



week2



<버전관리 개요>

- 버전관리의 필요성과 개념
- 버전관리 도구의 인터페이스 방식
- 커밋과 버전관리
- 원격 저장소와 지역저장소
- Clone, push, pull, add, commit

=====

<깃과 깃허브 개요>

- 깃 개요와 기능
- 깃 설치 : 두 개의 SW 제공 Git Bash, Git GUI
- 깃의 내부 저장소 구조와 이동 명령
- 브랜치 이해
- 깃허브 이해

버전관리 개념 : 파일의 추가 및 수정 이력(추적) 관리



인터페이스 방식

-CLI : Command Line Interface(명령어 줄 인터페이스)

-Git Bash : 텍스트로 명령을 입력하고 결과도 텍스트로 표시되는 방식

ex) 윈도의 prompt 나 유닉스의 shell이 있음

```

PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]
$ git init repo
Initialized empty Git repository in C:/[smart Git]/repo/.git/
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]
$ cd repo
  
```

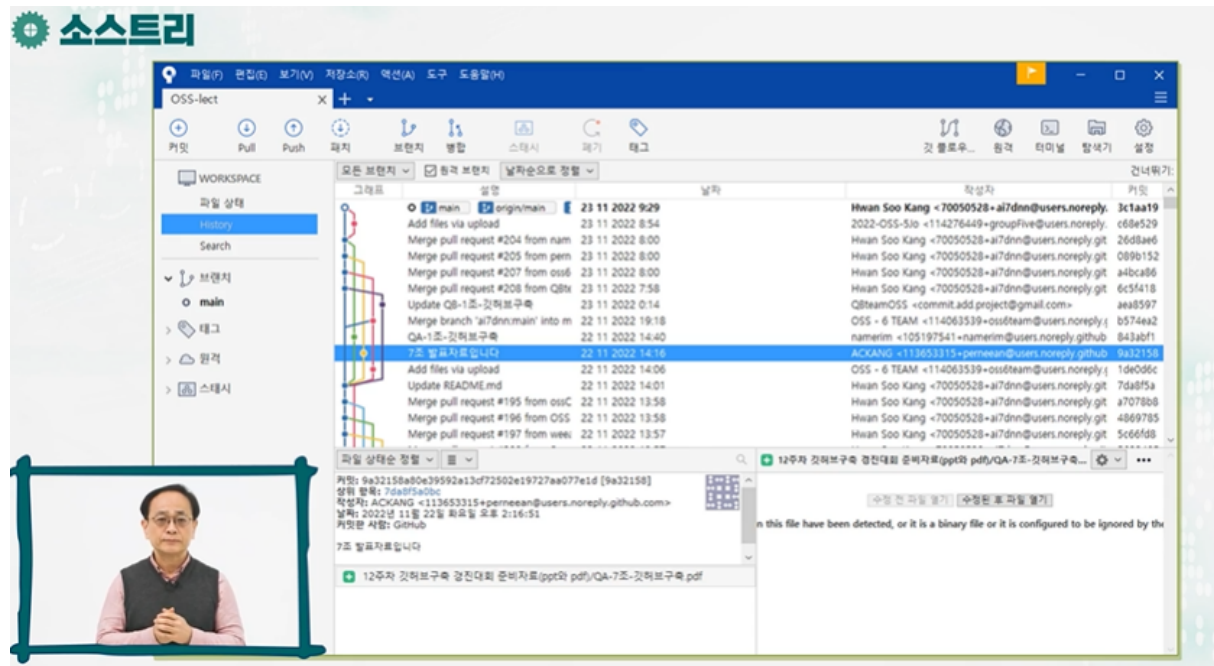
프롬프트
명령어
명령에 대한 결과가 표시

```

PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/repo (main)
$ git config --global core.autocrlf true
  
```

현재 작업 폴더
현재 작업 저장소의 브랜치 이름

-GUI : Graphic User Interface : 윈도우처럼 그래픽 대화 화면에서 마우스와 텍스트의 입력방식으로 명령을 입력하고, 결과가 표시되는 인터페이스 방식 ⇒ 클릭



“버전관리”의 커밋

✓ 저장소의 현 상태를 저장하는 행위



✓ 파일 집합의 변경 내용을 깃 저장소에 기록하는 작업

- 어느 시점의 파일 집합(폴더)의 추가/변경 사항을 저장소에 기록
- 이전 커밋 상태부터 현재 상태까지의 변경 이력이 기록된 커밋이 생성

02. 버전관리 개요

2 커밋과 버전관리

⚙ 커밋 추적 관리

✓ 커밋

- 저장소의 현재 상태를 저장

✓ 저장소

- 연속된 커밋으로 관리
- 파일이 달라지지 않았으면 파일을 새로 저장하지 않음
 - 단지 이전 상태의 파일에 대한 링크만 저장

✓ 저장소(Git repository)

- 파일이나 폴더를 저장해 두는 곳
 - 파일이 변경 이력 별로 구분되어 저장



✓ 저장소(Git repository)

원격 저장소
(Remote Repository)



파일이 원격 저장소 전용 서버에서 관리되며
여러 사람이 함께 공유하기 위한 저장소

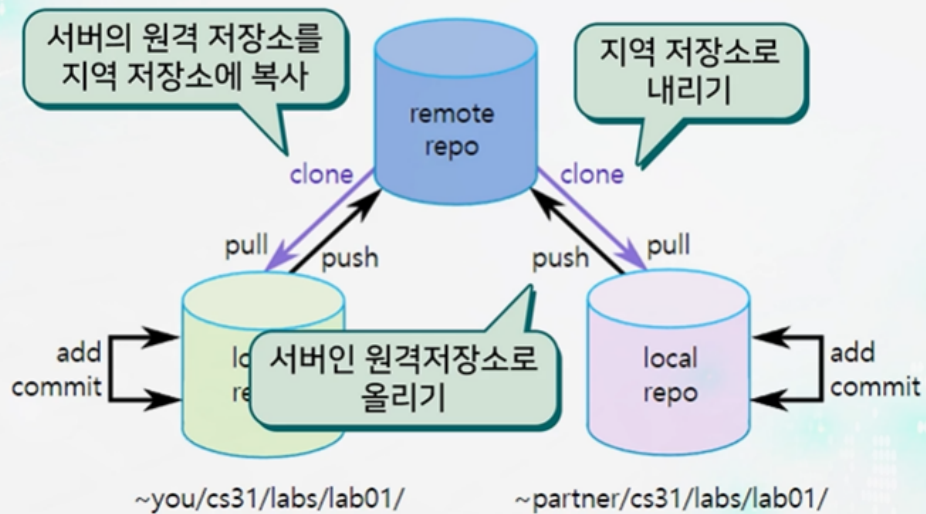
지역 저장소
(Local Repository)



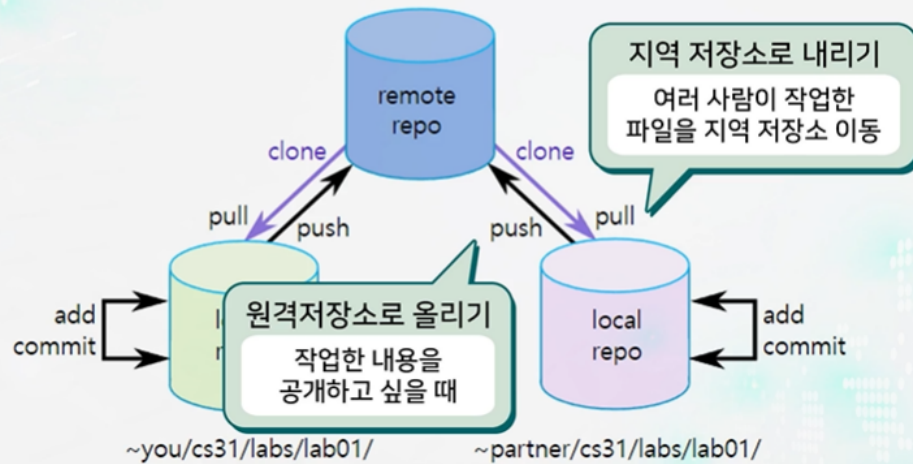
내 PC에 파일이 저장되는 개인 전용 저장소

2 커밋과 버전관리

⚙️ 원격 저장소와 지역 저장소의 명령 - Clone, Push, Pull



원격 저장소와 지역 저장소의 명령 - Push와 Pull 활용



정리하기

» 버전관리의 정의

- ◆ 시간 흐름에 따라 파일 집합에 대한 변경 사항을 추적, 관리

» 버전관리의 필요성

- ◆ 과거 지점의 버전으로 돌아가 누가, 무엇을 수정했는지 파악 가능

» 버전관리 도구의 인터페이스방식

- ◆ CLI와 GUI
 - Git: CLI
 - 소스트리: GUI

» 커밋(commit)

- ◆ 저장소의 현 상태를 저장하는 행위이자 깃 명령어

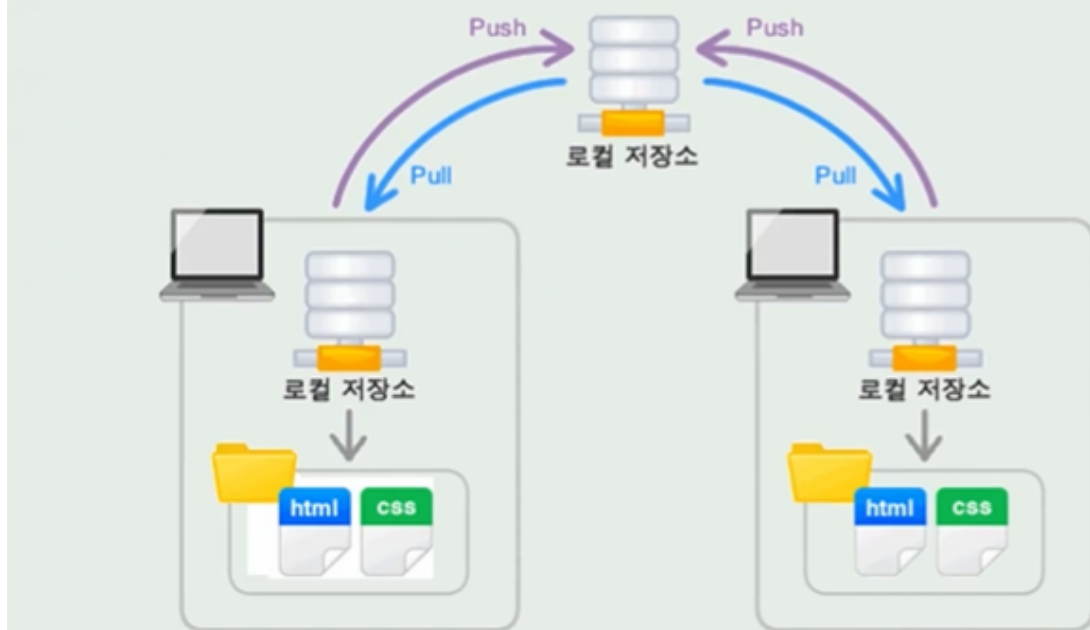
» HEAD

- ◆ 가장 최근의 커밋을 가리키는 포인터

» 저장소(Git repository)

- ◆ 파일이나 폴더, 버전관리를 위한 관련 파일을 저장해 두는 곳

» Push와 Pull



깃 개요

깃 사용 장점

- 모든 개발자는 지역 시스템에 코드의 전체 사본을 소유
 - 소스 코드에 대한 모든 변경 사항은 다른 사용자가 추적 가능

깃 기능

☑ 정의

- 컴퓨터 파일의 변경을 추적하는 데 사용되는 버전 관리 시스템

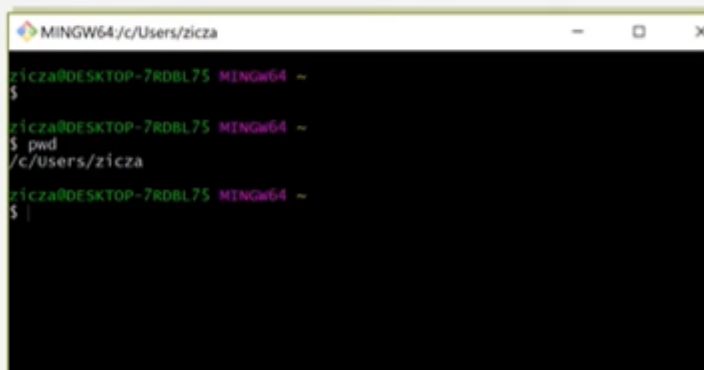
☑ 기능

- 여러 개발자가 함께 작업
- 소스 코드의 변경 사항을 추적하는 데 사용
- 소스 코드 관리에 분산 버전 제어 도구가 사용
- 여러 개의 평행 분기를 통해 비선형 개발을 지원

깃 설치: 두 개의 SW 제공 Git Bash, Git GUI

☑ Git Bash: CLI(Command Line Interface)

- 명령 행 인터페이스
 - 처음엔 어렵지만
 - CLI를 사용할 줄 알면 GUI도 사용할 수 있지만
 - ➡ 반대는 성립하지 않음
- Mac의 Terminal
- Windows의 CMD나 Powershell



```
MINGW64 /c/Users/zicza
zicza@DESKTOP-7RDBL75 MINGW64 ~
$
zicza@DESKTOP-7RDBL75 MINGW64 ~
$ pwd
/c/Users/zicza
zicza@DESKTOP-7RDBL75 MINGW64 ~
$
```


⚙️ 깃의 내부 주요 영역 구조

☑️ 깃 내부 저장소 상태

- 작업 디렉토리(working directory, working folder)
- 작업 공간(work space), 작업 트리(working tree)
 - modified, untracked
- 스테이징 영역(staging area, stage area, index)
 - staged, indexed
- 깃 저장소(git repository, repository, .git directory)
 - Committed
- 임시 저장소(stash)
 - stash