



# week4



1. 깃 커밋과 로그
2. 로그 이력과 과거 여행
- 3.[실습] 로그 이력과 과거 여행

## summary

1. 깃 커밋과 로그

## Summary

### » 깃 설정

- ◆ \$ git config --global user.name "hs kang"
- ◆ \$ git config --global user.email hsakng@gmail.com
- ◆ \$ git config --global core.autocrlf true
- ◆ \$ git config --global core.safecrlf false

### » 깃 저장소 생성과 버전관리 과정

- ◆ \$ git init basic
- ◆ \$ git add hello.txt
- ◆ \$ git commit -m A
- ◆ \$ git status
- ◆ \$ git log --oneline
- ◆ \$ git show --oneline

# Summary

## » Git init basic

## » Add, commit

## » Status

- ◆ 파일 처음 생성
  - Untracked file
    - ➡ 붉은 색 표시는 작업 디렉토리를 의미
- ◆ 처음 add한 파일
  - A new file
    - ➡ 녹색 표시는 스테이징 영역을 의미
- ◆ 다시 수정한 파일
  - Modified file
    - ➡ 붉은 색 표시는 작업 디렉토리를 의미
    - ➡ 스테이징 영역과 비교해서 작업 디렉토리의 파일이 수정된 것을 의미
- ◆ 수정한 파일을 add
  - Modified file
    - ➡ 녹색 표시는 스테이징 영역을 의미
    - ➡ 최근 커밋과 비교해서 스테이징 영역의 파일이 수정된 것을 의미



# Summary

## » 모두 커밋 이력 보기

- ◆ \$ git log
  - --oneline --graph --all -n

## » 특정한 커밋 이력 보기

- ◆ \$ git show [HEAD]
  - --oneline

## 2. 로그 이력과 과거 여행

## » 저장소 생선

- ◆ `$ git init basic`

## » 깃 저장소에 저장

- ◆ `$ git add, commit`

## » 모두 커밋 이력 보기

- ◆ `$ git log`

- `--oneline --graph --all -n`

## » 측정한 커밋 이력 보기

- ◆ `$ git show [HEAD]`

- `--oneline`

## » 바로 이전 버전으로 가기

- ◆ \$ git checkout HEAD~

## » 다시 최신 버전으로 돌아오기

- ◆ \$ git checkout main

## » 다시 checkout 이전으로 돌아오기

- ◆ \$ git checkout -

---

3.[실습] 로그 이력과 과거 여행

## » 저장소 생성

- ◆ \$ git init basic

## » 깃 저장소에 저장

- ◆ \$ git add, commit

## » 모두 커밋 이력 보기

- ◆ \$ git log
  - --oneline --graph --all -n

## » 특정한 커밋 이력 보기

- ◆ \$ git show [HEAD]
  - --oneline

## » 바로 이전 버전으로 가기

- ◆ \$ git checkout HEAD~

## » 다시 최신 버전으로 돌아오기

- ◆ \$ git checkout main

## » 다시 checkout 이전으로 돌아오기

- ◆ \$ git checkout -

### 깃 3 영역

#### - 작업 영역

- working directory(folder)
- working tree
- 탐색기 상의 폴더 하부

#### - 스테이징 영역

- staging area
- index
- 저장소의 .git 폴더의 파일 index

#### - 깃 저장소(git repository)

- 저장소
- 저장소의 .git 폴더의 여러 정보

## 버전관리를 위한 add, commit 명령

### \$git add

- 파일을 Working tree ⇒ Staging Area로 이동(복사)

### \$git commit

- Staging Area에 준비된 파일 → Git repository(버전관리)로 이동(복사)

## 깃 상태 보기 : \$git status



시험문제 예상

git add, git commit 을 하기 전에 오류가 나는 경우(untracked file) 어떻게 해야지 해결할 수 있는지 상황파악할 수 있기

### 커밋

#### ✓ 커밋(commit)

- 버전 관리를 위해 현재 스테이지 영역의 내용에 대해 스냅샷(snapshot)을 찍는 명령
  - 즉 커밋으로 버전 관리를 위해 인덱스에 추가된 파일들의 현재 상태를 저장
- 커밋에는 이력을 관리하기 위해 반드시 커밋 메시지의 저장이 필요

#### 주요 명령

\$ git commit	커밋 메시지를 입력할 기본 편집기 실행됨
\$ git commit -m 'message'	커밋 메시지를 직접 입력 [-m   --message]
\$ git commit -a -m 'message'	추가와 커밋을 함께 실행 [-a   --all]
\$ git commit -am 'message'	

#### ✓ Untracked file

- 추가하고 한 이후에 커밋
  - 깃에서 추가와 커밋을 동시에 실행 불가능

## ❗ 커밋 성공 후 상태 보기

### ✓ 첫 커밋이 성공한 후 상태를 조회

- 커밋할 것은 없고, 작업 트리는 깨끗한(clean) 상태

- nothing to commit, working tree clean ▪ 3 영역이 동일한 상태

PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic (main)



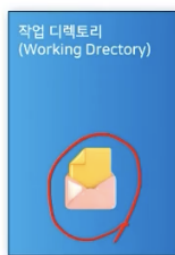
\$ git status

On branch main

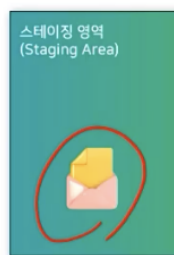
nothing to commit, working tree clean



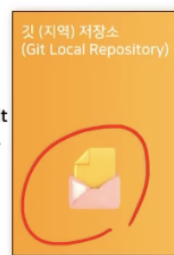
탐색기 저장소 폴더



add



commit



## 버전 로그 이력 확인



## ✓ \$ git log

### 주요 명령

\$ git log	로그 이력 정보를 표시
\$ git log --oneline	로그 이력을 한 줄로 표시
\$ git log [--patch   -p]	로그 이력과 함께 파일의 변화(이전 커밋과의 차이)를 표시

PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic (main)

\$ git log

commit dd3f28a76f6f6abe24b0e5cfc1b99c5ae0af8073 (HEAD -> main)

Author: ai7dnn <ai7dnn@gmail.com>

Date: Mon Dec 12 18:37:02 2022 +0900

A

1st commit

## ⚙ 커밋 정보 git show

✓ 특정한 커밋 정보를 확인하려면 명령 git show를 사용

### 주요 명령

\$ git show	마지막 커밋(HEAD)의 커밋 정보 표시
\$ git show --oneline	커밋 로그 한 줄과 파일 차이 표시
\$ git show -s	파일 차이는 표시되지 않음
\$ git show [HEAD]	지정한 HEAD의 커밋 정보 표시
\$ git show [commitID]	지정한 commitID의 커밋 정보 표시

## 로그 이력과 과거 여행

→ 굉장히 중요하고 시험에도 나올 거 같은데 어렵다.. pdf 파일을 보라



## 명령 checkout

✓ 현재 브랜치에서 과거 커밋 HEAD~로 이동

주요 명령	
\$ git checkout HEAD~	HEAD 이전 커밋으로 이동
\$ git checkout -	이전 checkout으로 이동
\$ git checkout main	브랜치의 마지막 커밋으로 이동

설명	git checkout	git switch
	↓	↓
이전 커밋으로 이동	\$ git checkout [이전커밋]	\$ git switch -d [이전커밋]
다른 브랜치로 이동	\$ git checkout [branch]	\$ git switch [branch]
	↓	↓
새로운 브랜치를 생성하고 이동	\$ git checkout -b [newBranch]	\$ git switch -c [newbranch]