

GIT & CLOUD & TOOL 활용

PART I

GIT & CLOUD

형상·버전관리시스템

형상·버전관리시스템

◆ 버전(Version)

- ❖ 특정 매체 제작/개발/작성 과정에서 부여하는 번호
- ❖ 이전 제작/개발/작성된 것에서 변경 및 추가 된 경우 식별 위해 사용하는 번호
- ❖ **이전과 다른 변화를 구분하는 표시**

- ❖ 표기 : 숫자, 연도, 기호를 활용한 규칙적인 형태

- Window10 Home **22H2**
- Coogle Chrome **버전 127.0.6533.101**
- Linux Kernel **6.10.5**

형상·버전관리시스템

◆ 버전(Version)

❖ 결과물 완성을 위한 과정들

프로젝트기획서

이름

2024_서비스_로봇_개발_기획서_240501_01.hwp
2024_서비스_로봇_개발_기획서_240501_02.hwp
2024_서비스_로봇_개발_기획서_240502_01.hwp
2024_서비스_로봇_개발_기획서_240503_01.hwp
2024_서비스_로봇_개발_기획서_240504_F.hwp

결과물



2024_서비스_로
봇_개발_기획서
_240504_F.hwp

형상·버전관리시스템

◆ 버전(Version)

❖ 공동 작업 통한 결과물 완성을 위한 과정들



형상·버전관리시스템

◆ 형상관리시스템(CMS:Configuration Management System)

❖ 형상

- 개발 단계의 각 과정에서 만들어지는 프로그램을 설명하는 문서, 데이터 등 통칭

❖ 형상관리

- 개발 전반 모든 산출물의 종합 및 변경 과정을 체계적으로 관리/유지하는 개발 관리 활동

❖ 형상관리 범위

- 버전 관리 ➔ 소스 코드, 관련 문서들
- 공유 및 권한 관리 ➔ 소스 코드, 관련 문서들

형상·버전관리시스템

◆ 버전관리시스템(VCS:Version Control System)

- ❖ 코드와 콘텐츠 변화를 관리 및 추적하는 소프트웨어

- ❖ 변경 사항(언제, 누가, 어떻게)을 시간에 따라 기록

- ❖ 필요할 때 특정 버전 다시 사용 가능

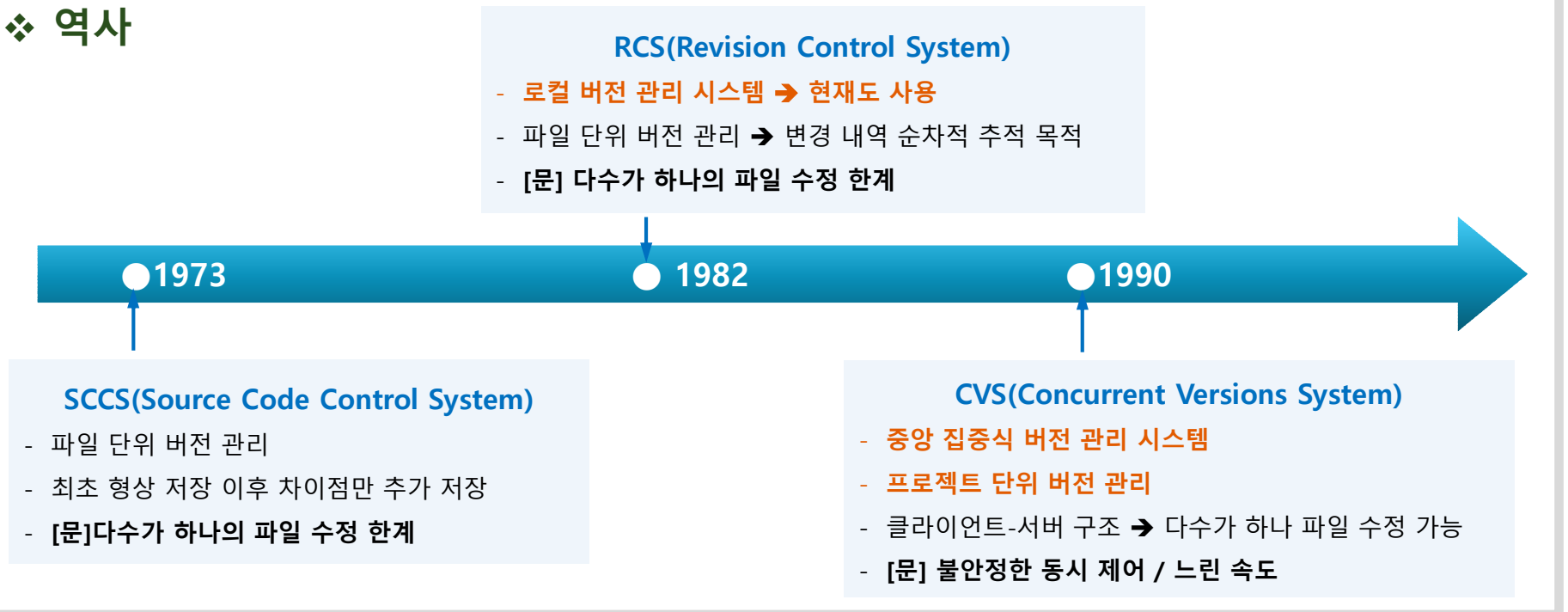
- ❖ [기존] 소스 코드 관리 소프트웨어

- ❖ [현재] 소스 및 관련문서 모두 관리하는 소프트웨어, 형상관리와 동일 의미로 사용

형상·버전관리시스템

◆ 버전관리시스템(VCS:Version Control System)

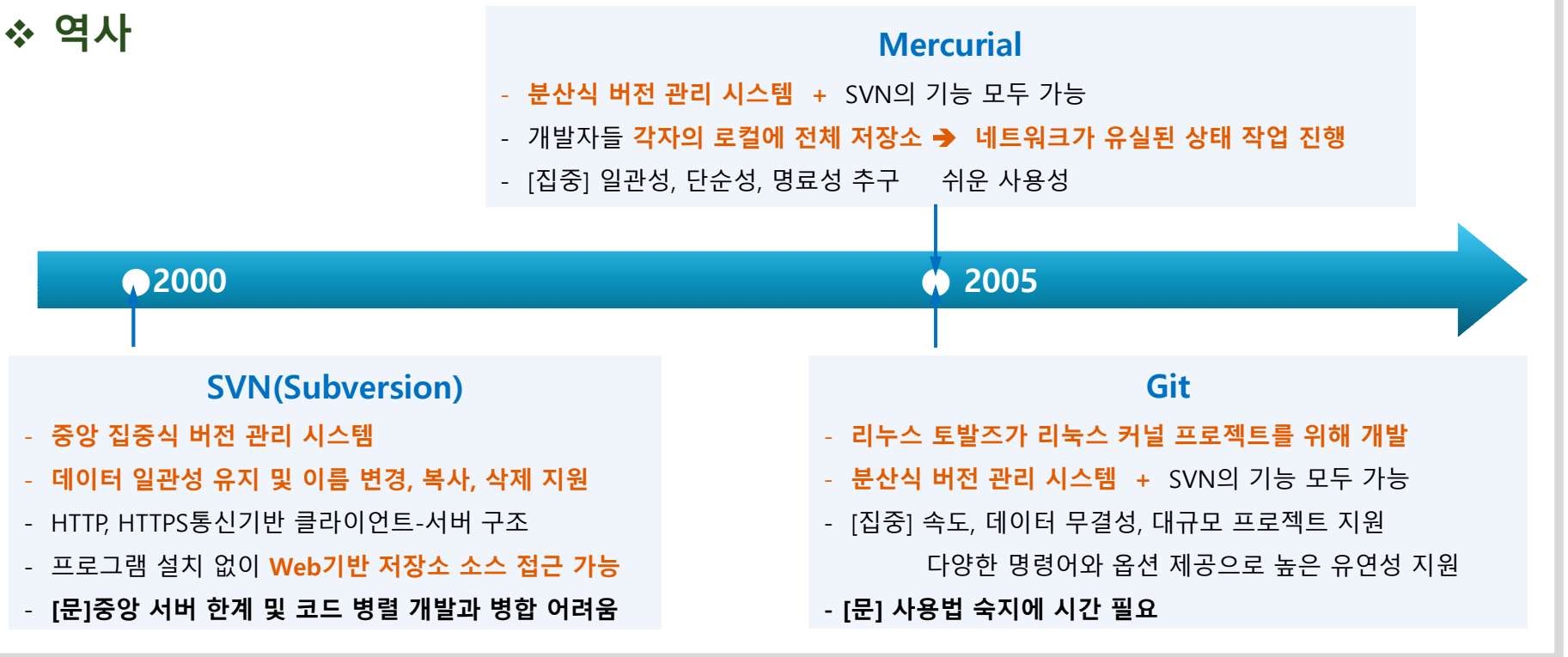
❖ 역사



형상·버전관리시스템

◆ 버전관리시스템(VCS:Version Control System)

❖ 역사

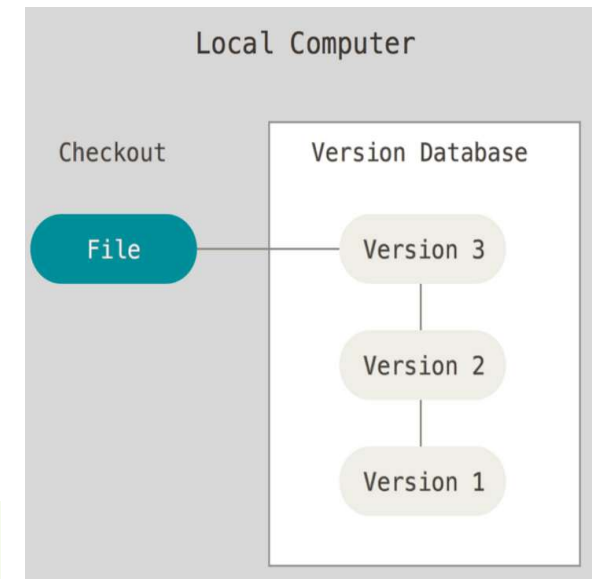


형상·버전관리시스템

◆ 관리시스템모델 - LVCS

❖ 로컬 버전 관리 시스템(Local Version Control System)

- 가장 기본적인 형태의 버전 관리 시스템
 - 로컬 컴퓨터 1대에 작업 수행, 데이터베이스 사용해 버전 저장 & 관리
 - 다수의 개발자가 동시 협업 하기 힘들
 - 모든 파일, 버전 같은 데이터가 하나의 로컬 컴퓨터 저장
➔ 데이터 손실 위험 매우 ↑
-
- 대표적인 예 : RCS, SCCS 시스템

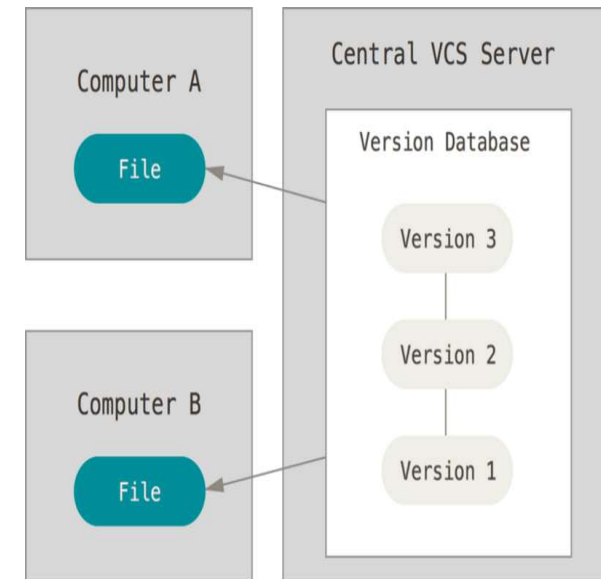


형상·버전관리시스템

◆ 관리시스템모델 - CVCS

❖ 중앙 집중식 버전 관리 시스템(Centralized Version Control System)

- 중앙 서버 존재하며 여기에 모든 파일과 변경 내역이 저장되고 관리
 - 클라이언트-서버 모델로 동작
 - **체크아웃(checkout)** → 서버로부터 특정 파일/버전 다운로드
 - **커밋(commit)** → 작업 후 변경 사항을 업로드
 - 데이터 유실 위험도 ↓, 안정성 ↑
 - 중앙 서버 문제 발생 시 전체 프로젝트 마비
-
- 대표적인 예 : CVS, Subversion, Preforce 시스템

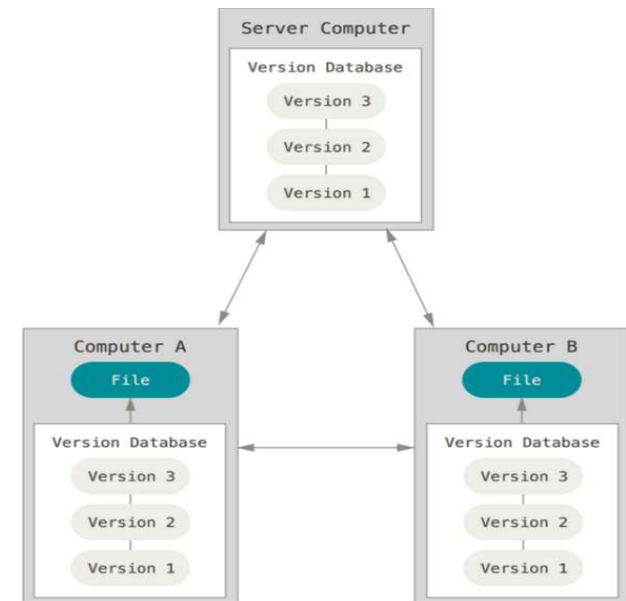


형상·버전관리시스템

◆ 관리시스템모델 - DVCS

❖ 분산 버전 관리 시스템(Distributed Version Control System)

- 로컬 / 중앙 집중식 버전 관리 시스템의 단점 극복
- **클론(clone)** → 개발자는 전체 저장소를 로컬에 복제해 작업
- **푸쉬(push)** → 작업 후 로컬 저장소에서 원격 저장소로 업로드
- **머지(merge)** → 변경 사항 통합
- 진입 장벽 높음 → 로컬/중앙집중식 시스템 비해 명령어 많음
- 대표적인 예 : Git, Mercurial, Bazaar, Darcs



GIT 환경구축

GIT 환경구축

◆ GIT 사용법

❖ CLI (Command Line Interface) : 명령어 입력기반 사용

- GIT에서 작업 실행 시 사용

❖ GUI (Graphical User Interface) : 그래픽 요소 입력기반 사용

- 프로젝트 상태 체크 시 사용

GIT 환경구축

◆ GIT 설치

① 설치파일 다운로드

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

Download for Windows

The screenshot shows the Git website homepage. At the top, the Git logo is followed by the tagline "--distributed-is-the-new-centralized". Below this, a search bar is visible. The main content area describes Git as a "free and open source distributed version control system" and highlights its "easy to learn" nature and "tiny footprint with lightning fast performance". To the right, there is a diagram illustrating the distributed nature of Git with multiple repositories connected. Below the main text, there are four sections: "About", "Documentation", "Downloads", and "Community". A green arrow points from the "Download for Windows" button in the "Downloads" section to the "Download for Windows" button on a computer monitor in the bottom right corner. The monitor displays the "Latest source Release 2.46.0" and the "Download for Windows" button.

GIT 환경구축

◆ GIT 설치

① 설치파일 다운로드

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

Click here to download

Download for Windows

[Click here to download](#) the latest (2.46.0) 64-bit version of Git for Windows. This is the most recent maintained build. It was released 17 days ago, on 2024-07-29.

Other Git for Windows downloads

Standalone Installer

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

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

GIT 환경구축

◆ GIT 설치

② 설치 파일 실행

- Git-2.46.0-64-bit.exe 파일 더블클릭

파일 열기 - 보안 경고

이 파일을 실행하시겠습니까?



이름: C:\Users\Wanece\Downloads\Git-2.46.0-64-bit.exe

게시자: [Johannes Schindelin](#)

유형: 응용 프로그램

시작: C:\Users\Wanece\Downloads\Git-2.46.0-64-bit.exe

클릭

실행(R)

취소

☒ 이 파일을 열기 전에 항상 확인(W)



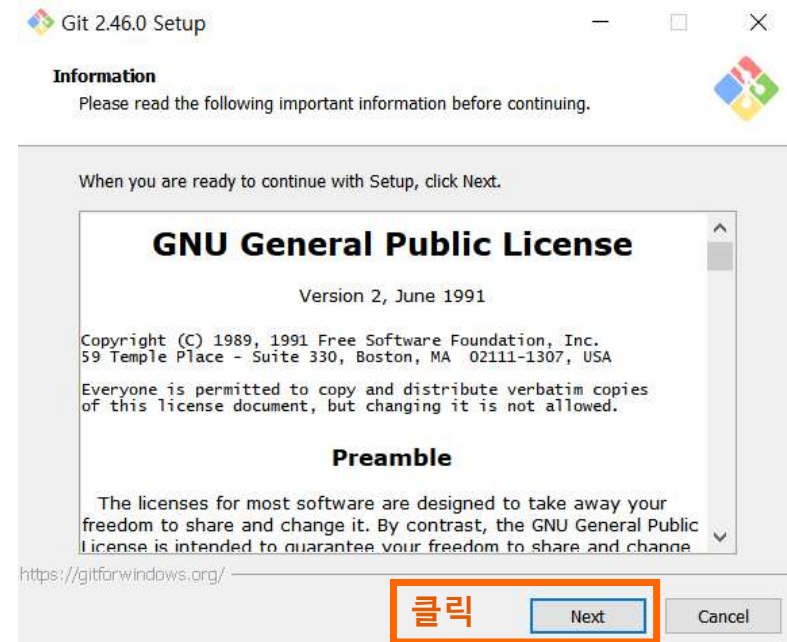
이 형식의 파일은 사용자의 컴퓨터에 피해를 줄 수 있습니다. 신뢰할 수 있는 게시자로부터의 소프트웨어만 실행하십시오. [위험성](#)

GIT 환경구축

◆ GIT 설치

③-1 설치 관련 설정

- 약관 확인

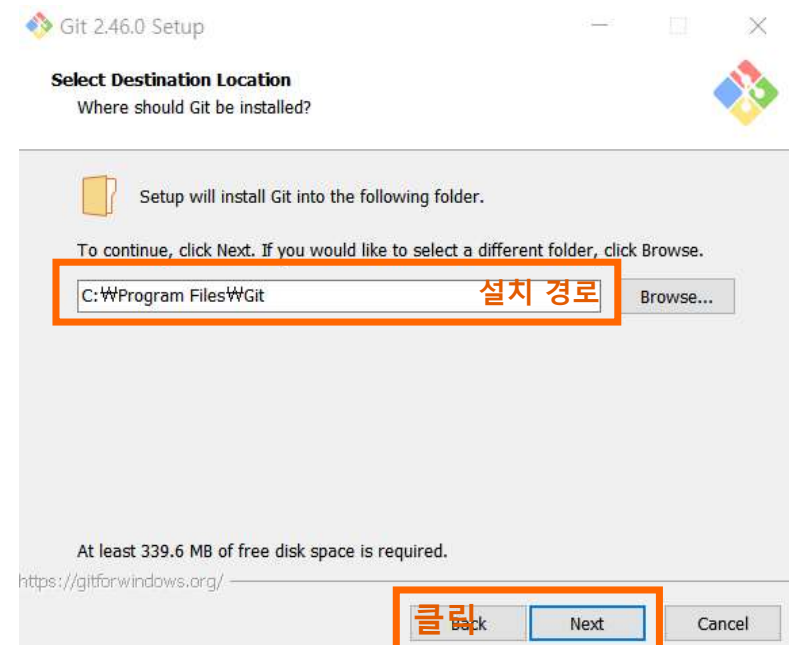


GIT 환경구축

◆ GIT 설치

③-2 설치 관련 설정

- 설치 경로 선택 설정



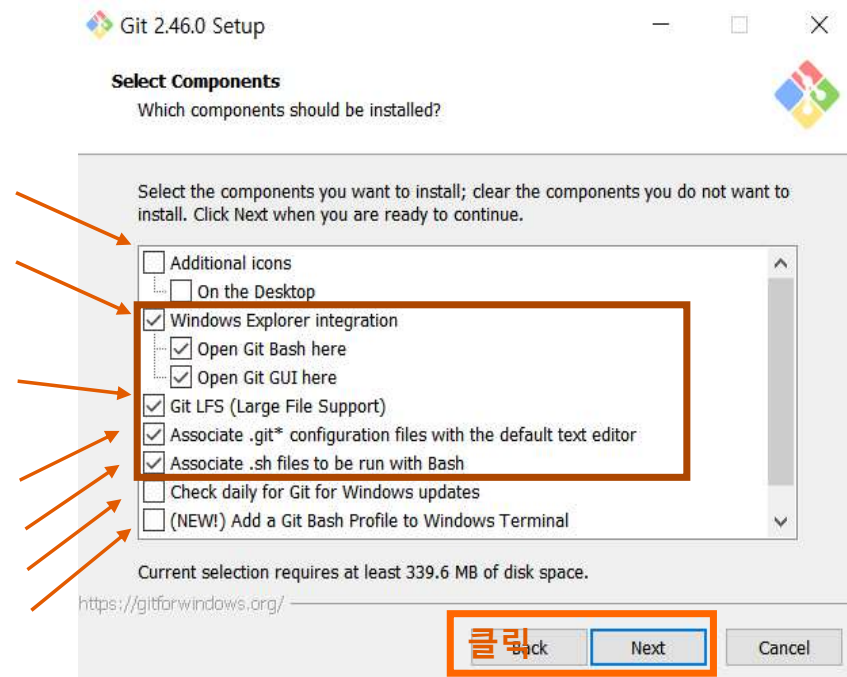
GIT 환경구축

◆ GIT 설치

③-3 설치 관련 설정

- 설치 구성요소 선택 설정

- 바탕화면에 바로가기 생성
- 폴더 오른쪽 클릭 메뉴에 Git Bash Here 추가
- 폴더 오른쪽 클릭 메뉴에 Git GUI Here 추가
- 대용량 파일 지원
- 기본 텍스트 에디터에 git 구성(.git 확장자) 연결
- Bash에 .sh 확장자 파일 연결
- 매일 새로운 업데이트 확인
- 윈도우 기본 터미널에 Git Bash 프로파일 추가

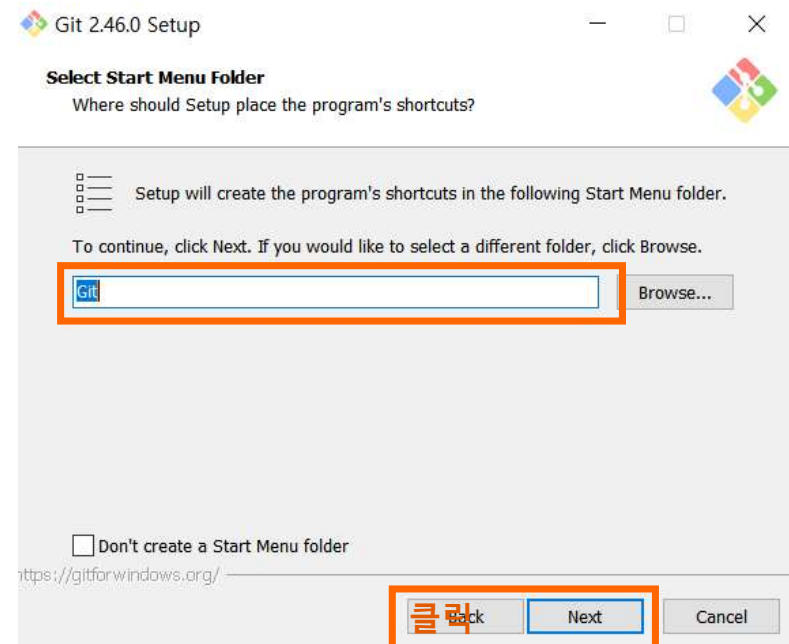


GIT 환경구축

◆ GIT 설치

③-4 설치 관련 설정

- 시작 메뉴 바로가기 및 폴더 경로 지정 설정

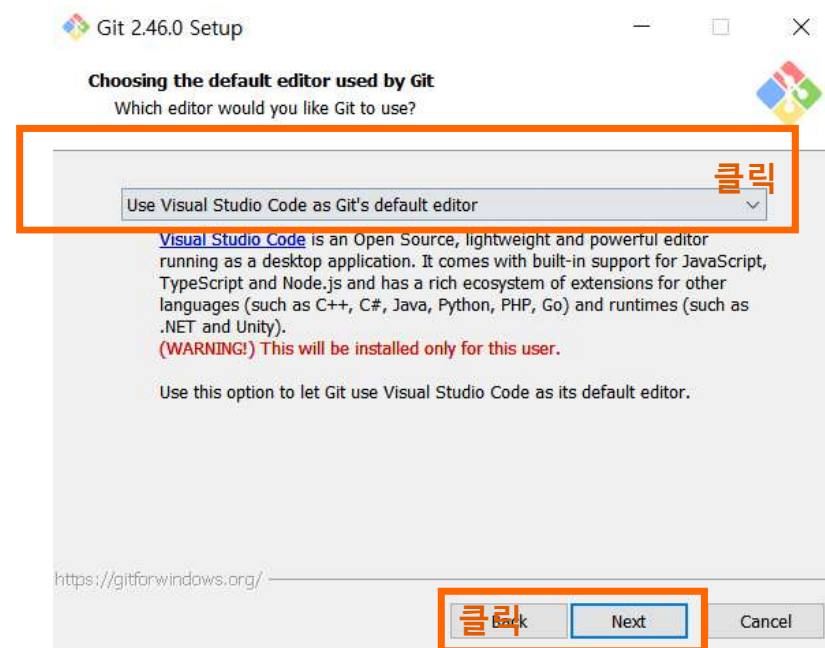


GIT 환경구축

◆ GIT 설치

③-5 설치 관련 설정

- Git 기본 편집기 선택 설정
 - : VSCODE를 Git 기본 편집기 지정
 - : 커밋 메시지와 기타 Git 관련 파일 편집 가능



GIT 환경구축

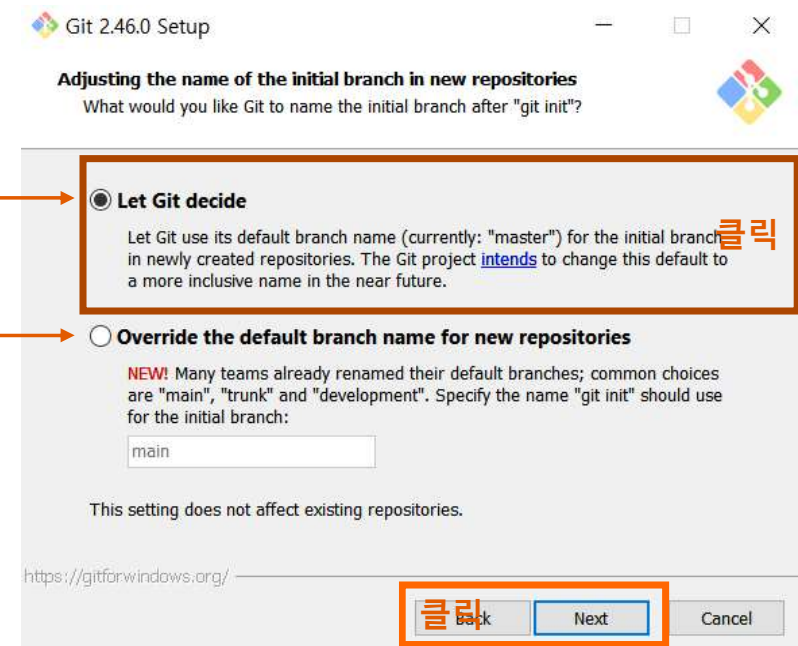
◆ GIT 설치

③-6 설치 관련 설정

- initial branch 이름 지정 방법 설정

기본 분기 이름("master") 사용

사용자 지정 분기 이름 사용



GIT 환경구축

◆ GIT 설치

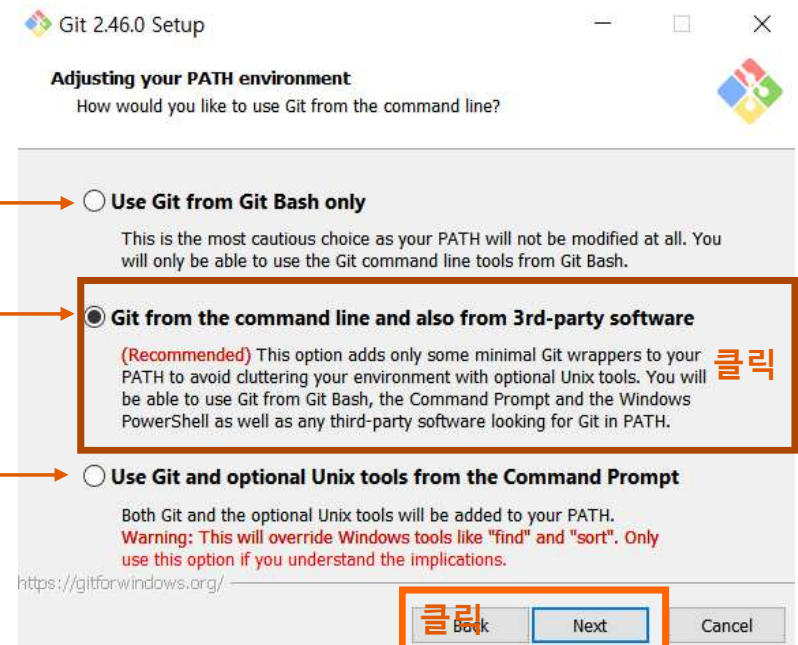
③-7 설치 관련 설정

- Git 커맨드 사용 위한 환경변수 설정

Git Bash에서만 Git 명령어 수행 가능

환경변수(PATH) 추가 → 윈도우 CMD에서 명령어 수행 가능

Git/Unix 도구 모두 환경변수(PATH) 추가



GIT 환경구축

◆ GIT 설치

③-8 설치 관련 설정

- SSH 실행 도구를 선택 설정

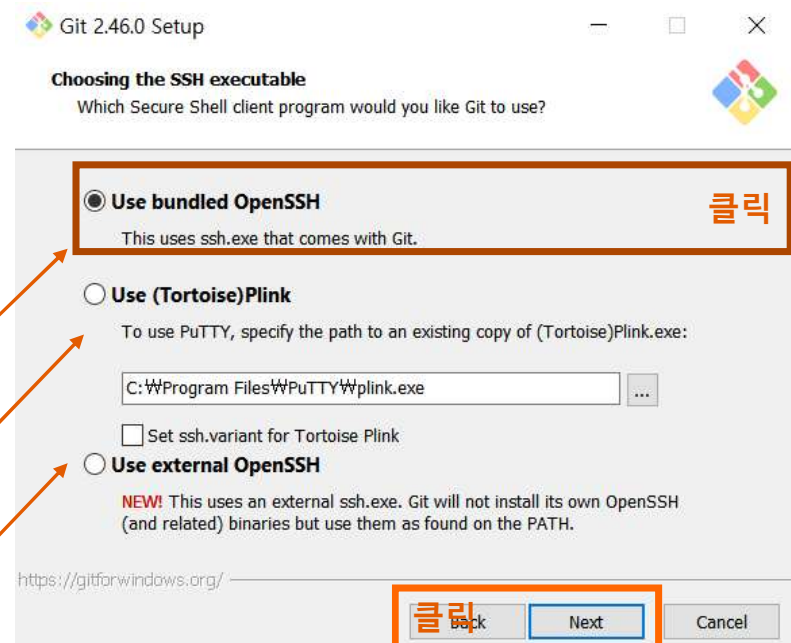
SSH(Secure Shell)

네트워크 상의 다른 컴퓨터에 로그인하여 명령 실행하고
정보 교환할 수 있도록 해 주는 **보안 프로토콜**

GIT 기본 제공 OpenSSH 사용

PuTTY 제품군 Plink 사용

외부 OpenSSH 사용



GIT 환경구축

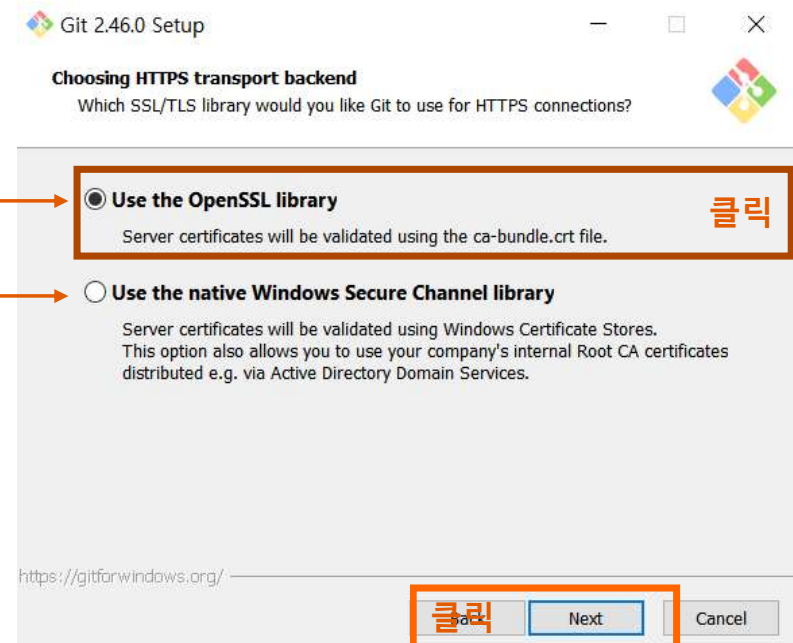
◆ GIT 설치

③-9 설치 관련 설정

- HTTP 연결 옵션 선택 설정

OpenSSL 라이브러리 ca-bundle.crt 사용 검증

Windows 인증서 저장소 사용하여 검증



GIT 환경구축

◆ GIT 설치

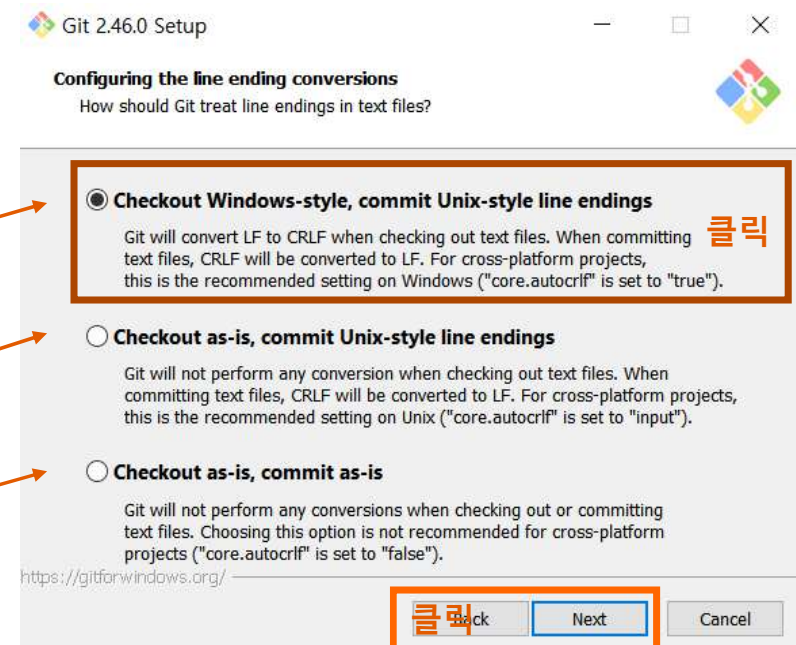
③-10 설치 관련 설정

- 저장소 체크인/아웃 시 줄 바꿈 방법 선택 설정
- 윈도우: `WrWn` 유닉스: `Wn`

체크아웃: 윈도우 / 커밋: 유닉스 자동 변경 설정

체크아웃: 변경 없이 / 커밋: 유닉스 설정

체크아웃 / 커밋 모두 스타일 변경 없이 진행



GIT 환경구축

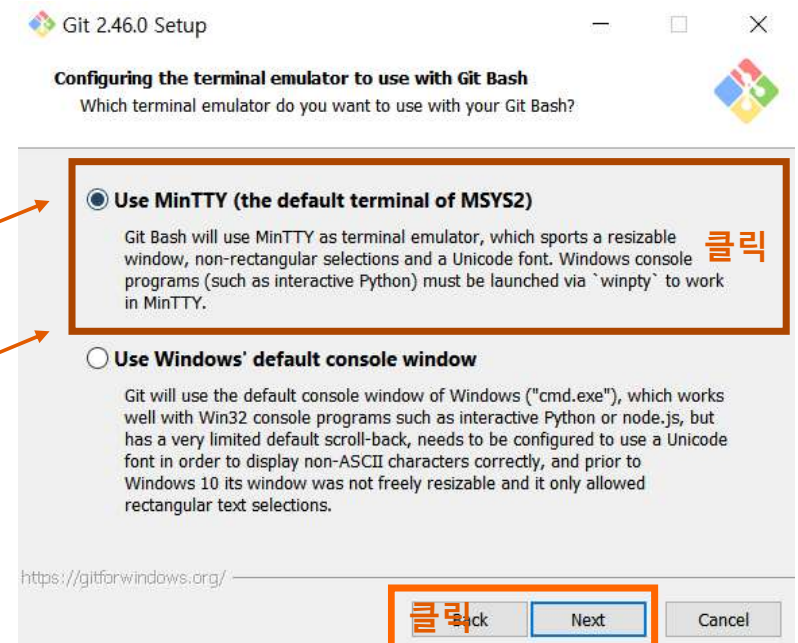
◆ GIT 설치

③-11 설치 관련 설정

- Git Bash 터미널 에뮬레이터를 선택 설정

Git Bash 기본 터미널 에뮬레이터(MinTTY) 사용

윈도우 기본 콘솔(cmd) 사용



GIT 환경구축

◆ GIT 설치

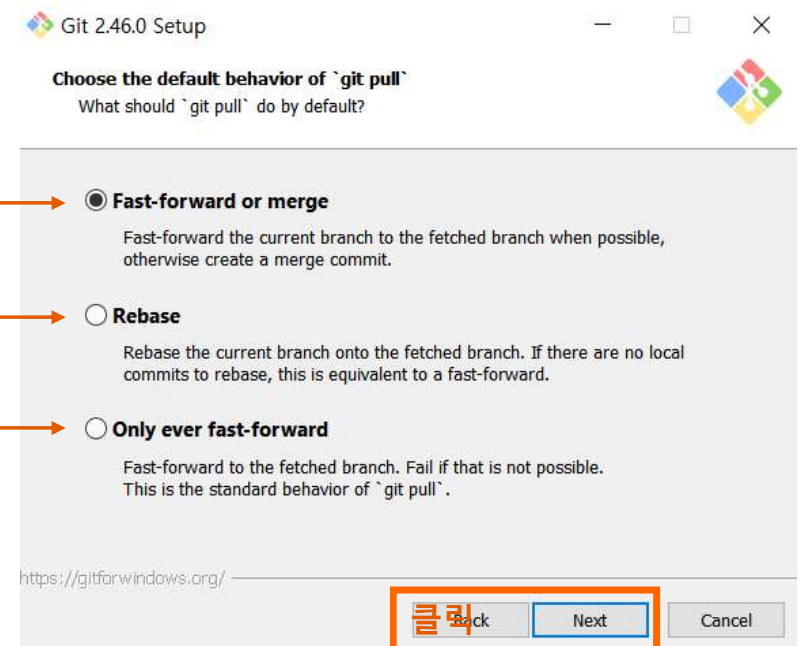
③-12 설치 관련 설정

- Git pull 명령어 수행 작업 선택 설정

기본으로 설정

현재 분기를 불러온 분기에 재배치

불러온 분기로 빠르게 이동 / 명령어 수행 실패 가능성



GIT 환경구축

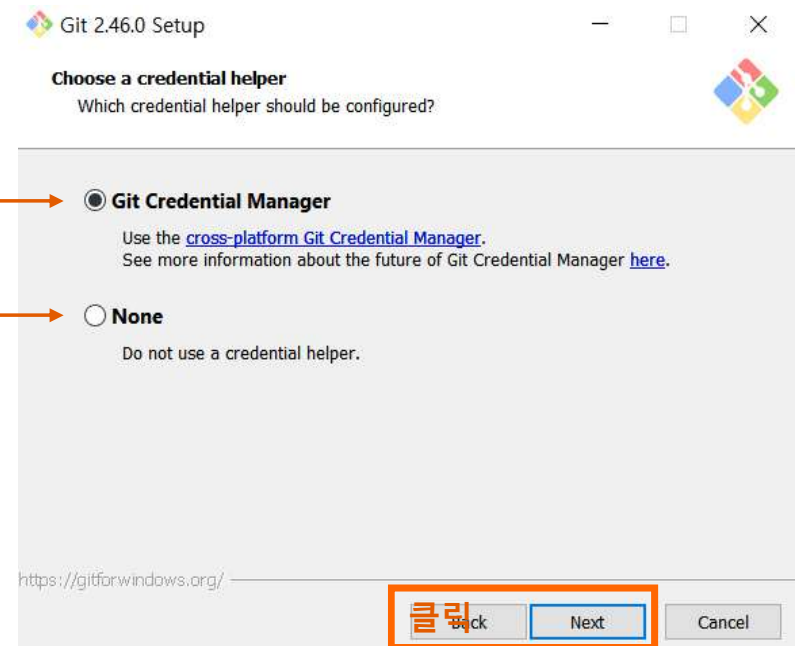
◆ GIT 설치

③-13 설치 관련 설정

- 자격 증명 도우미 선택 설정

Git의 자격 증명 도우미를 사용

자격 증명 도우미를 사용하지 않음



GIT 환경구축

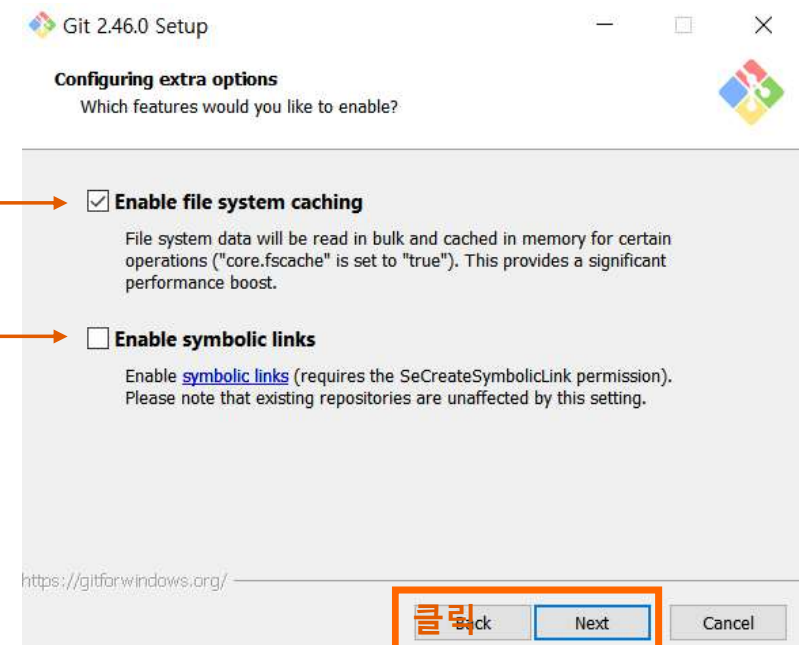
◆ GIT 설치

③-14 설치 관련 설정

- 기타 옵션 선택 설정

파일 시스템 캐싱 활성화 → 상당한 성능 향상 제공

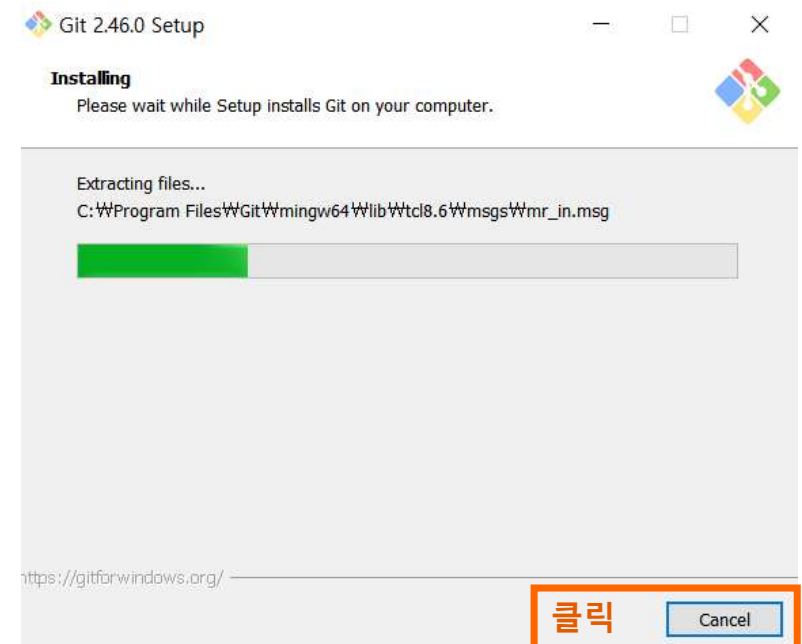
심볼릭 링크 활성화 → SeCreateSymbolicLink 권한 필요



GIT 환경구축

◆ GIT 설치

④ 설치 진행



GIT 환경구축

◆ GIT 설치

⑤ 설치 완료

Git 2.46.0 Setup

Completing the Git Setup Wizard



Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

- ☐ Launch Git Bash
- ☒ View Release Notes

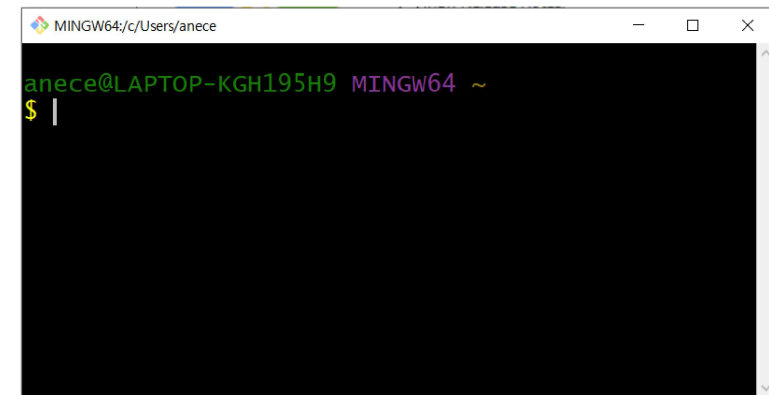
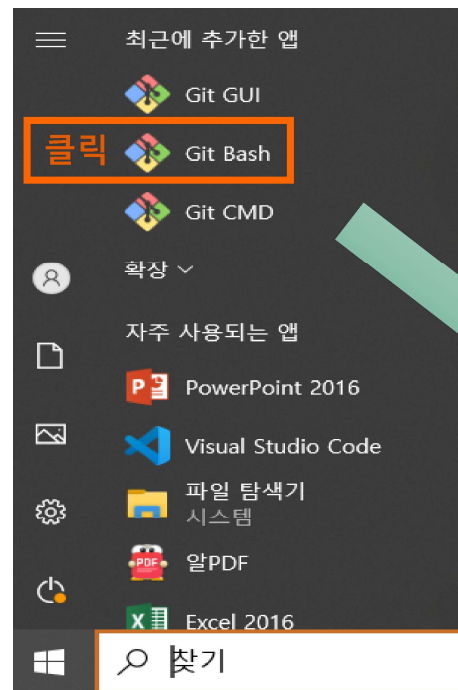
클릭

Finish

GIT 환경구축

◆ GIT 설치

⑥ 설치 확인

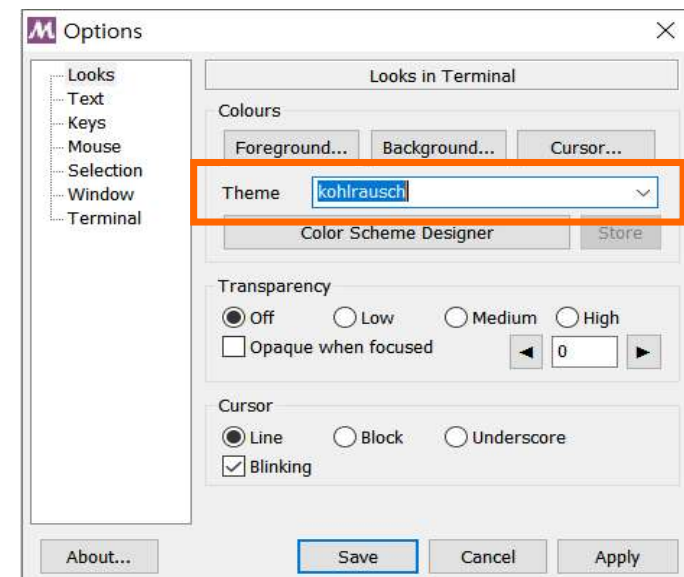
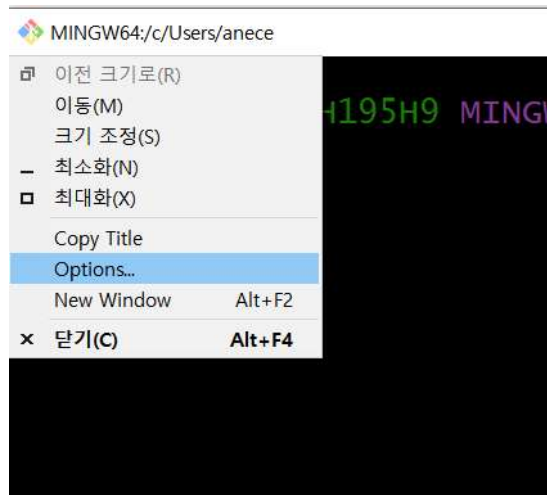


GIT 환경구축

◆ GIT 설치

⑦ Git Bash 환경설정

- 테마설정

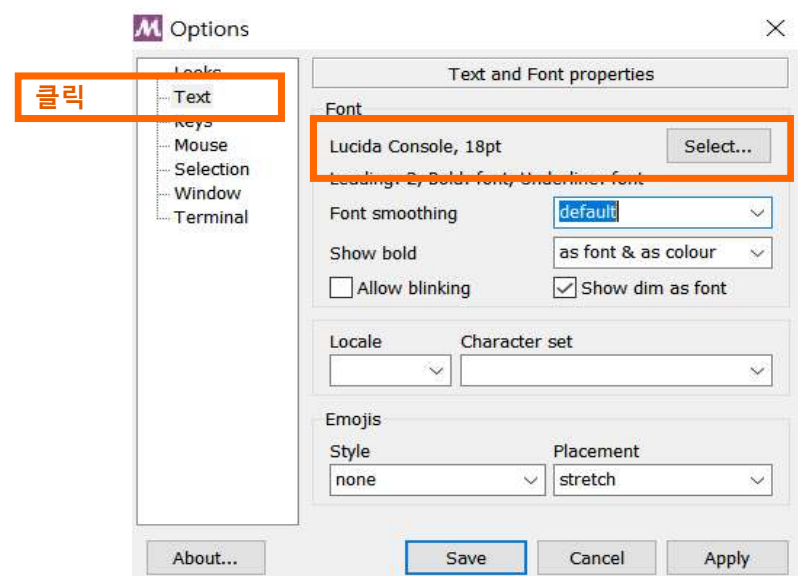


GIT 환경구축

◆ GIT 설치

⑦ Git Bash 환경설정

- 글자 크기/폰트 설정



GIT 환경구축

◆ GIT 설치

⑦ Git Bash 환경설정

- 줄바꿈 설정

* 윈도우 : `\\r\\n` ← 다름 → 맥/리눅스 : `\\n`

명령어 입력

MINGW64:/c/Users/anece

```
anece@LAPTOP-KGH195H9 MINGW64 ~  
$ git config --global core.autocrlf true
```

SourceTress 환경구축

SOURCETREE 환경구축

◆ GIT GUI - Sourcetree

① 설치 파일 다운로드

- <https://www.sourcetreeapp.com/>

Download for Windows

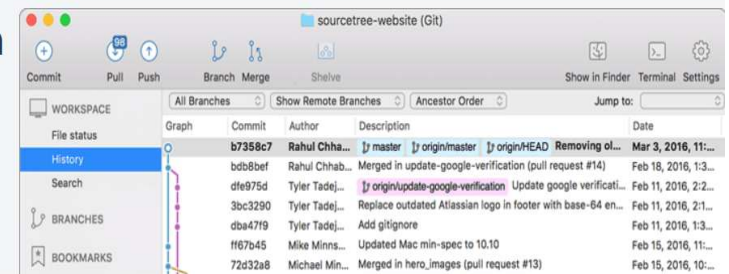
 Sourcetree

Download free

Simplicity and power in
a beautiful Git GUI

Download for Windows

Also available for Mac OS X



SOURCETREE 환경구축

◆ GIT GUI - Sourcetree

① 설치 파일 다운로드

- <https://www.sourcetreeapp.com/>

Important information ×

To continue downloading this product you must read and agree to the Atlassian Software License Agreement and Privacy Policy.

☒ I agree to the Atlassian Software License Agreement and Privacy Policy.

Click

Download

Cancel

SOURCETREE 환경구축

◆ GIT GUI - Sourcetree

② 설치 파일 실행

- SourceTreeSetup-3.4.18.exe 더블클릭

파일 열기 - 보안 경고

이 파일을 실행하시겠습니까?



이름: ...Users\Wanece\Downloads\SourceTreeSetup-3.4.18.exe

게시자: [Atlassian Pty Ltd](#)

유형: 응용 프로그램

시작: C:\Users\Wanece\Downloads\SourceTreeSetup-3.4.18...

클릭

실행(R)

취소

☒ 이 파일을 열기 전에 항상 확인(W)



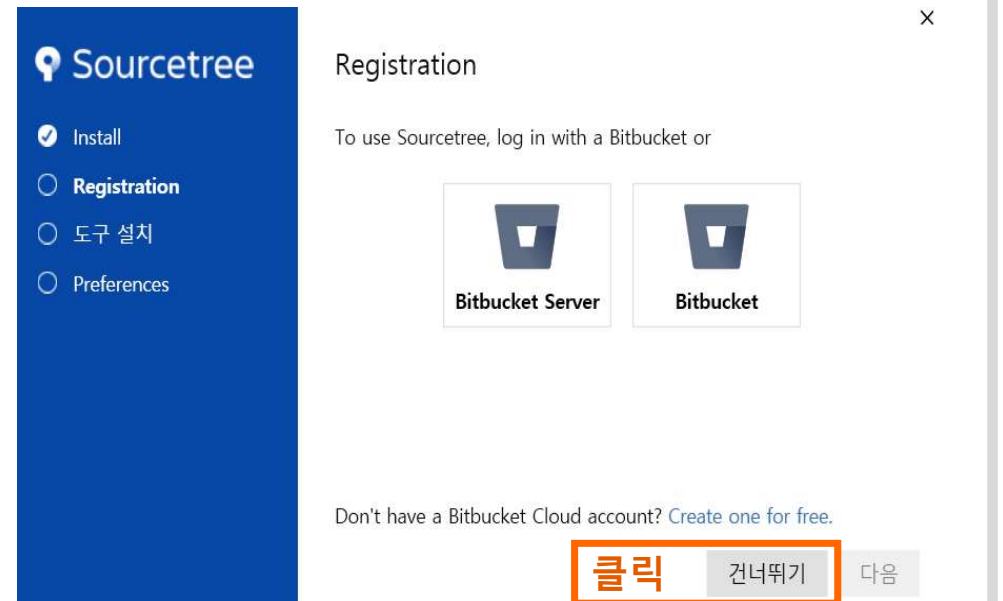
이 형식의 파일은 사용자의 컴퓨터에 피해를 줄 수 있습니다. 신뢰할 수 있는 게시자로부터의 소프트웨어만 실행하십시오. [위험성](#)

SOURCETREE 환경구축

◆ GIT GUI - Sourcetree

③-1 설치 관련 설정

- Bitbucket 서비스 사용 로그인 설정

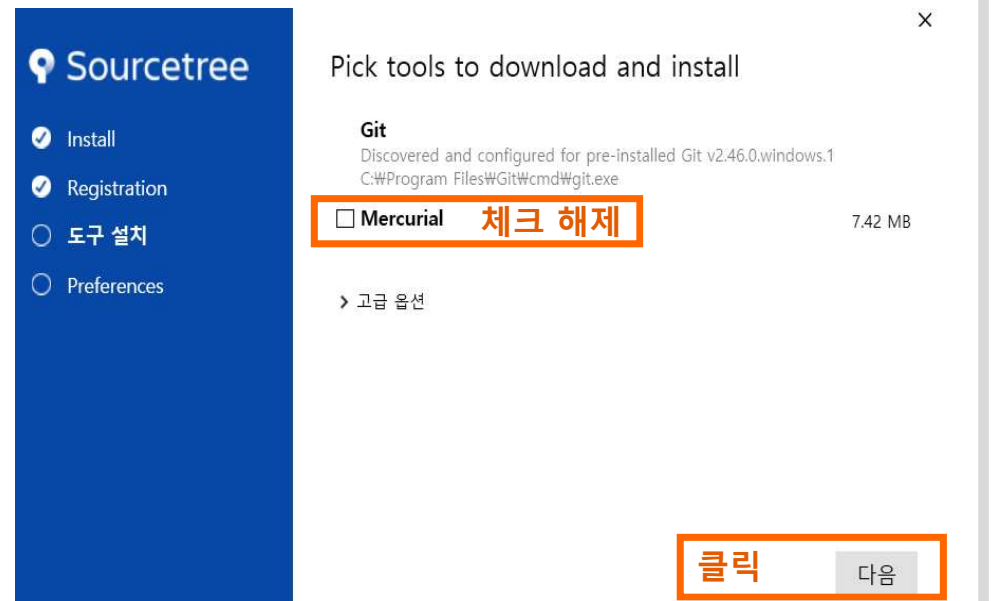


SOURCETREE 환경구축

◆ GIT GUI - Sourcetree

③-2 설치 관련 설정

- Mercurial 버전관리시스템 설정

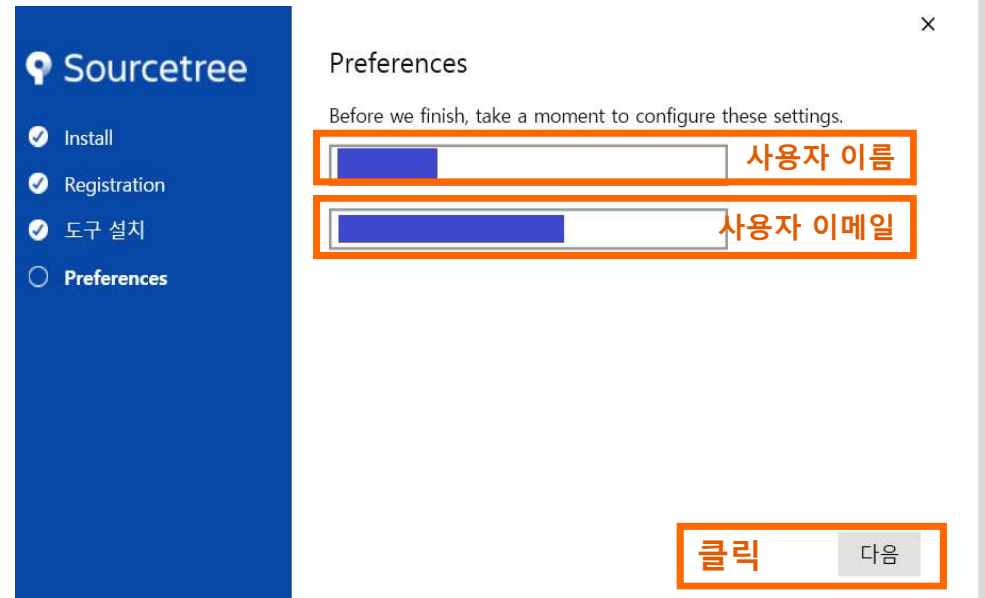


SOURCETREE 환경구축

◆ GIT GUI - Sourcetree

③-3 설치 관련 설정

- 사용자 설정



The image shows the 'Sourcetree Preferences' dialog box. On the left is a blue sidebar with the 'Sourcetree' logo and a list of settings: 'Install' (checked), 'Registration' (checked), '도구 설치' (checked), and 'Preferences' (selected). The main area is titled 'Preferences' and contains the text 'Before we finish, take a moment to configure these settings.' Below this are two text input fields, each with a blue cursor and an orange border. The first field is labeled '사용자 이름' (User Name) and the second is labeled '사용자 이메일' (User Email). At the bottom right, there are two buttons: '클릭' (Click) and '다음' (Next), both with orange borders.

Sourcetree

- ☒ Install
- ☒ Registration
- ☒ 도구 설치
- ☐ Preferences

Preferences

Before we finish, take a moment to configure these settings.

사용자 이름

사용자 이메일

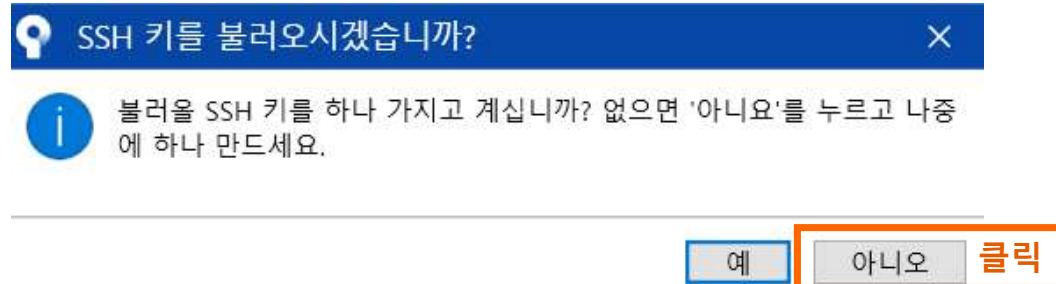
클릭 다음

SOURCETREE 환경구축

◆ GIT GUI - Sourcetree

③-4 설치 관련 설정

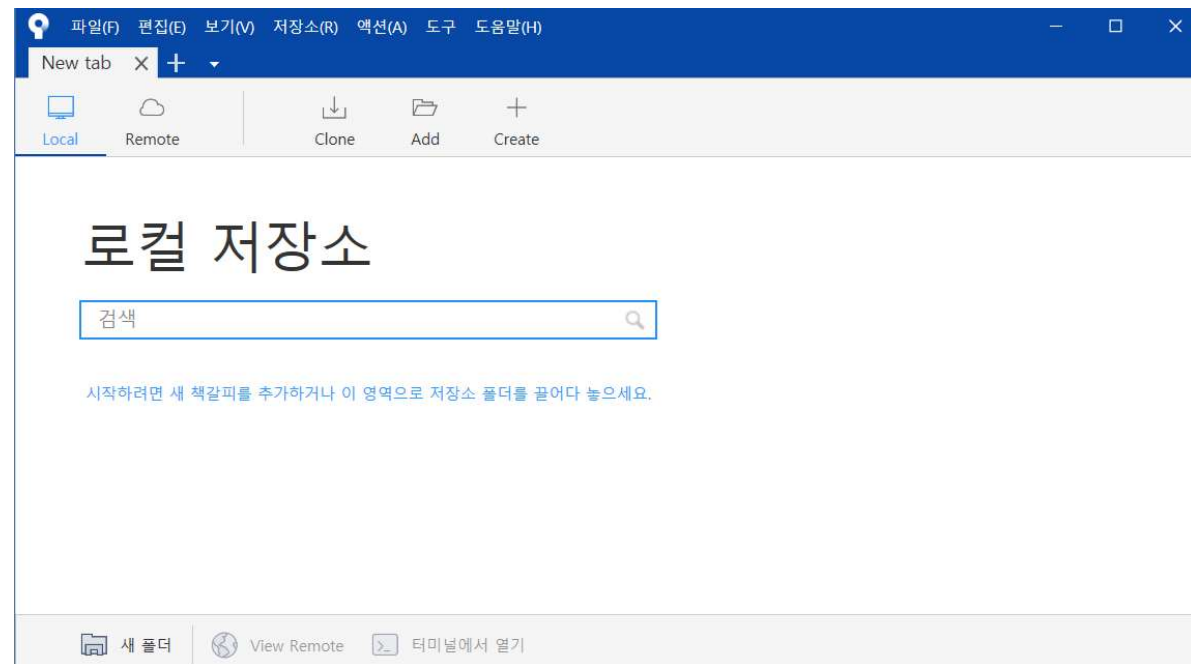
- SSH 키 로딩 여부 설정



SOURCETREE 환경구축

◆ GIT GUI - Sourcetree

④ 설치 완료



VSCODE 환경구축

VSCODE 환경구축

◆ VSCODE 설치

❖ CLI (Command Line Interface) : 명령어 입력기반 사용

- GIT에서 작업 실행 시 사용

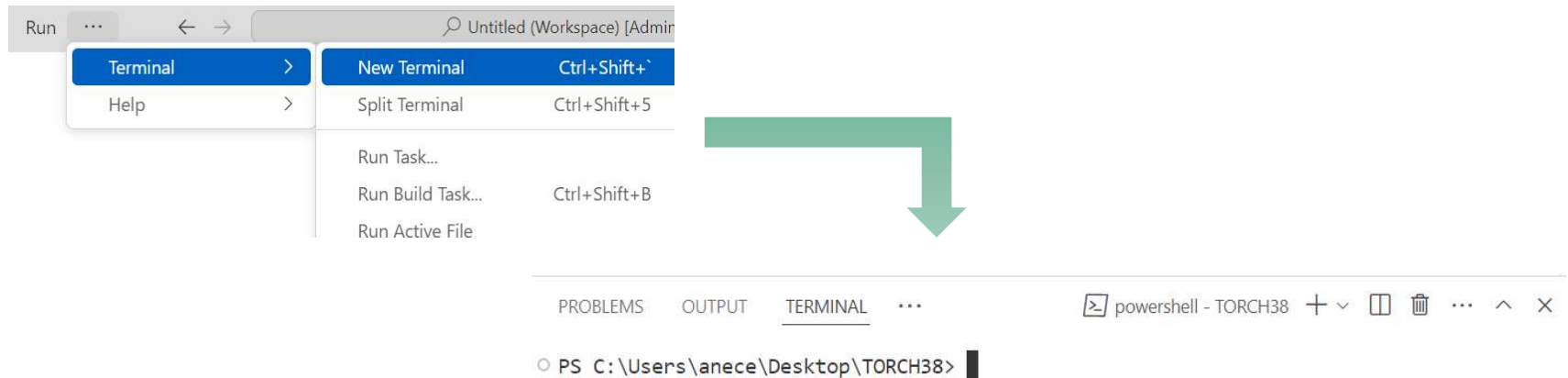
❖ GUI (Graphical User Interface) : 그래픽 요소 입력기반 사용

- 프로젝트 상태 체크 시 사용

VSCODE 환경구축

◆ VSCODE 설치 - Git Bash 터미널 설정

① Terminal 메뉴 클릭 >> New Terminal



VSCODE 환경구축

◆ VSCODE 설치 - Git Bash 터미널 설정

② Git Bash 연결

The image shows a sequence of steps in VS Code to set Git Bash as the default terminal. It starts with a terminal window where the command `>select default` is entered. A red box highlights the command, and a red label '클릭' (Click) points to it. Below the command, a list of suggestions is shown, with 'Terminal: Select Default Profile' highlighted in blue. A green arrow points from this suggestion to a 'Select your default terminal profile' dialog box. In this dialog, 'Git Bash' is selected and highlighted with a red box, with a red label '클릭' (Click) pointing to it. The dialog lists several profiles: 'Command Prompt', 'Git Bash', 'PowerShell', 'Ubuntu (WSL)', and 'Azure Cloud Shell (Bash)'. The 'Git Bash' profile is the one being selected.

>select default

클릭 Terminal: Select Default Profile similar commands ⚙️

Preferences: Open Default Keyboard Shortcuts (JSON)
Preferences: Open Default Settings (JSON)
Show or Focus Standalone Color Picker
Tasks: Configure Default Build Task

Select your default terminal profile

profiles

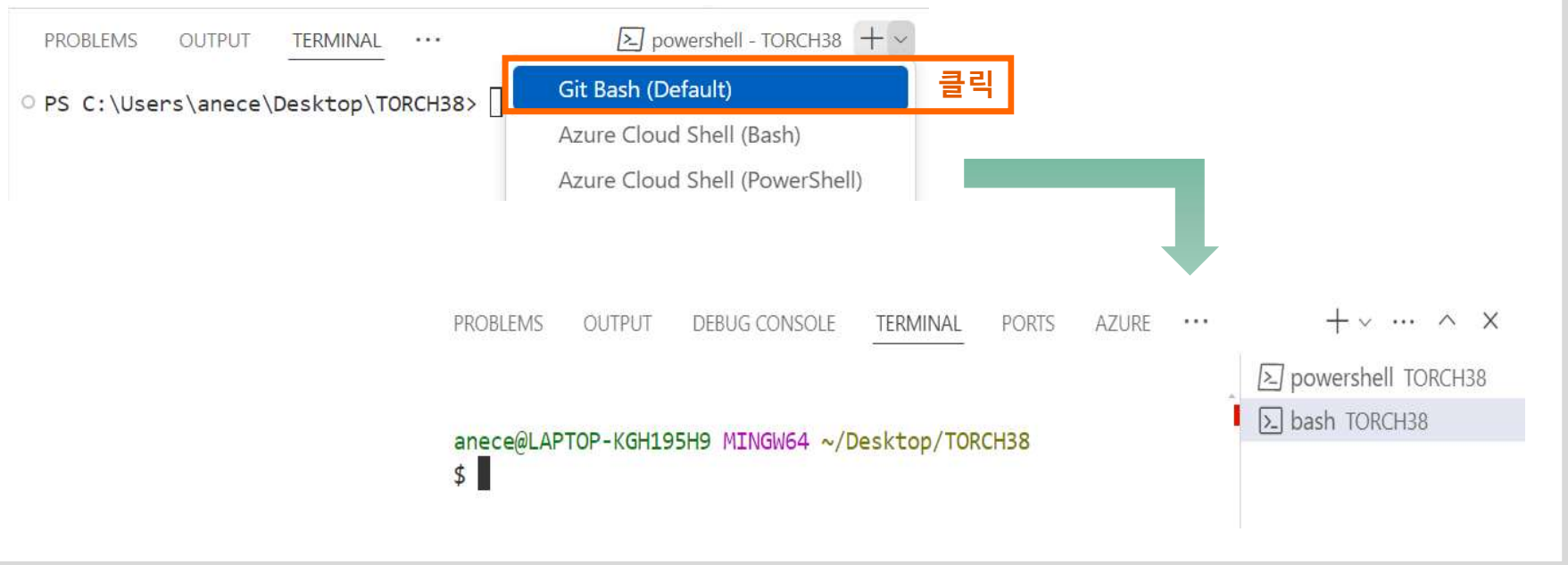
클릭 Git Bash C:\Program Files\Git\bin\bash.exe --login -i ⚙️

PowerShell C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe
Ubuntu (WSL) C:\WINDOWS\System32\wsl.exe -d Ubuntu
Azure Cloud Shell (Bash) contributed

VSCODE 환경구축

◆ VSCODE 설치 - Git Bash 터미널 설정

③ Git Bash 연동 터미널 실행



GIT 프로젝트 생성

GIT 프로젝트 생성

◆ GIT 최초 설정 → 사용자 설정

❖ GIT 사용자 이름/이메일 주소 설정

- GIT 설치 시 가입한 계정과 별개
- 다른 사람과 협업 시 표시되는 사용자

명령어

이름 설정 : `git config --global user.name OOOO`

이메일 설정 : `git config --global user.email OOOO`

GIT 프로젝트 생성

◆ GIT 최초 설정 → 사용자 설정

❖ GIT 사용자 이름/이메일 주소 설정

설정

```
anece@LAPTOP-KGH195H9 MINGW64 ~  
$ git config --global user.name "anece"  
  
anece@LAPTOP-KGH195H9 MINGW64 ~  
$ git config --global user.email "anece00@naver.com"
```

확인

```
anece@LAPTOP-KGH195H9 MINGW64 ~  
$ git config --global user.name  
anece  
  
anece@LAPTOP-KGH195H9 MINGW64 ~  
$ git config --global user.email  
anece00@naver.com
```

GIT 프로젝트 생성

◆ GIT 최초 설정 → 기본 브랜치 설정

❖ 기본 브랜치 이름 변경 설정

- [기존] master → [변경] main 또는 trunk

명령어

이름 변경 : `git config --global init.defaultBranch` OOO

```
anece@LAPTOP-KGH195H9 MINGW64 ~  
$ git config --global init.defaultBranch main
```


GIT 프로젝트 생성

◆ GIT 프로젝트 생성 → Git 관리 프로젝트

❖ .git 폴더 생성

① 윈도우 탐색기 실행

- 위치 : 원하는 곳
- 이름 : git_practice
- 주의 : 한글 포함되면 안됨!!!

> 사용자 > anece

이름

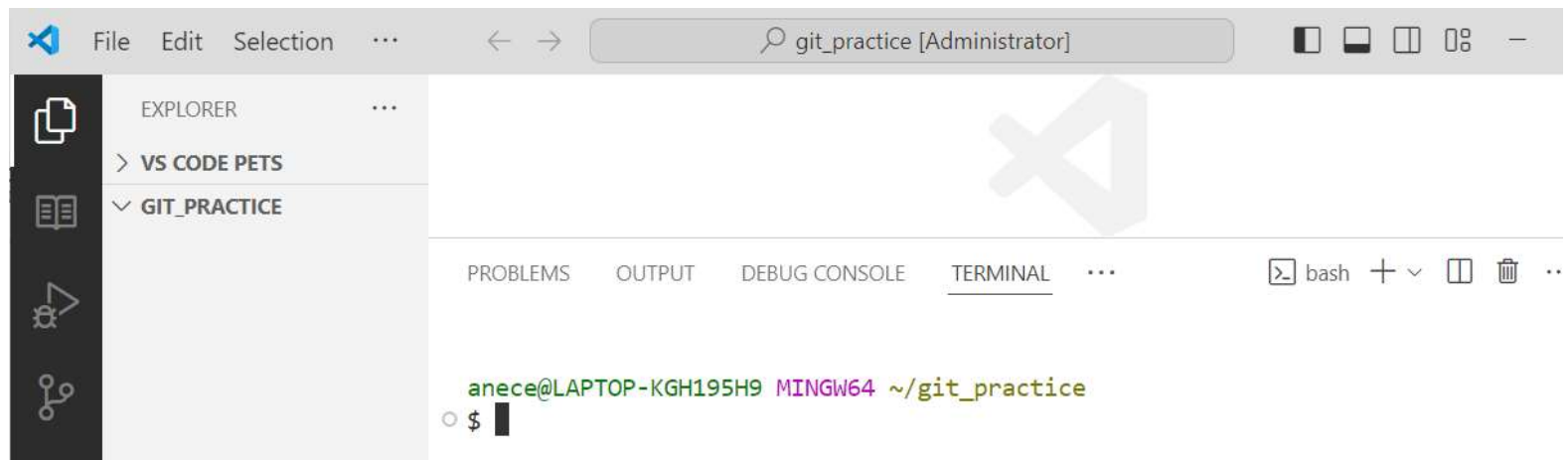
- EXAM_ML2
- EXAM_NLP
- EXAM_SERVICE_ML
- EXAM-ML
- git_practice

GIT 프로젝트 생성

◆ GIT 프로젝트 생성 → Git 관리 프로젝트

❖ .git 폴더 생성

② VSCODE에서 프로젝트 열기



GIT 프로젝트 생성

◆ GIT 프로젝트 생성 → Git 관리 프로젝트

❖ .git 폴더 생성

③ git_practice 폴더 관리 설정

명령어 git 관리 설정: **git init**

```
anece@LAPTOP-KGH195H9 MINGW64 ~/git_practice
● $ git init
Initialized empty Git repository in C:/Users/anece/git_practice/.git/

anece@LAPTOP-KGH195H9 MINGW64 ~/git_practice (main)
○ $ █
```

GIT 프로젝트 생성

◆ GIT 프로젝트 생성 → Git 관리 프로젝트

❖ .git 폴더 생성

③ git_practice 폴더 관리 설정

anece > git_practice

관리 내역 저장 폴더



GIT 프로젝트 생성

◆ 실습

❖ 파일 생성

- 도 구 : VSCODE
- 파일명 : tigers.yaml

```
team: Tigers

manager: John

member:
- Linda
- William
- David
```

GIT 프로젝트 생성

◆ 실습

❖ 파일 생성

- 도 구 : VSCODE
- 파일명 : lions.yaml

```
team: Lions

manager: Mary

member:
- Thomas
- Karen
- Margaret
```

GIT 프로젝트 생성

◆ 실습

❖ 파일 & Git 연동

명령어

현재 git 상태 확인 : **git status**

```
anece@LAPTOP-KGH195H9 MINGW64 ~/git_practice (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    linons.yaml
    tigers.yaml

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

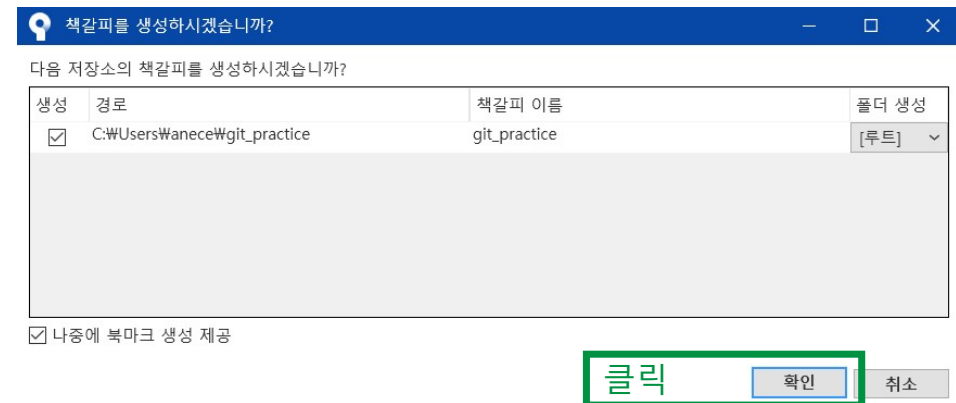
GIT 프로젝트 생성

◆ GIT 프로젝트 생성 → SourceTree

❖ git_practice 폴더 연동

① SourceTree 실행 >> 파일 메뉴 클릭

- git_practice 폴더 선택

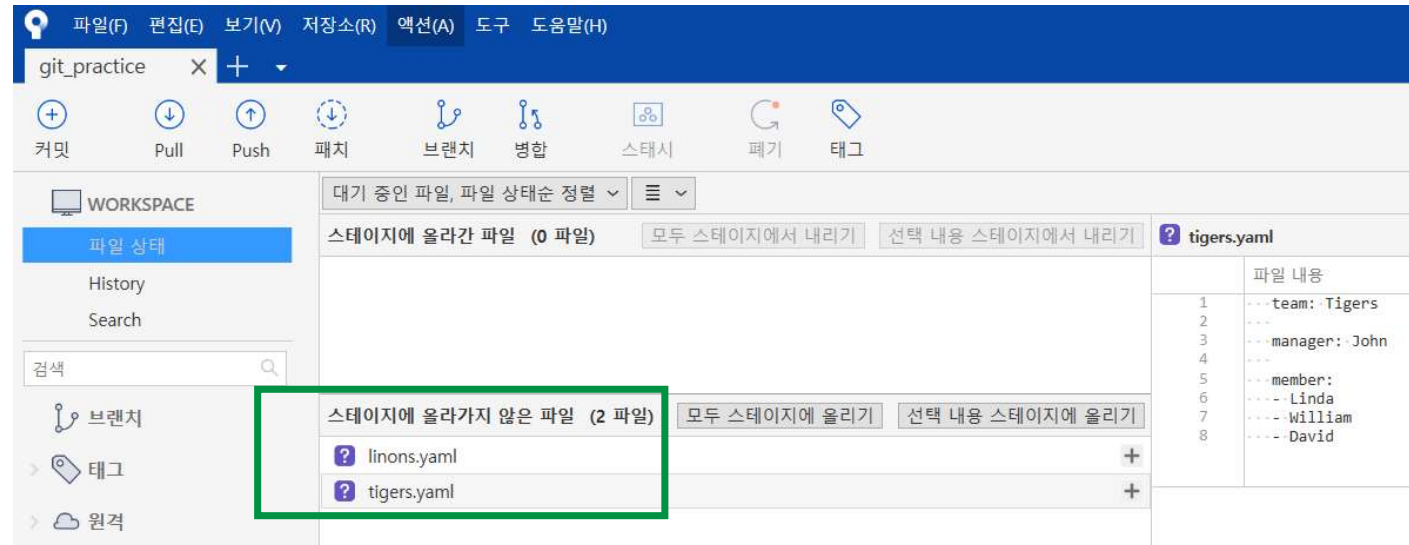


GIT 프로젝트 생성

◆ GIT 프로젝트 생성 → SourceTree

❖ git_practice 폴더 연동

① SourceTree 실행 >> 파일 메뉴 클릭

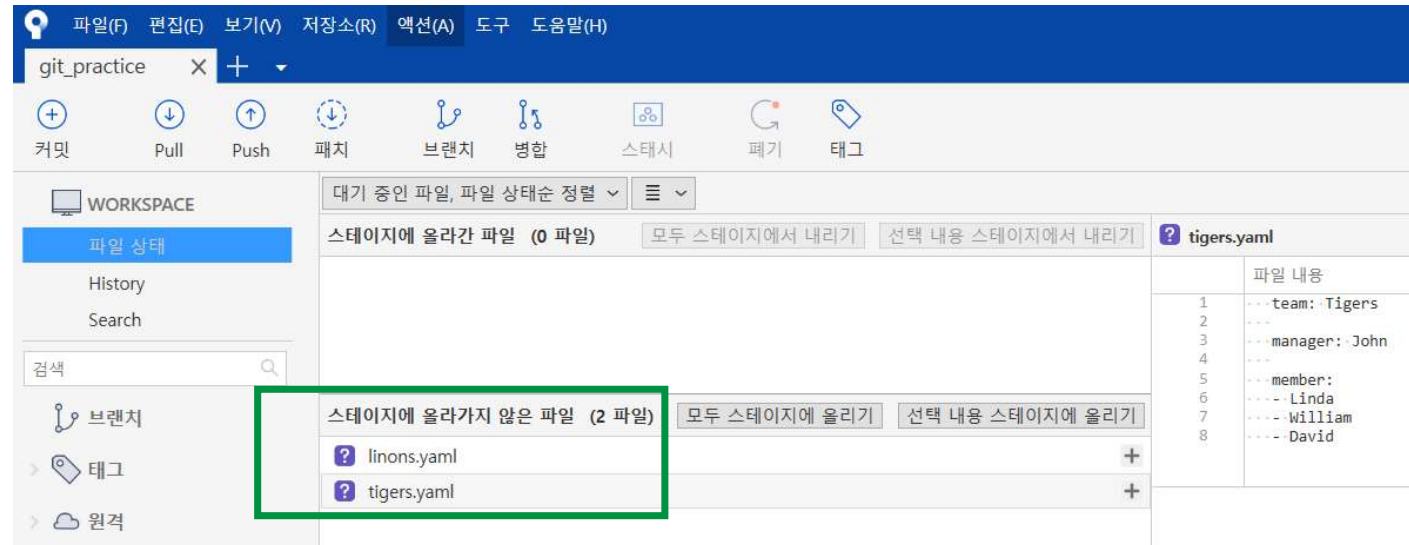


GIT 프로젝트 생성

◆ GIT 프로젝트 생성 → SourceTree

❖ git_practice 폴더 연동

① SourceTree 실행 >> 파일 메뉴 클릭



GIT 프로젝트 생성

◆ 실습

❖ 파일 & Git 연동 해제

- 프로젝트 폴더 → .git 폴더 삭제
 - 상태 확인

`git status`

```
anece@LAPTOP-KGH195H9 MINGW64 ~  
$ git status  
fatal: not a git repository (or any of the parent directories): .git  
  
anece@LAPTOP-KGH195H9 MINGW64 ~  
$
```

GIT 프로젝트 생성

◆ GIT 관리 제외 목록 생성

❖ .gitignore 파일

- Git 버전 관리에서 제외할 파일/폴더 목록을 지정하는 파일
- Git의 root 디렉터리에 저장
- git status 를 이용했을 때 보이지 않음 → tracking되지 않음!

- 관리 불필요 파일 : 빌드 결과물, 다운로드 가능한 라이브러리
- 보안상 중요 파일 : 서버 비밀번호, 보안 관련 파일/폴더 등등

GIT 프로젝트 생성

◆ 실습 - GIT 관리 제외 목록 생성

❖ 보안관련 파일 GIT 관리 제외

- 프로젝트 폴더 → git_practices 폴더

- secrets.yam 파일 생성

```
id: admin  
pw: 12341234
```

- GIT 상태 확인

```
anece@LAPTOP-KGH195H9 MINGW64 ~/git_practice (main)  
$ git status  
On branch main  
  
No commits yet  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
linons.yaml  
secrets.yaml  
tigers.yaml
```

GIT 프로젝트 생성

◆ 실습 - GIT 관리 제외 목록 생성

❖ 보안관련 파일 GIT 관리 제외

- 프로젝트 폴더 → git_practices 폴더

- .gitignore 파일 생성

secrets.yaml

- GIT 상태 확인

```
anece@LAPTOP-KGH195H9 MINGW64 ~/git_practice (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        lions.yaml
        tigers.yaml
```

GIT 프로젝트 생성

◆ GIT 관리 제외 목록 생성

❖ .gitignore 파일 형식

- 특정 파일 관리 제외/무시

```
.gitignore
```

```
secrets.yaml
```

```
# 해당 파일명과 동일한 파일 관리에서 제외/무시
```

GIT 프로젝트 생성

◆ GIT 관리 제외 목록 생성

❖ .gitignore 파일 형식

- 특정 위치의 특정 파일 관리 제외/무시

`.gitignore`

`/[파일 명] 또는 .[파일 명]`

`/secrets.yaml`

`# 최상위 폴더 안 해당 파일은 관리에서 제외/무시`

GIT 프로젝트 생성

◆ GIT 관리 제외 목록 생성

❖ .gitignore 파일 형식

- 특정 확장자/특정 파일명 파일 관리 제외/무시

.gitignore

/*.c

확장자가 c인 파일은 관리에서 제외/무시

/secrets.*

secrets파일명 파일은 관리에서 제외/무시

GIT 프로젝트 생성

◆ GIT 관리 제외 목록 생성

❖ .gitignore 파일 형식

- 특정 확장자/특정 파일명 파일 관리 제외/무시

.gitignore

/*.c

확장자가 c인 파일은 관리에서 제외/무시

!/information.c

확장자가 c인 파일이지만 관리에서 제외 되지 X

GIT 프로젝트 생성

◆ GIT 관리 제외 목록 생성

❖ .gitignore 파일 형식

- 특정 파일명의 파일 모든 폴더에서 관리 제외/무시

```
.gitignore
```

```
information
```

```
# 프로젝트 내 모든 폴더에서 해당 파일명 파일 제외/무시
```

GIT 프로젝트 생성

◆ GIT 관리 제외 목록 생성

❖ .gitignore 파일 형식

- 특정 폴더의 모든 파일들 관리 제외/무시

`.gitignore`

`[디렉터리 명]/`

`logs/`

`# logs폴더 내 모든 내용들 관리에서 제외/무시`

GIT 프로젝트 생성

◆ GIT 관리 제외 목록 생성

❖ .gitignore 파일 형식

- 특정 폴더의 바로 하위에 모든 파일들 관리 제외/무시

`.gitignore`

[디렉터리 명]/[파일 명]

`logs/debug.log`

`# logs폴더 내 debug.log 파일 관리에서 제외/무시`

`logs/*.yaml`

GIT 프로젝트 생성

◆ GIT 관리 제외 목록 생성

❖ .gitignore 파일 형식

- 특정 폴더의 내부 하위 폴더 내 파일 관리 제외/무시

`.gitignore`

[디렉터리 명]/[파일 명]

`logs/debug.log` # logs폴더 내 바로 아래 debug.log 파일 관리에서 제외/무시

`logs/**/*.yaml` # logs폴더 내 모든 하위 폴더 내의 yaml 파일 관리에서 제외/무시

GIT 프로젝트 생성

◆ GIT 관리 제외 목록 생성

❖ .gitignore 파일 형식

명령어

파일 삭제 명령어 : `git rm [파일명]`

`git commit -m [메시지]`

- 커밋으로 올라간 파일은 gitignore을 하기 위해서는 먼저 파일 제거

.gitignore

```
git rm secrets.yaml
```

```
git commit -m "secrets.yaml 제거"
```

GIT 활용

GIT 프로젝트 관리 활용

◆ GIT 영역

❖ git clone 명령어

- 원격 저장소(remote repository)에서 로컬 저장소(local repository)로 프로젝트를 복제
- 원격 저장소의 전체 히스토리와 파일을 로컬 저장소로 가져오는 데 사용

명령어	<code>git clone <repository></code>	# 원격 저장소를 로컬로 복제
-----	---	------------------

명령어	<code>git clone <repository> <directory></code>	# 원격 저장소 -> 지정 폴더로 복제
-----	---	-----------------------

GIT 프로젝트 관리 활용

◆ GIT 영역

❖ Working Directory

- 작업공간
- 준비 되지 않은 파일들 존재

❖ Staging Area

- 대기 공간
- GIT에 기록될 파일들 존재

❖ Repository

- 저장 공간
- GIT에 기록된 파일들 존재

GIT 프로젝트 관리 활용

◆ GIT 영역

❖ git init 명령어

- 세가지 영역과 Git을 구성하기 위한 파일과 메타데이터 생성
- 프로젝트 폴더에 GIT 관리를 위한 .git 폴더 생성
- Working Directroy, Staging Area, Repository 모두 비어있는 상태

명령어

`git init`

Enter

GIT 프로젝트 관리 활용

◆ GIT 영역

❖ git add 명령어

- Working Directory의 파일을 Staging Area로 옮기기 위한 명령어

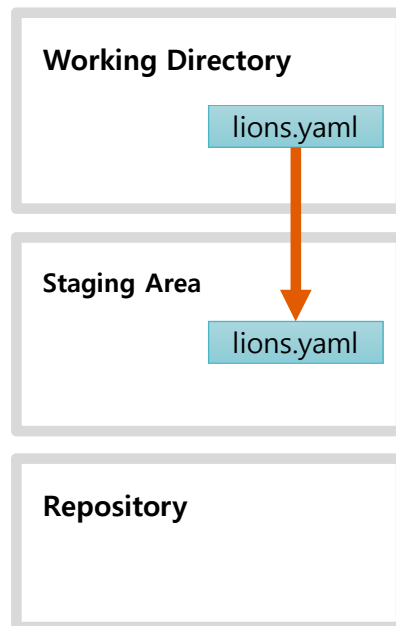
명령어	<code>git add [file1] [file2]</code>	<input type="text" value="Enter"/>
-----	---	------------------------------------

명령어	<code>git add .</code>	<input type="text" value="Enter"/>	# 폴더 안 모든 파일
-----	------------------------	------------------------------------	--------------

GIT 프로젝트 관리 활용

◆ GIT 영역

❖ git add 명령어



```
anece@LAPTOP-KGH195H9 MINGW64 ~/git_practice (main)
$ git add lions.yaml

anece@LAPTOP-KGH195H9 MINGW64 ~/git_practice (main)
$ git status
on branch main

No commits yet

changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   lions.yaml

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    test.txt
```

GIT 프로젝트 관리 활용

◆ GIT 영역

❖ git commit 명령어

- Staging Area의 파일을 Repository로 옮기는 명령어
- commit → 관련된 작업의 묶음

명령어

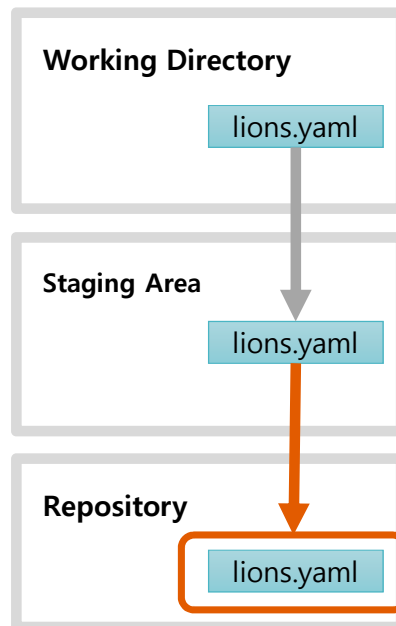
`git commit -m "커밋메시지"`

Enter

GIT 프로젝트 관리 활용

◆ GIT 영역

❖ git commit 명령어



```
anece@LAPTOP-KGH195H9 MINGW64 ~/git_practice (main)
$ git commit -m "TEST"
[main (root-commit) 66af98d] TEST
3 files changed, 17 insertions(+)
create mode 100644 lions.yaml
create mode 100644 test.txt
create mode 100644 tigers.yaml
```

← commit 단위

GIT 프로젝트 관리 활용

◆ GIT 영역

❖ git commit 명령어

- commit hash 값 : 커밋을 구분하기 위한 구분자
- git log 명령어로 확인 가능

명령어

git log

Enter

GIT 프로젝트 관리 활용

◆ GIT 영역

❖ git commit 명령어

```
anece@LAPTOP-KGH195H9 MINGW64 ~/git_practice (main)
$ git log
commit 66af98d18948b7fa03fa0d13f83c7878e45d5add (HEAD -> main)
Author: anece <anece00@naver.com>
Date:   Fri Aug 16 11:55:28 2024 +0900

    TEST
```

GIT 프로젝트 관리 활용

◆ GIT 영역

❖ git diff 명령어

- 현재와 이전 변경 사항 상세히 출력
- commit 전에 확인
- 출력 결과 보기 : K (위로) / J(아래로)

명령어

git diff

Enter

GIT 프로젝트 관리 활용

◆ [실습]

❖ git diff 명령어

```
anece@LAPTOP-KGH195H9 MINGW64 ~/git_practice (main)
$ git status
On branch main

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    lions.yaml
        modified:   tigers.yaml

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        leopards.yaml
        secrets.yaml

no changes added to commit (use "git add" and/or "git commit -a")
```

GIT 프로젝트 관리 활용

◆ GIT 영역

❖ git commit + add → git commit -am 명령어

- commit과 add를 한꺼번에 실행

명령어

git commit -am

Enter