



경성대학교  
KYUNGSUNG UNIVERSITY

# GIT



2024-07-25



Intel AI 융합 DX 마스터 클래스



소프트웨어학과 서보형










# 목차



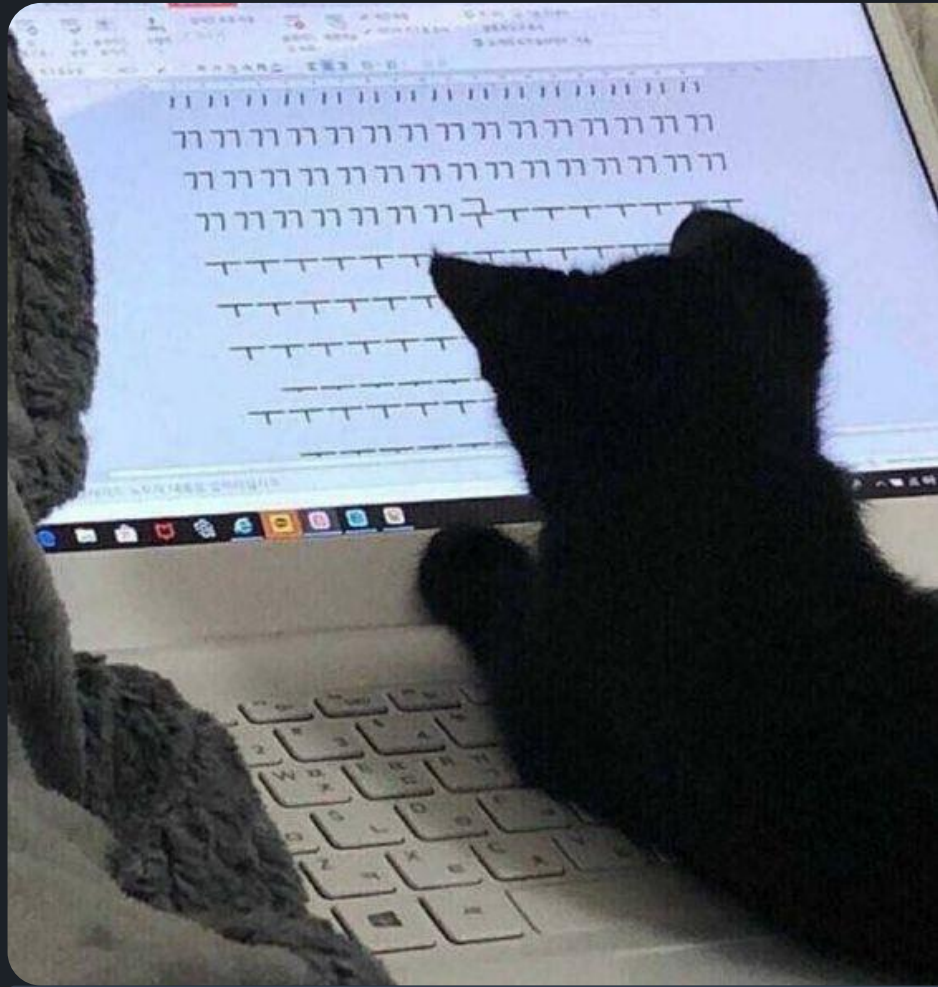
- GIT
- GIT의 특징
- GIT 설치
- GIT 명령어

**GIT**



-  프로젝트 결과 보고서\_1126
-  프로젝트 결과 보고서\_1126\_수정
-  프로젝트 결과 보고서\_1126\_수정2
-  프로젝트 결과 보고서\_최종
-  프로젝트 결과 보고서\_최종\_보고용
-  프로젝트 결과 보고서\_최종\_보고용\_1127 수정
-  프로젝트 결과 보고서\_최종\_보고용\_1127 최종
-  프로젝트 결과 보고서\_최종\_보고용\_1127 최종\_진짜진짜진짜 최종
-  프로젝트 결과 보고서\_최종\_보고용\_1127 최종\_진짜진짜최종

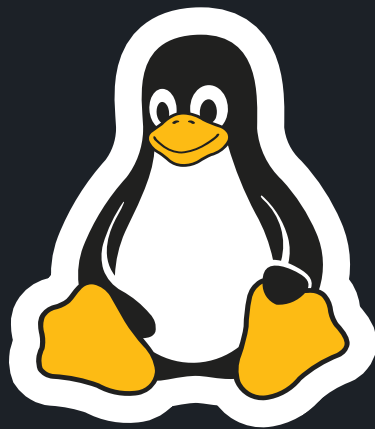




## GIT 등장



리누스 토르발스  
(Linus Torvalds)





# GIT 등장

Linux  
kernel  
mailing  
list



CVS



SVN



GIT

# GIT의 특징



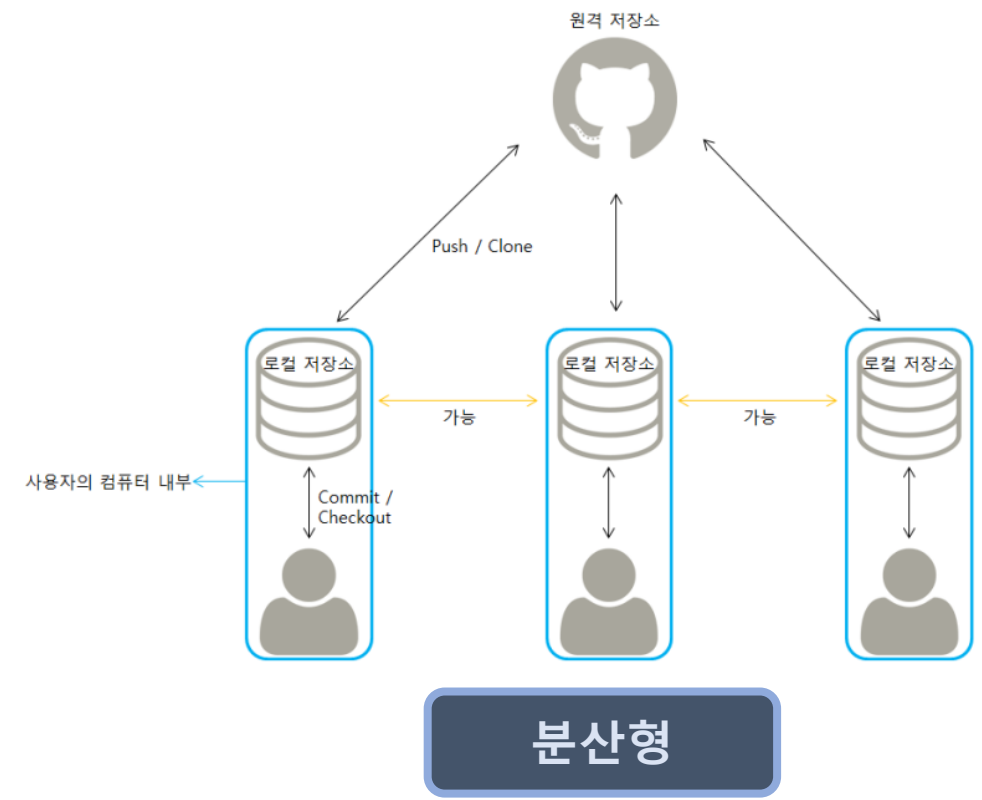
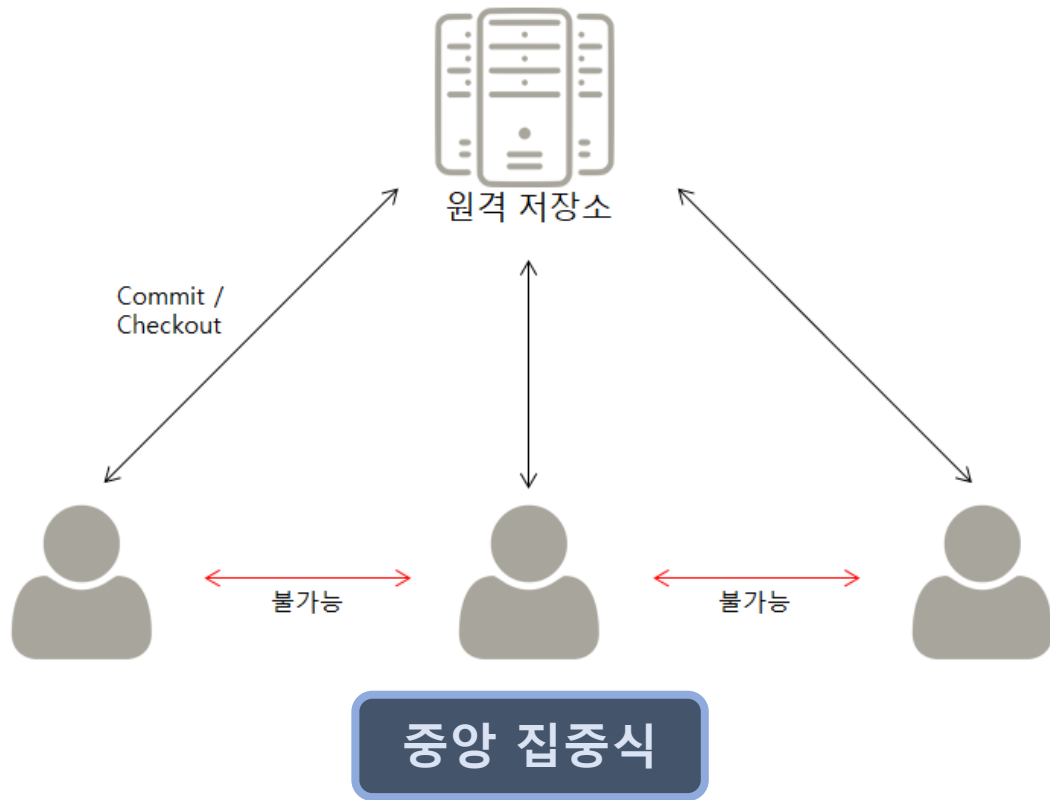


## GIT의 특징

- 오픈소스
  - 분산형 버전 관리 시스템
-



# GIT의 특징





# GIT의 핵심 기능



버전 관리



백업



협업



# GIT 클라이언트

```
MINGW64:/c/Users/singh/Desktop/newRepo
nothing to commit, working tree clean

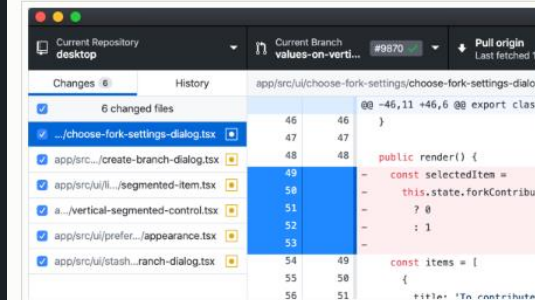
singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)
$ git add .

singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)
$ git commit -m "second commit"
[master 9e7f7d0] second commit
1 file changed, 1 insertion(+)
create mode 100644 abc/jhvjhb.txt

singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 326 bytes | 65.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/taran910/NewRepo.git
7a5d54b..9e7f7d0 master -> master

singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)
$ |
```

## Git Bash

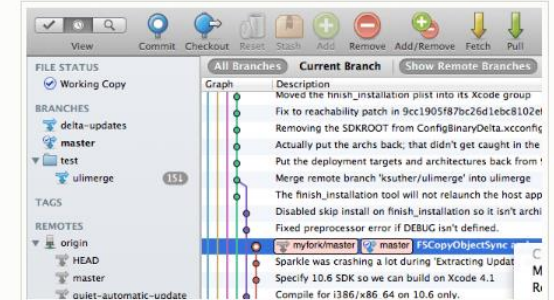


### GitHub Desktop

Platforms: Mac, Windows

Price: Free

License: MIT

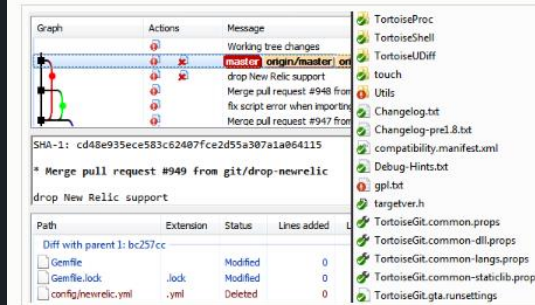


### SourceTree

Platforms: Mac, Windows

Price: Free

License: Proprietary

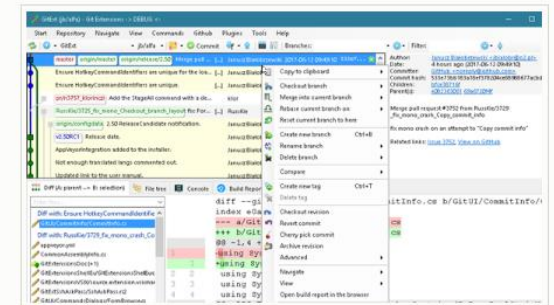


### TortoiseGit

Platforms: Windows

Price: Free

License: GNU GPL



### Git Extensions

Platforms: Windows

Price: Free

License: GNU GPL

# GIT 설치





# GIT 설치하기



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



# GIT 설치하기



**git** --distributed-is-the-new-centralized

 Search entire site...

About

Documentation

**Downloads**

GUI Clients

Logos

Community

The entire **Pro Git book**  
written by Scott Chacon and  
Ben Straub is available to [read](#)

## Downloads



macOS



Windows



Linux/Unix

Older releases are available and the [Git source repository](#) is on GitHub.

Latest source Release

**2.45.2**

[Release Notes](#) (2024-05-31)

Download for Windows



# GIT 설치하기



**git** --everything-is-local



Search entire site...

About

Documentation

**Downloads**

GUI Clients

Logos

Community

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on

## Download for Windows

[Click here to download](#) the latest (**2.45.2**) **64-bit** version of **Git for Windows**. This is the most recent [maintained build](#). It was released **about 2 months ago**, on 2024-06-03.

### Other Git for Windows downloads

#### Standalone Installer

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)

Portable ("thumbdrive edition")

[32-bit Git for Windows Portable.](#)

[64-bit Git for Windows Portable.](#)

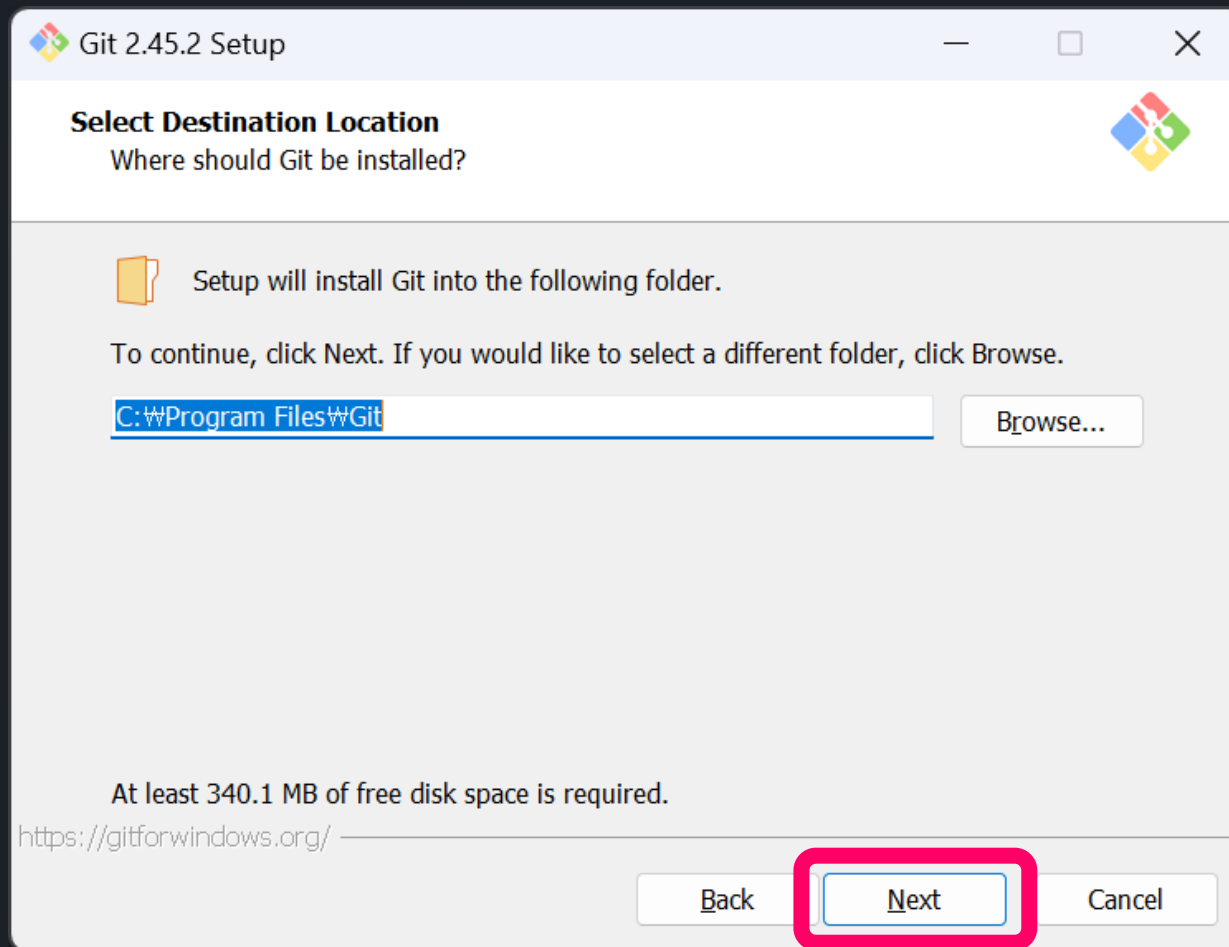


# GIT 설치하기





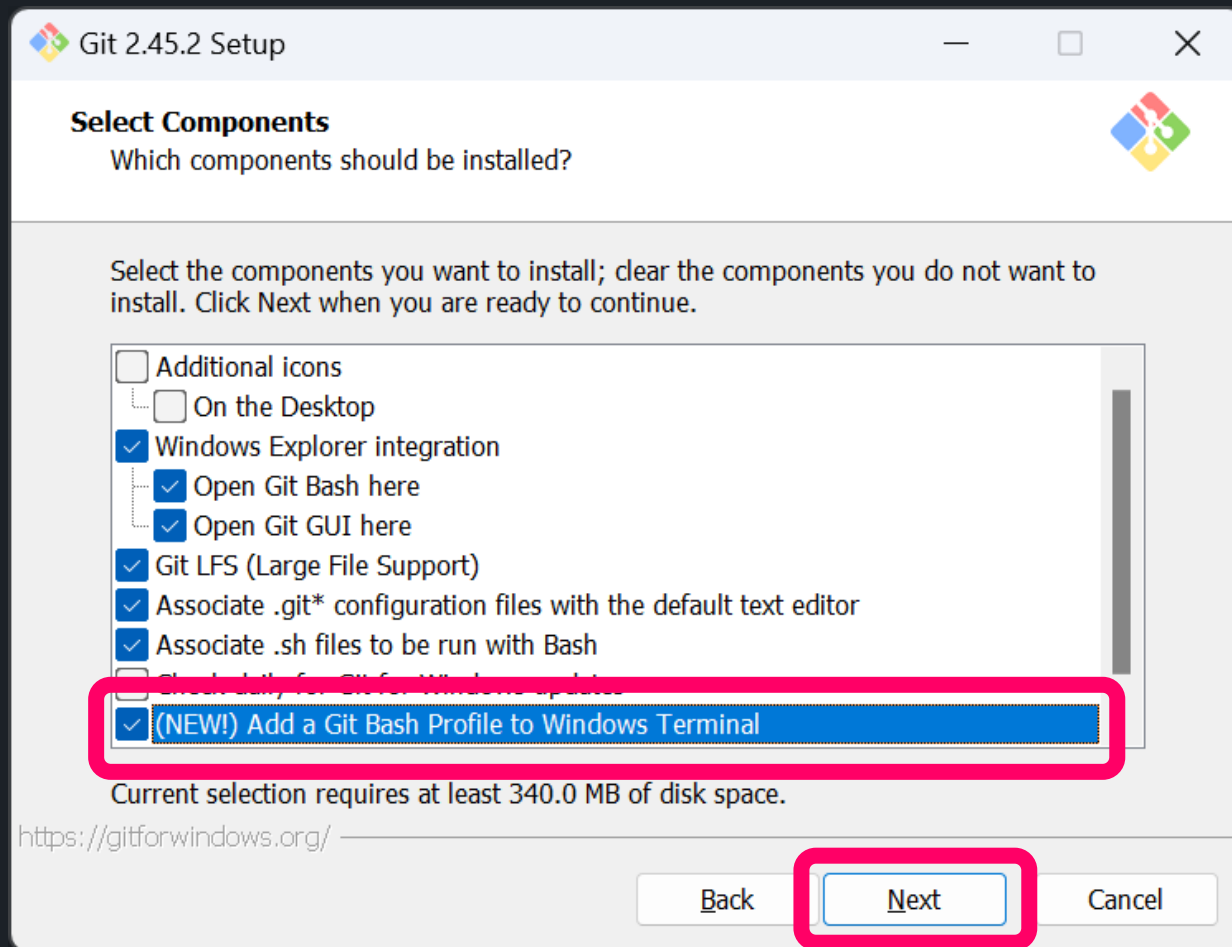
# GIT 설치하기





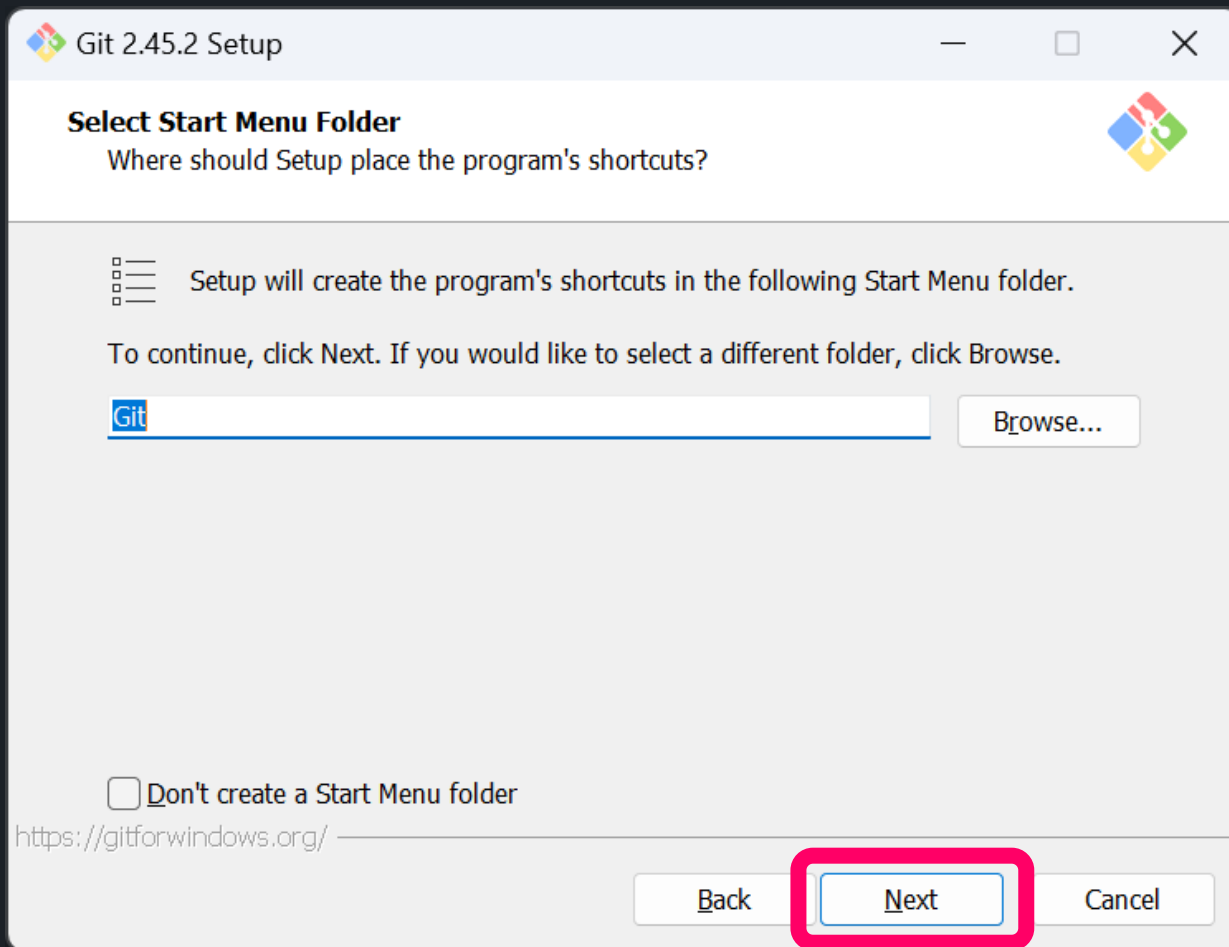


# GIT 설치하기



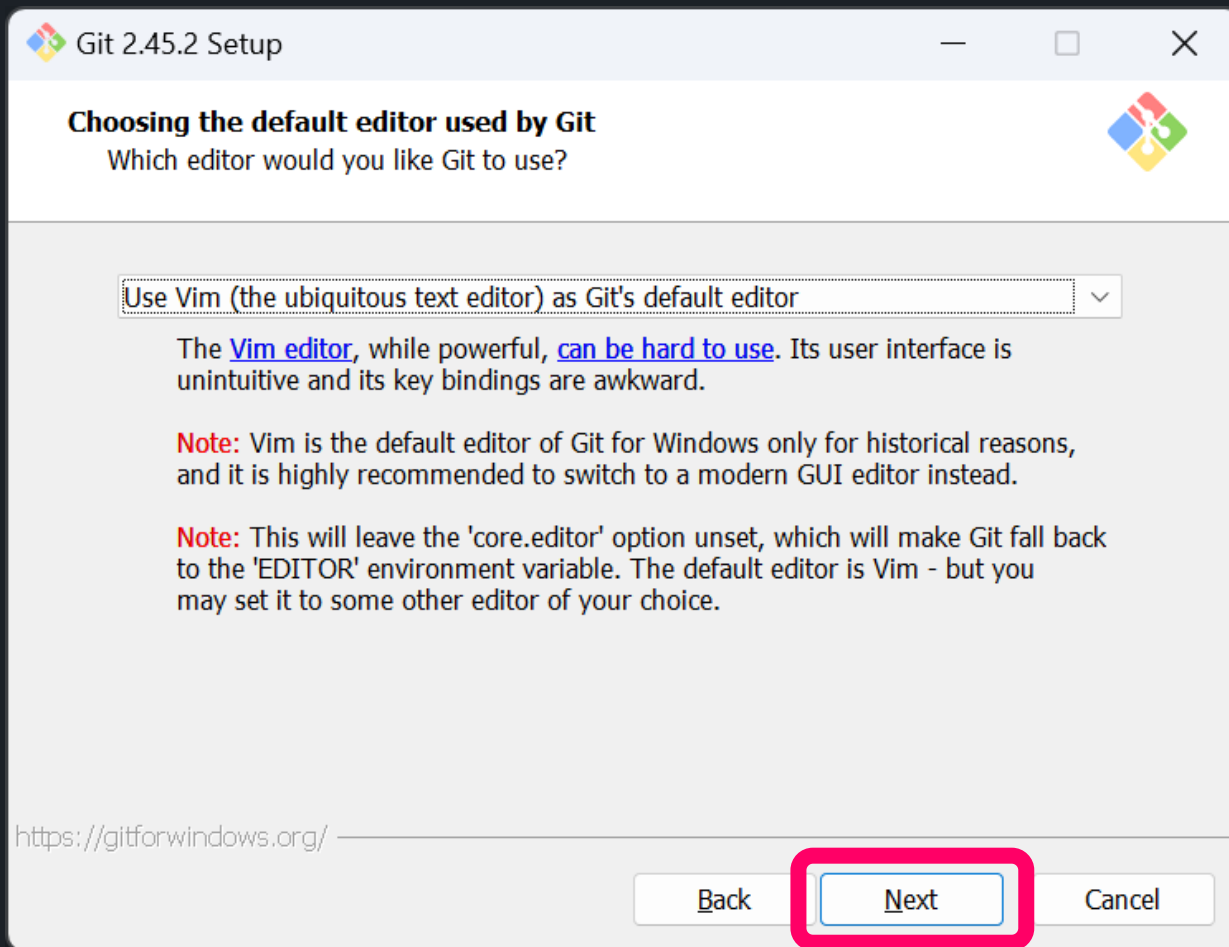


# GIT 설치하기





# GIT 설치하기





# GIT 설치하기

Git 2.45.2 Setup

**Adjusting the name of the initial branch in new repositories**  
What would you like Git to name the initial branch after "git init"?

☐ **Let Git decide**  
Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project [intends](#) to change this default to a more inclusive name in the near future.

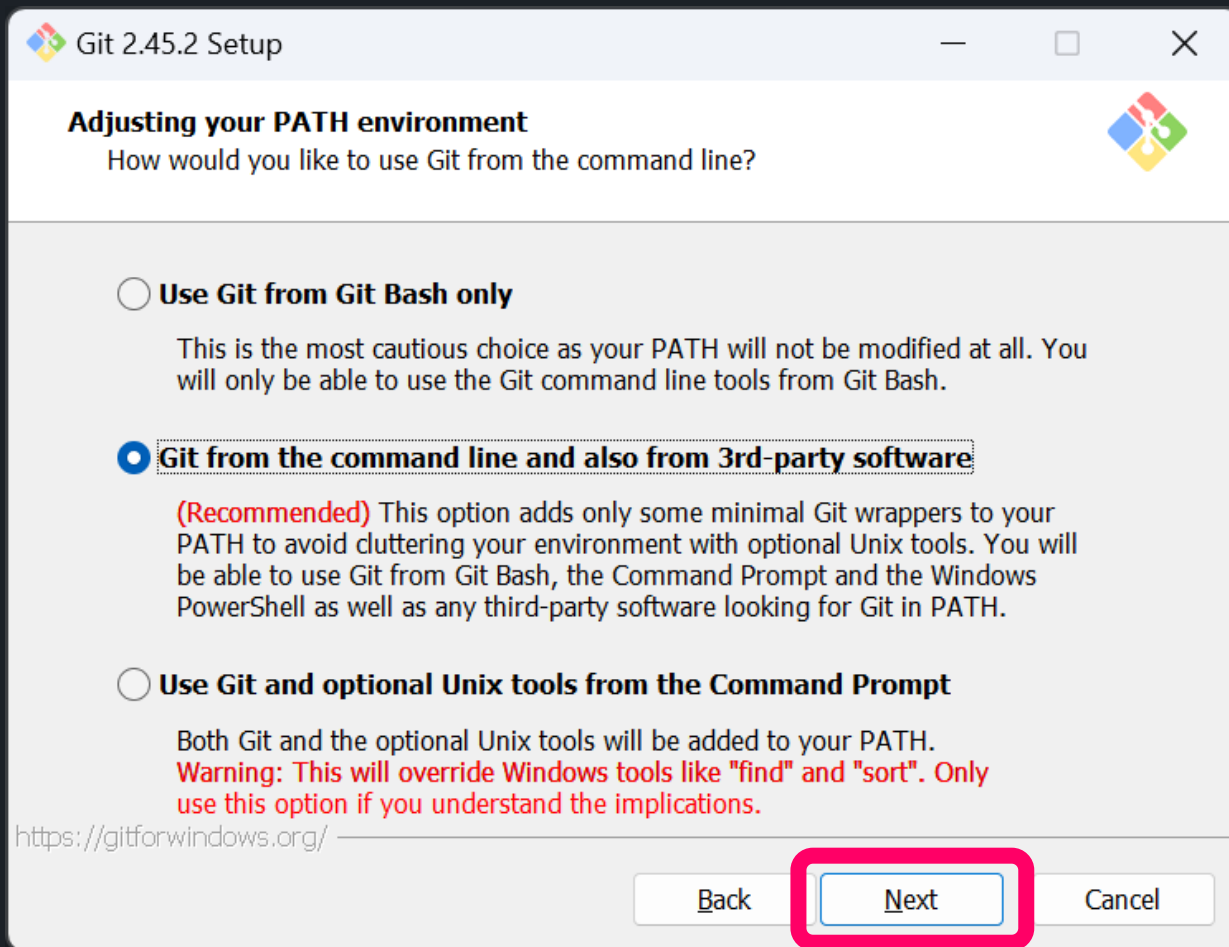
☒ **Override the default branch name for new repositories**  
**NEW!** Many teams already renamed their default branches; common choices are "main", "trunk" and "development". Specify the name "git init" should use for the initial branch:

This setting does not affect existing repositories.

<https://gitforwindows.org/>



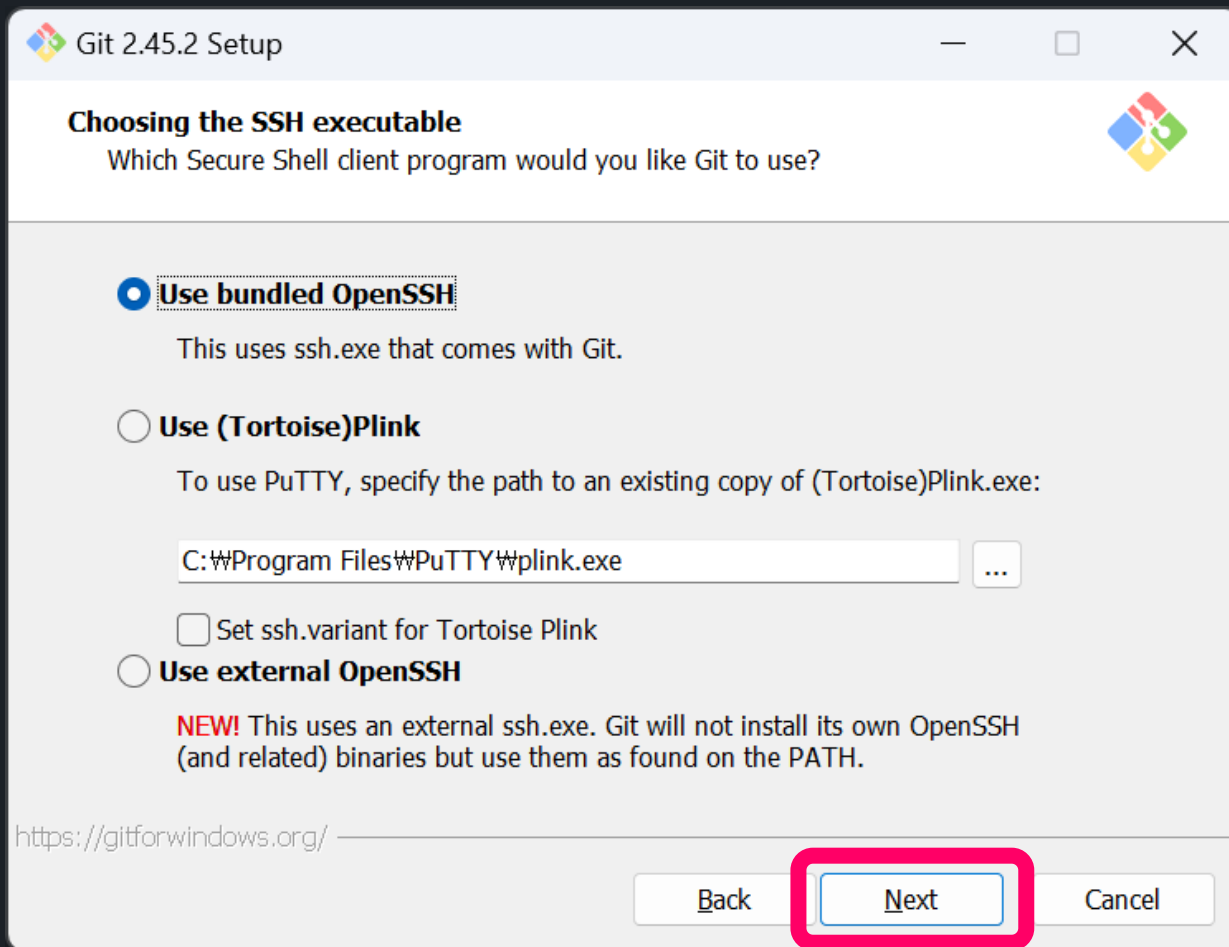
# GIT 설치하기





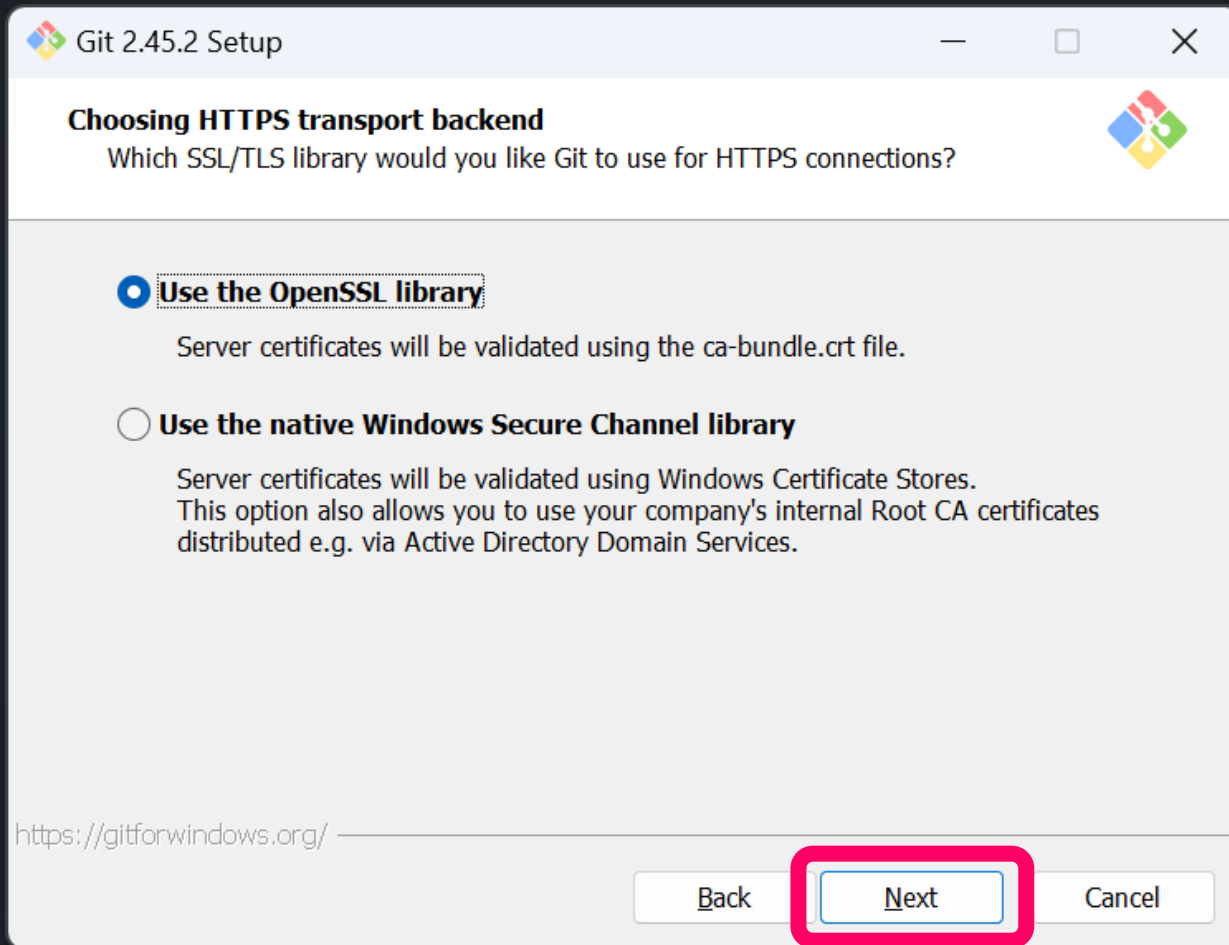


# GIT 설치하기



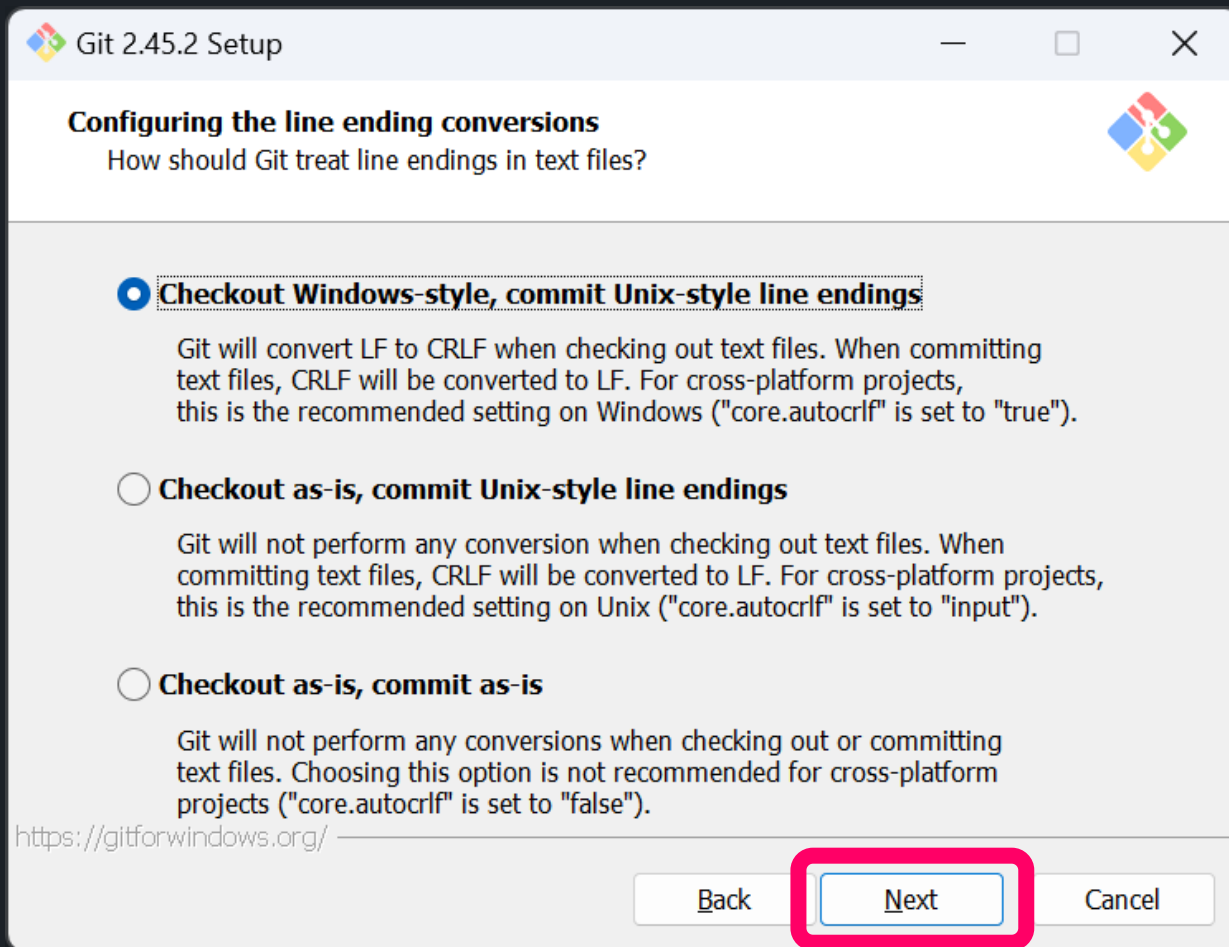


# GIT 설치하기



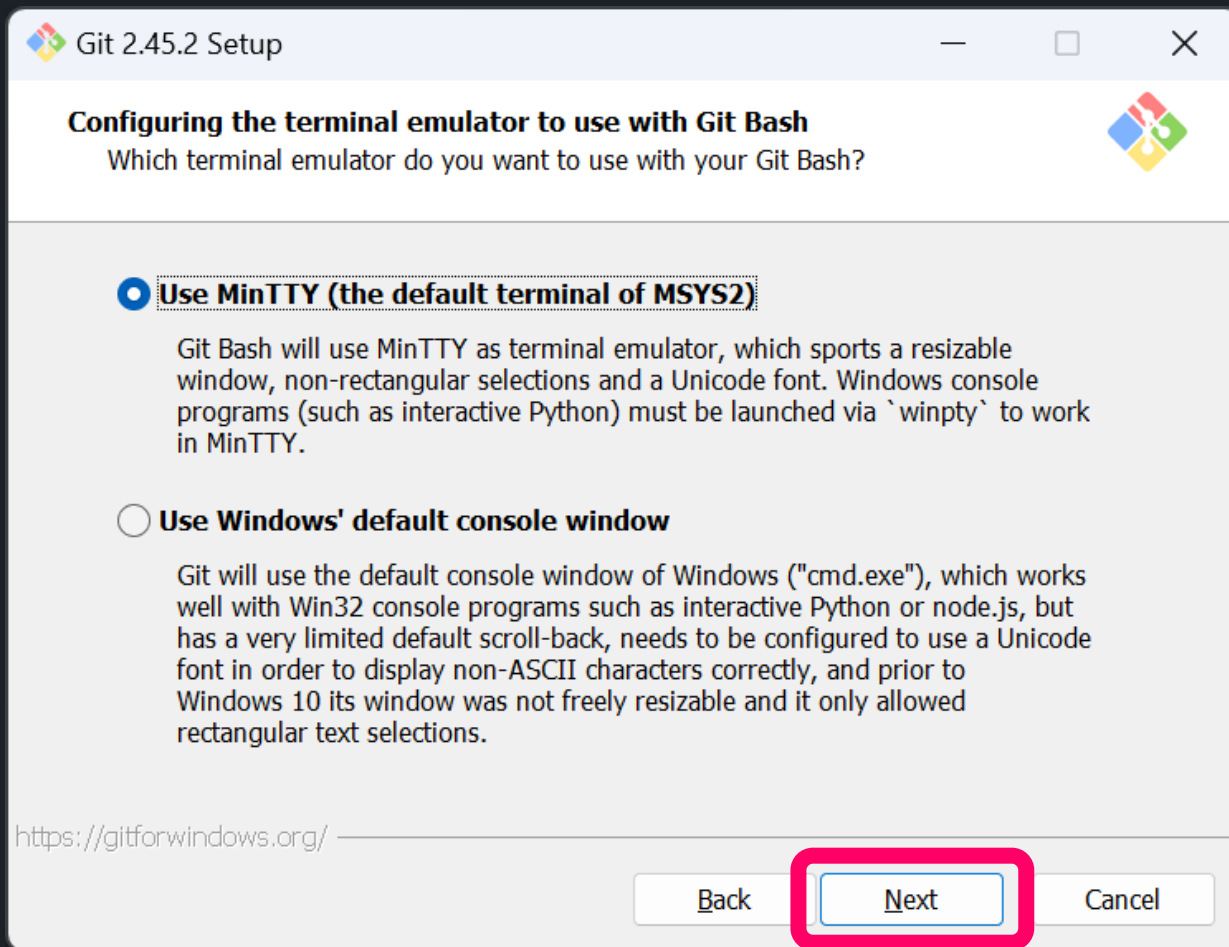


# GIT 설치하기



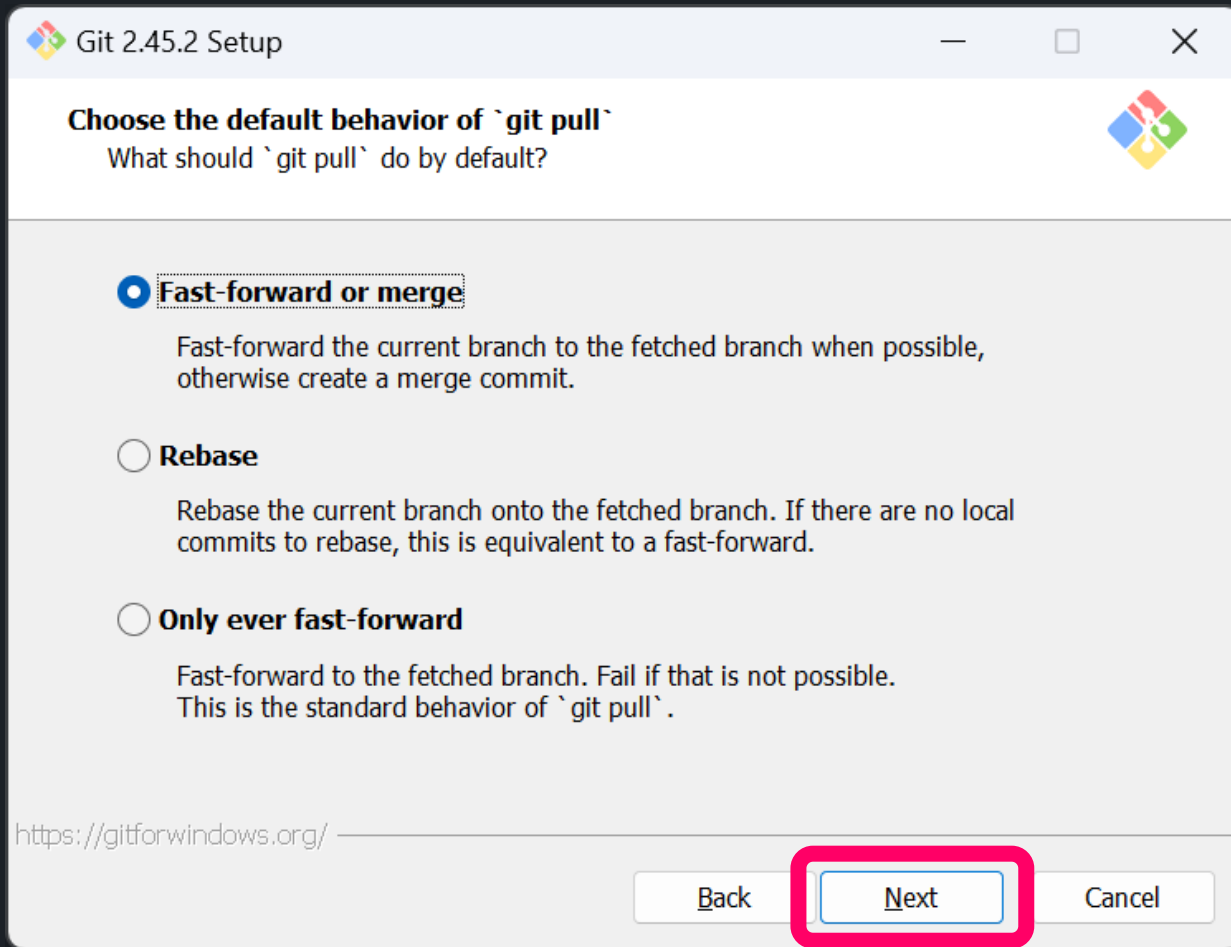


# GIT 설치하기





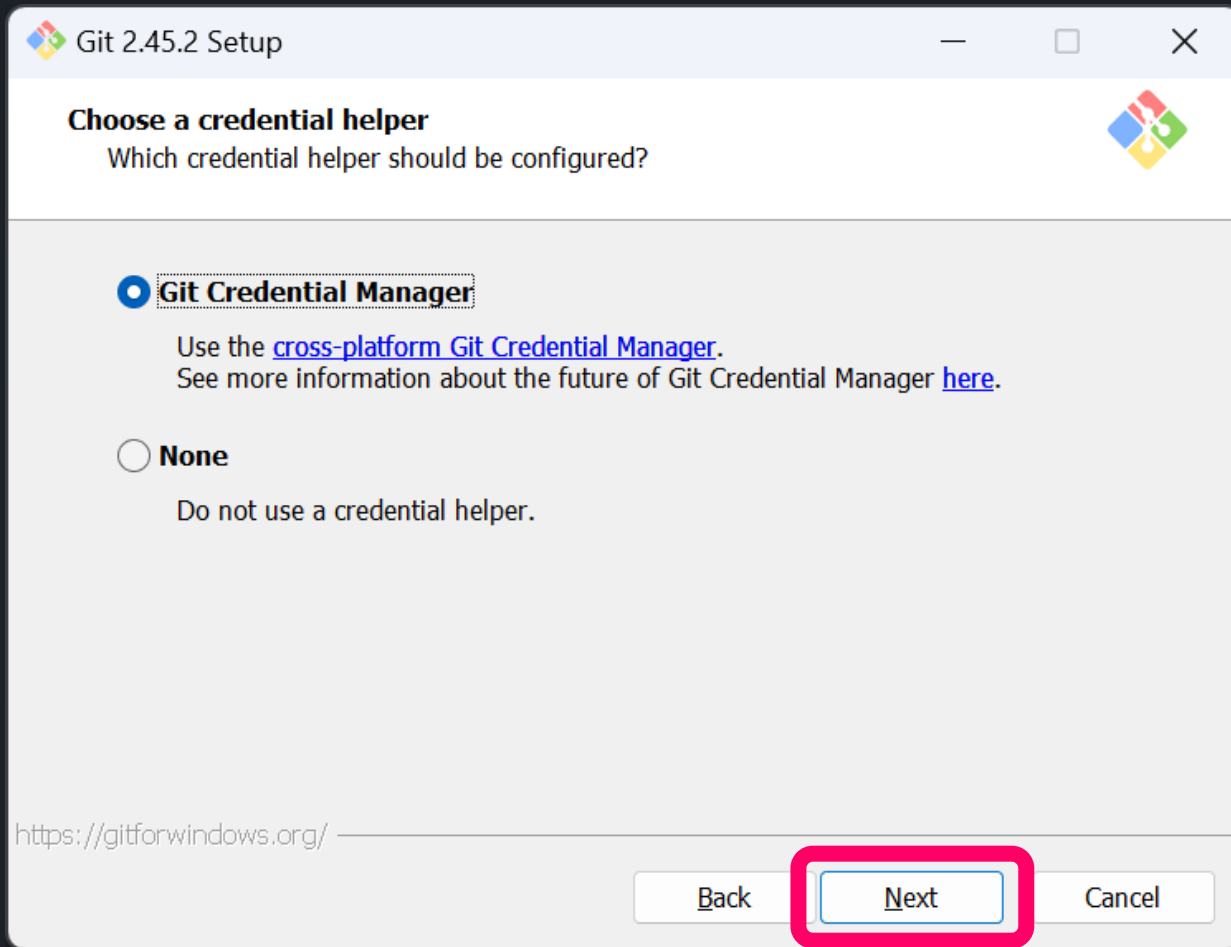
# GIT 설치하기





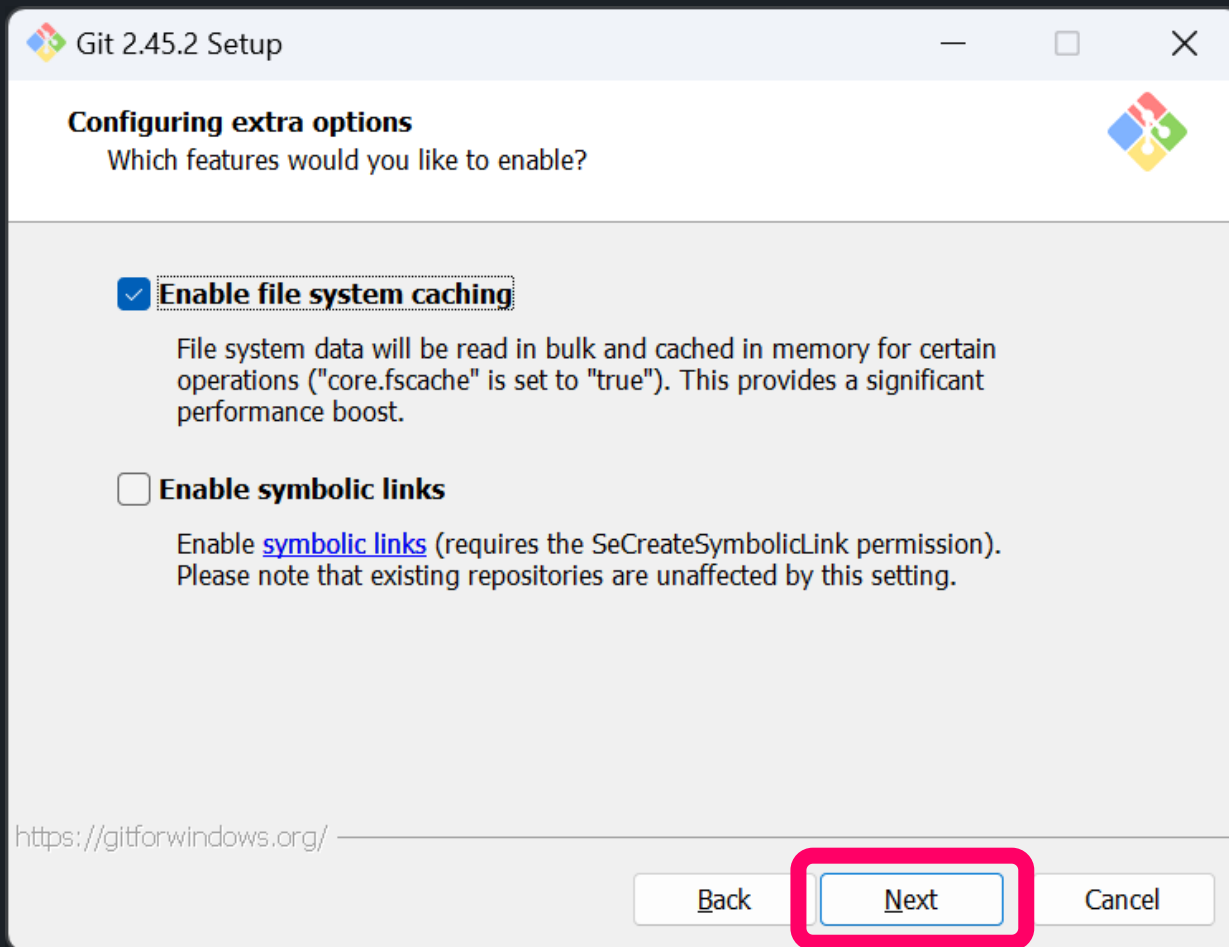


# GIT 설치하기



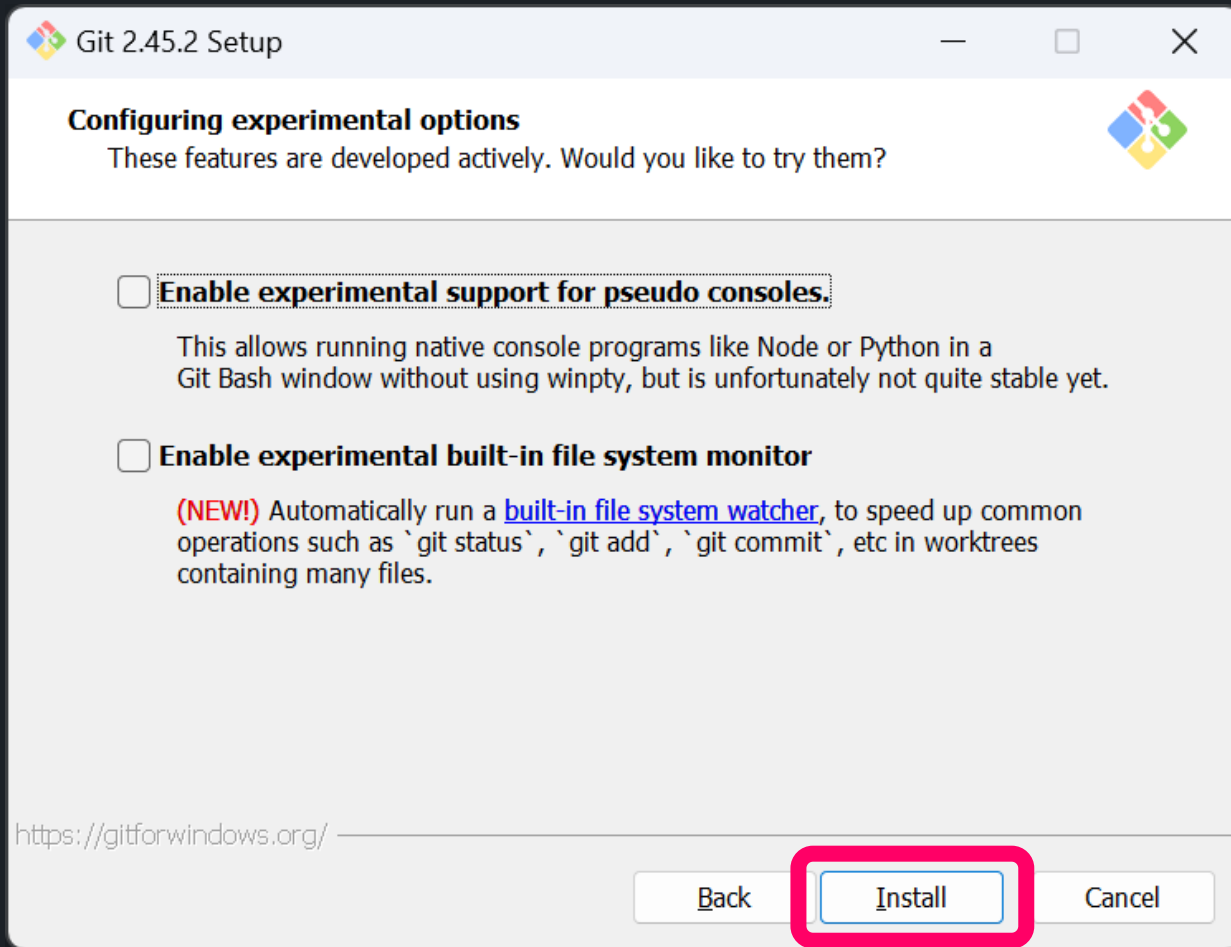


# GIT 설치하기



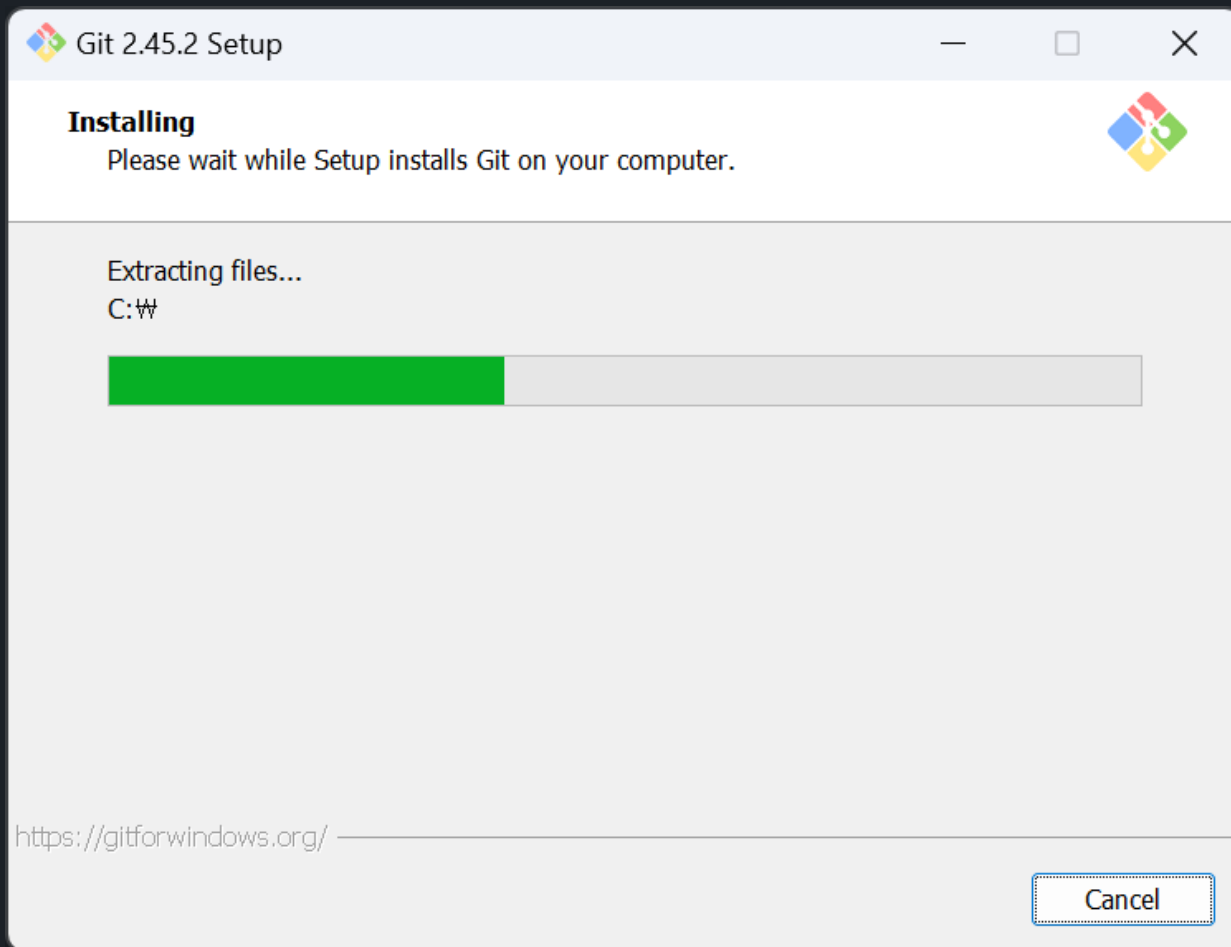


# GIT 설치하기



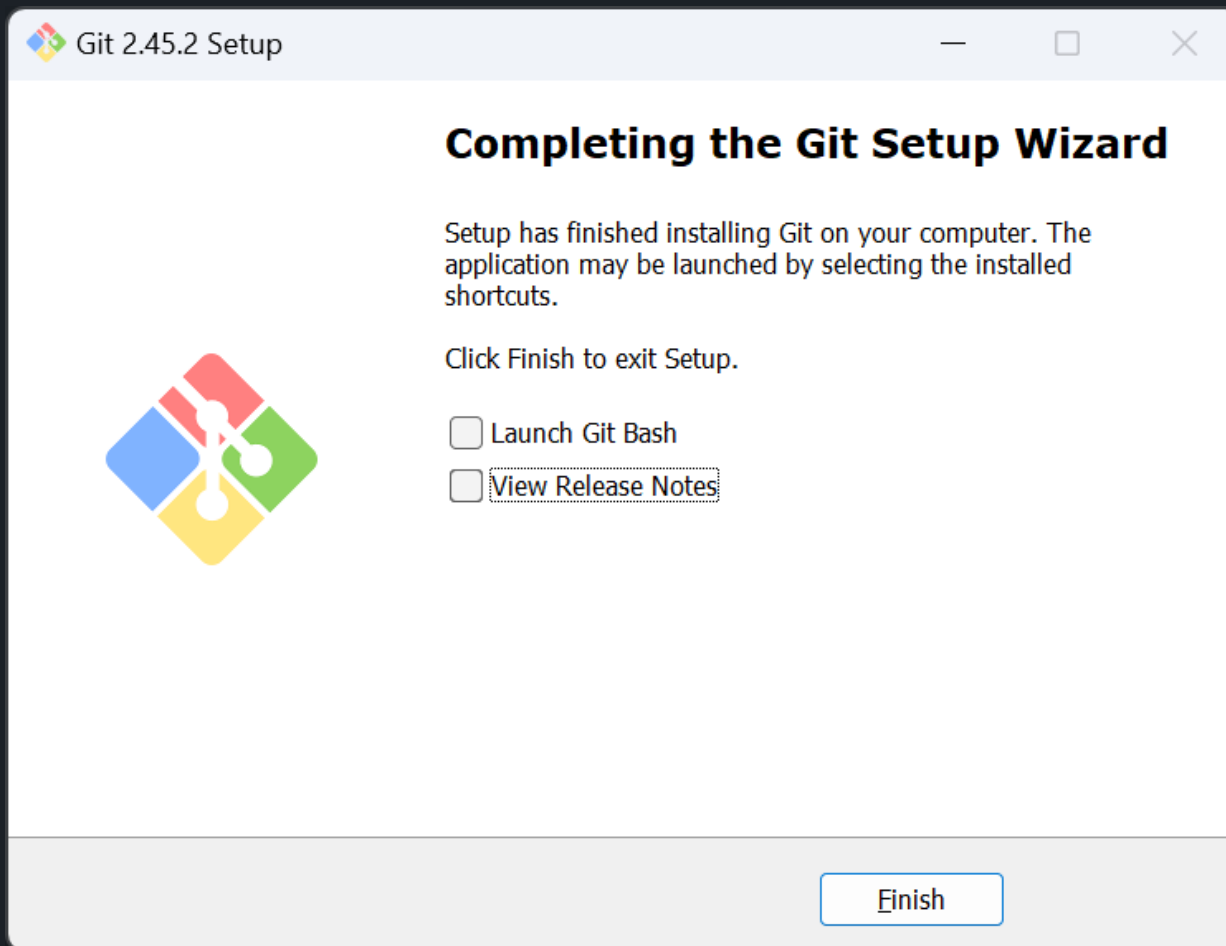


# GIT 설치하기





# GIT 설치하기



# GIT 명령어





# 저장소 생성



1. `mkdir` 폴더이름 : 폴더 생성
2. `cd` 폴더이름 : 해당 폴더로 이동
3. `ls -la` : 폴더 내용 자세히 보기
  - 아직 `.git` 이 생성되지 않음
4. `git init` : 깃 저장소 초기화
  - initialize '초기화하다'의 의미
  - 현재 디렉터리에 깃을 위한 저장소 생성
5. `ls -la` : `.git` 디렉터리 생성 확인하기
  - 저장소(repository)
  - 버전이 저장되는 곳



# 버전 생성 (commit)



**버전:** 이전 상태와 구별하기 위해 번호 등으로 구별하는 것

- 커밋 → 버전 생성

## 깃의 버전 관리

- 생성 시간, 수정 내용 기록
- 원래 파일의 이름을 유지하면서 변경 시점마다 저장 가능
- 버전마다의 작업 내용 확인
- 특정 버전으로 되돌리기



# 버전 생성 (commit)



작업 폴더

작업 트리



git add



.git

작업 트리



git commit

저장소



...



# 버전 생성 (commit)



## 커밋 과정

작업 트리 → 스테이지 → 저장소(리포지터리)

- **작업 트리**

- 파일을 만들고 작업하는 공간

- **스테이지(스테이징 영역)**

- 버전으로 만들 파일이 대기하는 곳
- `.git` 내부

- **저장소(리포지터리)**

- 스테이지에서 대기하고 있던 파일들을 버전으로 만들어 저장하는 곳
  - `.git` 내부
-



# 버전 생성 (commit)



## 커밋하기

1. `git add 파일명` : 스테이징 영역에 추가
2. `git commit -m "커밋메시지"` : 스테이징 영역의 파일들로 버전 생성

⇔ `git -am "커밋메시지"`

- a 는 all을 의미
- 이전에 커밋했던 파일들을 대상으로만 `-am` 옵션 사용 가능

## 커밋 메시지 수정

- `git commit --amend` : 직전에 커밋한 메시지 수정



# 버전 생성 (commit)



## 커밋 기록 보기

- `git log` : 커밋 기록 보기
- `git log --oneline` : 요약된 커밋 기록 보기
- `git log --stat` : 커밋과 관련된 파일목록 출력
- `git log --branches` : 브랜치들도 포함해서 출력하기
- `git log --graph` : 브랜치와 커밋 관계를 그래프 형태로 표시
- `git log --oneline --branches --graph`



# 버전 생성 (commit)



## 커밋 기록 보기

- `git log` : 커밋 기록 보기
- `git log --oneline` : 요약된 커밋 기록 보기
- `git log --stat` : 커밋과 관련된 파일목록 출력
- `git log --branches` : 브랜치들도 포함해서 출력하기
- `git log --graph` : 브랜치와 커밋 관계를 그래프 형태로 표시
- `git log --oneline --branches --graph`





# 버전 생성 (commit)



## 특정 버전의 커밋의 파일 내용 출력하기

- `git log --stat` : 커밋과 관련된 파일 목록 출력
- `git show 커밋해시:파일명` : 해당 커밋의 파일 내용 출력
- `git show :파일명` : 스테이징의 파일 내용 출력하기



## 커밋 전 변경 취소하기

1. `git add 파일명` : 스테이징 영역에 추가
2. `git restore --staged 파일명` : 스테이징 영역에서 제거
3. `git restore 파일명` : 변경 사항 취소



# 되돌리기



## 커밋 후 변경 취소하기

### • 마지막 커밋 되돌리기 (가장 최신 커밋)

1. `git reset HEAD^`
2. `git restore` 파일명

### • 특정 버전으로 되돌리기

- 되돌린 시점 이후 커밋 기록 보존 안됨

1. `git reset` 커밋해시
2. `git restore` 파일명

### • 커밋 취소하기 (커밋 기록은 보존됨)

1. `git revert` 커밋해시 (취소할 커밋의 해시)
2. 커밋 메시지 수정
3. `git log` 를 확인하면, 취소한 커밋이 보존되어 있고, revert한 커밋이 생성됨



# 되돌리기



Tip

터미널의 커밋 해시 복사는 블록 지정 후 `Ctrl` + `Insert` 키 누르기



# 브랜치 (branch)



## 브랜치 (branch, 분기)

1. `git branch` : 브랜치 목록 출력
2. `git branch 브랜치이름` : 브랜치 생성
3. `git switch 브랜치이름` : 브랜치 전환

## 브랜치 비교

- `git log 기준브랜치..대상브랜치`
  - 브랜치 사이의 차이점 비교하기
  - 기준브랜치와 비교했을 때, 대상 브랜치에만 있는 커밋 표시됨



# 브랜치 (branch)



## 브랜치 삭제

- `git branch -d 브랜치이름`



# 머지 (merge)



## 머지 (merge, 병합)

새 브랜치에 있던 파일을 원래 main 브랜치에 합치기

- `git merge` 브랜치이름 : 브랜치 병합하기
- `git cherry-pick` 커밋해시 : 특정 버전의 내용과 병합하기 (브랜치 전체를 합치지 않음)







## 머지 상황

- 서로 다른 파일 병합
- 같은 파일의 다른 부분을 수정
- 같은 파일의 같은 부분을 수정 → 충돌 발생



Q & A

# 참고 자료

- 이고잉 & 고경희, Do it! 지옥에서 온 문서관리자 깃&깃허브 입문, 이지스퍼블리싱, 2019.
-  깃 (소프트웨어) - 위키백과, 우리 모두의 백과사전 ([https://ko.wikipedia.org/wiki/깃\\_\(소프트웨어\)](https://ko.wikipedia.org/wiki/깃_(소프트웨어))) 
-  47. GIT | 만화로 나누는 자유/오픈소스 소프트웨어 이야기 (<https://joone.net/2022/10/02/47-git/>) 
-  [[Git] VCS에 대해서 알아보자 (CVCS와 DVCS) (<https://velog.io/@sisofiy626/Git-VCS에-대해서-알아보자-CVCS와-DVCS>)] 

감사합니다