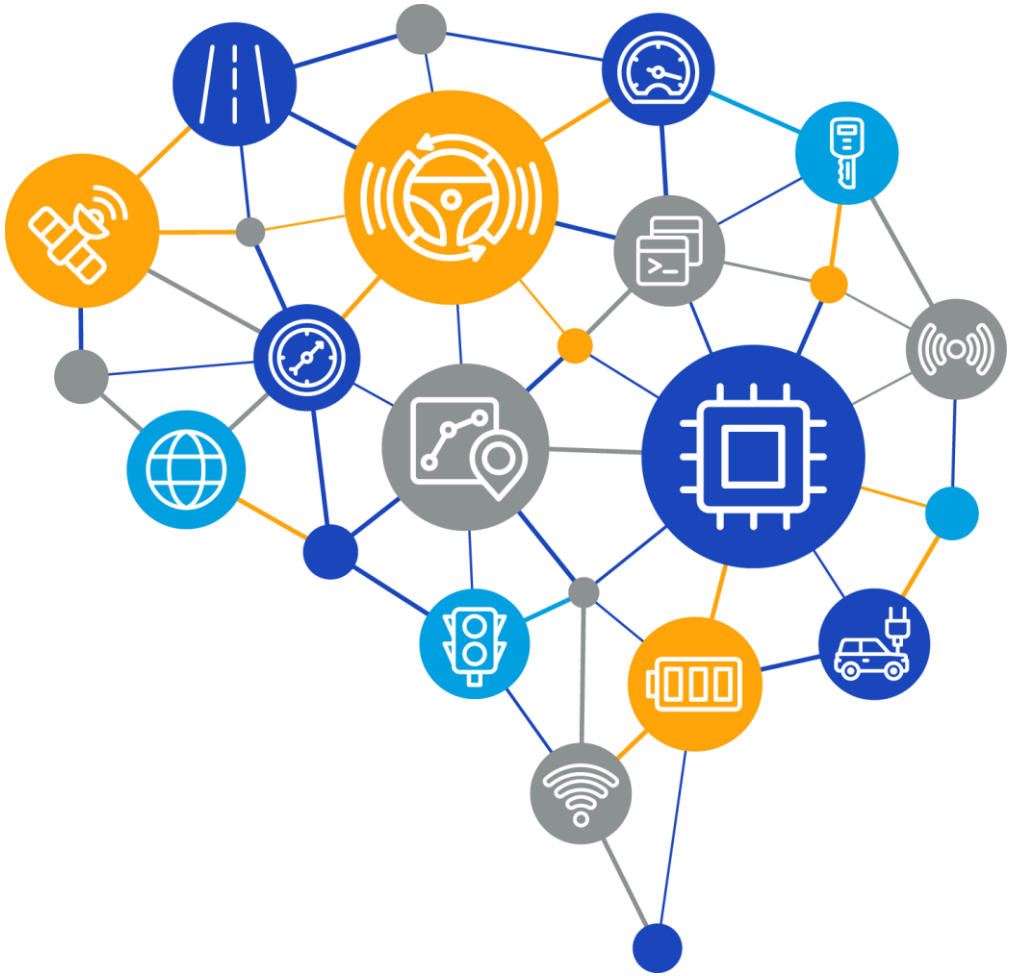
깃 설정과 기본 명령어 정리

Version Control and
Git & Github basics



강환수 교수

DMUAi 동양미래대학교
인공지능소프트웨어학과

AI Experts Who Lead
The Future

AI Experts
Who Lead
The Future

01

깃 설정 명령어

지역 저장소 생성 후 작업 시 다음 설정 반드시 확인

- 맥(lf)과 윈도(crlf) 간의 자동 변환
  - git config --global core.autocrlf true
- 사용자 설정
  - $ git config --global user.name "John Doe"
  - $ git config --global user.email johndoe@example.com
- 뉴라인 경고 발생 없애기(옵션)
  - git config --global core.safecrlf false

편집기 설정

- git config core.editor code
- git config core.editor “code --wait”
- git config core.editor notepad
- git config --global core.editor code

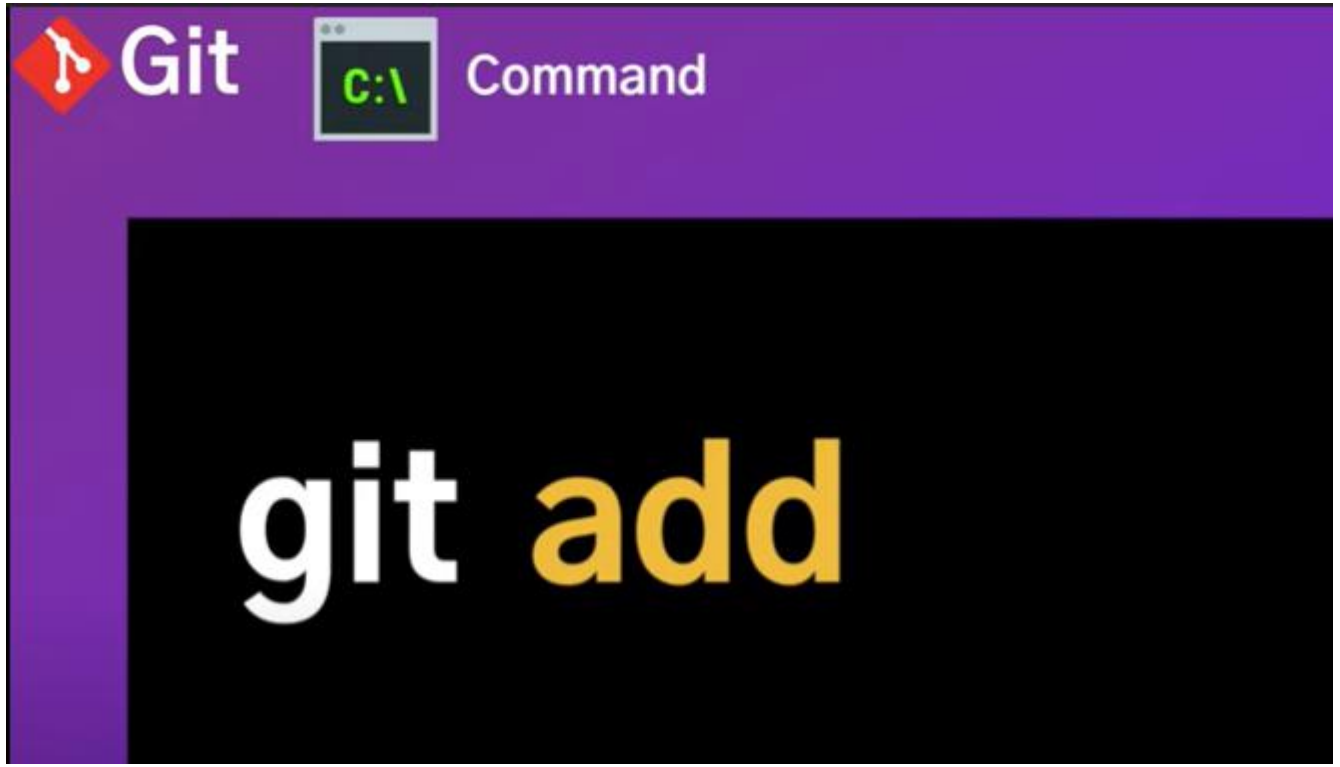
AI Experts
Who Lead
The Future

02

깃 저장소 생성과 삭제

명령어 구조

- 맨 앞에 항상 git
  - $ git add
  - $ git commit



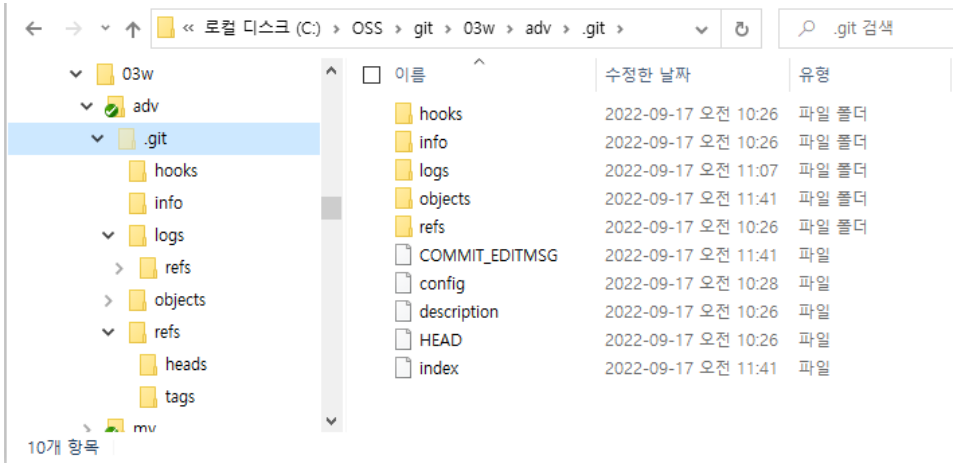
git init

- "git init" 명령어 실행
  - 프로젝트(소스코드들이 있는 디렉토리)를 git repository로 만들기 위해서 사용
  - 디렉토리를 git repository로 만들어야 git으로 버전 관리가 가능

저장소 생성

- **$ git init [project-name]**
  - 현재 하부 폴더에 새로운 저장소 [project-name]를 생성
  - [project-name] 없다면, 현재 폴더를 저장소로 생성, 다음과같은 의미
    - $ git init .

- **새로운 로컬 저장소를 생성하고 이름을 정합니다.**
  - Initialized empty Git repository in [저장소전체경로]/.git/
    - .git 폴더는 복잡



환경 설정

- **Git을 사용한다면 설치 후 꼭 미리 설정해두기를 권장**
  - 커밋을 할 때 사용할 이름과 이메일을 지정
    - 저장소에 변경 사항을 추가하는 Commit 작업을 할 때 누구의 작업인지를 기록하는 것이 매우 중요
  - GitHub의 사용자를 연결할 때도 사용

- **저장소로 이동 환경설정**

  - 현재 로컬 저장소에 적용할 사용자 정보를 설정합니다
  - $ git config user.name "[name]"
    - 자신이 생성한 커밋(commit)에 들어갈 이름을 설정합니다
  - $ git config user.email "[email address]"
    - 자신이 생성한 커밋에 들어갈 이메일 주소를 설정합니다

지역 저장소 설정파일

- [지역저장소]/.git/config 파일

```
PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/02w/basic (main)
$ cat .git/config
[core]
        repositoryformatversion = 0
        filemode = false
        bare = false
        logallrefupdates = true
        symlinks = false
        ignorecase = true
[user]
        name = hskang
        email = hskang
```

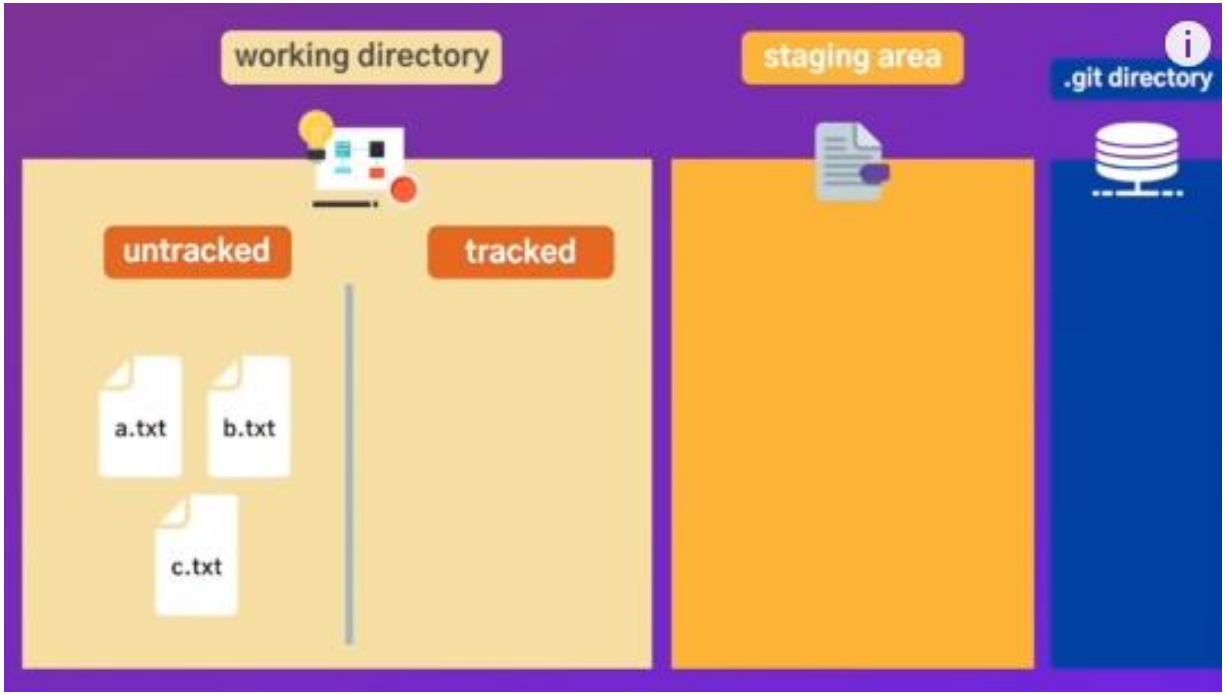
AI Experts
Who Lead
The Future

03

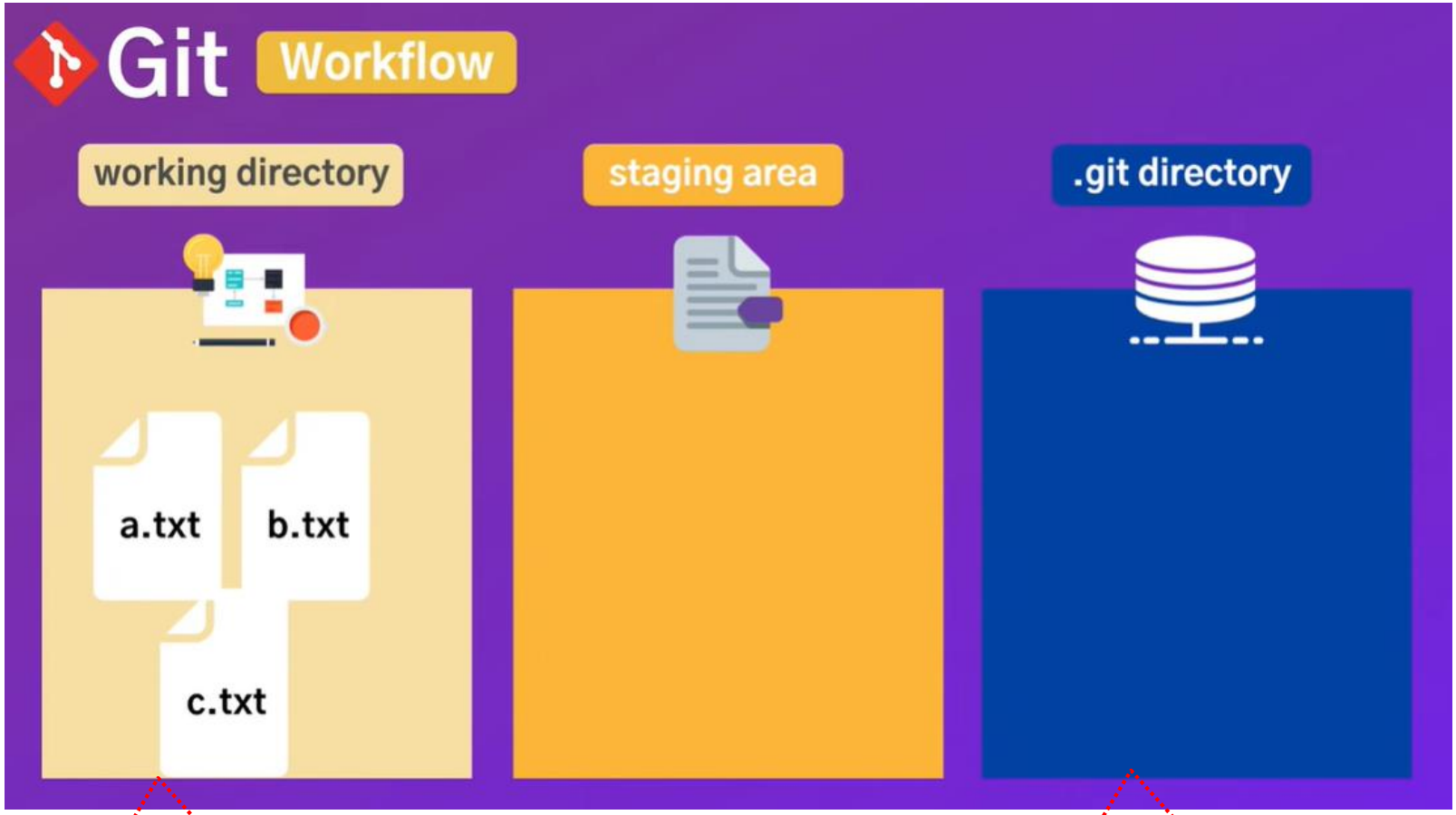
깃 커밋 명령어

관리 파일

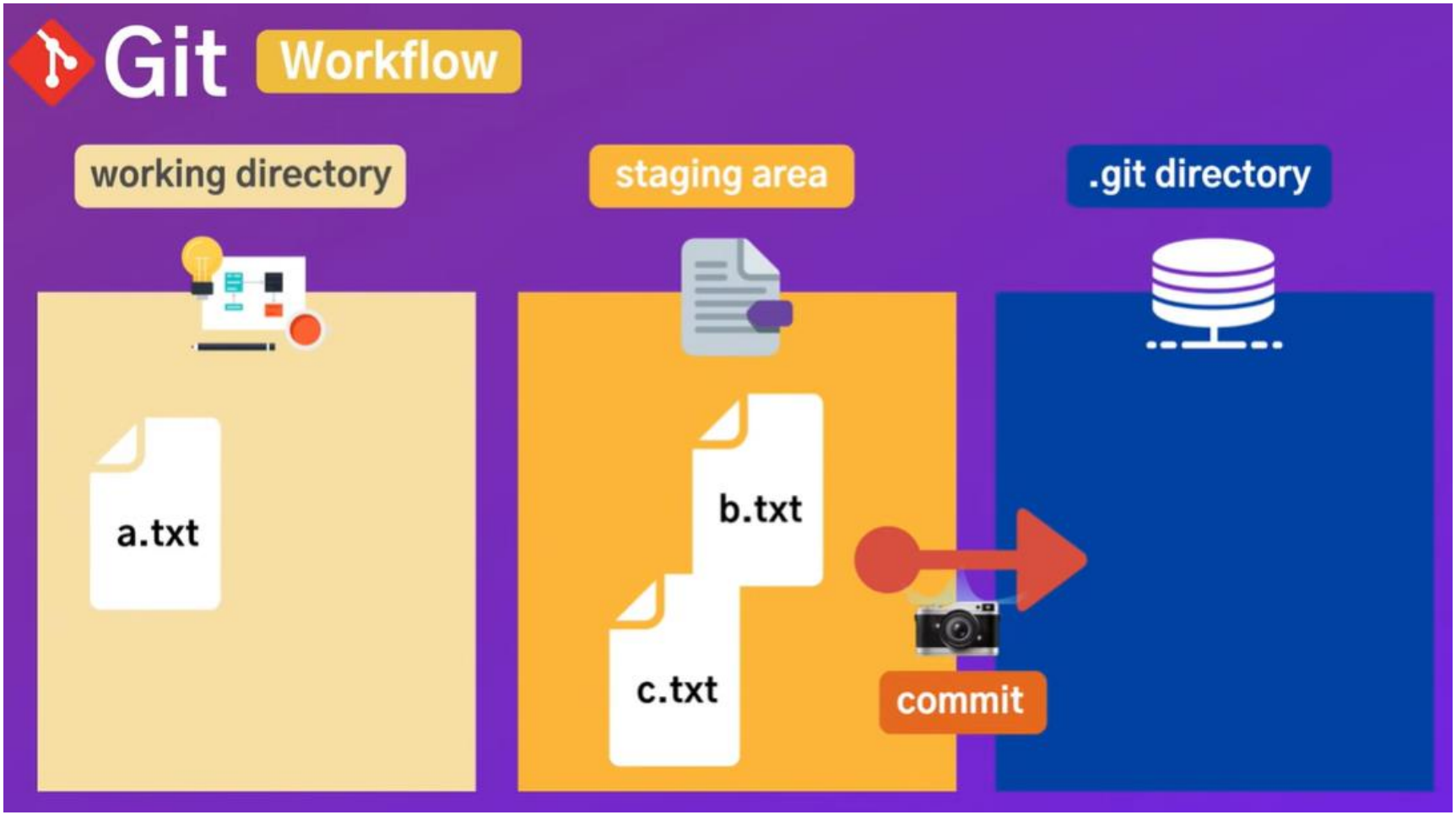
- **Untracked vs tracked**
  - 깃의 관리 대상 인지 아닌지
    - 형상(버전) 관리의 대상인지 아닌지
  - Untracked
    - 처음 생성만 해 놓은 파일
    - Tracked로 이동
      - $ git add [file]



세 가지 flow 상태

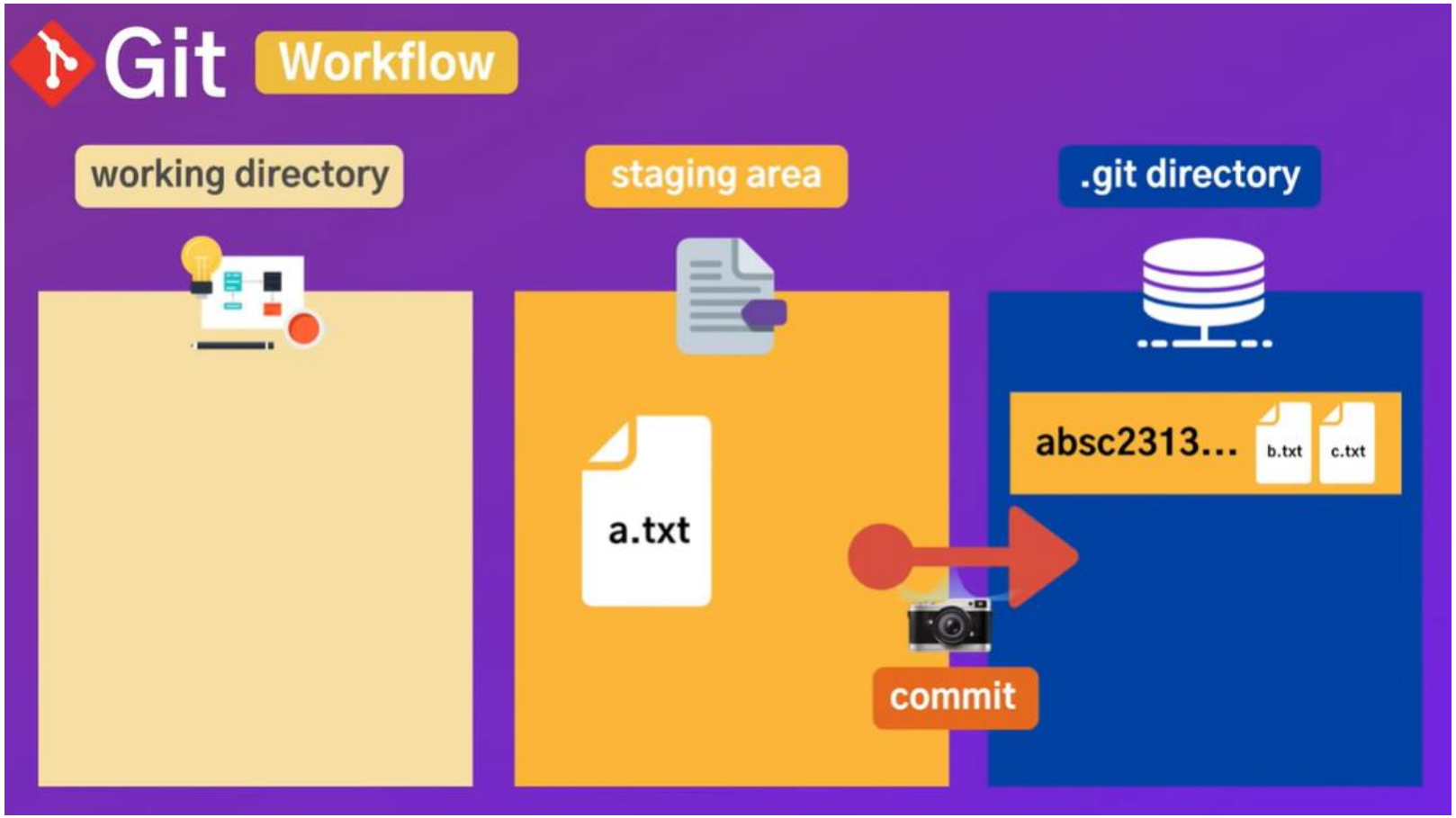

Git Workflow
working directory
staging area
.git directory
a.txt
b.txt
c.txt
저장소 폴더
.git 폴더

명령어 add commit



커밋 결과

- 버전 관리를 위한 git 디렉토리에 반영



파일의 이동

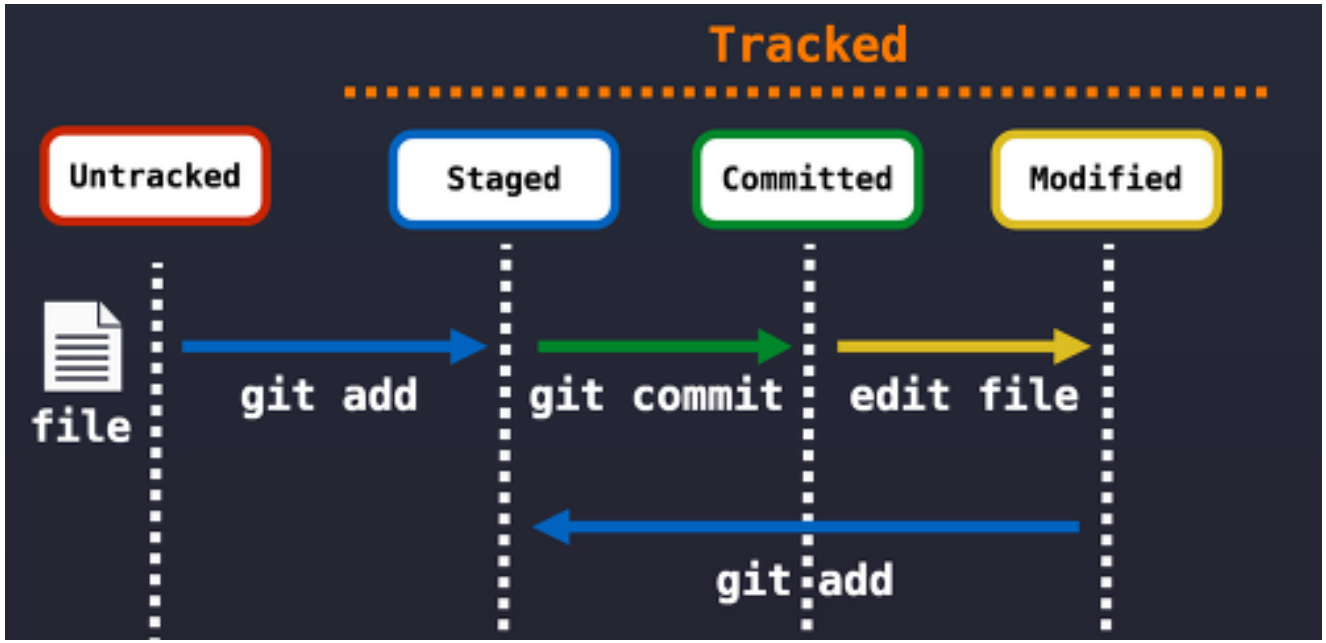

Working Directory
Staging Area
Git Repository
Add command
Commit command
Push command
Untracked File
Tracked File
Committed File
Remote File

git log, git show

- **Git log [head]**
  - commit history 확인
    - 마지막 커밋인 head부터 이전 모든 이력 로그 표시
    - Author 영역의 이름과 이메일 주소
    - git config 명령을 통해 세팅했던 user.name / user.email 값이 표기

- **Git show [head]**
  - commit 자세한 정보 확인
    - 지정한 커밋인 head의 정보 확인
    - Author 영역의 이름과 이메일 주소
    - git config 명령을 통해 세팅했던 user.name / user.email 값이 표기
    - 파일의 변화(patch)도 표시

Untracked vs Tracked



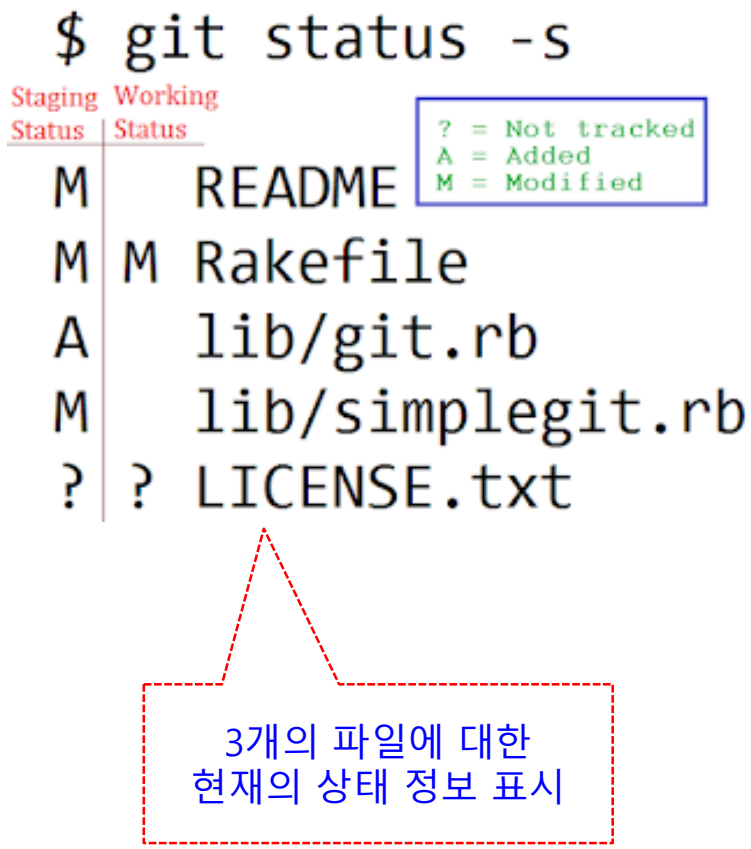
AI Experts
Who Lead
The Future

03

깃 저장소 간략한 상태 정보

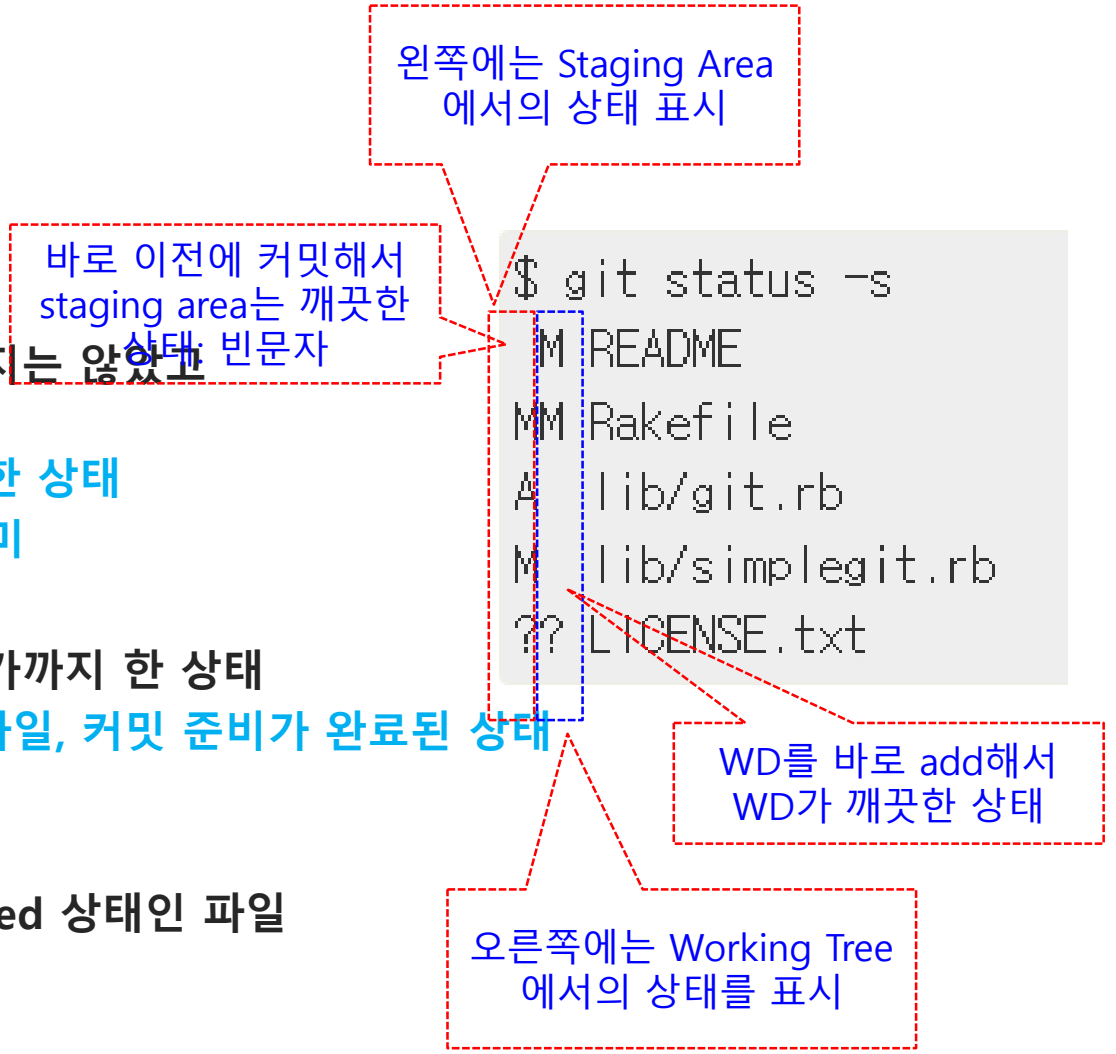
간략한 상태 정보

- **$ git status -s**
  - 옵션 --short [file]
    - 파일 상태를 짤막하게 확인하기
    - File 있으면 file에 대한 것만 표시, 없으면 모든 파일 표시



$ git status -s

- ??
  - 아직 추적하지 않는(untracked) 새 파일 앞에 표시
- A□ 표시
  - 새로 생성해 add해 Staged된 파일 의미
- M□
  - 수정한 파일을 add한 상태
- □M README
  - 커밋한 이후에 아직 Staged 상태로 추가하지는 않았고 내용은 수정되었다는 의미
    - Add 한 이후에 파일 수정하고, 커밋한 상태
    - 결국 add, commit이 필요하다는 의미
- M□ lib/simplegit.rb
  - 파일은 내용을 변경하고 Staged 상태로 추가까지 한 상태
    - 최초 생성이 아닌 파일로서 수정한 파일, 커밋 준비가 완료된 상태
- MM Rakefile
  - 변경하고 Staged 상태로 추가한 후 또 내용을 변경해서 Staged 이면서 Unstaged 상태인 파일



주요 상태의 변화

- ?? => A□
  - Untracked에서 SA로 add된 상태의 변화
- □M => M□
  - WD에서 수정된 파일을 add한 상태의 변화
- □□ => □M
  - 커밋한 이후에 파일 수정 시 상태의 변화
- M□ => MM
  - Add한 이후에 다시 파일 수정한 상태의 변화
- MM => □M
  - MM에서 커밋한 상태의 변화

AI Experts
Who Lead
The Future

04

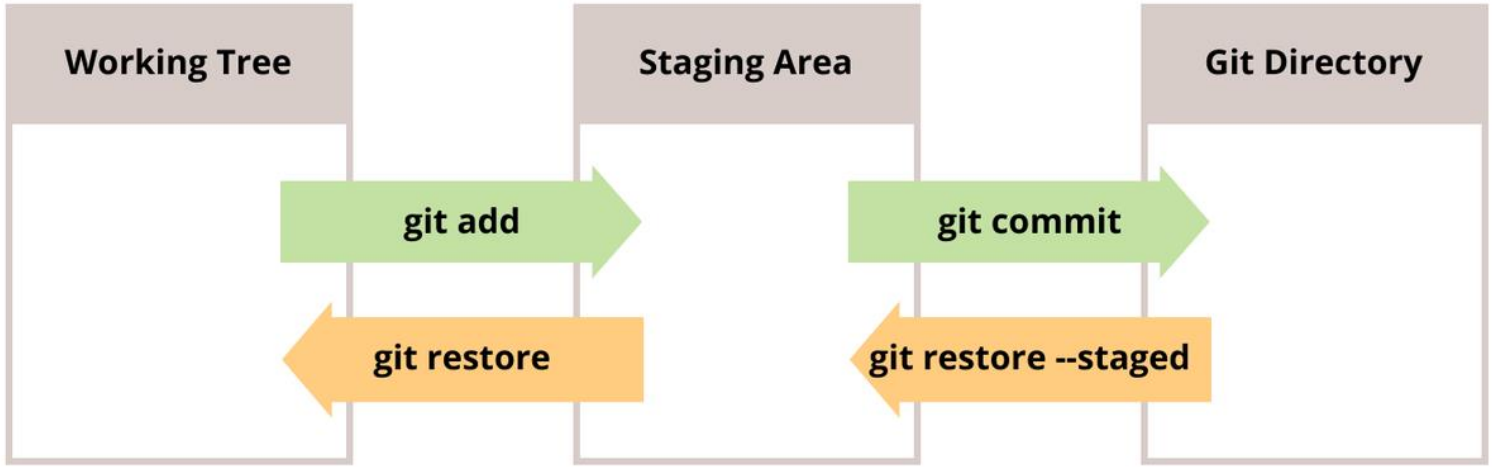
깃 저장소 파일 수정과 staging 취소

파일의 수정 후 취소와 파일의 add 후 취소: 깃 버전 2.23부터

- 스테이지 파일의 수정 후 취소
  - 수정 후
    - Modified
  - WD의 수정을 취소(결국 SA의 내용으로 수정) 하려면
    - $ git restore fname

https://bufferings.hatenablog.com/entry/2020/05/01/013451

- 스테이지 파일로 add한 것을 취소
  - Add 후
  - SA로 올린 것을 취소(HEAD에 있는 내용으로 수정) 하려면
    - $ git restore --staged fname



AI Experts
Who Lead
The Future

05

작업 공간과 staging 영역에서 파일 삭제

작업공간과 스테이징 영역에서 파일 삭제

- **$ git rm fname**
  - 작업공간과 스테이징 영역에서 모두 파일 삭제
    - 즉 파일을 작업영역에서 제거하고 그 제거한 상태를 index에 반영(add)
      - rm + git add

- **$ git rm --cached fname**
  - 작업공간에는 남겨두고 스테이징 영역에서만 파일 삭제
  - 파일 fname이 untracked 됨

강제 삭제 옵션 -f

- 만약 워킹디렉터리에서 파일을 수정한 직후 커밋하기 전에 수정하는 경우 다음의 두 에러 발생
  - -f 옵션을 추가하여 git rm -f 명령을 통해 강제로 삭제

- 워킹 디렉터리에서 파일 수정 직후 삭제하려는 경우 오류 내용
  - $ git rm testFile
    - error: the following file has local modifications: testFile

- 워킹 디렉터리에서 파일 수정 후 git add 하고 커밋 직전에 삭제하려는 경우 오류 내용
  - $ git rm testFile
    - error: the following file has changes staged in the index: testFile

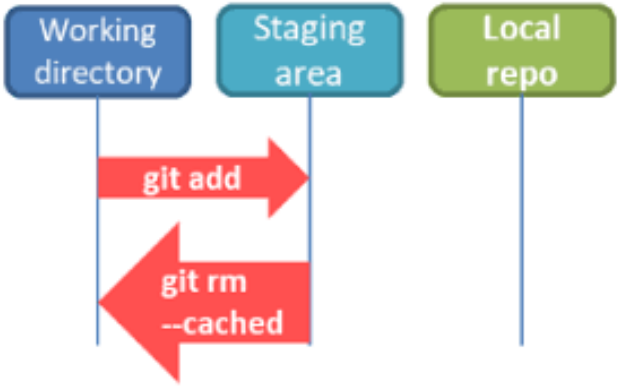
  - 원인은 변경 내역을 커밋하기 전에 삭제하려 했기 때문
  - Git은 안전장치로 실수로 인해 데이터를 삭제하지 못하도록 해 두었기 때문

- 따라서 이때에는 위에서 말했듯이 git rm -f 명령을 통해 파일을 삭제

Add를 취소하는데 사용 가능

- **$ git rm --cached fname**

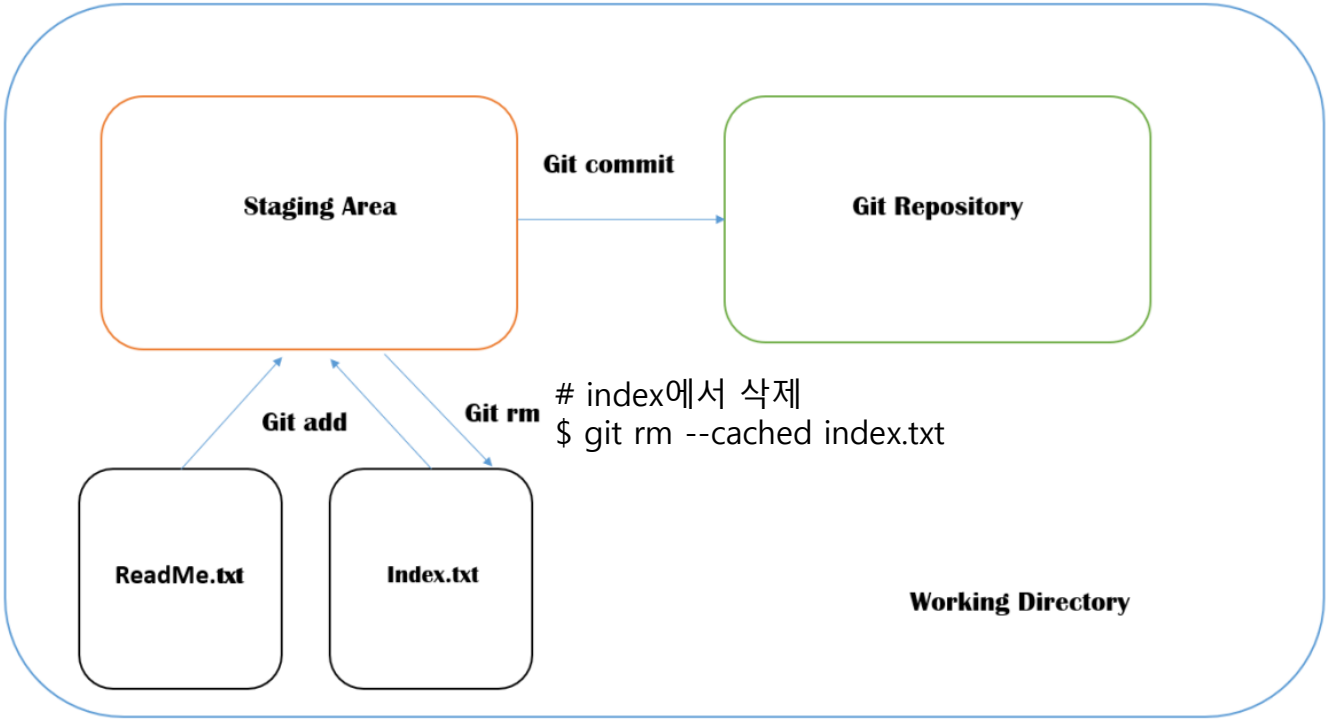
```
PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/exec/soo (main)
$ echo A > hel

PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/exec/soo (main)
$ git add hel

PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/exec/soo (main)
$ git st
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   hel

PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/exec/soo (main)
$ git rm --cached hel
rm 'hel'

PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/exec/soo (main)
$ git st
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hel

nothing added to commit but untracked files present (use "git add" to track)
```



$ git rm --cached fname 사례

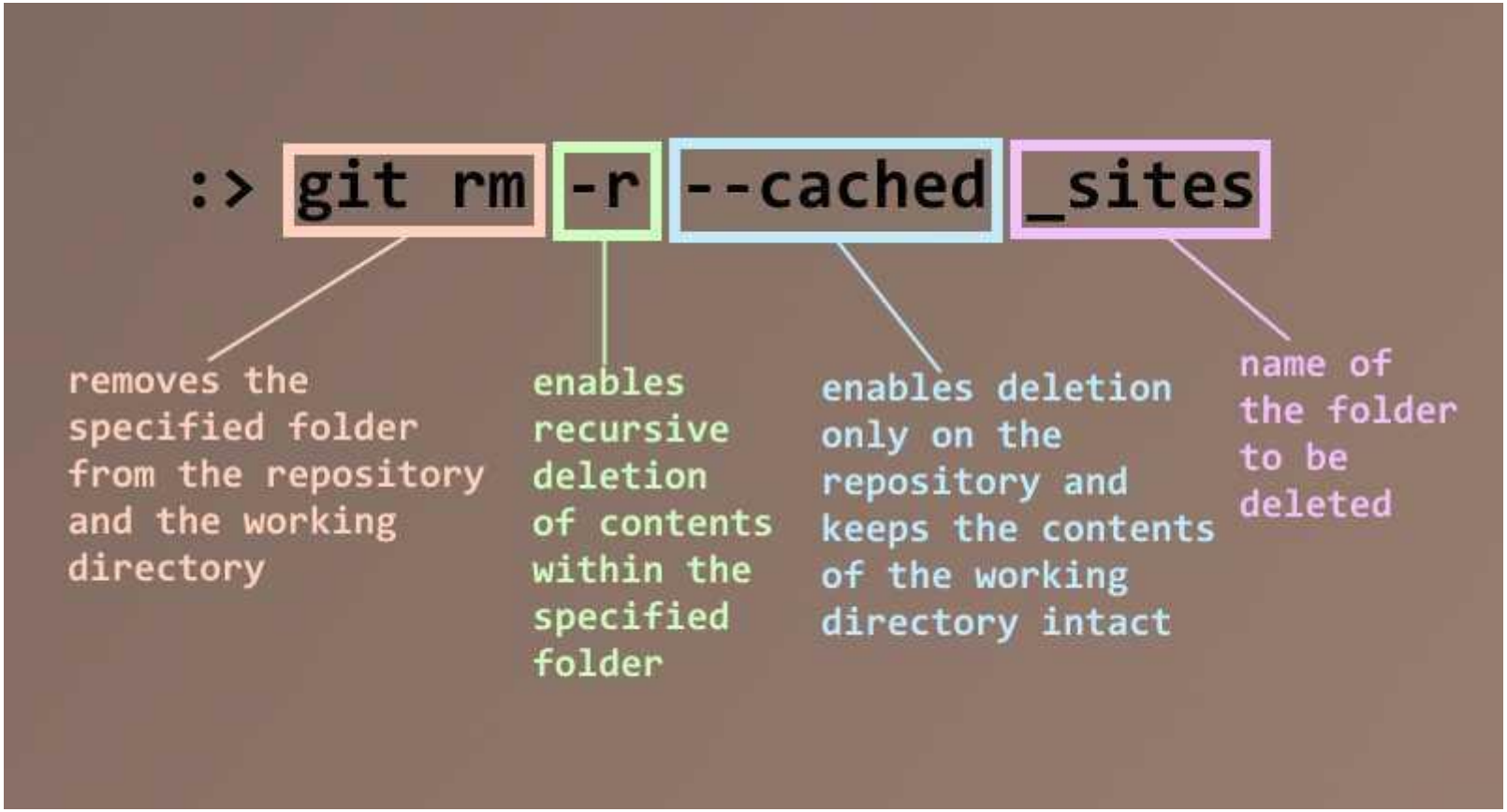
MINGW64:/c/GitTest

```
                    MINGW64 /c/GitTest (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Index.txt
        new file:   ReadMe.txt


                    MINGW64 /c/GitTest (master)
$ git rm --cached Index.txt
rm 'Index.txt'

                    MINGW64 /c/GitTest (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   ReadMe.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Index.txt
```

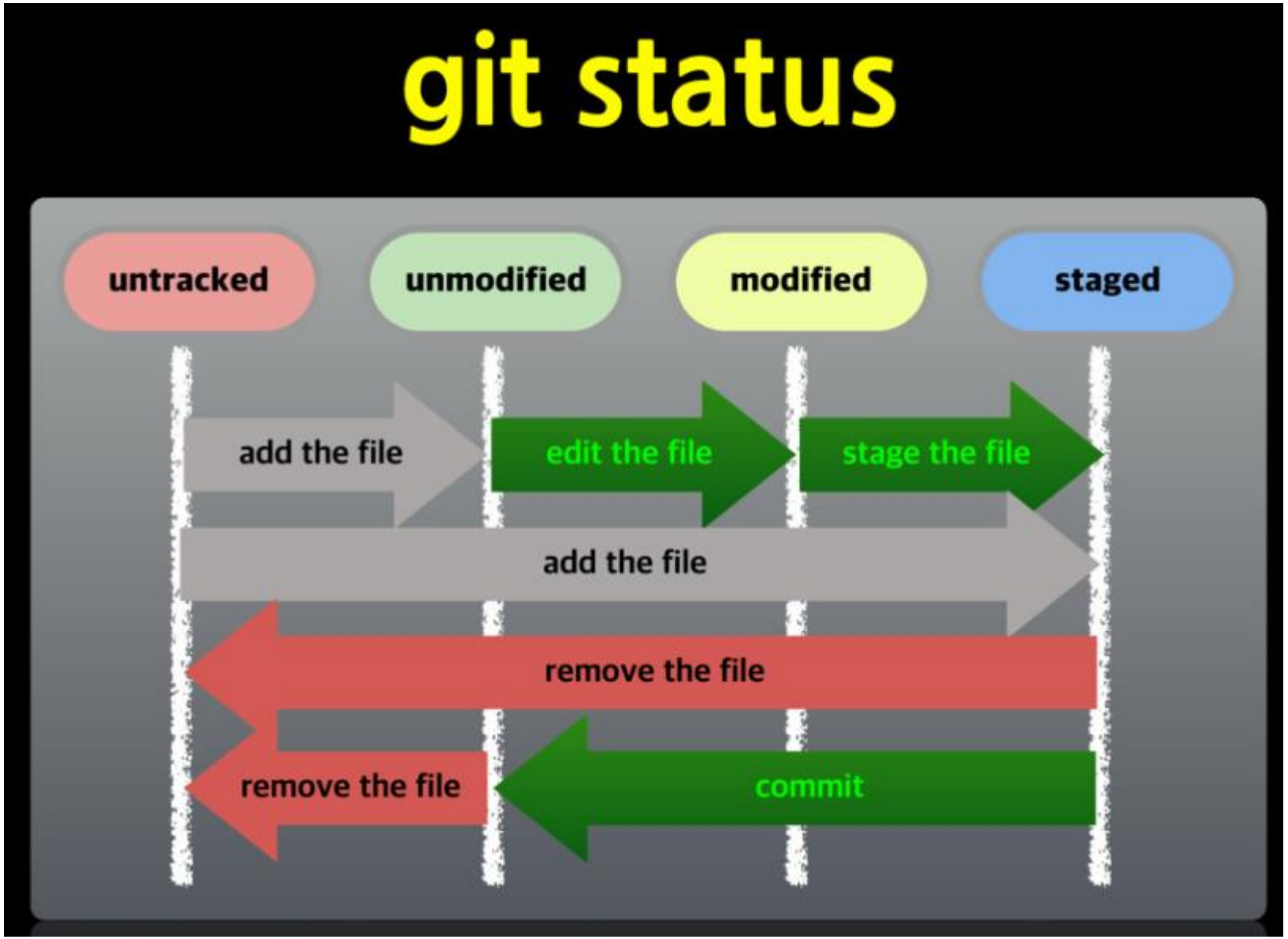


폴더와 모든 파일 삭제

- 스테이징 영역만 삭제, 작업공간에는 남음



깃 상태 변화



https://speakerdeck.com/rorlab/git-iyagi?slide=17