버전 관리 시스템 (git, subversion, 머큐리얼, cvs...)

1. 버전이란

2.Git

3.source tree (for gui)

4. 저장소만들기 & commit

5. 되돌리기

Discard <커밋전 이전상태로 돌리기>
Reset <이미 커밋한 사항 돌리기 + 이전버전 삭제>
Reset <이후 버전을 지우면서 현재상태 커밋X인 상태>
Revert 되돌리기

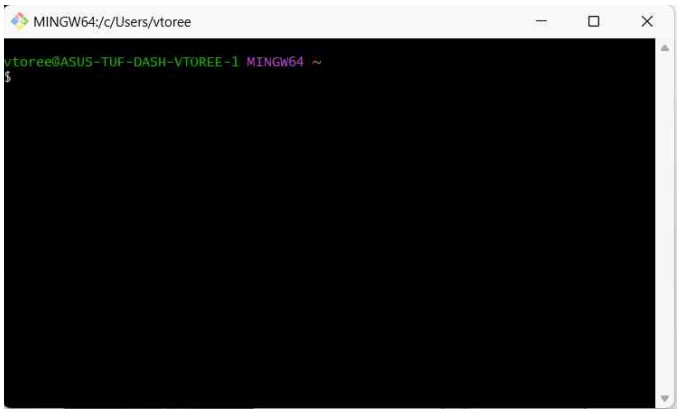
버전이란?

어떠한 의미있는 변화들
<기능 개선, 버그 수정, 커스터 마이징>

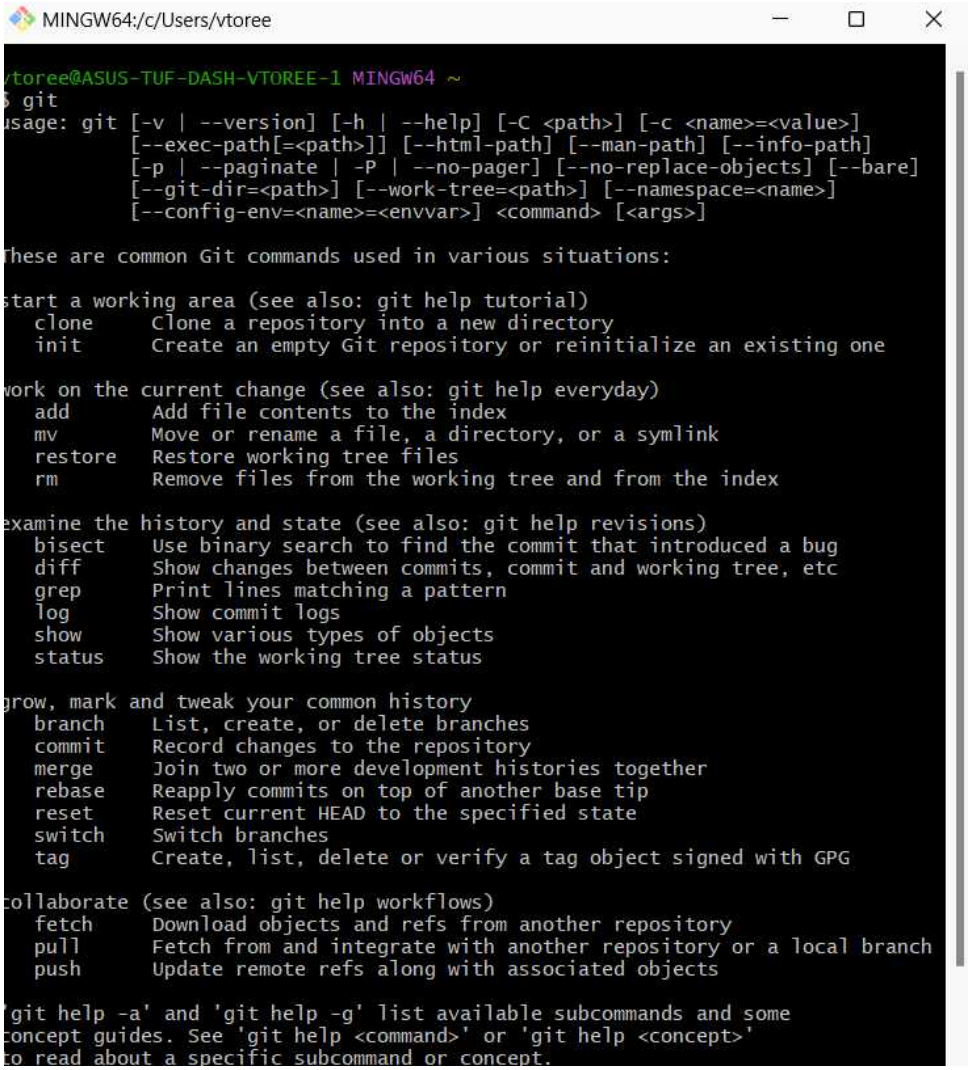
버전 관리 시스템 -> 변화의 관리
(악의적인 편집의 대처, 수정 컨펌X시 되돌리기)

이전 상태로 돌리기 위한 기능으로 필수적이다!

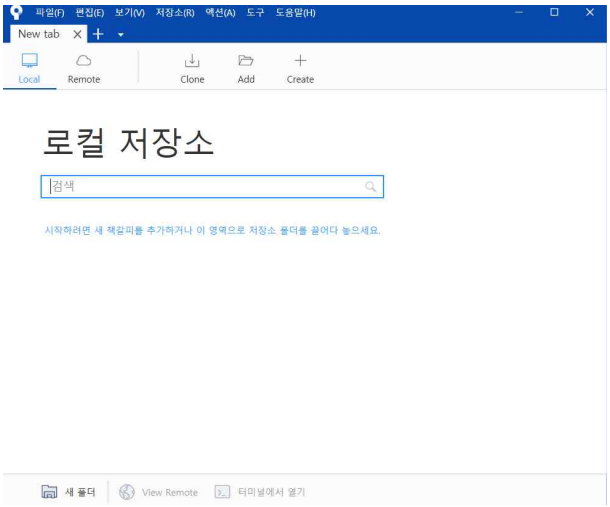
Git



git 명령어 입력시 사용가능한 기본 명령어들을 확인 할 수 있다.
>> git


MINGW64:/c/Users/vtoree
vtoree@ASUS-TUF-DASH-VTOREE-1 MINGW64 ~
$ git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
[--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
[-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
[--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
[--config-env=<name>=<envvar>] <command> [<args>]
These are common Git commands used in various situations:
start a working area (see also: git help tutorial)
clone Clone a repository into a new directory
init Create an empty Git repository or reinitialize an existing one
work on the current change (see also: git help everyday)
add Add file contents to the index
mv Move or rename a file, a directory, or a symlink
restore Restore working tree files
rm Remove files from the working tree and from the index
examine the history and state (see also: git help revisions)
bisect Use binary search to find the commit that introduced a bug
diff Show changes between commits, commit and working tree, etc
grep Print lines matching a pattern
log Show commit logs
show Show various types of objects
status Show the working tree status
grow, mark and tweak your common history
branch List, create, or delete branches
commit Record changes to the repository
merge Join two or more development histories together
rebase Reapply commits on top of another base tip
reset Reset current HEAD to the specified state
switch Switch branches
tag Create, list, delete or verify a tag object signed with GPG
collaborate (see also: git help workflows)
fetch Download objects and refs from another repository
pull Fetch from and integrate with another repository or a local branch
push Update remote refs along with associated objects
'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.

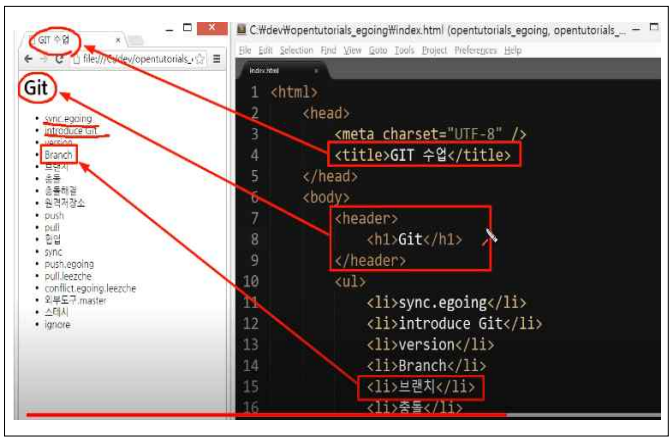
Source Tree <gui>



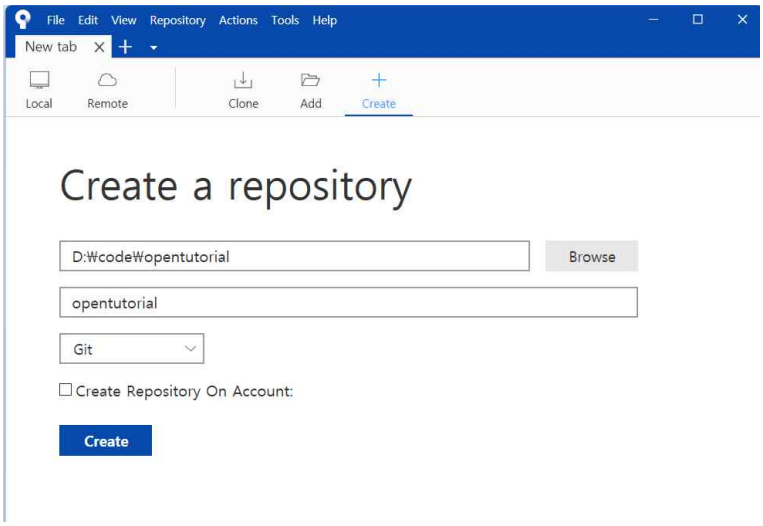
예제

```
<html>
        <head>
                <meta charset ="UTF-8" />
                <title> GIT 수업</title>
        </head>
        </body>
                <header>
                        <h1>Git</h1>
                </header>
                <u1>
                        <li>sync.egoing</li>
                        <li>introduce Git</li>
                        <li>version</li>
                        <li>Branch</li>
                        <li>브랜치</li>
                        <li>충돌</li>
                        <li>충돌해결</li>
                        <li>원격저장소</li>
```

…

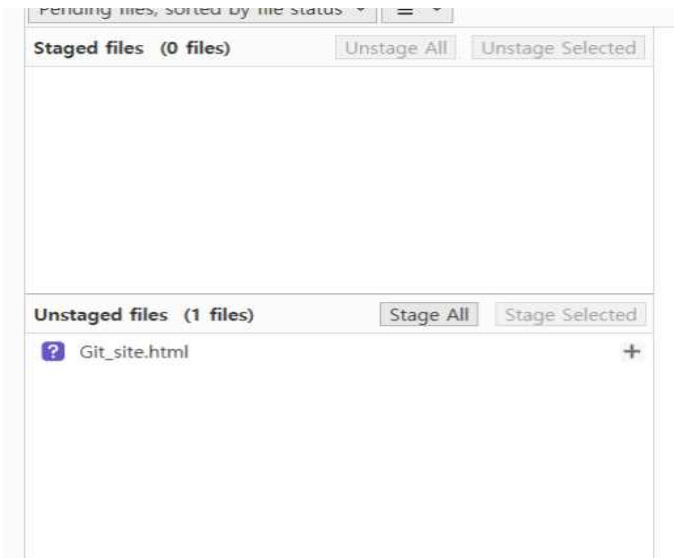


저장소 만들기 (Add repository)



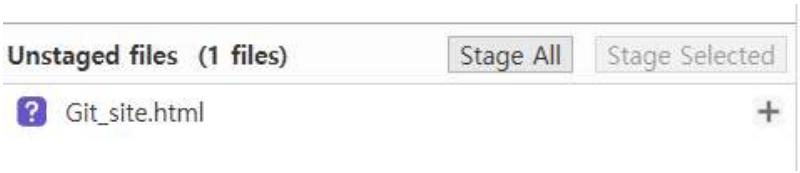
add : 기존 레포지토리 추가

create : to 레포지토리 생성

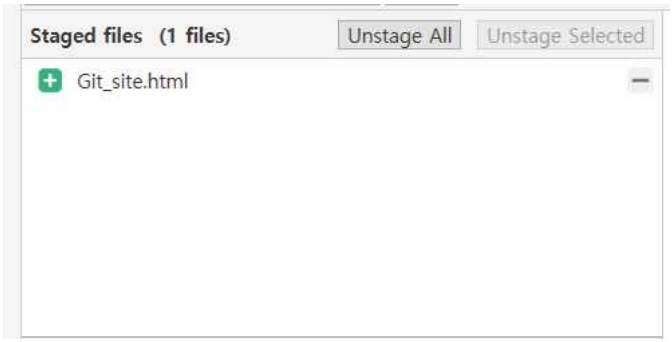


해당 폴더에 신규 파일 생성시 unstaged files에 변화

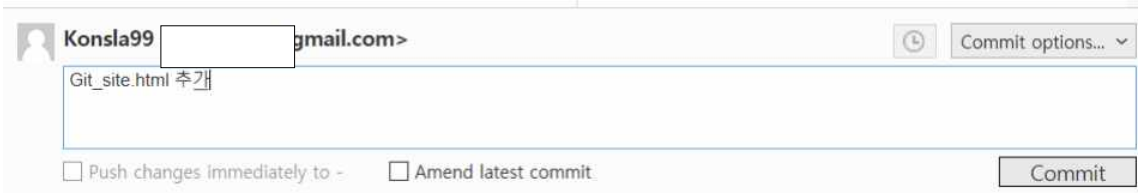
git 은 작업중인 저장소에서 일어나는 일들 즉 file, directory등의 추가, 제거, 수정 등의 이벤트를 모니터링 하고 있다.



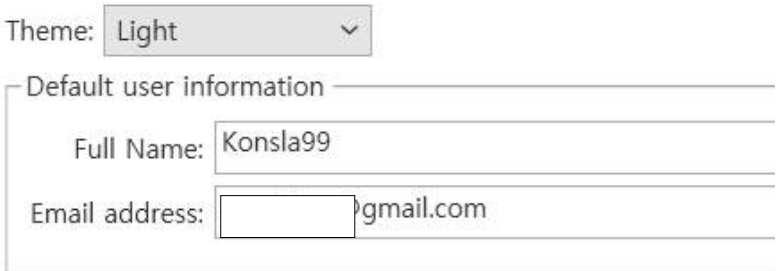
+를 눌러 주면



해당 파일이 이동한다.

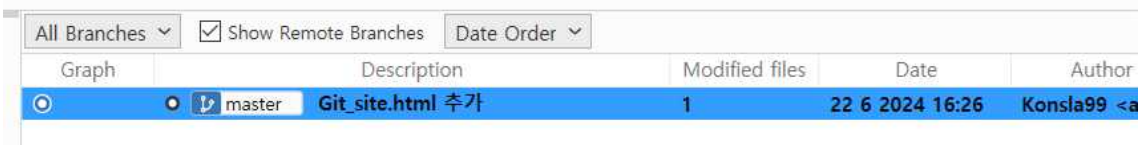


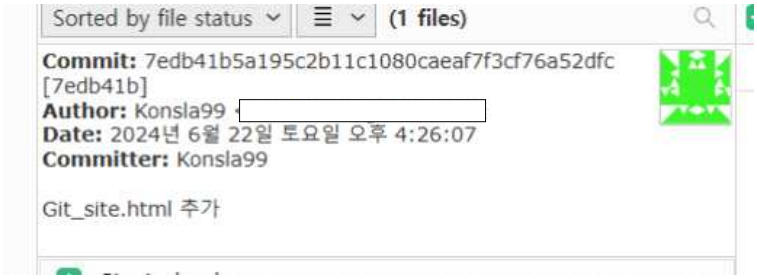
commit = 버전을 하나 만드는 행위



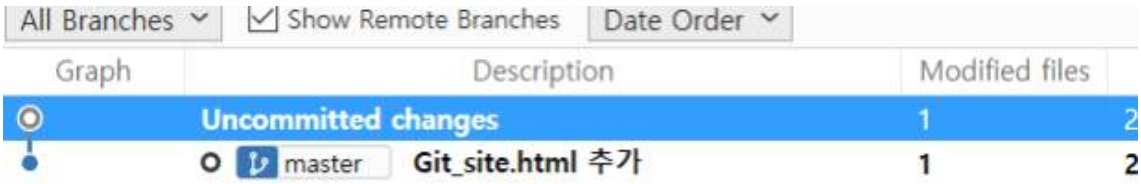
tools -> option
해당 버전을 만든유저의 정보를 자동으로 기입

실제로는 git config 명령어를 사용하지만 gui source tree를 사용하므로

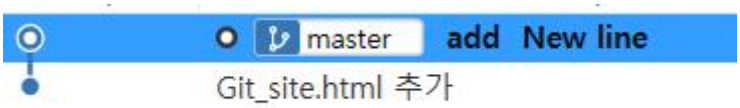




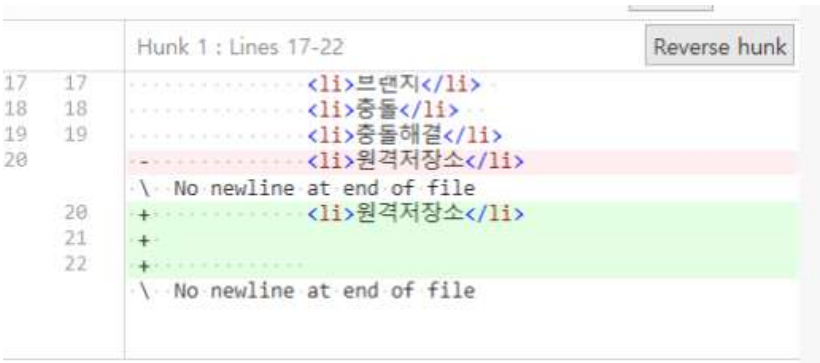
해당 파일 Git_site.html을 수정해보자.



uncommitted changes라는 문구가 새로 등장하는 것을 확인 할 수 있다.
commit되지 않은 버전을 감지한 것



실저 문서에서 enter를 눌러 new line이 추가 된것이라 해당 내용을 적어 커밋을 진행 하면 다음과 같이 변화하는 것을 확인 할 수 있다.



history를 확인하면 각 버전 별 변경이 된 내용은 녹색으로 표기 되어 버전간 변화된 내용을 확인 할 수 있다.

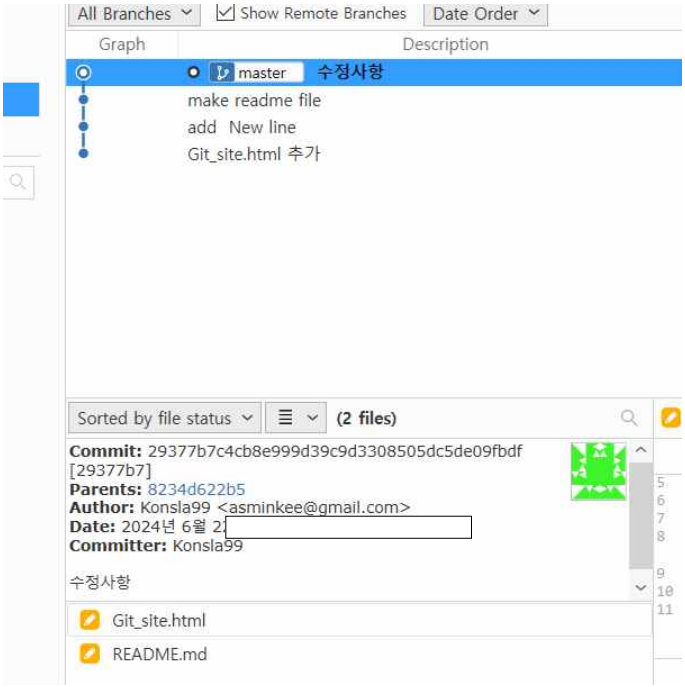
되돌리기 – 버전을 이전상태로-

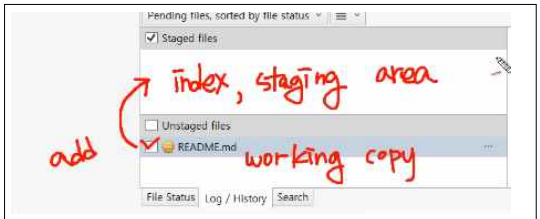
동일 폴더에 각기 다른 파일을 수정해 주었다.

두파일의 앞에 위치한 아이콘의 모양이 다른 것을 확인 할 수 있다.
Git입장에서 둘은 서로 다른 파일이라는 의미다.

Git_site.html = 이미 git에 의해 추적되는 파일 이미 커밋된 버전이 존재
README.md = 아직 git에 의한 추적이 안되는 상태의 파일

커밋 이후 둘 모두 같은 아이콘임을 확인 가능하다.





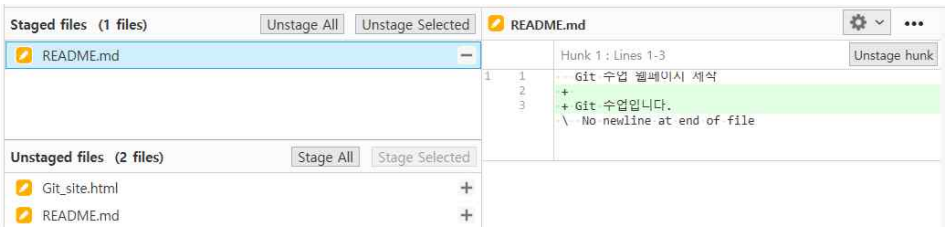
working copy : 수정내역이 나오는 곳
<unstage 단계>
add : stage에 추가하는 작업
<check행위>

add시 index(staging area로 이동)

두파일을 한번에 커밋할 경우
수정사항이라는 버전에는 이전 버전들과 다르게 두가지 파일이 모두 확인된다.

각각의 파일에 대한 수정파일을 원한다면
stage -> commit단계를 각각 진행해 주면된다.

Stage = 커밋전 중간단계 각각의 변경 사항중 필요한, 완결된 파일만을 grouping 하여 하나의 버전으로 만드는 것이 가능하다!



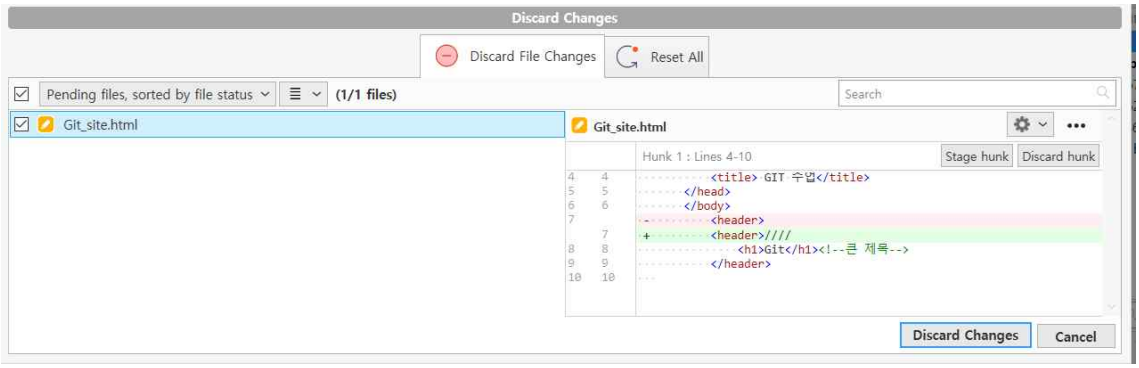
a 파일을 수정하고 staging area에 올린후
commit없이 a파일을 다시 수정하면

하나의 파일(a)가 staging area와 working space에 공존이 가능하다.
커밋시 staging area에 있는 내용만 새로운 버전으로 추가된다.

Discard <커밋전 이전상태로 돌리기>

```
Git_site.html > html > u1 > li
<html>
    <head>
        <meta charset ="UTF-8" />
        <title> GIT 수업</title>
    </head>
    </body>
        <header>////
            <h1>Git</h1><!--큰 제목-->
        </header>
```

////를 추가



붉은색 - 제거된 내용
초록색 + 추가된 내용

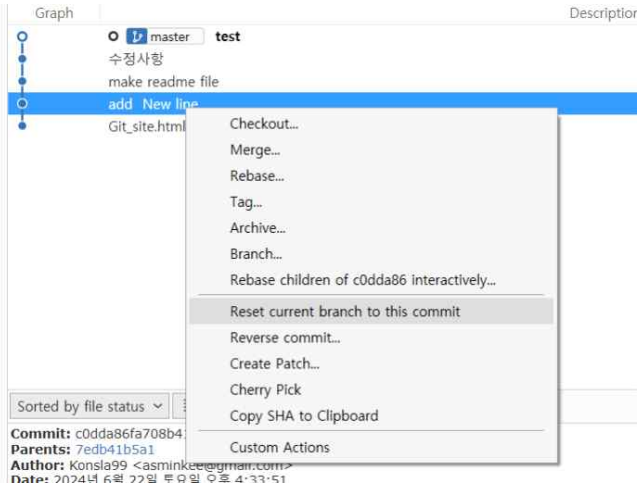
discard -> discard change

```
<html>
    <head>
        <meta charset ="UTF-8" />
        <title> GIT 수업</title>
    </head>
    </body>
        <header>
            <h1>Git</h1><!--큰 제목-->
        </header>
```

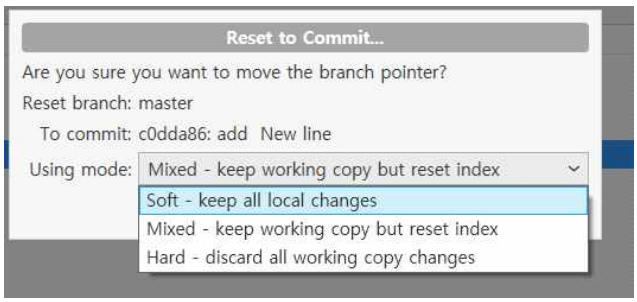
커밋 전에 이전에 커밋한 상태로 돌아가려면 (Discard)폐기를 하면 된다.

변경된 내용을 확인하여
커밋전에 변경사항을 리뷰할 수 있다. 꼭 확인 할 것!

Reset <이미 커밋한 사항 돌리기 + 이전버전 삭제>

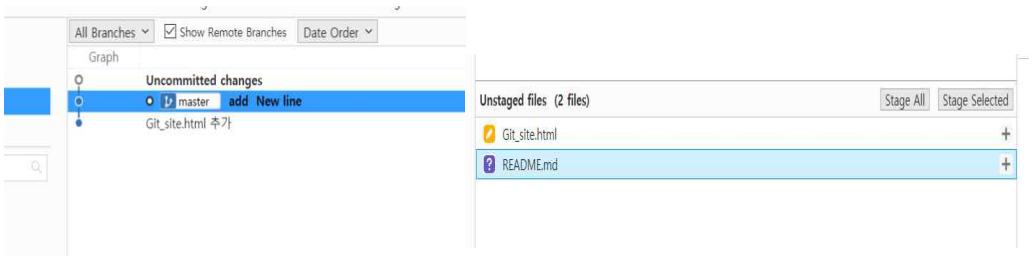


원하는 버전을 선택하고 우클릭 -> Reset current branch to this commit



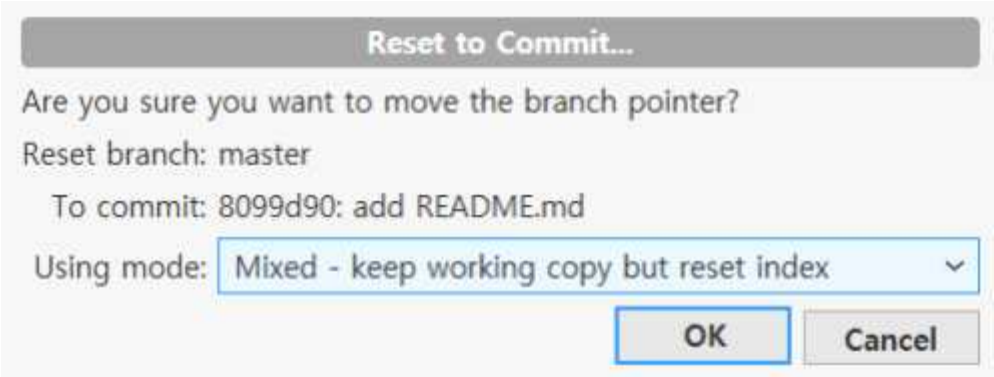
using mode -> Hard 선택

선택한 버전 이후의 모든 버전 삭제, 커밋하지 않은 내용 및 스테이지에 있는 내용 삭제.

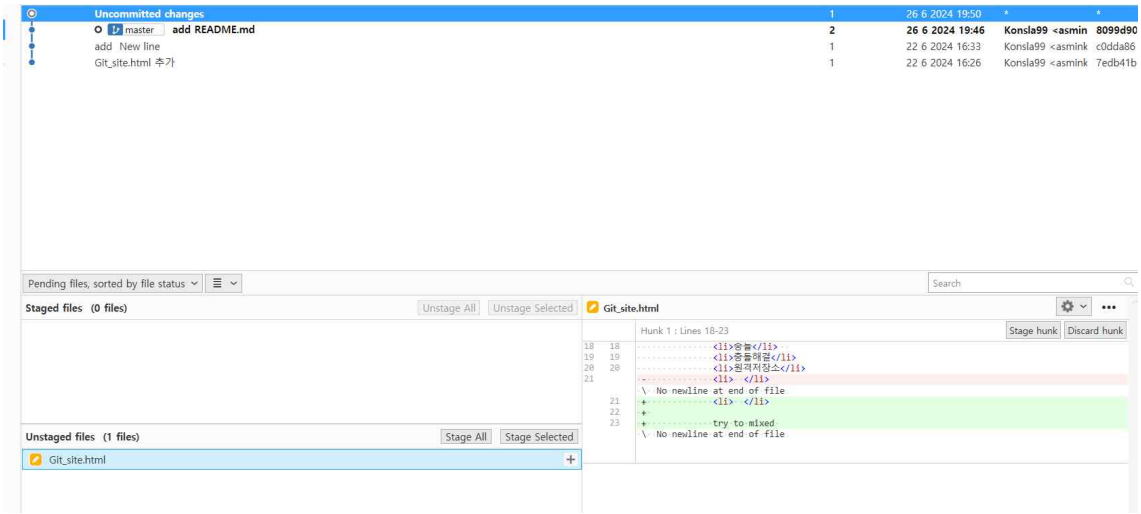


버전이 삭제된 것을 확인할 수 있다.
Readme.md는 커밋전의 상태.

Reset <이후 버전을 지우면서 현재상태 커밋X인 상태>

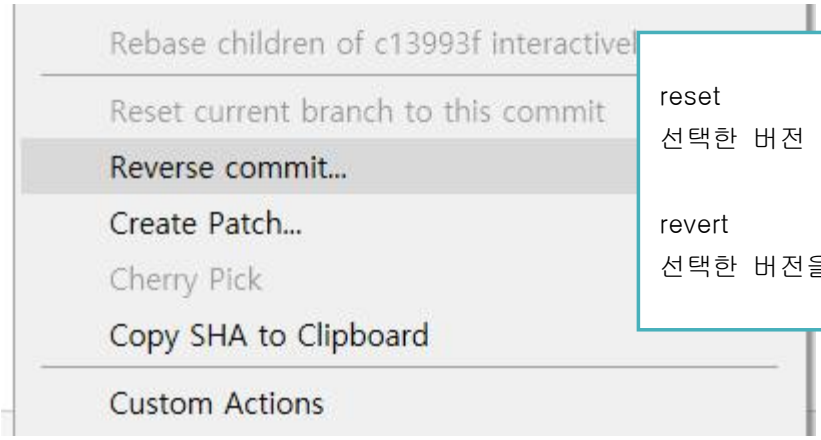


이전과 같은 상태에서 mixed를 선탣하면
이후버전 삭제 , 파일은 커밋 전의 버전으로



Revert 되돌리기

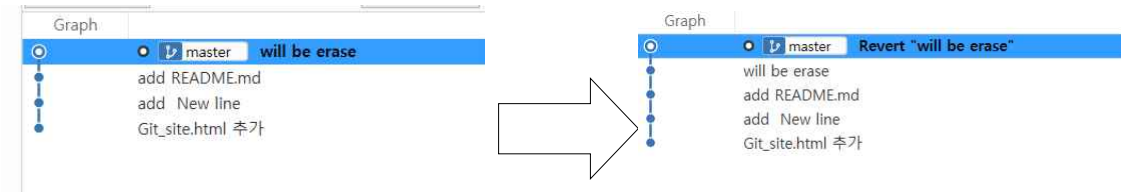
커밋을 취소하지만 버전을 유지하기 위해 사용



reset
선택한 버전 이후 버전 삭제

revert
선택한 버전을 취소해 그 이전으로

reverse commit 선택



새로운 버전이 생긴다.
아래서부터 버전 i 라고 할때 will be erase는 ver4
ver4에서 revert를 진행시 그 이전버전인 ver3의 상태를 ver5로 생성

즉 선택한 버전을 취소하고 싶지만 해당 버전을 지우지 않기위해
이전버전과 동일한 버전을 새로 커밋