W05
버전 관리와 변경 추적

코드의 세이브 포인트 만들기

코드 복구가 필요한 상황

개발하는 도중

- AI가 여러 파일을 동시에 수정했는데 뭐가 바뀌었는지 모르겠다
- 잘 되던 코드가 갑자기 안 된다
- AI한테 "원래대로 해줘"라고 했는데 더 이상해졌다
- Ctrl+Z를 눌러도 한계가 있다

커서의 복구 기능 사용

- 커서는 작업을 할 때 코드를 스스로 저장하지 않고 작업한다.
- 작업한 내용을 확인하고 문제가 있다면 되돌릴 수 있다
- 현재 작업중인 코드에서만 변경이 가능

커서의 복구 기능 사용 방법

Cursor에서 AI가 코드를 수정하면:

- 채팅창에 변경된 파일 목록이 뜬다
- 각 파일 옆에 Reject 버튼 → 그 변경만 취소
- 전체 Restore → 이 대화 시작 전으로 복구

한계:

- 채팅창 닫으면? 끝. 복구 불가.
- 어제 작업으로 돌아가고 싶으면? 방법 없음.
- 여러 대화에 걸쳐 작업했으면? 어디서 잘못됐는지 모름.

→ "세이브 포인트"을 직접 만들어둬야 한다

버전 관리란?

"지금 이 상태를 저장해두고, 나중에 돌아올 수 있게 하는 것"

구글 독스의 버전 기록

- 파일 → 버전 기록 → 이전 버전 보기
- 언제 누가 뭘 바꿨는지 다 나온다
- 원하는 시점으로 복원할 수 있다

코드에서도 중요하다

- 잘 되던 시점을 저장해두면 잘못되어도 돌아갈 수 있다
- 저장 안 하면? 처음부터 다시 만들어야 한다

Git & Github

Git: 내 컴퓨터에서 코드의 버전을 관리하는 도구

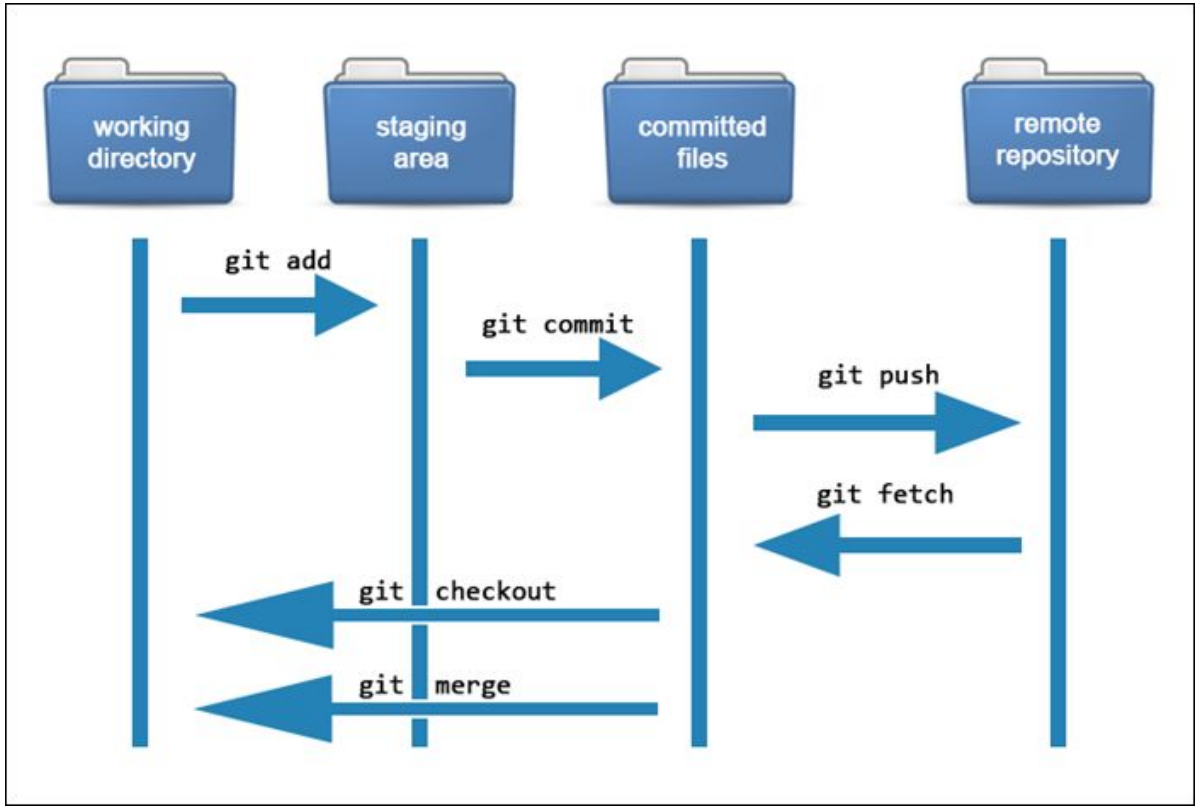
- 변경 사항을 기록한다
- 이전 상태로 되돌릴 수 있다

GitHub: Git으로 관리하는 코드를 인터넷에 저장하는 곳

- 컴퓨터가 고장 나도 코드가 살아있다
- 다른 컴퓨터에서도 작업을 이어갈 수 있다

Git = 버전 관리 도구

GitHub = 버전을 저장하는 클라우드 서비스


working directory
staging area
committed files
remote repository
git add
git commit
git push
git fetch
git checkout
git merge

용어 (1) - 리포지토리

- GitHub 들어가면 "Create Repository" 버튼
- AI한테 "커밋해줘" → "리포지토리가 없습니다" 에러

리포지토리 = 프로젝트 폴더 + 변경 기록

- Cursor에서 열어둔 그 폴더가 리포지토리가 된다.
- 단, git init을 해야 Git이 이 폴더를 추적하기 시작함.
- 리포지토리 없이 커밋 불가능.
- 하나의 서비스 = 하나의 리포지토리

용어 (2) - 스테이징, 로컬, 리모트

스테이징(Staging): 커밋할 파일을 고르는 단계

- "이 파일들을 저장하겠다"고 표시하는 것
- 장바구니에 담는 것과 비슷

로컬(Local): 내 컴퓨터

- 인터넷 없이도 작업 가능
- 커밋하면 여기에 저장됨

리모트(Remote): GitHub (인터넷 저장소)

- 푸시해야 여기에 올라감
- 백업 + 공유 역할

명령(1) - git init, git add

git init: 이 폴더를 Git으로 관리하기 시작한다

- 새 프로젝트 시작할 때 한 번만 실행
- 이미 GitHub에서 clone 해왔으면 필요 없음
- 실행하면 .git 폴더가 생김 (숨김 폴더)

git add: 커밋할 파일을 고른다

- git add . → 변경된 파일 전부
- 시나리오 단위로 작업하니까 보통 전체 추가
- Cursor Source Control에서 + 버튼 누르는 것과 같음

명령(2) - git commit

커밋: "지금 이 상태를 기록한다"

파일 저장(Ctrl+S)과 다르다

- 파일 저장: 이전 내용이 사라진다
- 커밋: 이전 기록이 모두 남아있다

언제 커밋하는가?

- 시나리오 하나가 완성됐을 때
- 테스트가 통과했을 때
- AI한테 큰 작업 시키기 전

"커밋은 '여기까지는 확실히 된다'는 표시다"

질문 1. 새로 리포지토리를 만들어서 커밋하기

“현재 프로젝트를 git으로 관리하고 싶어. 새 리포지토리를 만들어 줘"

“지금까지 한 작업을 git에 커밋해줘"

GitHub에도 올리려면:

- GitHub에서 빈 리포지토리 먼저 생성
- AI에게 "GitHub 리포지토리 연결하고 푸시해줘" 요청
- 또는 Cursor에서 "Publish to GitHub" 버튼 사용

명령(3) - git status

AI가 여러 파일 수정했는데 뭐가 바뀌었는지 모르겠다

커밋하기 전에 확인하고 싶다

보여주는 것:

- 수정된 파일 목록
- 새로 생긴 파일
- 삭제된 파일
- 아직 스테이징 안 된 것 / 된 것

명령(4) - git diff

git status로 파일 목록은 봤는데 정확히 어떤 코드가 바뀐 거야?

- 어떤 줄이 삭제됐고 (빨간색)
- 어떤 줄이 추가됐는지 (초록색)

AI 에이전트 맥락:

- AI가 "수정했어요" 했는데 뭔가 이상할 때
- "원래대로 해줘" 하기 전에 뭐가 바뀌었는지 먼저 확인
- 의도하지 않은 변경 발견 가능

명령 (5) - git push

커밋은 내 컴퓨터에만 저장된 것. 푸시해야 GitHub에 올라간다.

- git push origin main

언제 푸시하나:

- 오늘 작업 끝났을 때
- 컴퓨터 바꿔서 작업 이어갈 때
- "여기까지는 확실히 백업해두자" 싶을 때
- 커밋 = 로컬 저장점 / 푸시 = 클라우드 백업

.gitignore

gitignore = Git이 무시할 파일 목록

- 커밋하고 GitHub 봤더니 node_modules 폴더가 올라가 있다 (파일 수천 개)
- .env 파일의 API 키가 공개됐다

왜 중요한가:

- GitHub은 공개 저장소면 누구나 볼 수 있음
- API 키 노출되면 과금 폭탄 맞을 수 있음
- node_modules 올리면 푸시/풀 할 때마다 오래 걸림

.gitignore

올리면 안 되는 것들:

- node_modules/ - 용량만 크고, npm install로 재설치됨
- .env - 비밀번호, API 키, 민감한 정보
- .DS_Store - 맥 시스템 파일
- *.log - 로그 파일들

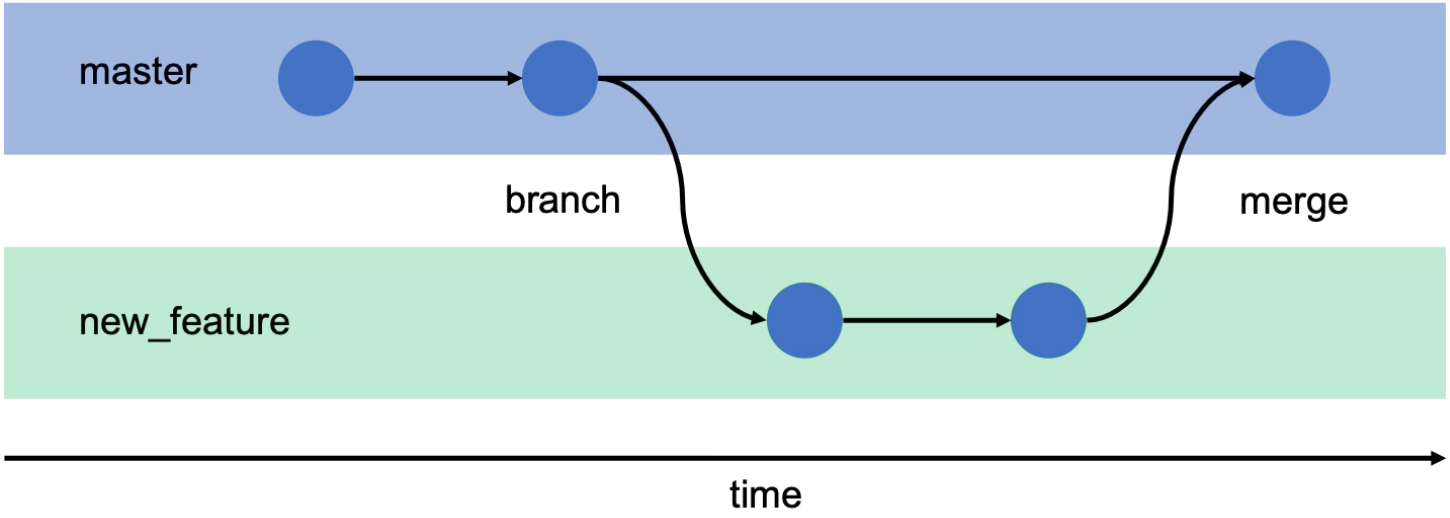
용어(3) - 브랜치

- 로그인 기능은 잘 된다
- 이제 "결제 기능" 만들려는데, 이거 복잡함
- AI한테 시켰다가 망해서 로그인까지 깨지면 어떡하지?

브랜치 = 안전한 실험 공간

- main 브랜치: 지금 잘 되는 코드
- feature/payment 브랜치: 결제 기능 작업 중
- 브랜치에서 망해도 main은 그대로.
- 잘 되면 main에 합친다.

"큰 작업 시키기 전에 브랜치부터"


master
branch
merge
new_feature
time

명령(7) - git branch

- 브랜치 목록 보기: git branch → 현재 브랜치에 * 표시됨
- 새 브랜치 만들기: git branch feature/payment
- 브랜치 이동: git checkout feature/payment
- 만들면서 바로 이동 : git checkout -b feature/payment
- 브랜치 삭제 (머지 후): git branch -d feature/payment

커서에서 브랜치를 확인하려면

좌측 하단에 현재 브랜치 이름이 표시된다

- "main"이면 main 브랜치에 있는 것
- 클릭하면 브랜치 목록, 전환, 새로 만들기 가능

Source Control 패널에서도 확인 가능

- 상단에 현재 브랜치 표시
- 브랜치 아이콘으로 전환

AI한테 브랜치 작업 시킬 때는 브랜치 이름을 정확히 말해줘야 한다.

명령(8) - git merge

브랜치에서 결제 기능 완성 + 테스트도 통과

이제 main에 합치고 싶다

- git checkout main # main으로 이동
- git merge feature/payment # 합치기

합친 후:

- main에 결제 기능이 들어옴
- 기능 개발이 모두 완료되면 브랜치는 삭제해도 됨

질문 3. 브랜치 작업

“결제 기능 작업 전에 브랜치를 새로 만들거야. 어떤 이름이 적절할까?”

“그 이름으로 브랜치 만들고 거기서 작업해줘. main은 건드리지 마.”

“이 브랜치 작업 다 버리고 main으로 돌아가줘.”

AI의 git 활용(1)

AI가 여러 파일 건드렸는데 뭐 한 건지 모르겠다면?

“마지막 커밋 이후로 뭐가 바뀌었는지 정리해줘”

AI가 해주는 것:

- git diff 실행
- 파일별로 뭐가 바뀌었는지 요약
- 코드 안 봐도 "로그인 검증 로직 추가, 에러 메시지 수정" 같이 설명
- 터미널 출력 직접 읽는 것보다 훨씬 편함.

AI의 git 활용(2)

커밋하려는데 뭐라고 써야 할지 모르겠다

- “변경사항 보고 커밋 메시지 작성해서 커밋해줘”

AI가 해주는 것:

- 변경 내용 분석
- 적절한 커밋 메시지 생성
- "SC-002 이메일 유효성 검증 추가" 같은 형태

더 좋은 방법:

- “시나리오 SC-002 작업 완료했어. 커밋 메시지에 시나리오 번호 넣어서 커밋해줘.”
- 로드맵 시나리오 번호 = 커밋 메시지 → 나중에 추적 가능

AI의 git 활용(3)

이 프로젝트에 맞는 .gitignore 만들어줘.

- 프로젝트 구조 확인 (Next.js? Python? 등)
- 해당 환경에 맞는 .gitignore 생성
- node_modules, .env, 빌드 결과물 등 자동 포함

이미 실수로 올렸으면:

- “node_modules가 GitHub에 올라가버렸어.
  gitignore에 추가하고 GitHub에서도 제거해줘.”
- 단 한 번 올라간 건 gitignore만 추가해선 안 지워진다.
- 직접 삭제 명령 필요.

AI의 git 활용(4)

커밋 전에 변경 취소하고 싶을 때:

- 방금 수정한 거 다 취소하고 마지막 커밋 상태로 돌아가줘.

특정 커밋으로 돌아가고 싶을 때:

- git log 보여주고, 어제 커밋으로 되돌리고 싶어.

푸시까지 해버린 걸 되돌릴 때:

- 방금 푸시한 커밋 취소하고 싶어. 어떻게 해야 해?

주의할 점

- 이미 푸시한 건 되돌리기 복잡함
- 그래서 "푸시 전에 확인"이 중요
- 브랜치에서 작업하면 이런 상황 자체를 줄일 수 있음

오늘 해결해야 할 문제

1. 현재 프로젝트를 Git 리포지토리로 만들고 첫 커밋을 완료합니다.
2. 시나리오 작업 후 변경사항을 확인하고 커밋합니다.
3. 커밋한 내용을 GitHub에 푸시해서 백업합니다.
4. 새 기능 작업 전에 브랜치를 만들어서 안전하게 진행합니다.
5. 완성된 브랜치를 main에 머지합니다.

Q&A