Git

Commit

커밋(commit)은 Git과 같은 버전 관리 시스템에서 특정 시점의 리포지토리 파일들의 수정 사항을 의미합니다. 커밋은 프로젝트의 버전 히스토리에 해당 시점의 변경 사항을 저장하는 것으로, 이전 커밋 이후로 이루어진 변경 사항들을 하나의 세트로 기록합니다.

ex: git commit -m “Git”

> 특정 버전 기록

Branch

브랜치(branch)는 Git에서 개발 작업을 분리하여 독립적으로 진행할 수 있도록 하는 기능을 제공하는 것입니다. 브랜치를 이용하면 다양한 작업을 병렬적으로 수행하고, 변경 사항을 각각 독립적으로 저장하며, 프로젝트의 다양한 기능을 개발할 때 유용하게 사용할 수 있습니다.

> 새로운 분기점 생성

ex: git branch zs

Merge

"Merge"는 Git에서 두 개의 브랜치를 하나로 합치는 작업을 말합니다. 즉, 두 개의 다른 브랜치에 있는 변경 사항을 하나의 브랜치에 통합하여 프로젝트를 합치는 것입니다.

> 두 분기점 통합하기 두 병합된 커밋들의 히스토리로 남는다

ex: git merge zs

Rebase

"Rebase"는 Git에서 브랜치를 이동하거나 다른 브랜치의 변경 사항을 현재 브랜치에 적용하는 작업을 말합니다. 기본적으로 `git rebase` 명령은 브랜치의 기점을 변경하거나 다른 브랜치의 변경 사항을 현재 브랜치에 적용하여 더 선형적인 히스토리를 만들어냅니다.

> 두 분기점 통합하기 선형된 히스토리로 남는다.

ex: git rebase zs

HEAD

`HEAD`는 Git에서 현재 작업 중인 브랜치의 최신 커밋을 가리키는 포인터입니다. 쉽게 말해, `HEAD`는 현재 작업 중인 커밋을 가리키는 '머리'라고 생각할 수 있습니다.

> 포인터

상대참조 ~, ^

- ^: 상위 커밋을 참조하는 상대참조
  - git checkout zs^
- ~n: n번째 조상을 가리키는 상대참조
  - git checkout zs~2

Fetch

`git fetch`는 원격 저장소에서 최신 변경 사항을 로컬 저장소로 가져오는 Git 명령어입니다. `git fetch`를 실행하면 원격 저장소의 변경 사항을 로컬에 반영하지 않고, 단지 최신 정보를 확인하고 다운로드합니다. 즉, 로컬 작업 히스토리나 브랜치는 변경되지 않으며, 원격 저장소의 변경 사항만 로컬에 가져오게 됩니다.

> 원격 저장소 변경사항 가져오기

ex: git fetch upstream

Pull

`git pull`은 Git에서 원격 저장소(remote repository)의 최신 변경 사항을 가져와서 로컬 브랜치에 병합하는 명령어입니다. `git pull`은 `git fetch`와 `git merge`를 합친 단축 명령어이므로, 원격 저장소의 변경 사항을 가져온 후 자동으로 로컬 브랜치에 병합합니다.

> Fetch + Merge

ex: git pull upstream main

Q. 우경이의 repo에 있는 backend-newbs의 커밋들을 내 개인 저장소랑 병합할려면?

Q. 왜 Readme.md를 수정하면 오류가 났던 것일까?