

1. rebase

(1) 재배치 명령어

(2) 한 브랜치의 커밋을 다른 브랜치에 합치는 데 사용

(3) merge와의 차이점: merge 사용 시 두 브랜치의 커밋을 합쳐서 새로운 커밋(merge 커밋) 생성

2. 브랜치 생성하기

(1) merge 커밋

① 일반 커밋과 달리 작업의 스냅샷이 아님: 메타 커밋
② merge 커밋 하나에서 충돌 발생 시 해결은 한 번만

(2) Rebase

① git rebase main 명령 입력 시 현재 브랜치에는 존재하지만 main 브랜치에 없는 커밋을 모두 main의 마지막 커밋 이후에 배치
② 재배치된 커밋은 이전 커밋과 내용은 동일하지만 다른 체크섬 값을 가지는 다른 객체
③ 단점
- 여러 개의 새로운 커밋 생성
- 충돌 해결 과정도 merge보다 까다로움

중급 명령어 익히기

3. gitignore

(1) 커밋 작업 시 불필요한 파일을 커밋하게 되는 상황을
예방하기 위한 명령

(2) 활용 방법
- .gitignore에 파일 또는 폴더 이름 기입
- 변경 내용을 스테이지에 추가 후 커밋
- 지정된 이름을 가진 파일과 폴더는 Git 저장소에서 무시

(3) GitHub에서 프로젝트 생성 시 사용 언어로 지정할 경우
.gitignore 자동 생성

(4) gitignore.io 사이트 활용

4. reset

(1) hard reset

① 브랜치를 특정 커밋으로 되돌릴 때 사용, 가장 보편적
② 작업한 내용을 잃을 가능성에 유의

(2) soft reset

- 작업 내용을 남기고 싶을 때 사용

(3) mixed reset

① 옵션 미설정 시 기본적으로 Mixed 모드로 동작
② 작업 디렉토리의 내용을 소스트리나 에디터를 통해 직접
확인하는 안전한 방법

중급 명령어 익히기

5. reset과 revert

(1) Reset
- 현재 브랜치 자체를 특정 커밋으로 되돌릴 때 사용

(2) Revert
- 취소 커밋을 새로 생성해 기존 커밋이 모두 히스토리에 보존되어 상대적으로 안전
- 주의 사항: 여러 개의 커밋을 취소할 경우 생성되어 있는 반대 순서로 revert
- 단점: 취소 커밋 생성
  장점: 안전한 작업

6. fork 기능

(1) 원격 저장소 추가 명령

① 원본 저장소에 있는 fork 버튼 클릭 시 저장소의 사본을 내 개인 저장소에 복사
② 내 로컬 PC에서 사용하려면 git clone 명령 이용
③ git remote add 명령을 사용하면 원격 저장소 추가 가능

(2) origin 저장소: 기본적인 원격 저장소

① 소유자는 개발자 자신
② 읽기, 쓰기 권한 모두 보유

(3) upstream 저장소: 원본 저장소

① 읽기 권한만 보유
② 기여하고 싶은 코드가 있다면 origin 저장소에 변경 내용 push > upstream 저장소로 pull request 보내기

중급 명령어 익히기

7. git stash 명령

(1) 현재 작업 내용이 stash라고 하는 임시 공간에 저장

(2) 작업 디렉토리는 변경사항이 없는 깔끔한 상태

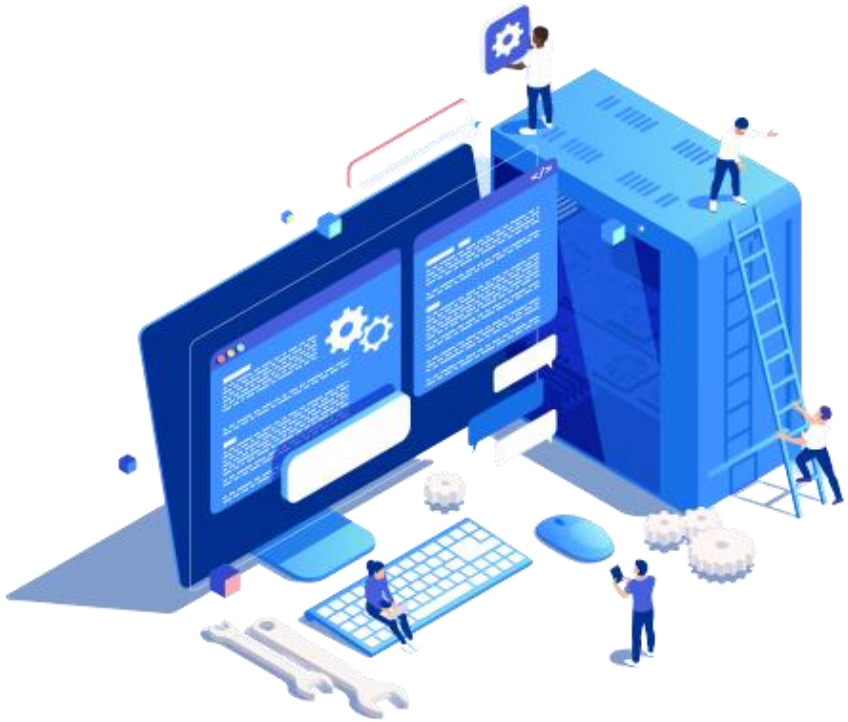
- stash 명령 시 문제 없이 브랜치 변경 가능

(3) First In First Out 자료 구조

- git stash 명령을 여러 번 입력 시 변경사항이 순서대로 저장

(4) git stash pop 명령의 기능

① stash 변경사항을 작업 디렉토리에 반영할 경우 사용
② stash 변경사항을 작업 디렉토리에 반영 없이 삭제하고 싶을 경우 사용



실무에서의 응용 사례

1. 사례1: 사라진 커밋을 찾아라!

사례

작업 중 잠시 다른 브랜치로 변경을 했다가
원래의 브랜치로 돌아오려고 했는데
지금까지 브랜치가 아닌 상태에서 커밋을 하고 있었다는 것을 깨달았습니다. 어떻게 해결할 수 있을까요?

원인 및 해결책

(1) 원인: detached branch 상태

- git checkout '커밋 체크섬' 명령 사용 시 브랜치 없이 커밋을 작업 디렉토리로 체크아웃 할 때 발생

(2) 해결책: git reflog 명령

① 브랜치와 HEAD가 과거 커밋의 체크섬 값을 모두 보여주는 명령
② 잃어버린 커밋을 찾은 후
> git branch [새로운 브랜치명] 커밋 체크섬 명령을 이용해 해당 커밋을 가리키는 브랜치 생성

실무에서의 응용 사례

2. 사례2: 범인을 찾아라!

사례

프로젝트에 갑자기 발생한 버그로 철야를 하던 박 대리.
원인을 몰라 한참을 고생하다가 담당한 코드 파일 중 하나에
자신이 짜지 않은 초보적인 실수가 있다는 걸 확인했습니다.
하지만 **모든 커밋 히스토리를 일일이 뒤지기에는**
너무 커밋이 많은 상황, 어떻게 해야 할까요?

해결책

(1) 해결책: git blame 명령

① 지정한 파일의 각 줄마다 수정된 커밋과 커밋의 작성자 표시
② 파일이 너무 클 경우
- git blame -L 10, 20 파일명 특정 줄 검사
- git blame -L : 함수 이름 지정된 함수명 검사

실무에서의 응용 사례

3. 사례3: PR 전체 취소하기

사례

철수는 코드 리뷰를 통과해서 main 브랜치에 머지가 된
본인의 PR에서 중대한 버그를 발견했습니다.
버그를 고치기에는 너무 어려워서
결국 PR 전체 커밋을 취소해야 하는데
이미 이후에 머지 된 다른 팀원들의 커밋이 너무 많아서
하드 Reset은 사용할 수 없는 상황입니다.
어떻게 해야 본인의 PR을 취소할 수 있을까요?

해결책

(1) 해결책: GitHub의 PR revert 명령

- 다른 커밋은 건드리지 않고 내가 작성한 PR만 취소할 수 있는 안전한 방법

(2) 차선책

- 직접 PR의 커밋들을 하나씩 Revert 후 Revert 커밋을 모아 새로운 PR 보내기

실무에서의 응용 사례

4. 사례4: 히스토리를 깔끔하게 만들고 싶어요!

사례

성철이네 팀은 개발한 소프트웨어를
오픈소스로 GitHub에 공개하기로 결정했습니다.
그런데 그동안 쌓인 **커밋 히스토리가 너무 꼬여 있어서**
깔끔하게 정리를 하려고 합니다.
어떤 기능을 사용할 수 있을까요?

해결책

(1) 해결책: git rebase –i 명령

① I의 의미
- interactive
- 사용자가 터미널 에디터 또는 GUI 기능을 통해 커밋 내용을 다양하게 조작

② rebase –i 장점
- 커밋의 순서 및 메시지 변경 가능
- 불필요한 커밋 삭제 가능
- 커밋 내용은 남기고 다음 커밋에 합치기 가능

실무에서의 응용 사례

5. 사례5: 일부 커밋만 필요해!

사례

수아는 기존에 개발하던 기능이 더 이상 필요하지 않게 되었습니다.
아쉬운 마음을 뒤로하고 해당 기능을 구현했던 커밋을
모두 취소하려고 합니다.
그런데 커밋들 중에 **몇 개의 유용한 커밋이 있어**
그들만 남겨 놓고 싶습니다.
어떻게 하면 쉽게 해결할 수 있을까요?

원인 및 해결책

(1) 원인: 특정 브랜치의 커밋 중 일부만 필요한 상황

(2) 해결책: git cherry-pick 명령

- 다른 브랜치에서 원하는 커밋만 현재 브랜치로 가져오는 명령

6. bisect 명령

(1) 갑작스레 원인을 알 수 없는 버그가 발생했을 경우
이 버그를 찾을 때 유용하게 사용되는 명령

① 문제가 생긴 커밋, 문제가 없는 커밋 지정
② 이분 검색을 통해 문제 발생 여부 확인
③ 원인이 되는 커밋 발견

(2) 버그 발생 확률이 1/100만일 경우
bisect로 20번 시도 내 버그 발견

오늘의 키워드

rebase, .gitignore, reset, revert, fork, git stash,
git reflog, git blame, PR revert, git rebase -i,
git cherry-pick 명령, bisect 명령

생각할 거리

실무에서 유용한 중급 명령어를 익혀
실무에서 발생하는 문제를 해결해 보자!