4차시. 이력 수정과 배제: amend, revert, reset과 .gitignore

Day 1 · 13:00-13:45 (수업 45분) / 13:45-14:00 (휴식 15분)

점심시간 완료: 12:00-13:00 휴식 후 오후 첫 수업

저장소: `portfolio-web`

선수 학습: 3차시 커밋 생성 및 이력 조회

이 차시에서 하는 일

- 방금 작성한 커밋의 메시지 오타나 누락된 파일을 `git commit --amend`로 안전하게 덮어쓴다.
- 이미 외부에 공유된 커밋을 취소할 때 안전한 `git revert`와, 로컬에서 포인터를 되돌리는 `git reset`의 3가지 모드 차이를 이해한다.
- `.gitignore` 파일을 작성해 운영체제 임시 파일, 비밀 키, 캐시 파일이 Git에 기록되지 않도록 완벽히 배제한다.
- 이미 추적 중인 파일이 생겼을 때 캐시를 해제하는 `git rm --cached`의 역할을 안다.

이력 수정과 불필요한 파일 배제가 필요한 이유

개발을 하다 보면 누구나 사소한 실수를 합니다. 방금 커밋을 마쳤는데 커밋 메시지에 오타가 발견되거나, 깜빡하고 함께 묶었어야 할 스타일 파일 하나를 빠뜨렸을 때가 있습니다.

이를 매번 새로운 커밋("오타 수정", "파일 누락 추가")으로 남기면 프로젝트 이력이 지저분해집니다. 원격 저장소에 올리기 전인 로컬 커밋이라면, 직전 커밋을 깔끔하게 하나로 합쳐서 수정하는 것이 좋습니다.

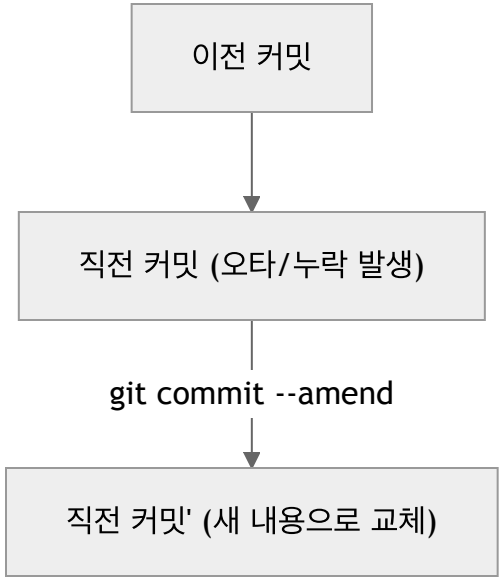
또한 프로젝트 폴더 안에는 Git이 절대 기록해서는 안 되는 파일들이 생깁니다:

- **보안 파일**: 비밀번호, 데이터베이스 접속 정보, API 키가 담긴 파일 (`.env` 등).
- **운영체제 임시 파일**: macOS의 `.DS_Store`, Windows의 `Thumbs.db` 등.
- **편집기/빌드 캐시 파일**: 자동 생성되는 로그나 임시 파일 (`*.tmp`, `*.log`).

이러한 파일들이 한 번이라도 커밋되면 저장소 용량이 불필요하게 커지고 무엇보다 심각한 보안 사고가 발생할 수 있습니다. 따라서 Git이 알아서 무시하도록 사전 차단막을 설치해야 합니다.

직전 커밋 교체와 되돌리기 삼총사

Git에서 과거 이력을 다루는 방법은 목적에 따라 명확히 나뉩니다:



1. **`git commit --amend` (직전 커밋 덮어쓰기)**
   - 가장 최근에 만든 커밋의 메시지를 바꾸거나, 새로 수정한 파일을 직전 커밋에 묶어 넣습니다.
   - 기존 커밋을 버리고 새로운 해시를 가진 커밋으로 교체합니다.
   - **주의**: 아직 GitHub 원격 저장소에 push하지 않은 로컬 커밋에만 사용해야 합니다.

직전 커밋 교체와 되돌리기 삼총사 · 계속

2. **`git revert` (안전한 되돌리기 커밋 추가)**
   - 과거 커밋을 지우는 대신, "해당 커밋에서 변경했던 내용을 정반대로 되돌리는 새로운 커밋"을 만듭니다.
   - 이력이 삭제되지 않고 누적되므로, 이미 원격에 공유된 커밋을 안전하게 취소할 때 사용하는 유일한 표준 방법입니다.
3. **`git reset` (로컬 포인터 되돌리기)**
   - 시간을 뒤로 감아 과거 커밋 시점으로 이동합니다. 3가지 모드가 있습니다:
     - `--soft`: 커밋만 취소하고, 수정했던 파일들은 스테이징 영역에 그대로 보존합니다.
     - `--mixed` (기본값): 커밋과 스테이징을 취소하고, 작업 공간에 파일 변경 상태를 남겨둡니다.
     - `--hard`: **주의!** 작업 공간의 파일 변경 내용까지 전부 강제로 지워버립니다.
     - **경고**: `--hard`는 미처 커밋하지 못한 소스코드를 영구적으로 날릴 위험이 있으므로 수업에서는 절대 직접 실행하지 않습니다.

.gitignore의 동작 원리와 작성 규칙

저장소 루트에 `.gitignore`라는 이름의 텍스트 파일을 두면, Git은 그 안에 적힌 패턴과 일치하는 파일이나 폴더를 없는 것처럼 취급합니다.

Git 실습 실행 결과

```
$ git log --oneline -1
39d281c chore: ignore local environment

$ git check-ignore .env
.env

$ git status --short
?? README.md
```

격리된 실습 저장소의 실제 실행 출력 (원격 배포 화면 아님)

[전체 예제 코드 · ignore-rules-1.gitignore]

.gitignore의 동작 원리와 작성 규칙 · 계속

- #으로 시작하는 줄은 주석입니다.
- 파일명(`filename.ext`)이나 확장자 패턴(`*.ext`), 폴더명(`folder/`)을 지정합니다.
- **핵심 주의점**: `.gitignore`는 **아직 Git이 추적하지 않은 Untracked 파일**에만 작동합니다. 이미 과거에 `git add`나 커밋으로 추적 대상이 된 파일은 `.gitignore`에 추가해도 계속 추적됩니다. 이미 추적된 파일을 제외하려면 `git rm --cached <파일명>`으로 인덱스 캐시를 먼저 제거해야 합니다.

따라하기

1. 직전 커밋 메시지에 오타가 있었다고 가정하고 amend로 메시지를 수정합니다: `git commit --amend -m "docs: 첫 화면 학습 목표 섹션 및 안내 문구 보완"`
2. `git log --oneline`으로 커밋 해시가 새로 갱신되고 메시지가 바뀐 것을 확인합니다.
3. 가짜 비밀 파일인 `.env`를 생성해 봅니다: `echo "API_KEY=sample_secret_key_12345" > .env`
4. `git status`를 실행하여 `.env` 파일이 빨간색 Untracked 파일로 나타나는 것을 확인합니다.

따라하기 · 계속

5. VS Code에서 새 파일 `.gitignore`를 만들고 위의 무시 규칙들을 입력한 뒤 저장합니다.
6. `git status`를 다시 실행합니다. 놀랍게도 방금 보였던 `.env`가 사라지고, 새로 만든 `.gitignore`만 목록에 나타납니다.
7. `.gitignore` 파일을 스테이징 영역에 올리고 커밋합니다: `git add .gitignore` `git commit -m "chore: .gitignore 파일 추가"`
8. `git status`로 모든 작업 트리가 깨끗해진 것을 확인합니다.

Git 명령

```
git commit --amend -m "docs: 첫 화면 학습 목표 섹션 및 안내 문구 보완"
git log --oneline
echo "API_KEY=sample_secret_key_12345" > .env
```

Git 명령 · 계속

Git 실습 실행 결과

```
$ git log --oneline -1
7d7b2d3 chore: ignore local environment

$ git check-ignore .env
.env

$ git status --short
?? README.md
```

격리된 실습 저장소의 실제 실행 출력 (원격 배포 화면 아님)

Git 명령 · 계속

[전체 예제 코드 · git-commands-1.sh]

선택 참고 명령

```
git rm --cached .env
git revert HEAD
```

에이전트 요청 예시

> .gitignore에 임시 파일과 캐시 파일 추가
> amend와 revert 차이 초보자 눈높이 설명
> reset 3가지 모드 soft mixed hard 차이 비교 정리

확인법

- ☐ `git log --oneline`에서 직전 커밋의 메시지가 수정되어 있다.
- ☐ 폴더 안에 `.env` 파일이 실제로 존재하지만 `git status`에 감지되지 않는다.
- ☐ `.gitignore` 파일 자체가 커밋 이력에 성공적으로 기록되어 있다.
- ☐ `reset --hard`가 왜 위험한지 그 이유를 설명할 수 있다.

오류 복구

증상	원인	대응
`.gitignore`에 적었는데도 파일이 계속 잡힘	이전에 이미 `git add` 또는 커밋되어 추적 중임	`git rm --cached <파일명>`을 실행해 추적 캐시를 해제한 뒤 다시 커밋한다.
원격에 push한 후 `amend`를 실행함	원격 이력과 로컬 이력의 불일치	이미 원격에 올라간 커밋은 amend하지 말고, 변경 사항을 새 커밋으로 작성하거나 `revert`를 사용한다.
실수로 `git reset --hard`를 생각함	작업 파일 유실 위험	절대 실행하지 말고, 파일 복원이 필요하면 강사에게 문의한다.

안전 주의

- `reset --hard` 명령은 작업 중이던 모든 파일의 변경을 영구히 지워버리므로 절대 임의로 실행하지 않습니다.
- 비밀번호, API 키, 인증 토큰은 절대로 커밋에 포함되어서는 안 되며, 반드시 `.gitignore`로 사전에 보호합니다.
- 상세한 Git 옵션과 명령어는 부록 2. Git 명령과 옵션 일람을 참조합니다.

안전 주의 · 계속

- 추가 심화 실습은 보충안 심화, 복습 퀴즈는 보충안 퀴즈, 에이전트 주도 Git은 보충안 에이전트에서 다룹니다.