



Git과 GitHub로 시작하는 오픈 소스 기초

1일차: Git의 기본 개념 및 로컬 저장소 사용 방법



1. Git 시작하기

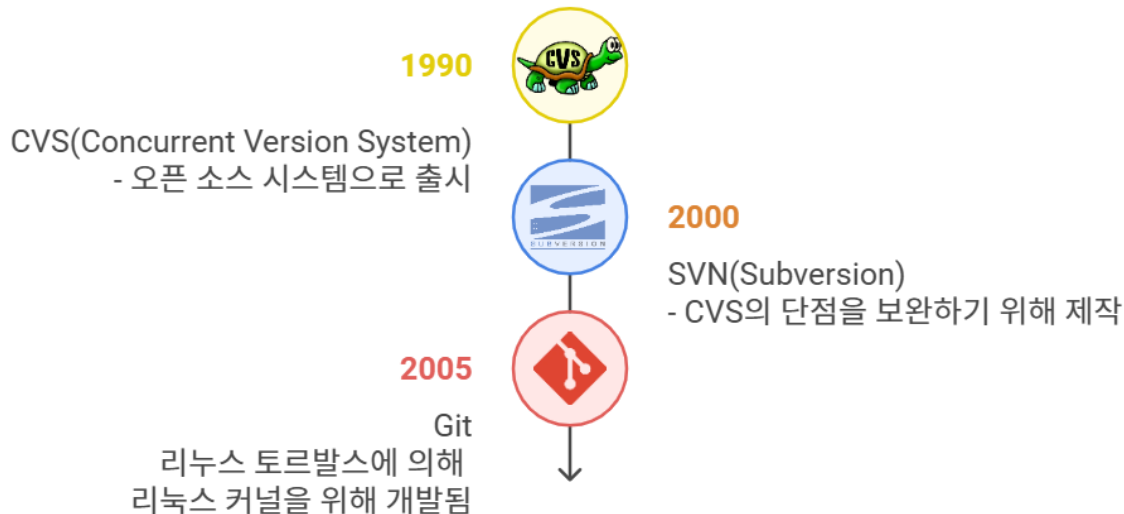
- 형상관리 개념 및 도구 별 특징 비교
- Git 설치 및 환경 설정



형상 관리

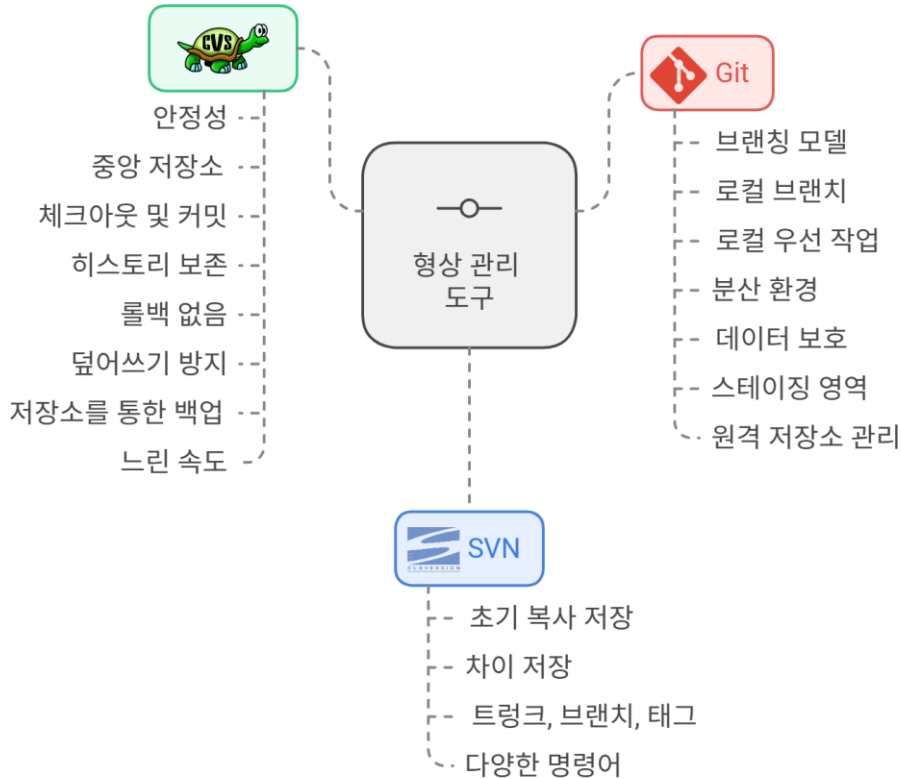
형상 관리(SCM, Software Configuration Management)

소프트웨어 개발 프로세스 각 단계에서 소프트웨어의 변경 점을 체계적으로 추적하고 관리하는 일련의 활동



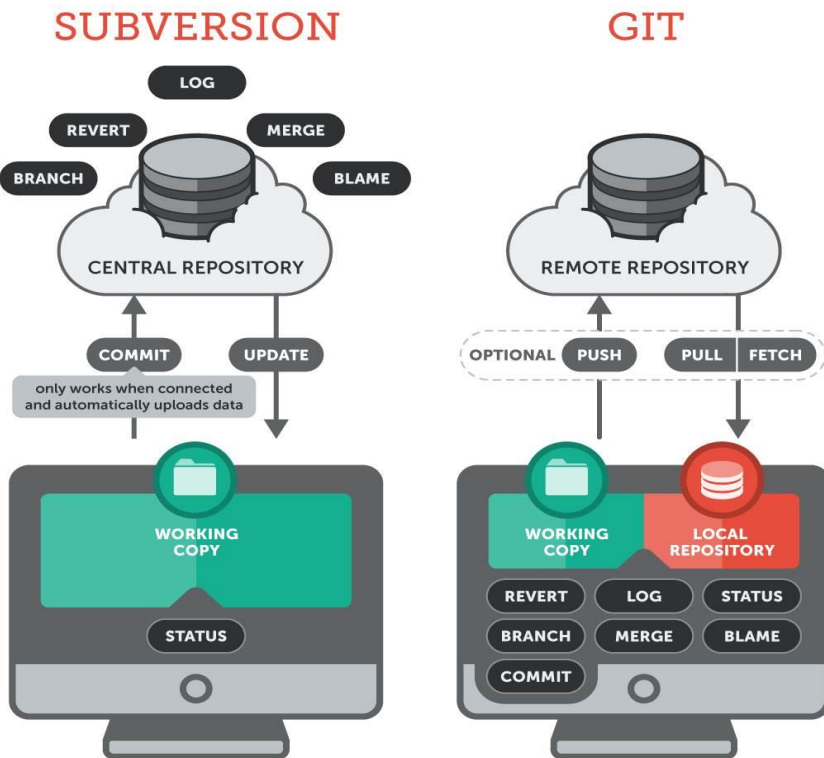
형상 관리 도구의 종류

버전 관리 시스템 비교: CVS, SVN, Git의 차이점



SVN vs Git: 버전 관리 시스템 비교

중앙 집중형과 분산형 버전 관리 시스템의 차이점



출처: https://elky84.github.io/2020/07/19/git_vs_svn/



깃(Git)이란 무엇인가?

'지옥'에서 온 문서 관리자??



Linus Torvalds



1. Global Information Tracker
2. It handles stupid content well

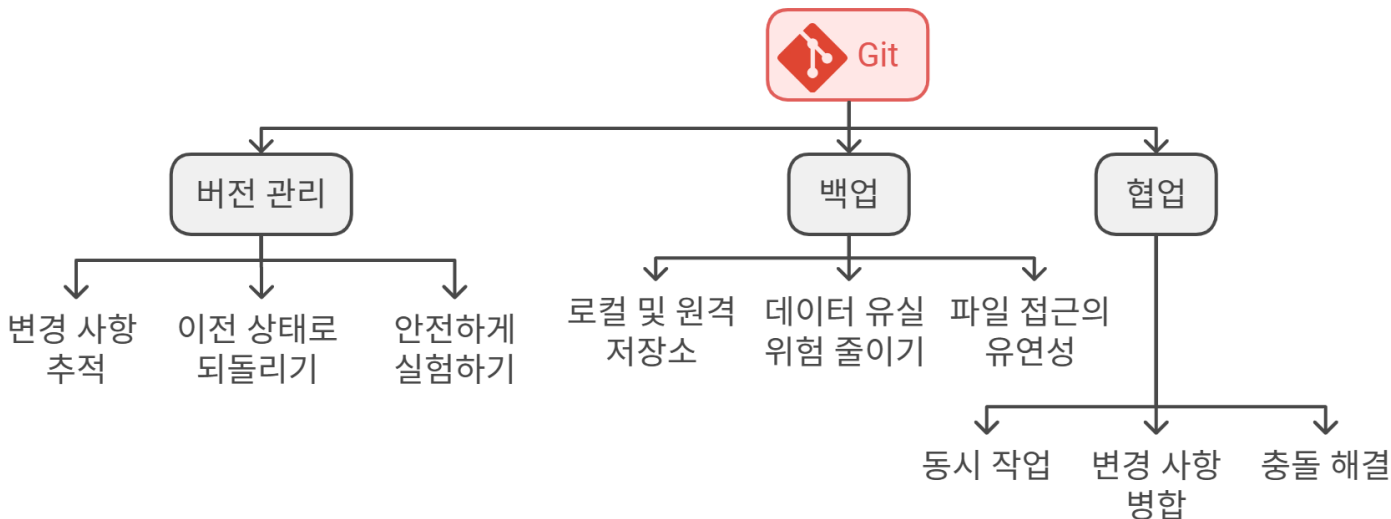
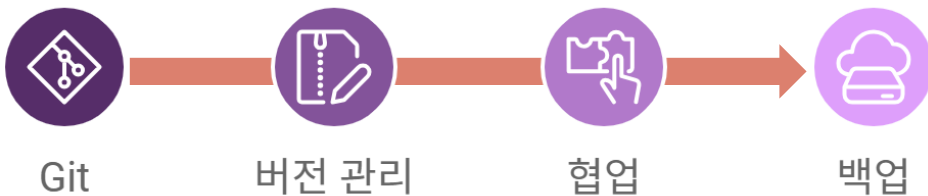


Linux

= Linus + Unix

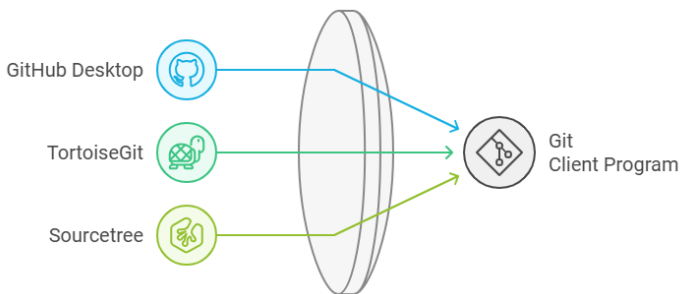
Git의 주요 기능과 이점

Git: 버전 관리, 백업, 협업을 위한 필수 도구













대표적인 Git 클라이언트 프로그램

GitHub Desktop, TortoiseGit, SourceTree



Git GUI 클라이언트 장단점

장점	vs	단점
 GitHub 통합		 기능 제한
 사용자 친화적인UI		 초보자에게 복잡함
 브랜치 관리		 Windows 전용
 무료 사용		 자원 집약적
 Windows 통합		 초기 설정 복잡성

Git 설치 시 환경 설정 필수 요소

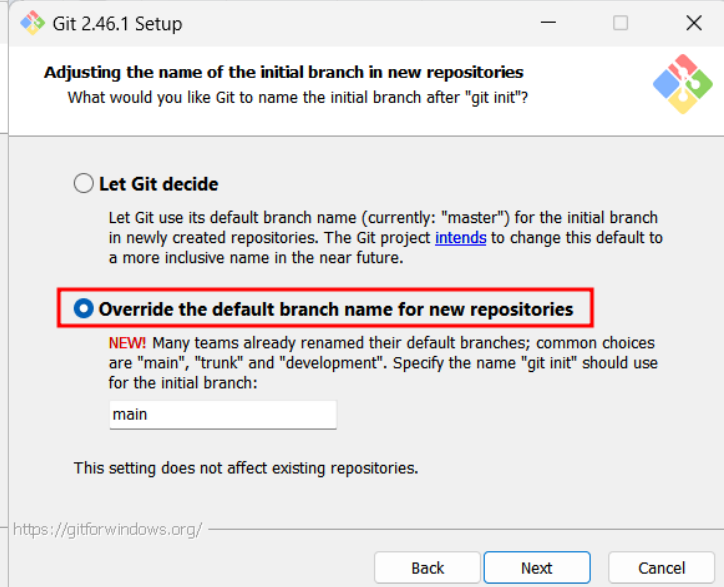
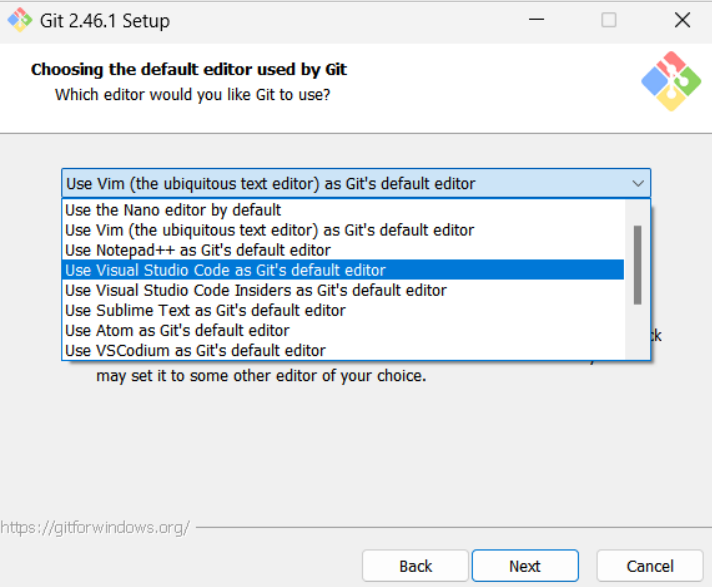
Git 사용을 위한 주요 설정과 옵션들



Git 설치 및 환경 설정

※ 설치 순서 및 유의 사항

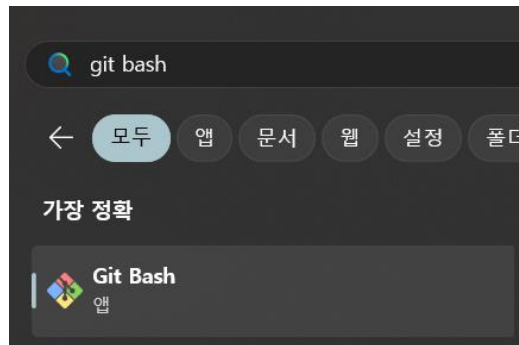
1. VS Code 설치 여부 확인
2. <http://git-scm.com>에 접속해 설치 파일 다운
3. 아래와 같은 설치 탭에서 옵션 변경



Git 설치 여부 확인

Git 버전 확인

1. Git Bash 앱 실행
2. git version 확인
3. git 명령어 동작 확인



```
MINGW64:/c/Users/LifeProfessor

LifeProfessor@DESKTOP-D222TL9 MINGW64 ~
$ git version
git version 2.46.1.windows.1

LifeProfessor@DESKTOP-D222TL9 MINGW64 ~
$ |
```

Git 환경 설정

Git 사용자 정보 설정

1. `git config --global user.name "이름"`
2. `git config --global user.email "메일 주소"`

* `--global` : 현재 컴퓨터에 있는 모든 저장소에서 같은 사용자 정보 사용

```
LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
$ git config --global user.name "LimSeongMin"

LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
$ git config --global user.email "lifeprof@naver.com"
```

* 어떤 사용자가 Commit한지 추적하기 위해서



2. Git으로 버전 관리

- Git 저장소 만들기
- 버전 만들기
- Commit 내용 확인하기
- 버전 단계 별 파일 상태 확인
- Commit 되돌리기



Git 저장소 만들기

Git 저장소 생성 절차

1. 드라이브 선정 → Git 저장소로 사용할 폴더 생성
2. 폴더 열기 → 파일 → PowerShell 열기
3. 명령어 입력 (code .) → VS Code 실행
4. VS Code 터미널 실행 (Ctrl+Shift+`) → Git Bash 열기
5. 명령어 입력 (git init)
6. 폴더 보기 → 숨긴 항목 표시 ON
7. .git 폴더 생성 확인

.git 폴더:

- Git이 관리하는 모든 파일과 메타데이터가 저장되는 곳입니다. 즉, 이 명령을 통해 해당 디렉토리가 Git으로 버전 관리를 할 수 있는 저장소로 변환

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Git-Test> git init
Initialized empty Git repository in D:/Git-Test/.git/
PS D:\Git-Test> █
```

```
LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
$ █
```

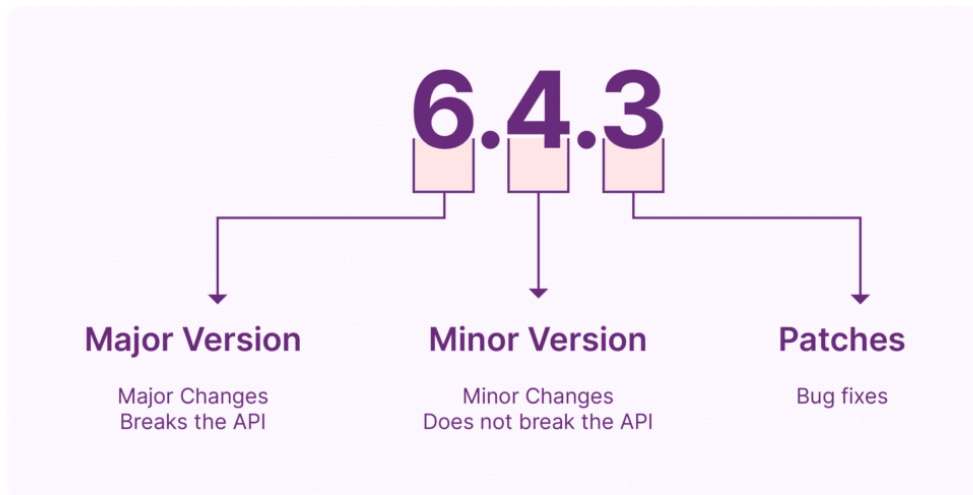


버전(Version)

버전 의미와 표기법

버전 : 소프트웨어 개발에서 **특정 시점의 상태나 변경된 내용**

예시) kakao i 기술 문서 – Release Note



버전 만들기

스테이징과 커밋 이해하기

작업트리(Working Tree)

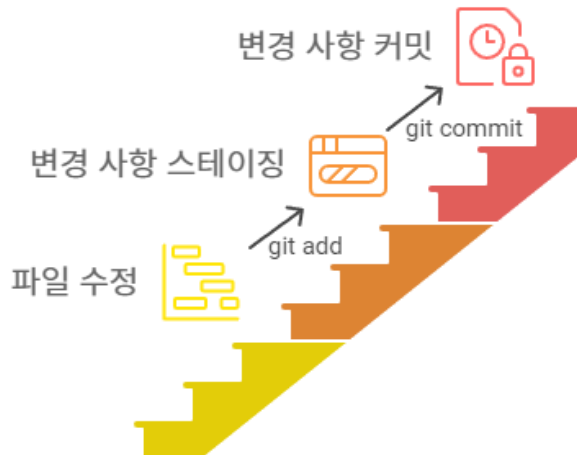
- 현재 사용자가 작업하고 있는 파일들이 위치하는 영역

스테이징(Staging Area)

- 작업 트리에서 수정된 파일 중에서 다음 커밋에 포함시킬 파일들을 임시로 저장하는 영역

저장소(Repository)

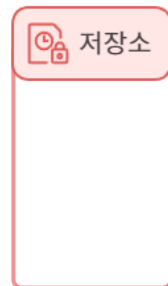
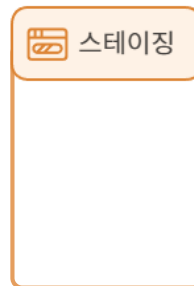
- 실제로 커밋된 내용이 저장되는 장소



Git 상태 확인(1)

git status

1. VS Code로 프로젝트 열기
2. 터미널 실행 → Git Bash로 열기
3. git status 입력



PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)

\$ git status

On branch main 현재 "main" branch에 있다

No commits yet 아직 이 저장소에 아무런 commit이 없다.

nothing to commit (create/copy files and use "git add" to track)

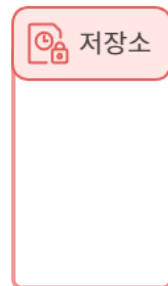
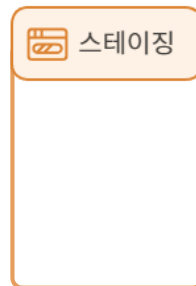
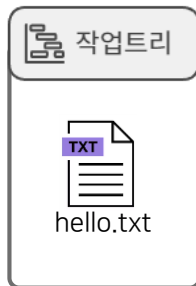
Git이 추적할 파일이 없다



Git 상태 확인(2)

파일 생성 후 git status

1. hello.txt 파일 생성
2. hello.txt 파일에
[자기소개], [각오] 작성 후 저장
3. git status 입력



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
$ git status
On branch main 현재 "main" branch에 있다

No commits yet 아직 이 저장소에 아무런 commit이 없다.

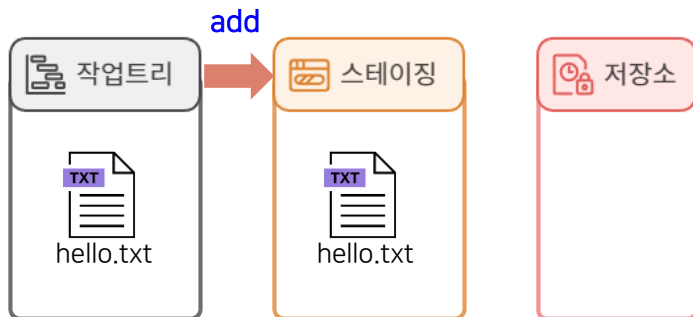
Untracked files: 버전을 아직 한 번도 관리하지 않은 파일
(use "git add <file>..." to include in what will be committed)
hello.txt

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

스테이징에 변경 사항 업로드

git add

1. git add hello.txt 입력
2. git status 입력



```
LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
$ git status
On branch main 현재 "main" branch에 있다

No commits yet 아직 이 저장소에 아무런 commit이 없다.

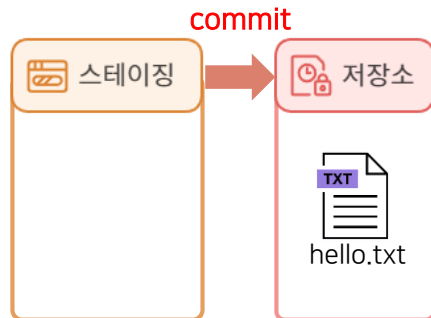
Changes to be committed: 커밋될 예정인 변경 사항
  (use "git rm --cached <file>..." to unstage)
    new file:   hello.txt 새 파일 hello.txt가 스테이징 영역에 있고
                                commit 준비 완료
```

저장소에 스테이징 자료 이동

git commit

1. git commit -m "first commit" 입력

- * commit : 확정하다, 기록하다
- * -m 옵션: 커밋 메시지 작성
- * git commit -m "커밋 메시지"

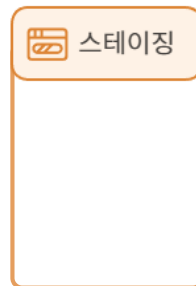


```
LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
$ git commit -m "first commit"
[main (root-commit) 20d2199] first commit
1 file changed, 5 insertions(+) 1개 파일 변경, 5줄 삽입
create mode 100644 hello.txt
```

Git 상태 및 로그 확인

git status / git log

1. git status 입력
2. git log 입력
3. .git/COMMIT_EDITMSG 파일 확인



```
LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
$ git status
On branch main 현재 "main" branch에 있다
nothing to commit, working tree clean commit 할 파일 없음,
작업 트리 수정사항 없이 깨끗

LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
$ git log
commit 20d2199a238e9633e99a3a4b9257b813d210802f (HEAD -> main)
Author: LimSeongMin <lifeprof@naver.com> commit Hash / commit ID
Date: Thu Sep 26 14:33:12 2024 +0900

first commit commit 메시지
```

Hash:

- 임의의 데이터를 고정된 길이로 변환하는 단방향 함수

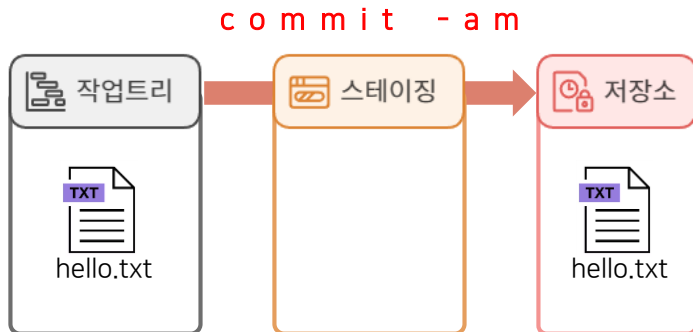
- 각 커밋은 해시 값(SHA-1)으로 고유하게 식별

스테이징과 커밋 한번에 실행

git commit -am

1. hello.txt 파일 내용 수정
2. git commit -am "second commit" 입력

* 이 방법은 1회라도 커밋한 적이 있는 파일만 가능



```
LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
$ git commit -am "second commit"
[main 1ffa769] second commit
1 file changed, 2 insertions(+), 1 deletion(-)
```

1개 파일 수정, 2개 라인 추가, 1개 라인 제거

Git 파일의 변경 사항 비교

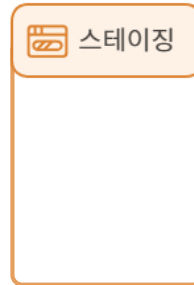
git diff

1. hello.txt 파일 내용 변경 후 저장

2. git diff 입력

* 빨간색(-): 삭제된 내용

초록색(+): 추가된 내용



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
$ git diff
```

```
LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
```

```
$ git diff
```

```
diff --git a/hello.txt b/hello.txt
```

```
index e22d9c4..d3e630d 100644
```

```
--- a/hello.txt
```

```
+++ b/hello.txt
```

```
@@ -3,4 +3,4 @@
```

```
[각오]
```

```
Git&GitHub 특강을 듣고 나만의 포트폴리오를 만들어보겠습니다.
```

```
-열심히 하겠습니다!
```

삭제된 내용

```
\ No newline at end of file
```

```
+정말 열심히 하겠습니다!
```

추가된 내용

```
\ No newline at end of file
```

4 [각오]

5 Git&GitHub 특강을 듣고 나만의 포트폴리오를 만들어보겠습니다.

6 정말 열심히 하겠습니다!

hello.txt Git local working changes - 1 of 1 change

5 5 Git&GitHub 특강을 듣고 나만의 포트폴리오를 만들어보겠습니다.

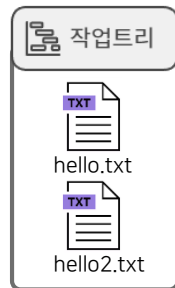
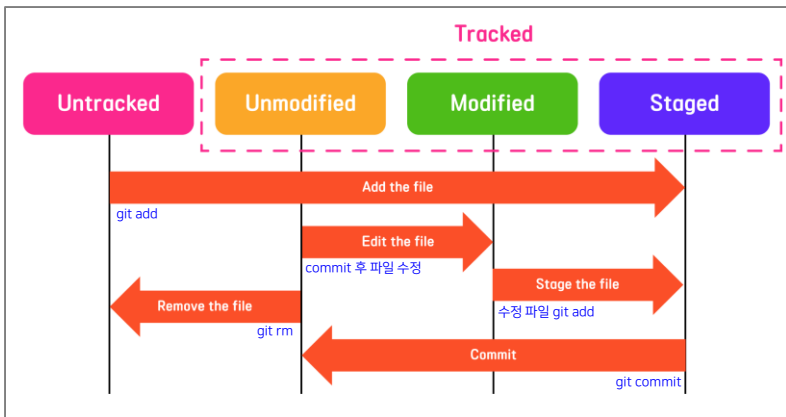
6 열심히 하겠습니다!

6 정말 열심히 하겠습니다!



버전 단계 별 파일 상태 확인(1)

Tracked vs Untracked 파일



Tracked	Untracked	<ul style="list-style-type: none"> - Git에서 추적하고 있지 않은 파일 - 생성 후에 한 번도 git add 한 적이 없는 파일
	Unmodified	<ul style="list-style-type: none"> - Commit을 하고 난 직후의 파일 - 최신 버전과 비교했을 때 변경된 부분이 없는 파일
	Modified	<ul style="list-style-type: none"> - Commit을 하고 내용을 수정한 파일 - 최신 버전과 비교했을 때 변경된 부분이 있는 파일
	Staged	<ul style="list-style-type: none"> - Staging Area에 존재하는 파일 - 새로 생성했거나 수정한 파일을 git add로 추가한 상태

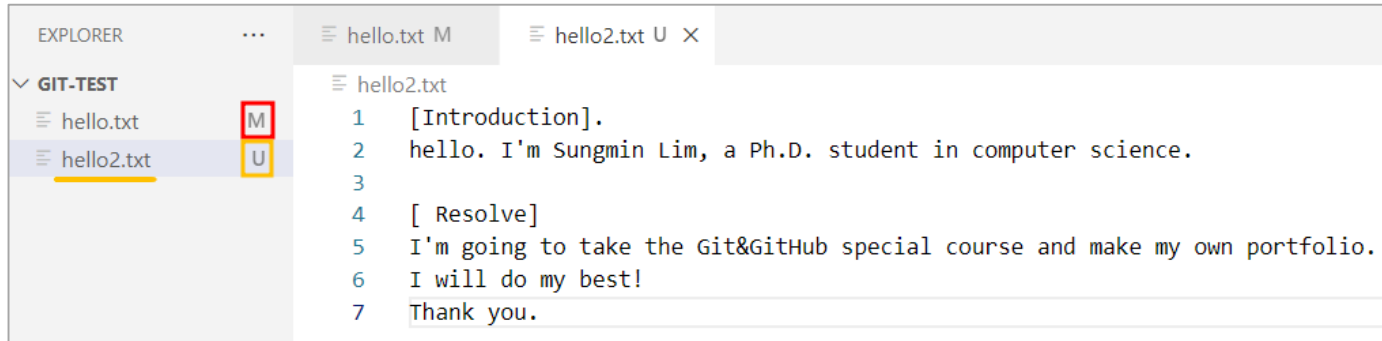
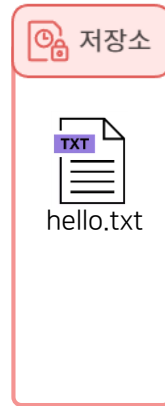
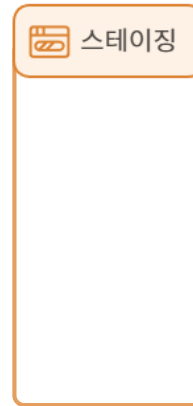
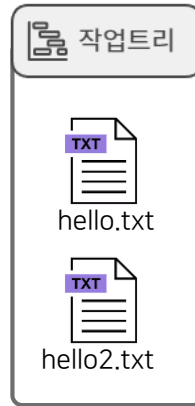
출처: <https://heina-fantasy.tistory.com/256>



버전 단계 별 파일 상태 확인(2)

Tracked vs Untracked 파일

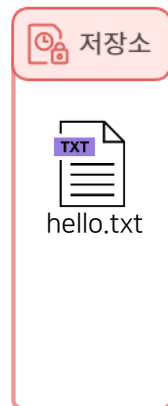
1. hello.txt 파일 내용 변경 후 저장
2. hello2.txt 파일 생성
3. hello2.txt 파일 내용 변경 후 저장



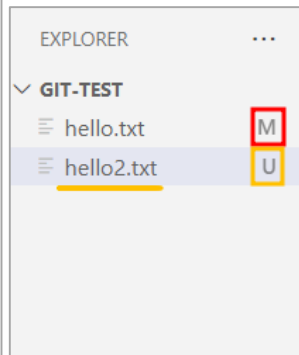
버전 단계 별 파일 상태 확인(3)

Tracked vs Untracked 파일

4. git status 입력



```
LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   hello.txt
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    hello2.txt
no changes added to commit (use "git add" and/or "git commit -a")
```

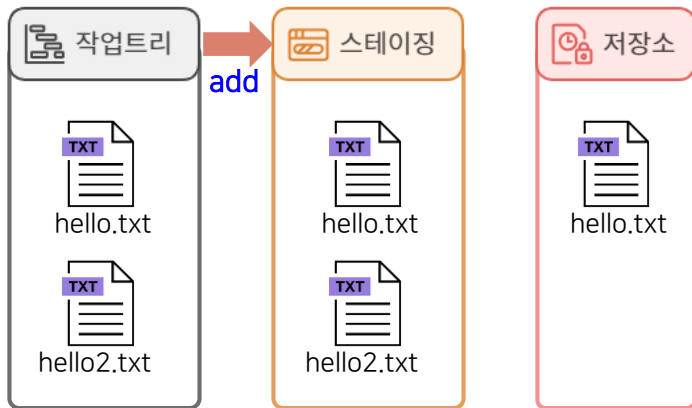


작업트리 수정 파일 한 번에 스테이징 업로드

git add .

1. git add . 입력

* (.)은 현재 폴더 내의 수정된 파일 전부 업로드



```
LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
$ git add .
```

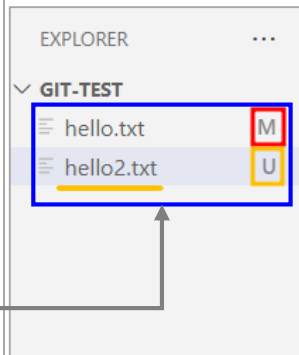
```
LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
$ git status
```

On branch main

Changes to be committed: **commit 될 변경 사항**

(use "git restore --staged <file>..." to unstage)

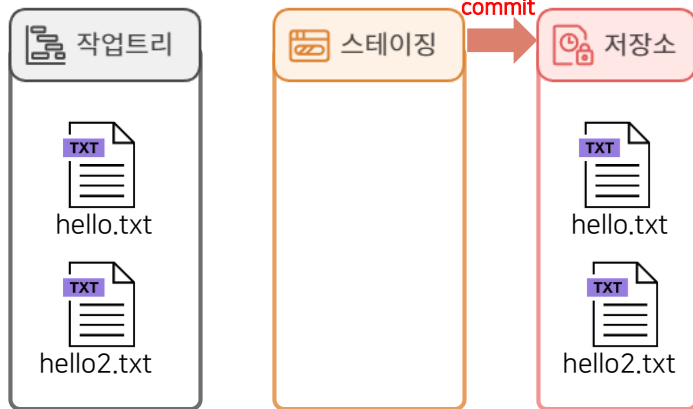
```
modified:   hello.txt
new file:   hello2.txt
```



세 번째 commit 실행

git commit

1. git commit -m "third commit"



```
LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
$ git commit -m "third commit"
[main 0a3325e] third commit
2 files changed, 9 insertions(+), 1 deletion(-)
create mode 100644 hello2.txt
```

Git 로그 확인

git log

1. git log 입력

```
LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
$ git log
commit 0a3325ed5b764e19c0654d7742d7fe0824c8b056 (HEAD -> main)
Author: LimSeongMin <lifeprof@naver.com>
Date: Thu Sep 26 15:28:11 2024 +0900

    third commit

commit 1ffa7690b2896498b2ae31f62a0599c79797a5df
Author: LimSeongMin <lifeprof@naver.com>
Date: Thu Sep 26 14:50:18 2024 +0900

    second commit

commit 20d2199a238e9633e99a3a4b9257b813d210802f
Author: LimSeongMin <lifeprof@naver.com>
Date: Thu Sep 26 14:33:12 2024 +0900

    first commit
```



Git 로그 상세 확인

git log --stat

1. git log --stat 입력

* stat: statistics 약어

```
LifeProfessor@DESKTOP-D222TL9 MINGW64 /d/Git-Test (main)
$ git log --stat
commit 0a3325ed5b764e19c0654d7742d7fe0824c8b056 (HEAD -> main)
Author: LimSeongMin <lifeprof@naver.com>
Date: Thu Sep 26 15:28:11 2024 +0900

    third commit

    hello.txt | 3 ++-      2줄 추가, 1줄 삭제
    hello2.txt | 7 ++++++  7줄 추가
    2 files changed, 9 insertions(+), 1 deletion(-)

commit 1ffa7690b2896498b2ae31f62a0599c79797a5df
Author: LimSeongMin <lifeprof@naver.com>
Date: Thu Sep 26 14:50:18 2024 +0900

    second commit

    hello.txt | 3 ++-
    1 file changed, 2 insertions(+), 1 deletion(-)

commit 20d2199a238e9633e99a3a4b9257b813d210802f
Author: LimSeongMin <lifeprof@naver.com>
Date: Thu Sep 26 14:33:12 2024 +0900

    first commit

    hello.txt | 5 +++++
    1 file changed, 5 insertions(+)
```



버전 관리 제외 방법

.gitignore

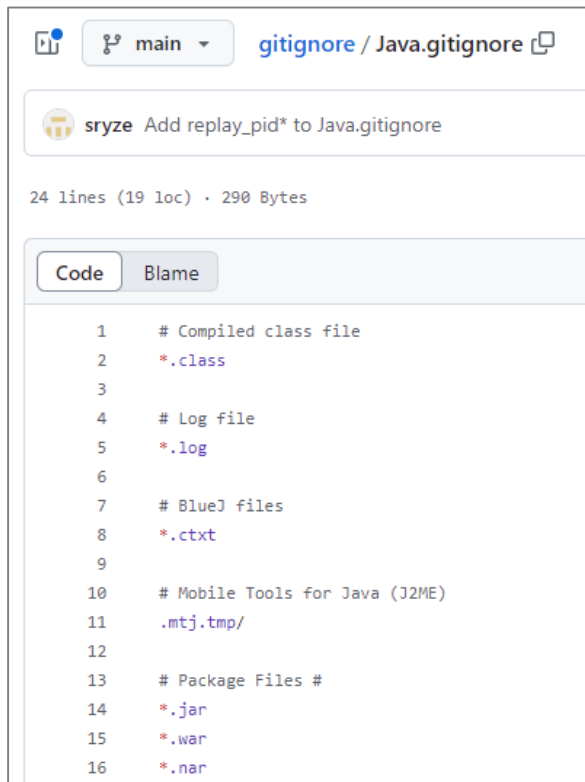
1. .gitignore 파일 생성

2. 버전 관리하지 않을

파일명/폴더명/파일 확장자 입력



.gitignore 파일 작성해주는 사이트



GitHub의 .gitignore Java 예시 일부

