
Open source Software
오픈소스 소프트웨어
08. 로그 이력과 과거 여행

학습 개요

1. 깃 설정과 저장소 생성
2. init, add, commit, status
3. log, show
4. 버전 이력을 표현
   - HEAD, HEAD~, HEAD~~, HEAD~2
   - HEAD, HEAD^, HEAD^1, HEAD^^
   - HEAD, HEAD~, HEAD^~
5. 과거 이전 버전으로 이동
   - Checkout HEAD~

학습 목표

1. 초기에 깃을 설정하고 저장소를 생성할 수 있다.
2. 파일을 만들어 버전관리를 위한 과정을 수행할 수 있다.
3. 버전관리 로그 이력을 확인할 수 있다.
4. 이전 버전으로 이동한 후 다시 최신 버전으로 돌아올 수 있다.

LESSON 01

여러 커밋과 로그 이력

1 여러 커밋과 로그 이력

깃 설정과 지역 저장소 생성

깃 설정과 깃 저장소 basic 생성 실습

순서	명령어	설명	비고
❶	$ git config --global user.name hskang $ git config --global user.email hskang@gmail.com	사용자 이름 전자메일 설정	
	$ git config --global core.autocrlf true $ git config --global core.safecrlf false	자동 줄바꿈 안전 줄바꿈	Carriage Return Line Feed
	$ git config --global core.editor 'code --wait' $ git config --global init.defaultBranch main	기본 편집기 설정 기본 브랜치 이름 설정	
❷	$ git init basic	저장소 basic 생성	
❸	$ cd basic $ ls -al	폴더 이동 파일 확인	.git 폴더 확인

1 여러 커밋과 로그 이력

파일 생성과 1st 커밋

파일 hello.txt 생성 후 커밋

수행 순서와 명령어	작업 디렉토리(폴더) [hello.txt]	스테이지(Index) 영역 [hello.txt]	깃 저장소와 커밋 이력	

❸ `$ git commit -m A`
`$ git status`
`$ git log`
`$ git log --oneline`
`$ git log [--patch | -p]`
		aaa	A
			aaa
				HEAD

❷ `$ git add hello.txt`
`$ git status`
`$ git status [--short | -s]`
`A hello.txt`
	aaa			

❶ `$ echo aaa > hello.txt`
`$ cat hello.txt`
`$ git status [--long]`
`$ git status -s`
`?? hello.txt`				

작업 디렉토리	스테이징 영역	깃 저장소
aaa	aaa	aaa

1 여러 커밋과 로그 이력

2nd 커밋과 파일 수정

- 파일 수정 후 커밋: 4 -> 5
- 다시 파일 수정: 6

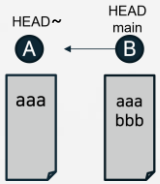
수행 순서와 명령어	작업 디렉토리(폴더) [hello.txt]	스테이지(Index) 영역 [hello.txt]	깃 저장소와 커밋 이력	

❻ `$ echo ccc >> hello.txt`
`$ cat hello.txt`
`$ git status`
`$ git status -s`
`M hello.txt`
	aaa
	bbb
	ccc
		aaa
		bbb		

❺ `$ git commit -am B`
`$ git status`
`$ git log`
`$ git log --oneline`
`$ git log [--patch | -p]`
	aaa
	bbb
			B
			aaa
			bbb
				HEAD

❹ `$ echo bbb >> hello.txt`
`$ cat hello.txt`
`$ git status`
`$ git status -s`
`M hello.txt`
		aaa	↓	

작업 디렉토리	스테이징 영역	깃 저장소
aaa bbb ccc	aaa bbb	aaa bbb



1 여러 커밋과 로그 이력

로그의 옵션

명령 git log

- 기본적으로 가장 최근의 커밋부터 표시

명령 git log 옵션

- --graph
- -n

주요 명령	
`$ git log --graph`	문자 그림으로 로그 이력 그리기
`$ git log --reverse`	오래된 커밋부터 표시 --graph와 함께 사용할 수 없음
`$ git log --all`	모든 브랜치의 로그 이력 표시
`$ git log -n`	최근 n개의 로그 이력 표시

1 여러 커밋과 로그 이력

3rd 커밋과 로그 이력

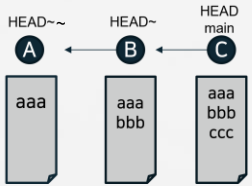
- $ git log
- $ git show

	수행 순서와 명령어	작업 디렉토리(폴더) [hello.txt]	스테이지(Index) 영역 [hello.txt]	깃 저장소와 커밋 이력	

	$ git commit -am C
	$ git status
	$ git log
	$ git log --oneline
	$ git log [--patch | -p]
	$ git log --reverse --onelin
	e
				C	

❼	$ git log -2
	$ git log -2 --oneline
	$ git log --graph --oneline
	$ git show
	$ git show --oneline
	$ git show HEAD
	$ git show HEAD~
	$ git show HEAD~2
		aaa
		bbb
		ccc
			aaa
			bbb
			ccc
				aaa
				bbb
				ccc
					HEAD

작업 디렉토리	스테이징 영역	깃 저장소
aaa bbb ccc	aaa bbb ccc	aaa bbb ccc



1 여러 커밋과 로그 이력

Log와 Show

세 번의 커밋 이력으로 실습

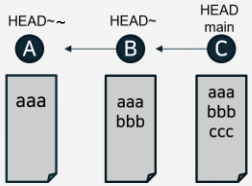
- HEAD, HEAD~, HEAD~~

수행 순서와 명령어	

❼	$ git commit –am C
	
	$ git status
	
	$ git log
	$ git log –oneline
	$ git log [--patch | -p]
	$ git log --reverse --oneline
	$ git log -2
	$ git log -2 --oneline
	$ git log --graph –oneline
	
	$ git show
	$ git show --oneline
	$ git show HEAD
	$ git show HEAD~
	$ git show HEAD~2

C	HEAD
aaa bbb ccc	07af7f6
↓	
B	**HEAD~** **HEAD^** **HEAD~1** **HEAD^1**
aaa bbb	3a2d414
↓	
A	**HEAD~~** **HEAD^^** **HEAD~^** **HEAD~2**
aaa	dd3f28a

작업 디렉토리	스테이징 영역	깃 저장소
aaa bbb ccc	aaa bbb ccc	aaa bbb ccc



1 여러 커밋과 로그 이력

파일 수정과 추가 후 다시 수정

- 파일 수정과 추가: 8, 9
- 다시 수정: 10

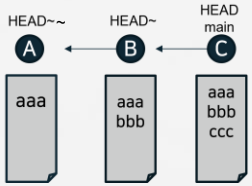
수행 순서와 명령어	작업 디렉토리(폴더) [hello.txt]	스테이지(Index) 영역 [hello.txt]	깃 저장소와 커밋 이력	

⑩ $ echo eee >> hello.txt
$ git status
$ git status -s
MM hello.txt
	aaa
	bbb
	ccc
	ddd
	eee
		aaa
		bbb
		ccc
		ddd
			최종커밋내용
			(hello.txt)
				aaa
				bbb
				ccc

⑨ $ git add hello.txt
$ git status
$ git status -s
M hello.txt
	aaa
	bbb
	ccc
	ddd			

⑧ $ echo ddd >> hello.txt
$ git status
$ git status -s
M hello.txt
		aaa
		bbb
		ccc		

작업 디렉토리	스테이징 영역	깃 저장소
aaa bbb ccc ddd eee	aaa bbb ccc ddd	aaa bbb ccc



1 여러 커밋과 로그 이력

현재 상태

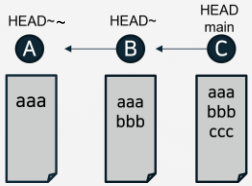
3번의 커밋이 수행된 상태와 커밋 로그 이력

- 커밋 이후에 수정(ddd 추가), 추가, 다시 수정(eee 추가)

작업 폴더 (hello.txt)	스테이지 영역 (hello.txt)	깃 저장소 (hello.txt)
aaa bbb ccc ddd eee	aaa bbb ccc ddd	aaa bbb ccc

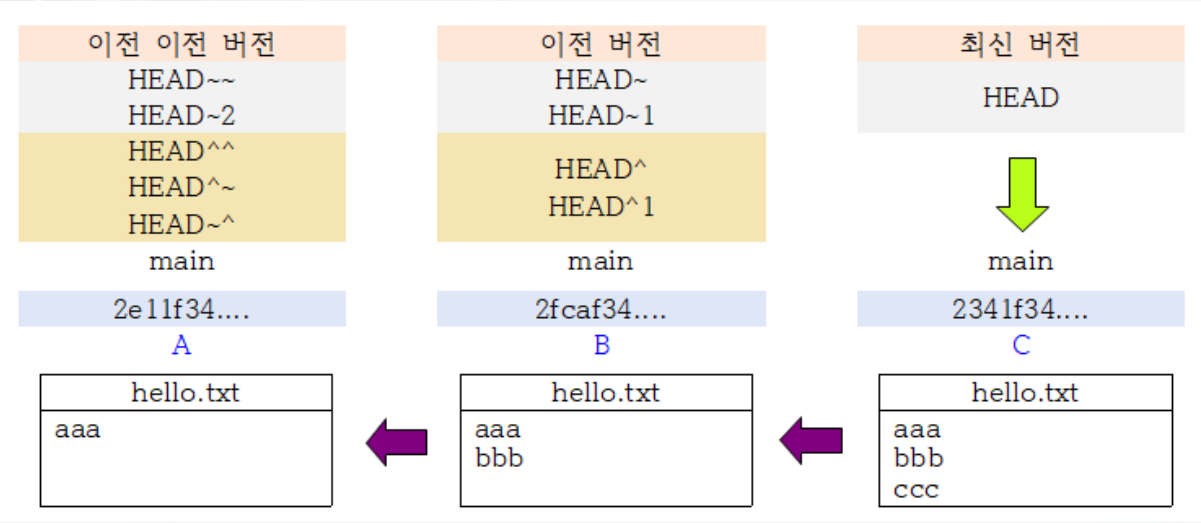
C	HEAD
aaa bbb ccc	07af7f6
↓	
B	HEAD~ HEAD^ HEAD~1 HEAD^1
aaa bbb	3a2d414
↓	
A	HEAD~~ HEAD^^ HEAD~^ HEAD~2
aaa	dd3f28a

작업 디렉토리	스테이징 영역	깃 저장소
aaa bbb ccc ddd eee	aaa bbb ccc ddd	aaa bbb ccc



1 여러 커밋과 로그 이력

현재 상태 다른 표현



작업 폴더 (hello.txt)	스테이지 영역 (hello.txt)	깃 저장소 (hello.txt)
aaa bbb ccc ddd eee	aaa bbb ccc ddd	aaa bbb ccc

LESSON 02

과거로의 시간 여행

2 과거로의 시간 여행

현재 상태

3번의 커밋이 수행된 상태와 커밋 로그 이력

작업 폴더 (hello.txt)	스테이지 영역 (hello.txt)	깃 저장소 (hello.txt)
aaa bbb ccc ddd eee	aaa bbb ccc ddd	aaa bbb ccc

C	HEAD
aaa bbb ccc	07af7f6
↓	
B	**HEAD~** **HEAD^** **HEAD~1** **HEAD^1**
aaa bbb	3a2d414
↓	
A	**HEAD~~** **HEAD^^** **HEAD~^** **HEAD~2**
aaa	dd3f28a

2 과거로의 시간 여행

명령 checkoout

현재 브랜치에서 과거 커밋 HEAD~로 이동

- 그 상태에서 당시의 파일 내용을 확인 가능

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

2 과거로의 시간 여행

$ git checkout HEAD~

과거 커밋으로 이동

- 현재 상태가 깨끗해야 checkout이 가능
- 떨어진 HEAD(detached HEAD) 상태
  - HEAD가 마지막 커밋이 아닌 그 이전을 가리킨다는 의미
- 프롬프트 브랜치 이름이 떨어진 HEAD 상태를 표시
  - HEAD가 가리키는 커밋ID ((3a2d414...))로 표시

```
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic (main)
$ git checkout HEAD~
Note: switching to 'HEAD~'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

git switch -c <new-branch-name>

Or undo this operation with:

git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 3a2d414 B

PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic ((3a2d414...))
$
```

C	HEAD
aaa bbb ccc	07af7f6
↓	
B	**HEAD~** **HEAD^** **HEAD~1** **HEAD^1**
aaa bbb	3a2d414
↓	
A	**HEAD~~** **HEAD^^** **HEAD~^** **HEAD~2**
aaa	dd3f28a

2 과거로의 시간 여행

detached HEAD

명령 git status에서도 떨어진 HEAD(detached HEAD) 상태를 확인

- 명령 git log 옵션 --oneline --all

```
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic ((3a2d414...))
$ git status
HEAD detached at 3a2d414
nothing to commit, working tree clean

PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic ((3a2d414...))
$ git log --oneline
3a2d414 (HEAD) B
dd3f28a A



PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic ((3a2d414...))
$ git log --oneline --all
07af7f6 (main) C
3a2d414 (HEAD) B
dd3f28a A



PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic ((3a2d414...))
$ cat hello.txt
aaa
bbb
```

C	HEAD
aaa bbb ccc	07af7f6
↓	
B	**HEAD~** **HEAD^** **HEAD~1** **HEAD^1**
aaa bbb	3a2d414
↓	
A	**HEAD~~** **HEAD^^** **HEAD~^** **HEAD~2**
aaa	dd3f28a

2 과거로의 시간 여행

되돌아오기 git checkout

브랜치 이름 main이나 - 사용

- 이전 위치를 나타내는 -을 사용

```
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic ((3a2d414...))
$ git checkout -
Previous HEAD position was 3a2d414 B
Switched to branch 'main'

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

PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic (main)
$ git log --oneline
07af7f6 (HEAD -> main) C
3a2d414 B
dd3f28a A
```

C	HEAD
aaa bbb ccc	07af7f6
↓	
B	HEAD~ HEAD^ HEAD~1 HEAD^1
aaa bbb	3a2d414
↓	
A	HEAD~~ HEAD^^ HEAD~^ HEAD~2
aaa	dd3f28a

2 과거로의 시간 여행

현재의 전전 커밋으로 이동

다음은 커밋 메시지 A로 이동하기 위해 git checkout HEAD~2를 실행한 화면

```
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic (main)
$ git log --oneline
07af7f6 (HEAD -> main) C
3a2d414 B
dd3f28a A

PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic (main)
$ git checkout HEAD~2
Note: switching to 'HEAD~2'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

git switch -c <new-branch-name>

Or undo this operation with:

git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at dd3f28a A

PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic ((dd3f28a...))
$ git log --oneline --all
07af7f6 (main) C
3a2d414 B
dd3f28a (HEAD) A
```

C	HEAD
aaa bbb ccc	07af7f6
↓	
B	**HEAD~** **HEAD^** **HEAD~1** **HEAD^1**
aaa bbb	3a2d414
↓	
A	**HEAD~~** **HEAD^^** **HEAD~^** **HEAD~2**
aaa	dd3f28a

2 과거로의 시간 여행

현재의 전전 커밋(HEAD~~)으로 이동

$ git checkout HEAD~2를 실행 이후

```
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic ((dd3f28a...))
$ git log --oneline --all
07af7f6 (main) C
3a2d414 B
dd3f28a (HEAD) A
```

```
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic ((dd3f28a...))
$ cat hello.txt
aaa

PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic ((dd3f28a...))
$ git checkout main
Previous HEAD position was dd3f28a A
Switched to branch 'main’

PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/basic (main)
$ cat hello.txt
aaa
bbb
ccc
```

2 과거로의 시간 여행

명령 checkout 정리

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]

Summary

저장소 생성

- $ 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 -