

Git과 GitHub 고급 (1)

Git의 설치, Git과 Git의 Concept 소개

이 강의를 듣기에 좋은 사람

- Git으로 Commit을 1회 이상 해 본 경험이 있는 사람.
- Git에 대해 더 자세히 알고 싶은 사람.

주차 별 활동 계획 (1안)

주차	날짜	주제
1주차	9/25 (수)	Git 기초 (1): Git의 설치, Git과 Git의 Concept 소개
2주차	10/2 (수)	Git 기초 (2): 브랜치 분기와 원격 저장소 & 자주 사용되는 Git 명령어
3주차	10/16 (수)	Git 협업 (3): 브랜치 병합과 GitFlow, 그리고 GitHub Issue
4주차	10/30 (수)	Git 협업 (4): GitHub Pull Request & Code Review
5주차	11/6 (수)	Git 심화 (5): 병합 전략 및 병합 충돌 해결 전략 (Git Rebase 명령 알아보기)
6주차	11/13 (수)	Git 심화 (6): Git의 자료구조와 실수로 삭제한 커밋 복구하기
7주차	11/20 (수)	Git 응용 (7): GitHub Actions를 활용한 자동화 워크플로 생성
8주차	11/27 (수)	Git 응용 (8): Git Submodule & Git Hook (로컬에서의 자동화)

중간고사 전 주에 멘토링을 진행해야 함.

주차 별 활동 계획 (2안)

주차	날짜	주제
1주차	9/25 (수)	Git 기초 (1): Git의 설치, Git과 Git의 Concept 소개
2주차	10/2 (수)	Git 기초 (2): 브랜치 분기와 원격 저장소 & 자주 사용되는 Git 명령어
3주차	10/30 (수)	Git 협업 (3): 브랜치 병합과 GitFlow, 그리고 GitHub Issue
4주차	11/6 (수)	Git 협업 (4): GitHub Pull Request & Code Review
5주차	11/13 (수)	Git 심화 (5): 병합 전략 및 병합 충돌 해결 전략 (Git Rebase 명령 알아보기)
6주차	11/20 (수)	Git 심화 (6): Git의 자료구조와 실수로 삭제한 커밋 복구하기
7주차	11/27 (수)	Git 응용 (7): GitHub Actions를 활용한 자동화 워크플로 생성
8주차	12/4 (수)	Git 응용 (8): Git Submodule & Git Hook (로컬에서의 자동화)

기말고사 전 주에 멘토링을 진행해야 함.

목차

1. Git의 설치 및 Github 계정 생성
2. Git과 Git의 Concept 소개
3. 파일의 LifeCycle
4. 실습
5. FAQ

이번 회차의 목표: 멘티가 과제를 수행할 수 있는 역량을 기르는 것

1. Git의 설치 및 GitHub 계정 생성

멘토링을 시작하기에 앞서...

다음 프로그램을 설치/가입했다고 가정합니다.



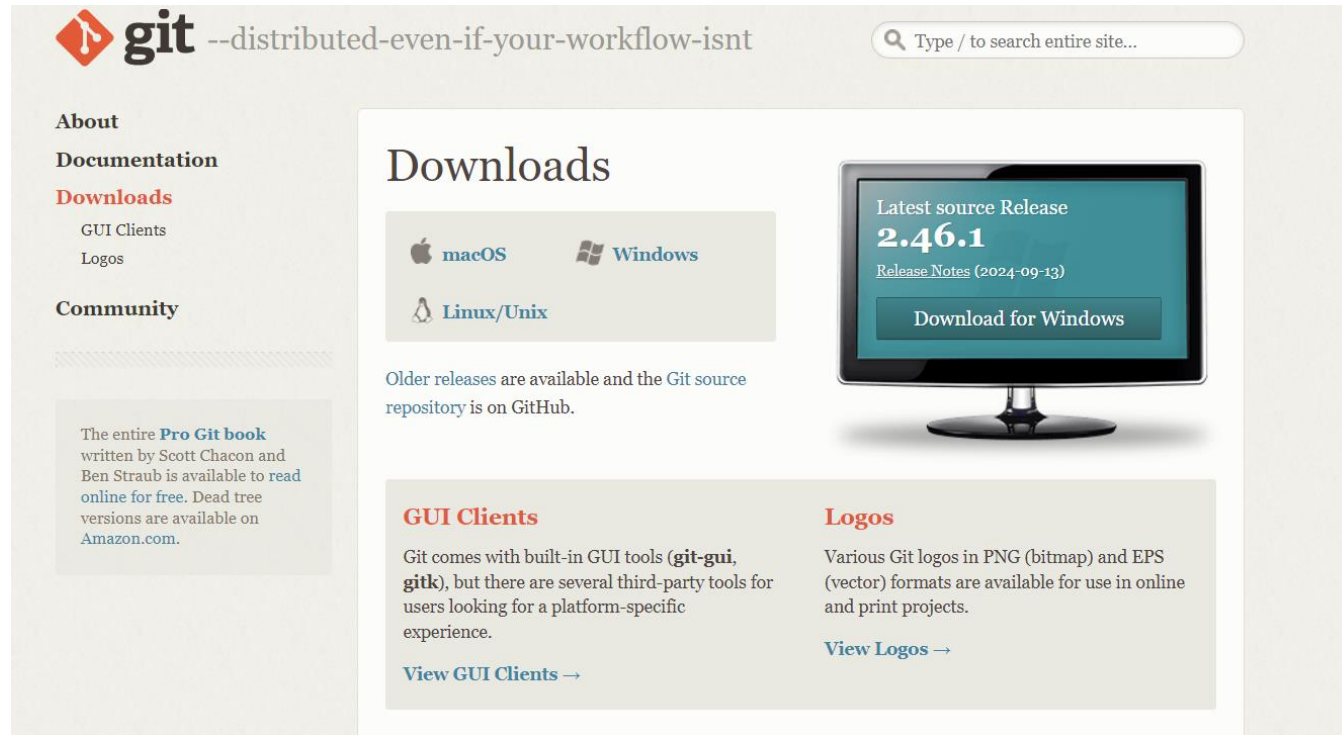
VS Code



Baekjoon Online Judge

설치/가입되어 있지 않은 사람은 별첨 부록을 확인해 주세요

Git 설치 (윈도우 기준)

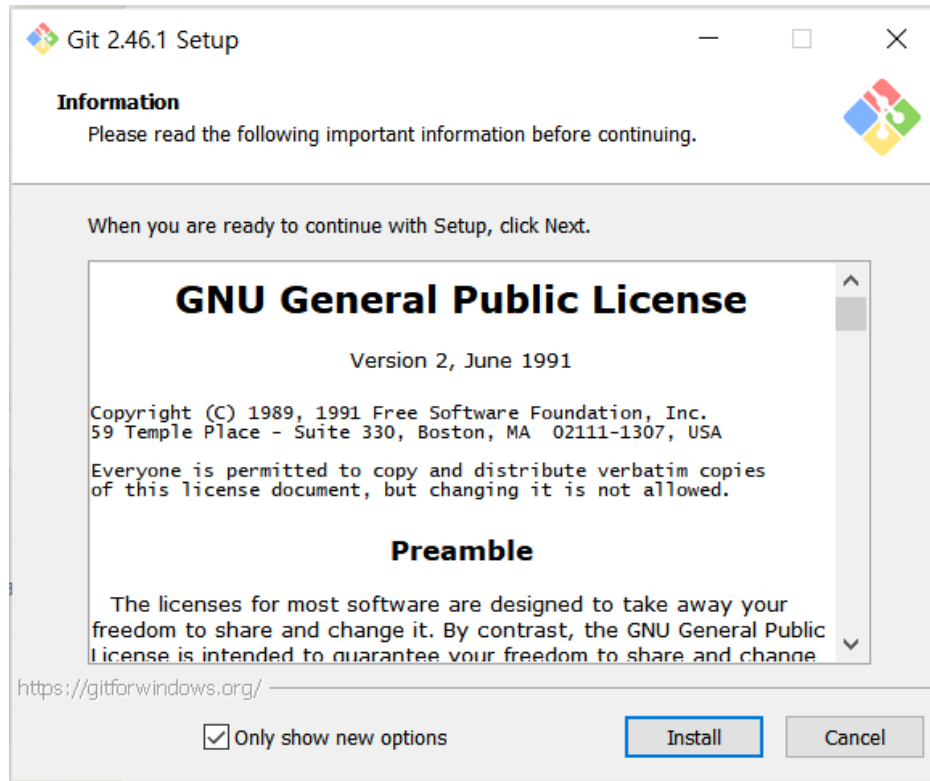


<https://git-scm.com/downloads>

1. Git 다운로드 페이지에 방문합니다.
2. 윈도우 버튼을 클릭합니다.



Git 설치 (윈도우 기준)



3. Setup 파일을 실행합니다.
4. Install 버튼을 누르고,
설치가 완료될 때까지 기다립니다.



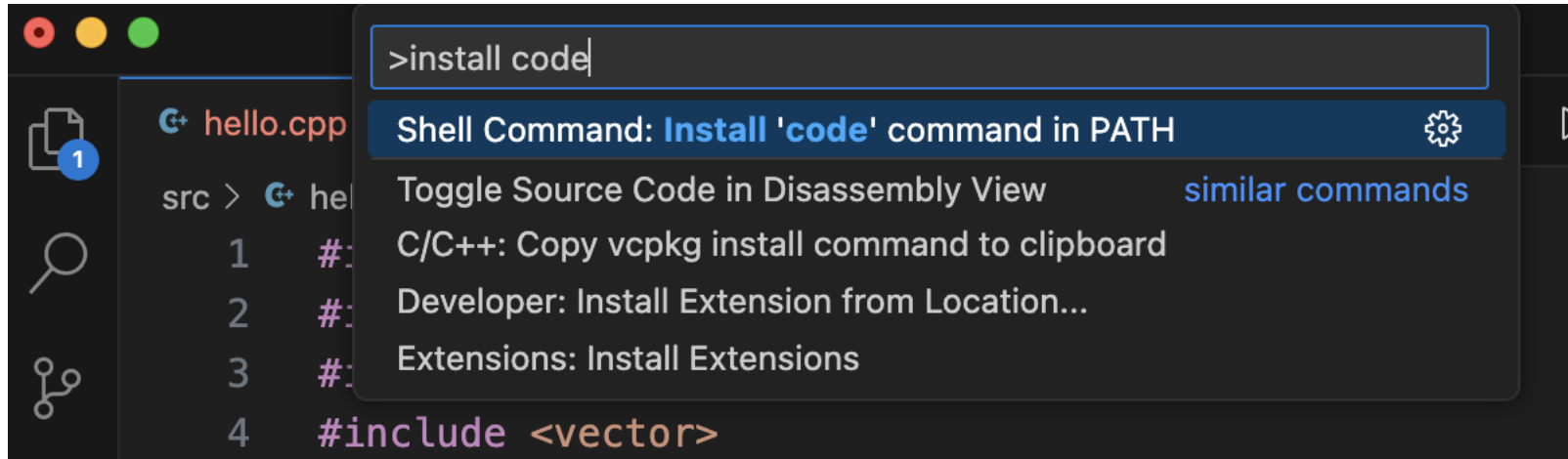
GitHub 계정 생성

<https://github.com>

1. GitHub 홈페이지로 이동
2. 앞으로 로그인할 때 사용할 이메일을 작성 후, 인증



환경 변수에 code(Vscode) 추가



1. Vscode로 아무 파일이나 엽니다.
2. 운영체제에 따라 아래의 키를 누릅니다.
Window os: ctrl + shift + p
Mac os: command + shift + p
3. '>' 옆에 "install code"를 입력합니다.

2. Git 과 Git의 Concept 소개

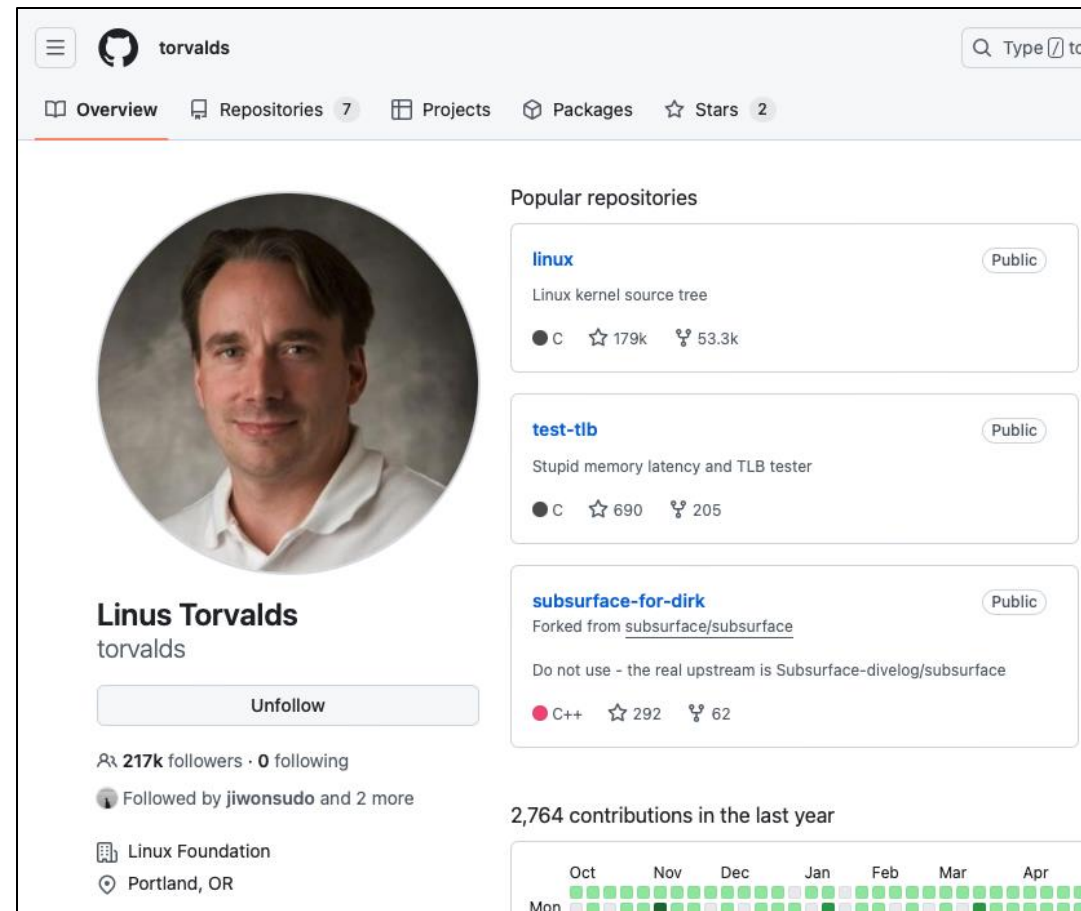
- 1주차에서는 로컬 저장소에서 한 명의 사용자가 작업 하는 것을 진행됩니다.
- 원격 저장소 및 협업을 위한 Git은 2주차에서 자세하게 소개될 예정입니다.

Git이란?

- 2005년에 Linux 커널 개발을 위해 초기 개발에 기여한 다른 커널 개발자들과 함께 **리누스 토르발스**(Linus Torvalds)가 처음 개발.
- GNU 일반 공중 사용 허가서(GNU General Public License) v2 하에 배포되는 자유 **오픈소스 소프트웨어**.

GNU 일반 공중 사용 허가서란?

1. 컴퓨터 프로그램을 어떤 목적으로든지 사용할 수 있다
2. 컴퓨터 프로그램의 복사를 언제나 프로그램의 코드와 함께 판매 또는 무료로 배포할 수 있다
3. 컴퓨터 프로그램의 코드를 용도에 따라 결정할 수 있다
4. 변경된 컴퓨터 프로그램 역시 프로그램의 코드와 함께 자유로이 배포할 수 있다라는 네 가지 조항을 명시하고 있다.



The screenshot shows the GitHub profile of Linus Torvalds. At the top, there's a navigation bar with 'torvalds' and a search bar. Below the navigation bar, there are tabs for 'Overview', 'Repositories' (7), 'Projects', 'Packages', and 'Stars' (2). The main profile section features a circular profile picture of Linus Torvalds, his name 'Linus Torvalds', and his GitHub handle 'torvalds'. Below this is an 'Unfollow' button. The profile also shows '217k followers · 0 following' and 'Followed by jiwonsudo and 2 more'. The location is listed as 'Linux Foundation, Portland, OR'. To the right of the profile picture, there's a section titled 'Popular repositories' which lists three repositories: 'linux' (Linux kernel source tree, 179k stars, 53.3k forks), 'test-tlb' (Stupid memory latency and TLB tester, 690 stars, 205 forks), and 'subsurface-for-dirk' (Forked from subsurface/subsurface, 292 stars, 62 forks). At the bottom right, there's a section titled '2,764 contributions in the last year' with a calendar visualization showing contributions from October to April.

Git이란?

- 깃(Git)은 컴퓨터 **파일의 변경사항**을 추적하고 **여러 명의 사용자들** 간에 해당 파일들의 작업을 조율하기 위한 **스냅샷 스트림 기반의 분산 버전 관리 시스템**.

```

#####

Git - the stupid content tracker

#####



"git" can mean anything, depending on your mood.

- random three-letter combination that is pronounceable, and not
  actually used by any common UNIX command. The fact that it is a
  mispronunciation of "get" may or may not be relevant.
- stupid. contemptible and despicable. simple. Take your pick from the
  dictionary of slang.
- "global information tracker": you're in a good mood, and it actually
  works for you. Angels sing, and a light suddenly fills the room.
- "goddamn idiotic truckload of sh*t": when it breaks

Git is a fast, scalable, distributed revision control system with an
unusually rich command set that provides both high-level operations
and full access to internals.

Git is an Open Source project covered by the GNU General Public
License version 2 (some parts of it are under different licenses,
compatible with the GPLv2). It was originally written by Linus
Torvalds with help of a group of hackers around the net.
```

Git 개발 저장소의 README.md 머릿말

git
noun [C] UK informal
UK  /git/ US  /git/ [Add to word list](#)

a person, especially a man, who is stupid or unpleasant:

- *You stupid/lying git!*
- *He's a miserable old git.*

동의어들
rotter mainly UK old-fashioned
skunk
stinker old-fashioned informal

(Cambridge Advanced Learner's Dictionary & Thesaurus의 git 정의 © Cambridge University Press)

캠브리지 어학 사전에 정의된 Git

Git이란? (4 Key Words)

- 깃(Git)은 컴퓨터 **파일의 변경사항**을 추적하고 **여러 명의 사용자들** 간에 해당 파일들의 작업을 조율하기 위한 **스냅샷 스트림** 기반의 **분산 버전 관리 시스템**.

파일의 변경사항

여러 명의 사용자들

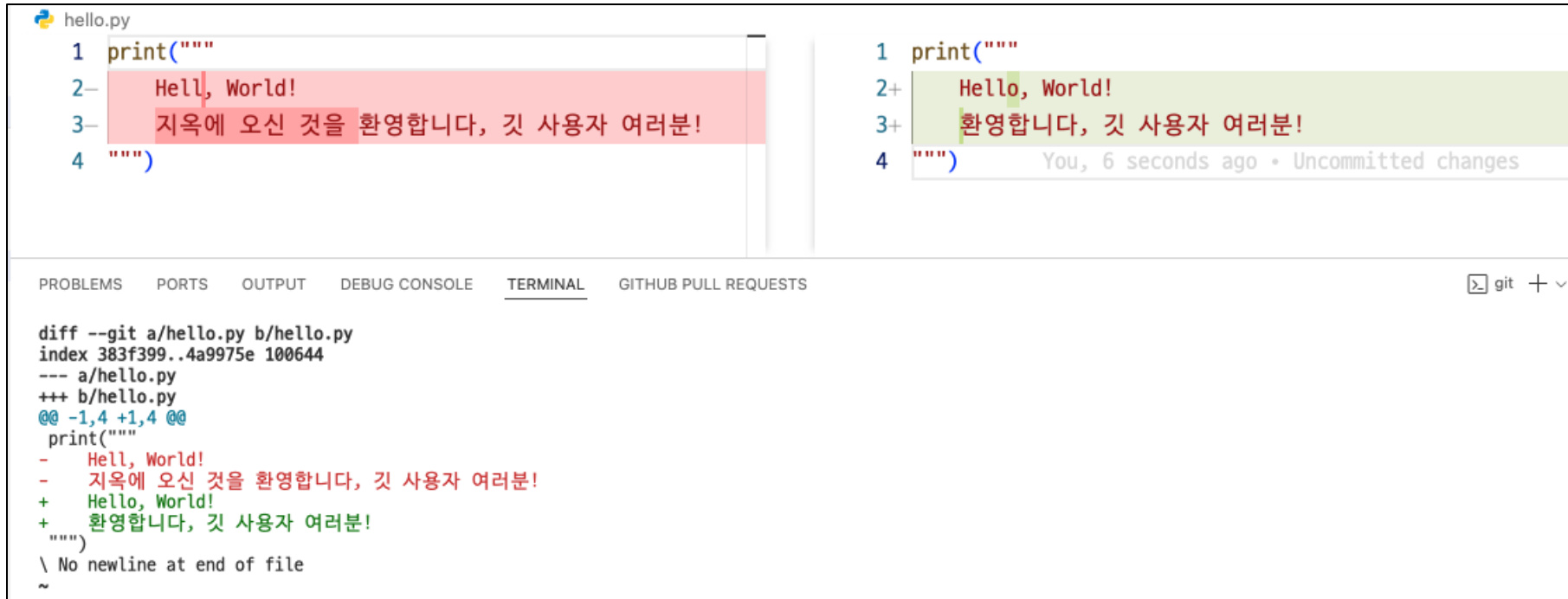
스냅샷 스트림

분산 버전 관리 시스템

Git의 네 가지 키워드를 중심으로 Git이 무엇인지 디테일하게 알아보시다.

변경사항(Diff)이란?

- Git에서의 **변경사항(diff)**이란, 아래와 같이 두 개 파일 간의 차이점으로, **제거된 내용**과 **추가된 내용**을 아울러 일컫는 말이다.



```
hello.py
1 print("""
2-     Hell, World!
3-     지옥에 오신 것을 환영합니다, 깃 사용자 여러분!
4 """)

1 print("""
2+     Hello, World!
3+     환영합니다, 깃 사용자 여러분!
4 """) You, 6 seconds ago • Uncommitted changes

PROBLEMS  PORTS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITHUB PULL REQUESTS  git + v

diff --git a/hello.py b/hello.py
index 383f399..4a9975e 100644
--- a/hello.py
+++ b/hello.py
@@ -1,4 +1,4 @@
 print("""
-     Hell, World!
-     지옥에 오신 것을 환영합니다, 깃 사용자 여러분!
+     Hello, World!
+     환영합니다, 깃 사용자 여러분!
 """)
\ No newline at end of file
~
```

파일의 변경사항

여러 명의 사용자들

스냅샷 스트림

분산 버전 관리 시스템

버전 관리(Version Control)란?

- 파일 변화(Diff)를 시간에 따라 기록했다가 나중에 특정 시점의 **버전**을 다시 꺼내어 올 수 있는 시스템.
- 버전 관리 시스템(VCS, Version Control System)을 사용하면, 큰 노력 없이...
 - 각 파일을 이전 상태로 되돌릴 수 있다.
 - 시간에 따라 수정 내용을 비교해 볼 수 있다.
 - 누가 문제를 일으켰는지 추적할 수 있다. (Git Blame 명령 사용)
 - 언제 만들어진 이슈인지 알 수 있다.
 - 파일을 잃어버리거나 잘못 고쳤을 때 쉽게 복구할 수 있다.

여러 명의 개발자들이
협업하여 작업하려면
필수인 기능들

파일의 변경사항

여러 명의 사용자들

스냅샷 스트림

분산 버전 관리 시스템

스냅샷(Snapshot) 이란?

- In Git, a **snapshot** refers to a complete representation of the **state of a project** (or a set of files) **at a specific point in time**.
 - 스냅샷은 특정 시점의 프로젝트(혹은 파일 집합)의 상태
- Each snapshot captures all the files and their contents as they exist in the repository at that moment, allowing users to track changes over time.



컴퓨터 파일 시스템에서 스냅샷은 과거의 한 때 존재하고 유지시킨 컴퓨터 파일과 디렉터리의 모임이다. 이 용어는 사진술에서 말하는 스냅샷에서 비롯한 말이다. [위키백과](#)

파일의 변경사항

여러 명의 사용자들

스냅샷 스트림

분산 버전 관리 시스템

Snapshot vs Diff(Delta) 의 차이

SNAPSHOT 을 저장하는 경우

docs > 📄 애국가.md	docs > 📄 애국가.md
1 1. 동해물과 백두산이 마르고 닳도록	1 1. 동해물과 백두산이 마르고 닳도록
2 하느님이 보우하사 우리나라 만세	2 하느님이 보우하사 우리나라 만세
3 무궁화 삼천리 화려 강산	3 무궁화 삼천리 화려 강산
4 대한 사람 대한으로 길이 보전하세	4 대한 사람 대한으로 길이 보전하세
5 2. 남산 위에 저 소나무 철갑을 두른 듯	5 2. 남산 위에 저 소나무 철갑을 두른 듯
6 무궁화 삼천리 화려 강산	6 바람 서리 불변함은 우리 기상일세
7 대한 사람 대한으로 길이 보전하세	7 무궁화 삼천리 화려 강산
8 3. 가을 하늘 공활한데 높고 구름 없이	8 대한 사람 대한으로 길이 보전하세
9 밝은 달은 우리 가슴 일편단심일세	9 3. 가을 하늘 공활한데 높고 구름 없이
10 무궁화 삼천리 화려 강산	10 밝은 달은 우리 가슴 일편단심일세
11 대한 사람 대한으로 길이 보전하세	11 무궁화 삼천리 화려 강산
12 4. 이 기상과 이 맘으로 충성을 다하여	12 대한 사람 대한으로 길이 보전하세
13 괴로우나 즐거우나 나라 사랑하세	13 4. 이 기상과 이 맘으로 충성을 다하여
14 무궁화 삼천리 화려 강산	14 괴로우나 즐거우나 나라 사랑하세
15 대한 사람 대한으로 길이 보전하세	15 무궁화 삼천리 화려 강산
	16 대한 사람 대한으로 길이 보전하세

277 Characters

296 Characters

(snapshot) +277 Characters

(snapshot) +296 Characters

Total 850 Characters in Memory

DIFF 를 저장하는 경우

docs > 📄 애국가.md	docs > 📄 애국가.md
1 1. 동해물과 백두산이 마르고 닳도록	1 1. 동해물과 백두산이 마르고 닳도록
2 하느님이 보우하사 우리나라 만세	2 하느님이 보우하사 우리나라 만세
3 무궁화 삼천리 화려 강산	3 무궁화 삼천리 화려 강산
4 대한 사람 대한으로 길이 보전하세	4 대한 사람 대한으로 길이 보전하세
5 2. 남산 위에 저 소나무 철갑을 두른 듯	5 2. 남산 위에 저 소나무 철갑을 두른 듯
6 무궁화 삼천리 화려 강산	6+ 바람 서리 불변함은 우리 기상일세
7 대한 사람 대한으로 길이 보전하세	7 무궁화 삼천리 화려 강산
8 3. 가을 하늘 공활한데 높고 구름 없이	8 대한 사람 대한으로 길이 보전하세
9 밝은 달은 우리 가슴 일편단심일세	9 3. 가을 하늘 공활한데 높고 구름 없이
10 무궁화 삼천리 화려 강산	10 밝은 달은 우리 가슴 일편단심일세
11 대한 사람 대한으로 길이 보전하세	11 무궁화 삼천리 화려 강산
12 4. 이 기상과 이 맘으로 충성을 다하여	12 대한 사람 대한으로 길이 보전하세
13 괴로우나 즐거우나 나라 사랑하세	13 4. 이 기상과 이 맘으로 충성을 다하여
14 무궁화 삼천리 화려 강산	14 괴로우나 즐거우나 나라 사랑하세
15 대한 사람 대한으로 길이 보전하세	15 무궁화 삼천리 화려 강산
	16 대한 사람 대한으로 길이 보전하세

277 Characters

296 Characters

(snapshot) +277 Characters

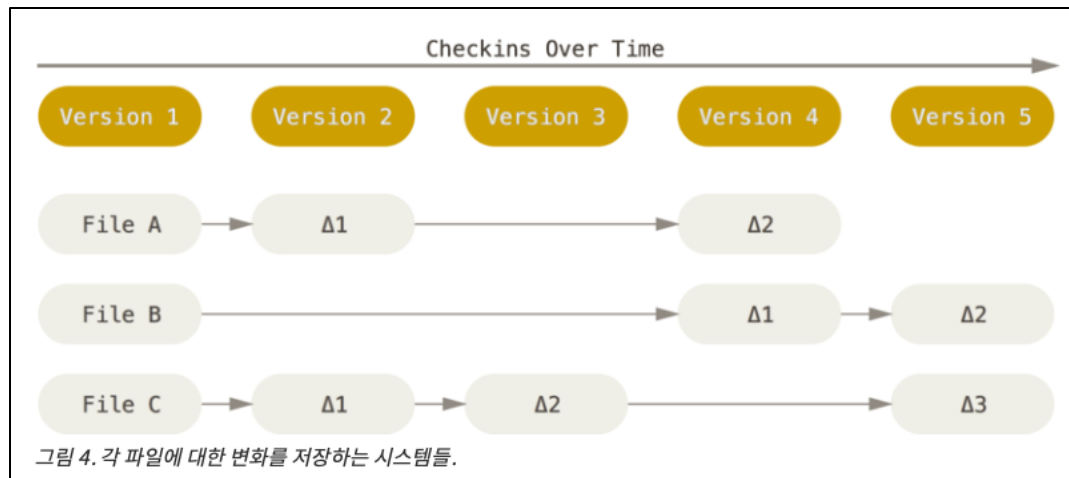
(delta) +19 Characters

Total 296 Characters in Memory

Git, 차이가 아니라 스냅샷

- Subversion(혹은 다른 VCS) 같은 프로그램과 Git의 가장 큰 차이점은 데이터를 다루는 방식.
 - CVS, Subversion, Perforce, Bazaar 등의 시스템은 파일의 변화(Diff)를 시간순으로 관리. (Git은 조금 다르다.)
 - 각 파일의 변화를 시간순으로 추적하여 파일들의 집합을 관리하는 것을 **델타 기반 버전관리 시스템**이라고 함.

*CVS: Concurrent Versioning System



<델타 기반 버전관리>

파일의 변경사항

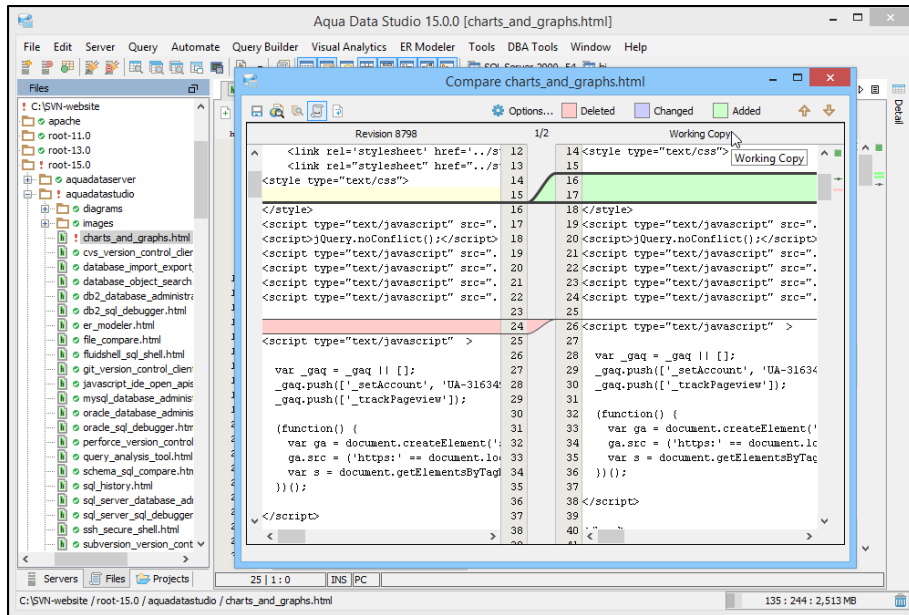
여러 명의 사용자들

스냅샷 스트림

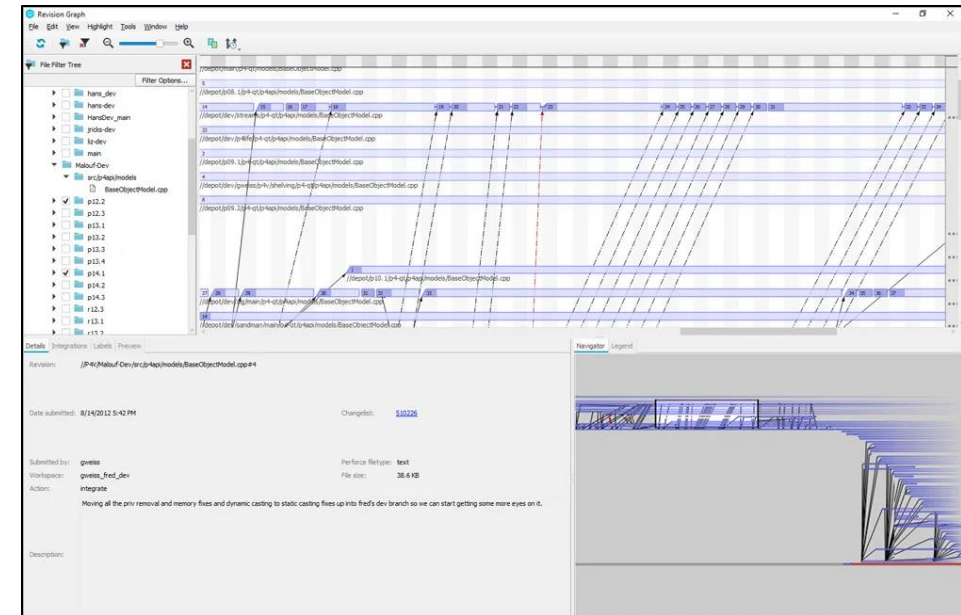
분산 버전 관리 시스템

Git, 차이가 아니라 스냅샷

<델타 기반 버전관리 시스템 클라이언트>



<Aqua Data Studio: Subversion Client>



<Perforce Helix Core>

파일의 변경사항

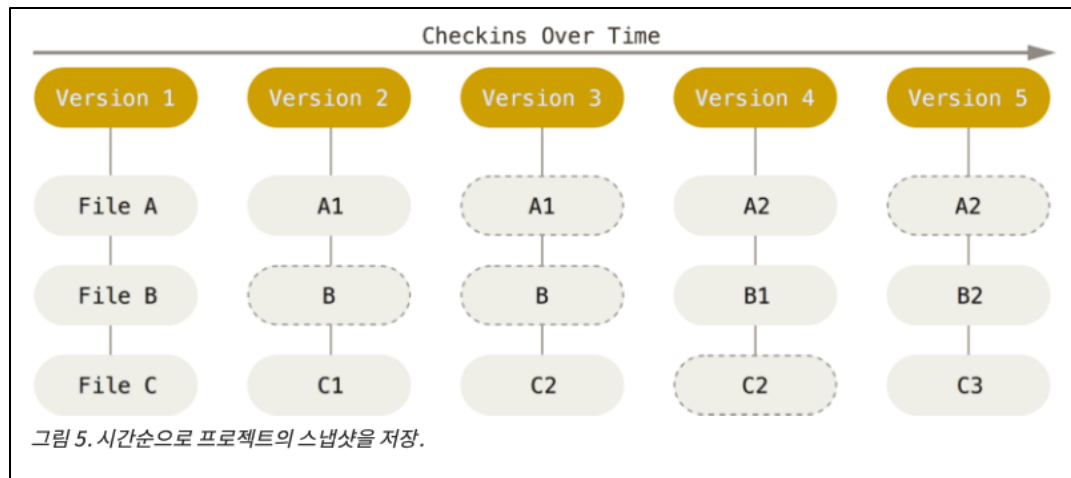
여러 명의 사용자들

스냅샷 스트림

분산 버전 관리 시스템

Git, 차이가 아니라 스냅샷

- Git은 파일이 달라지지 않았으면 성능을 위해 파일을 새로 저장하지 않는다.
 - 단지, 이전 상태의 파일에 대한 링크만 저장한다.
- Git은 **데이터를 스냅샷의 연속으로** 취급하고, 크기가 매우 작다.



<데이터를 시간에 따른 스냅샷의 연속으로 저장>

파일의 변경사항

여러 명의 사용자들

스냅샷 스트림

분산 버전 관리 시스템

Git, 차이가 아니라 스냅샷

- Git은 파일이 달라지지 않았으면 성능을 위해 파일을 새로 저장하지 않는다.
 - 단지, 이전 상태의 파일에 대한 링크만 저장한다.
- Git은 **데이터를 스냅샷의 연속으로** 취급하고 크기가 매우 작다.

Git은 데이터를 스냅샷의 스트림처럼 취급한다.

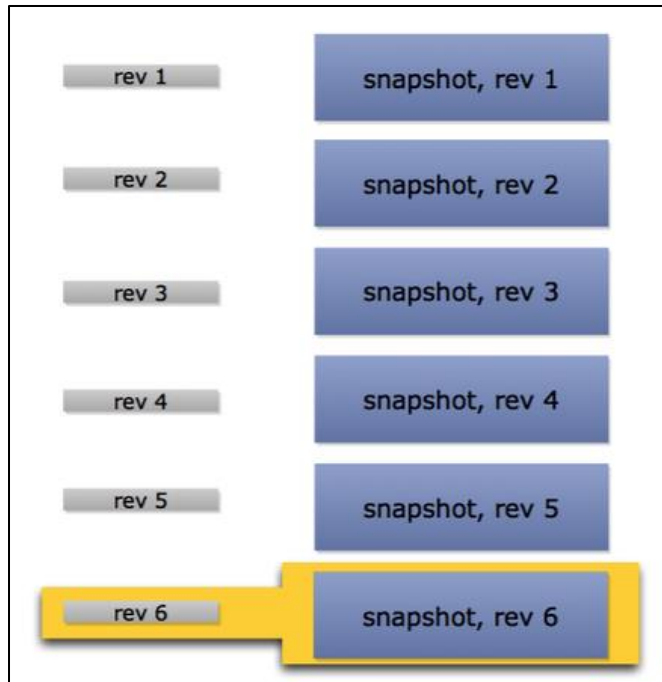
파일의 변경사항

여러 명의 사용자들

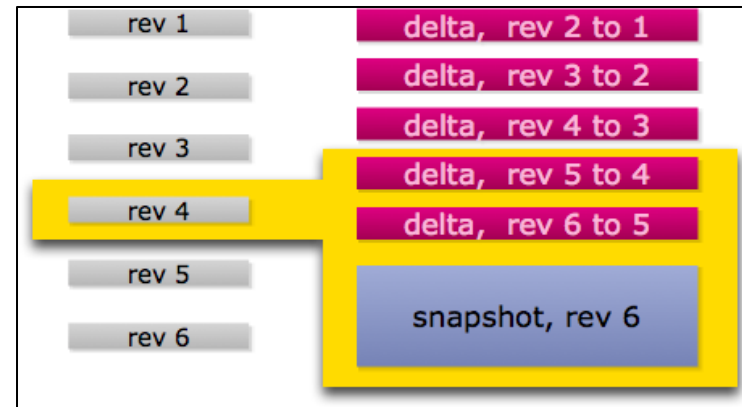
스냅샷 스트림

분산 버전 관리 시스템

Git, 차이가 아니라 스냅샷 (심화)



- Git은 각 시점의 스냅샷을 저장 (revision)



- Git은 Garbage Collector를 적용하여 **델타를 저장하도록** 할 수 있다.
- 가장 최신 버전의 스냅샷을 저장한다.

파일의 변경사항

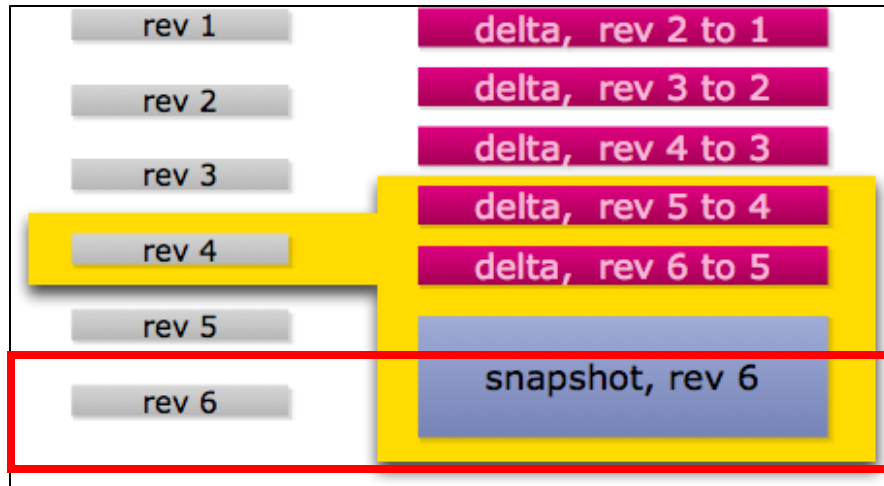
여러 명의 사용자들

스냅샷 스트림

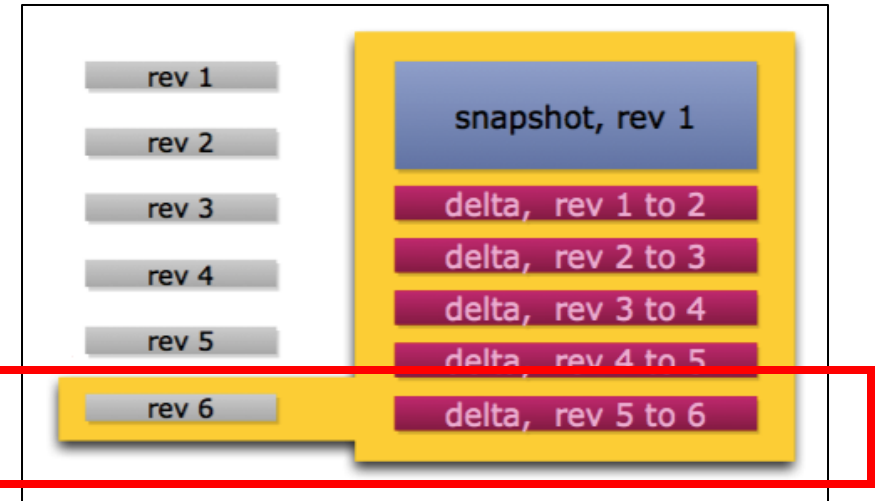
분산 버전 관리 시스템

Git, 차이가 아니라 스냅샷 (심화)

가장 최신 버전을 스냅샷 하는 경우 (e.g. Git)



가장 오래된 버전을 스냅샷 하는 경우 (e.g. Mercurial)



가장 최신 시점의 데이터(revision)가 가장 많이 조회된다.

가장 마지막 시점의 스냅샷을 저장하는 것이
성능 상의 이점으로 이어질 가능성이 높다.

파일의 변경사항

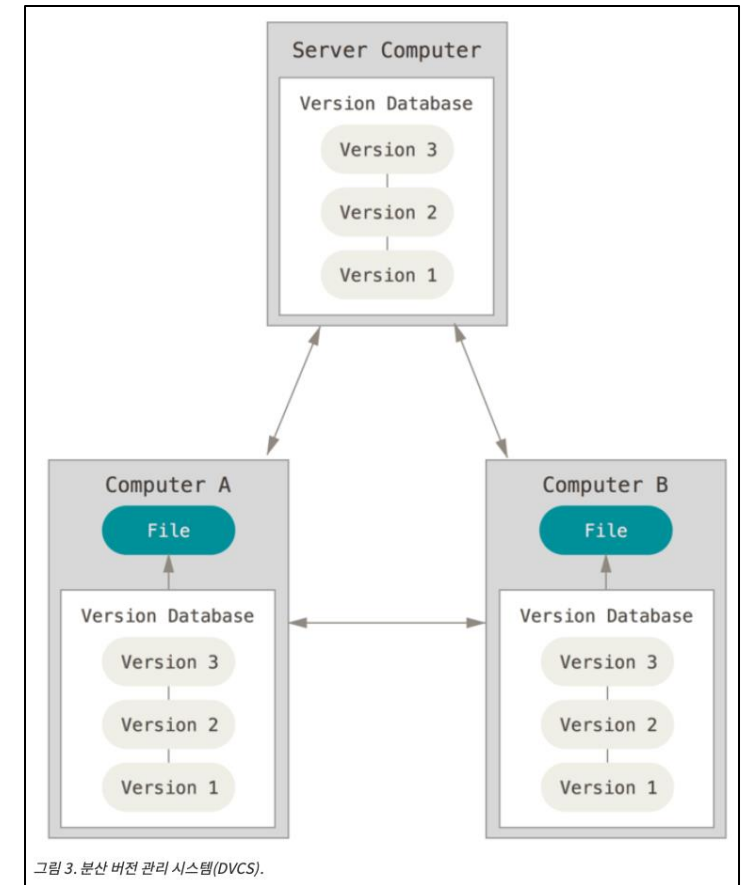
여러 명의 사용자들

스냅샷 스트림

분산 버전 관리 시스템

분산 버전 관리 시스템 (DVC(S))

- 분산 버전 관리(DVC, 혹은 DVCS 등 다양한 방법으로 불린다.)
 - Distributed revision control
 - Distributed Version Control (Systems)
 - Decentralized Version Control (Systems)
- 각 개발자가 **중앙 서버**에 접속하지 않은 상태에서도 코드 작업을 할 수 있는 것이 특징.
- 분산 모델은 일반적으로 리눅스 커널 프로젝트 같은 대형 (혹은 오픈소스) 프로젝트에 더 적합하다.
 - 개발자들이 독립적으로 작업한 다음에 변경사항을 병합(또는 거절)할 수 있기 때문.



파일의 변경사항

여러 명의 사용자들

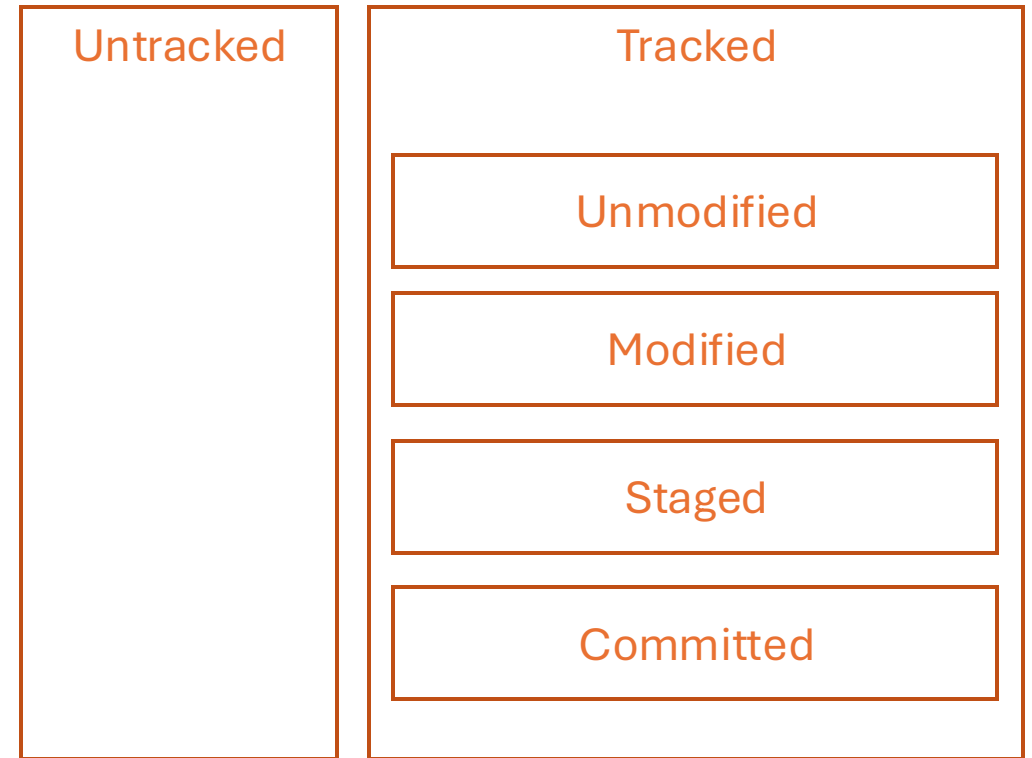
스냅샷 스트림

분산 버전 관리 시스템

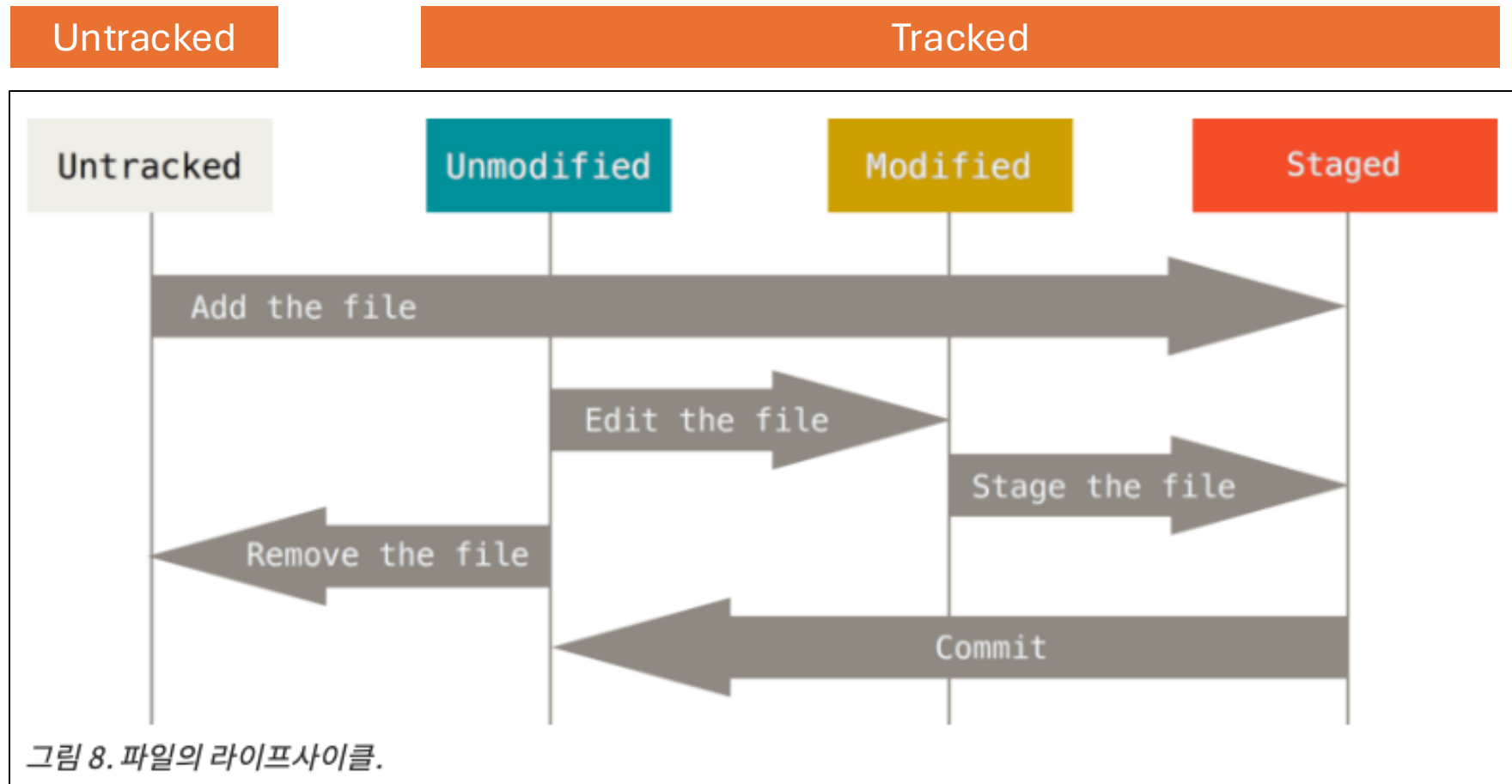
3. 파일 LifeCycle

파일의 Life Cycle

- 작업 디렉토리(Working Directory)의 모든 파일은 크게 Tracked와 Untracked로 나눈다.
- **Tracked(관리대상인 파일)**
 - 이미 스냅샷에 포함돼 있던 파일이다.
 - Git이 알고 있는 파일이다.
 - Tracked 파일은 Unmodified와 Modified 그리고 Staged 상태 중 하나이다.
 - **Unmodified (수정하지 않음)**
 - **Modified (수정함, 다음 커밋에 포함되지 않는 예정인 상태)**
 - **Staged (커밋으로 저장소에 기록할 예정인 상태)**
 - **Committed (커밋으로 저장소에 기록한 상태)**
- **Untracked(관리대상이 아닌 파일)**
 - Tracked에 해당하지 않는 나머지 모든 파일.
 - 스냅샷에도 Staging Area(Index)에도 포함되지 않은 파일이다.



파일의 Life Cycle



Tracked 파일의 상태 (Working Directory)

- **Tracked이지만 Unmodified**

- 작업 디렉터리에서 수정되지 않은 파일.
- Commit 후 변경 사항이 없는 파일.

- Git은 파일을 **로컬 데이터베이스**(.git directory)에 세 가지 상태로 관리.

- **Modified**

- 수정한 파일을 아직 로컬 데이터베이스에 **커밋하지 않은 것**을 말한다.
 - Commit 후 변경 사항이 없는 파일

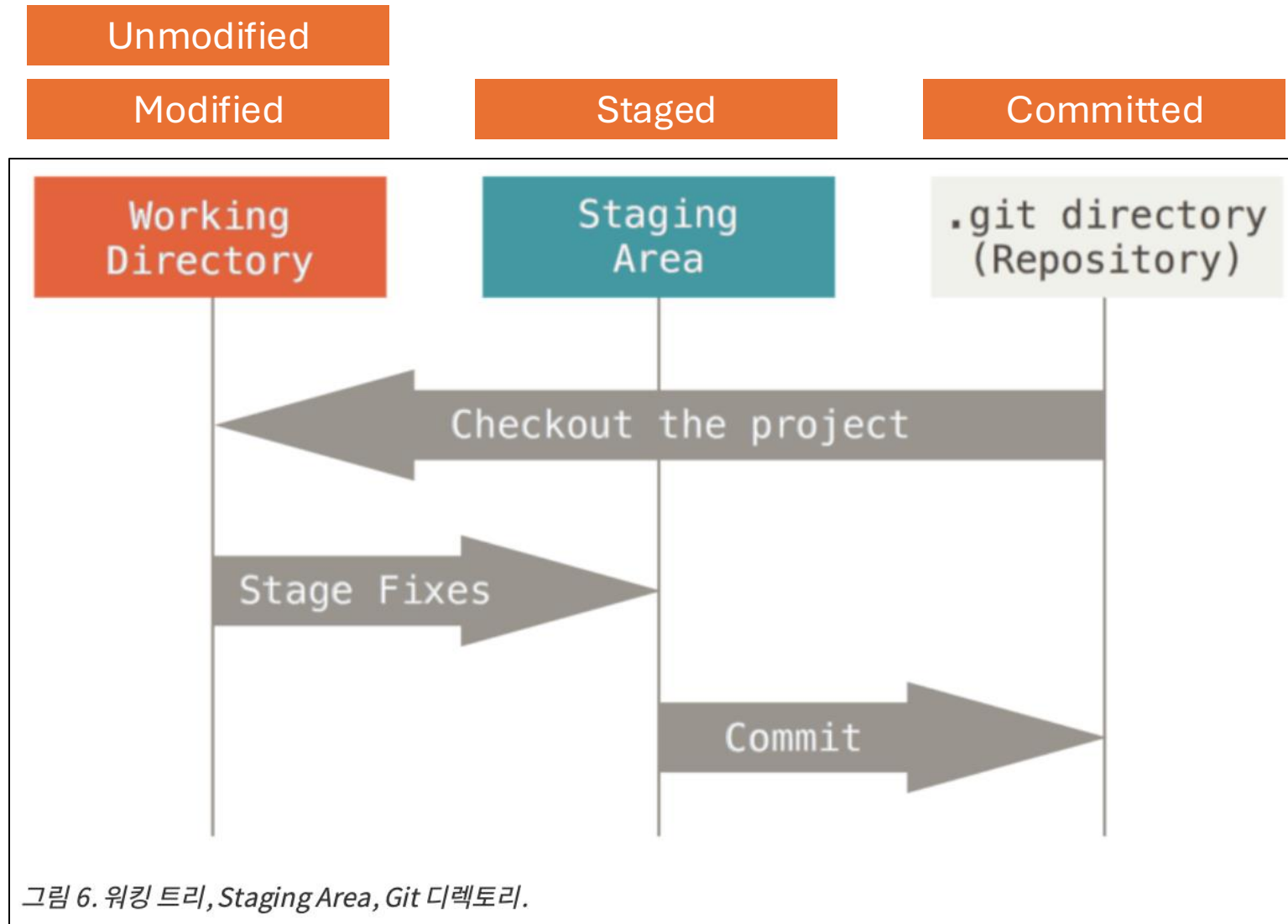
- **Staged**

- 현재 수정한 파일을 **곧 커밋할 것**이라고 표시한 상태를 의미한다.

- **Committed**

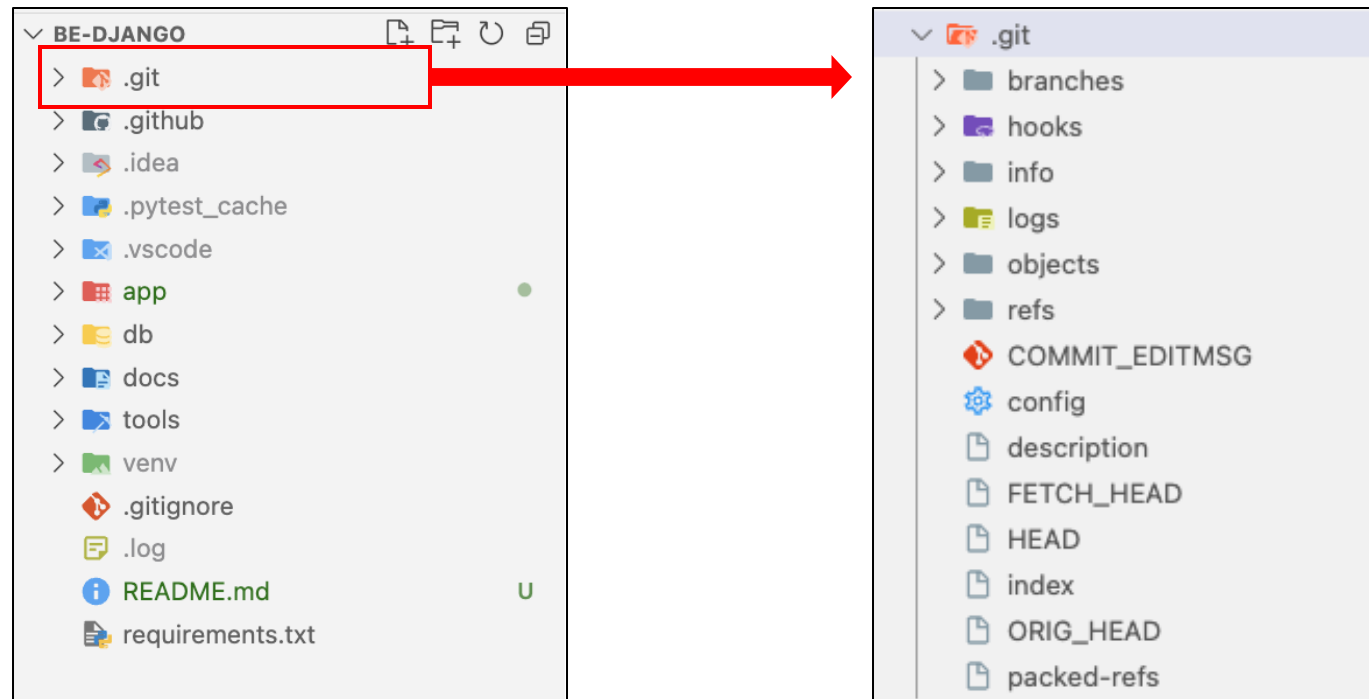
- 데이터가 로컬 데이터베이스에 **안전하게 저장되었다**는 것을 의미.

Tracked 파일의 세 가지 상태



“.git” Directory

- Git 저장소(Repository)를 추가하면 자동으로 만들어진다.
- Git 에서 관리하는 메타 데이터와 Snapshot들이 저장되어 있다.



실습: Git Clone 하기

- Git 저장소 클론하기
- Git 커밋 생성하기

Git 저장소 클론하기

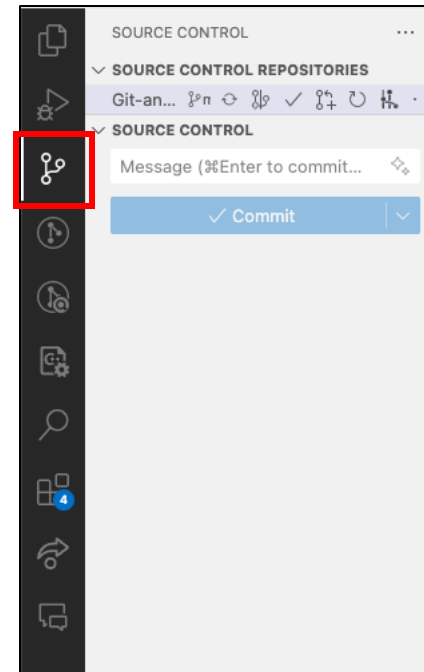
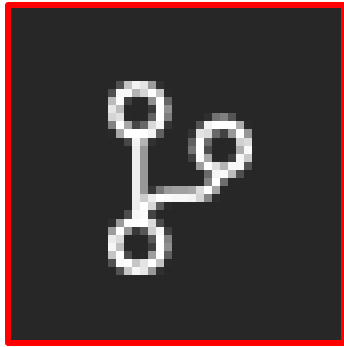
Git이 설치 되어있지 않다면 부록(Git 설치하기)을 참고해주세요.

```
git clone https://github.com/smu-202115064/Git-and-GitHub-Advanced.git
```

- 성공적으로 Clone을 마쳤다면, 저장소에 /week-01/README.md가 존재해야 합니다.
- Clone 작업이 모두 완료되었다면, VS Code를 이용하여 저장소 폴더를 엽니다.

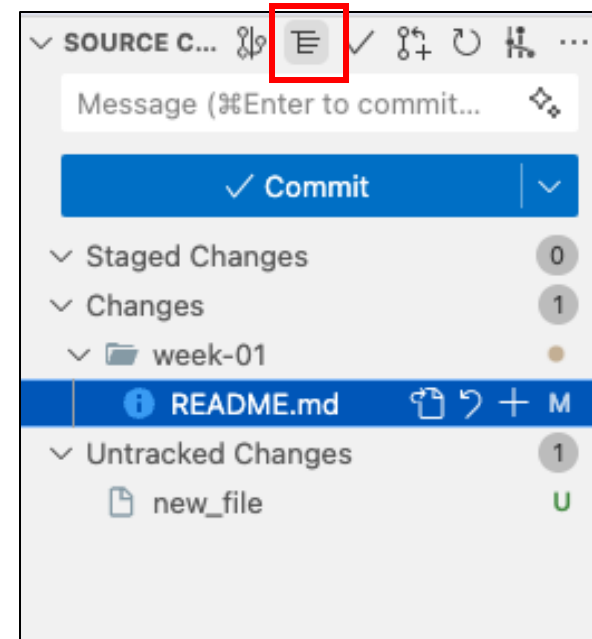
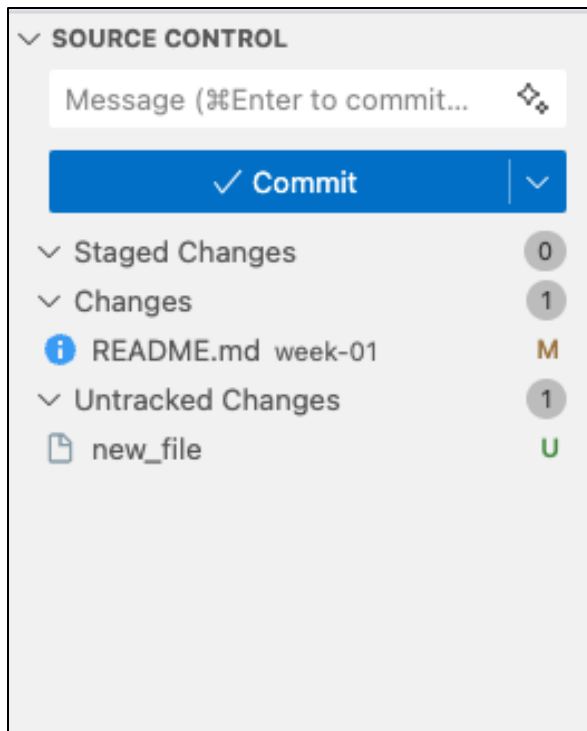
VS Code에서 Git 이용하기

- VS Code는 기본적으로 Git을 지원합니다.
- 만약, 본인의 PC에 Git이 올바르게 설치되어 있다면, 창의 좌측 아이콘에 “Source Control” 메뉴가 나타납니다.



VS Code에서 Git 이용하기 (2)

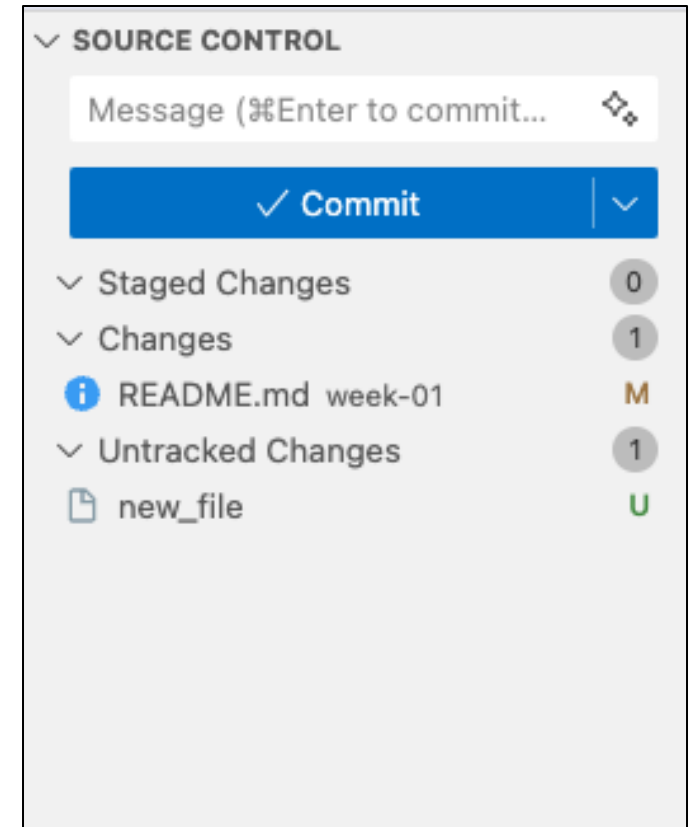
- 새로운 문서를 추가하거나, 기존의 문서를 수정하면 아래 화면과 같이 파일의 상태에 따라 파일 들이 표시됩니다.



View as Tree 아이콘을 누르면 파일 트리 형태로 표시

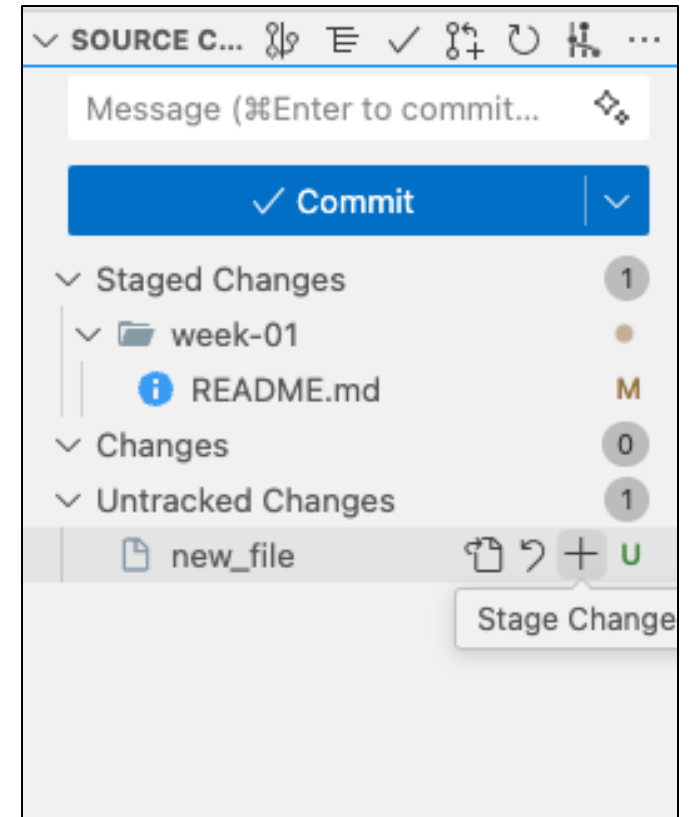
VS Code에서 Git 이용하기 (2)

- **Untracked**
 - 새로 생성된 파일
- **Changes(Modified)**
 - 기존에 존재하던 파일이 수정된 것
- **Staged Changes**
 - Modified 된 파일 중 다음 Commit에 포함될 부분만 선택하여 .git directory에 저장한 것(Staged Changes 에 해당하는 부분은 Modified에 표시되지 않음)



VS Code에서 Git 이용하기 (2)

- **Stage:**
 - 파일 우측의 '+' 버튼을 눌러 스테이징 합니다.
- **Commit:**
 - 상단의 메시지박스를 작성하고, 'Commit' 버튼을 눌러 커밋을 생성합니다.



준비가 완료되었다면,

- 클론한 저장소에서 `/week-01/README.md` 를 열고, 문서의 지시에 따라 작업을 수행해주세요.

실습 과제

백준 온라인 저지의 문제 1개 풀이.

- 풀이할 문제 선택과 풀이에 사용할 프로그래밍 언어는 자유.
- 제출한 문제 풀이 소스코드를 저장소에 커밋할 것.
 - Commit 시 파일명은 `/이름/BOJ_문제번호.*`으로 통일.

- 예) 김동주, 1000번 문제를 Python 언어로 풀이 시, `/김동주/BOJ_1000.py`
 - 예) 서동혁, 2444번 문제를 C언어로 풀이 시, `/서동혁/BOJ_2444.c`

- 문제 풀이 소스코드 1개 외의 파일은 커밋하지 말 것.
- (원격 저장소에 푸시는 2-3주차에 할 것이므로 로컬에만 커밋)

FAQ