



# git

## Git과 GitHub

버전 관리 시스템



# GitHub

김기정 (bangry313@gmail.com)

# 버전 관리 시스템 (Version Control System) 이란?

## ✓ 가장 간단한(?) 파일 버전 관리

- 수작업 직접 관리
  - 예) 자소서 V1.0 -> 자소서 V1.1 -> 자소서 V1.2 -> ... -> 자소서\_최종본 -> 이력서\_진짜 최종본

## ✓ 프로젝트 소스 코드 버전을 편리하게 관리할 수 있도록 도와주는 도구(Version Control System)이다.

- 프로젝트 소스 파일들을 누가? 언제? 어떻게? 변경하였는지 기록하고
- 기록한 내용을 쉽게 조회하고
- 특정 버전으로 되돌려 복구하고
- 여러 개발자들이 함께 개발하고, 공유할 수 있도록 도와주는 편리한 시스템이다.

## ✓ 종류

- 중앙 집중형 버전 관리 시스템
- 분산 버전 관리 시스템

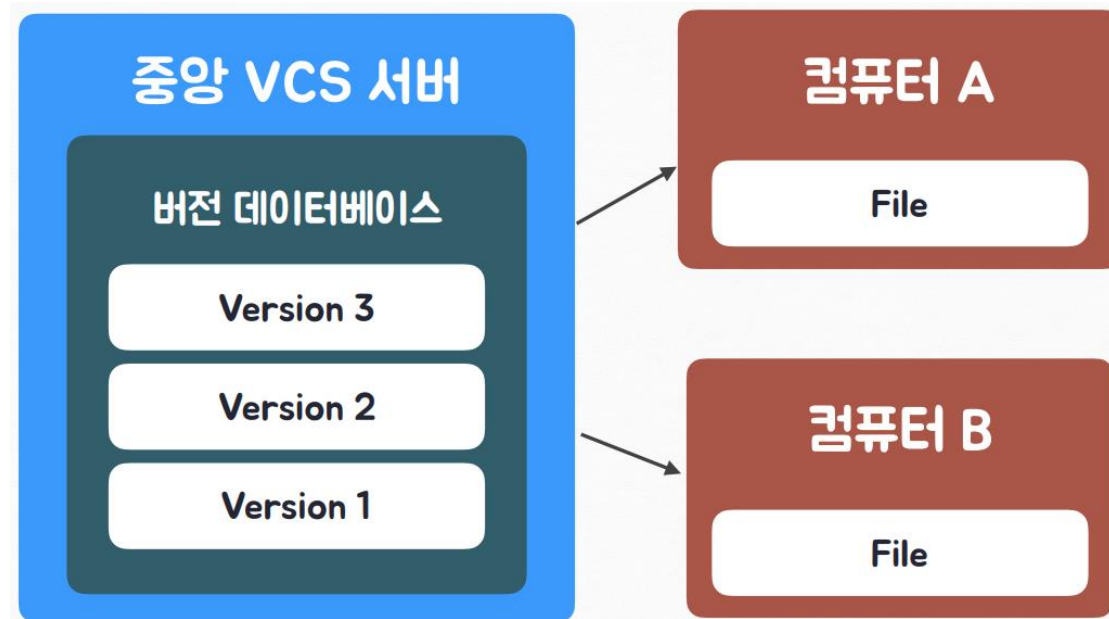
# 버전 관리 시스템 종류 (1/2)

## ✓ 중앙 집중형 버전 관리 시스템

- CVS, Subversion(SVN), Perforce

## ✓ 단점

- 오프라인 또는 서버 장애 시 사용 불가능



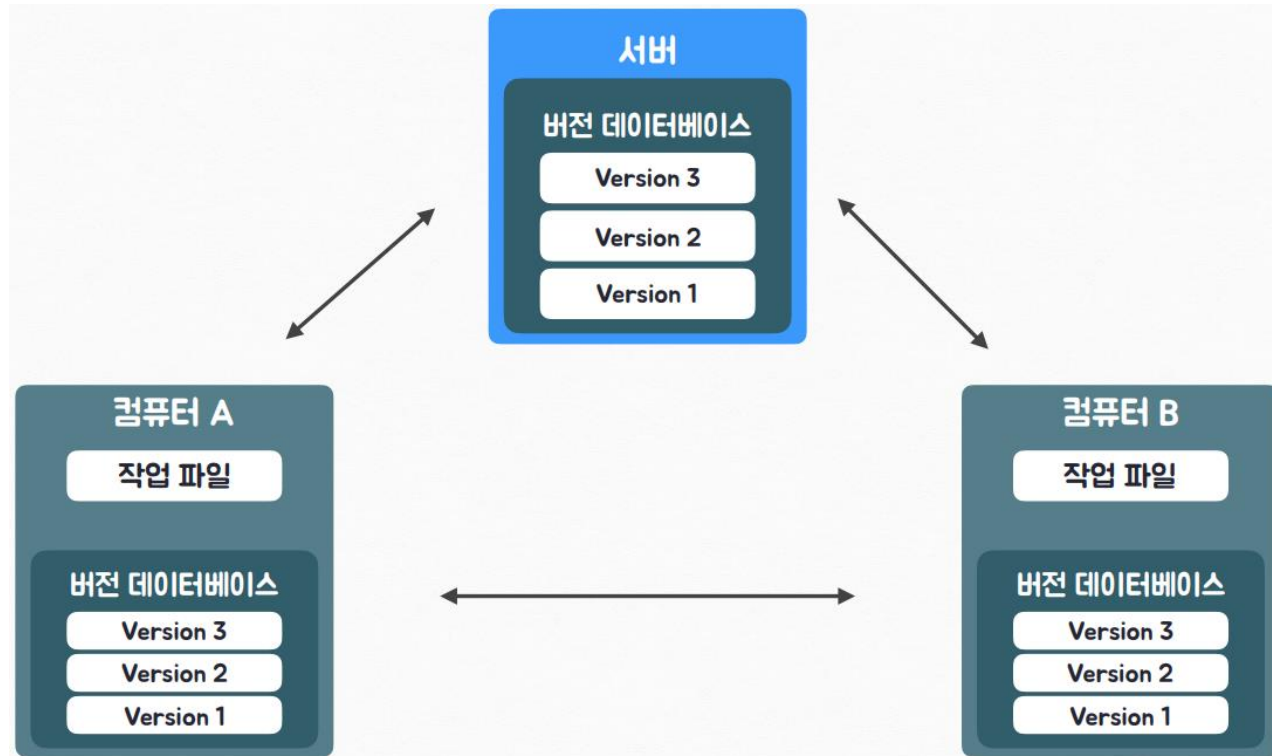
## 버전 관리 시스템 종류 (2/2)

### ✓ 분산 버전 관리 시스템

- **Git**, Mercurial, Darcs

### ✓ 장점

- 모든 개발자들이 서버와 동일한 히스토리 정보를 가지고 있기 때문에 서버 장애나 오프라인 시 사용 가능하다.
- 로컬에서 개별 버전 관리가 가능하며, 로컬 히스토리 정보를 이용하여 서버 복원도 가능하다.
- 서버는 회사별로 Private Server를 구축할 수 있으며, **GitHub**나 Gitlab, Bitbucket과 같이 서버를 이용할 수도 있다.





# Git 소개

- ✓ Git은 대표적인 분산 버전 관리 시스템이다.
- ✓ 컴퓨터 파일의 변경사항을 지속적으로 추적 관리하며, 여러 명의 사용자들이 파일을 공유하며, 협업 할 수 있도록 도와준다.
- ✓ 프로그램 개발에서 소스 코드 관리에 주로 사용되지만 그래픽 파일 등 대부분의 파일들을 추적 관리할 수 있다.

- ✓ 특징
  - 버전 관리 시스템의 사실상 표준으로 가장 많이 사용한다.
  - 오픈 소스이며 가볍고 빠르다.
  - 작업하고 있는 파일들을 과거 버전으로 다시 되돌릴 수 있다.
  - 저장 및 히스토리를 관리할 수 있으며, 협업이 가능하다.
  - 모든 작업을 오프라인 로컬에서 개인별로 관리할 수 있으며, 원격 저장소(예: GitHub)을 사용하여 많은 개발자들이 협업할 수 있다.
  - Branch 기능을 이용하여 여러 개발자들이 동시에 다양한 기능의 작업을 독립적으로 진행할 수 있으며, 작업한 내용을 하나의 branch로 합칠 수 있다.

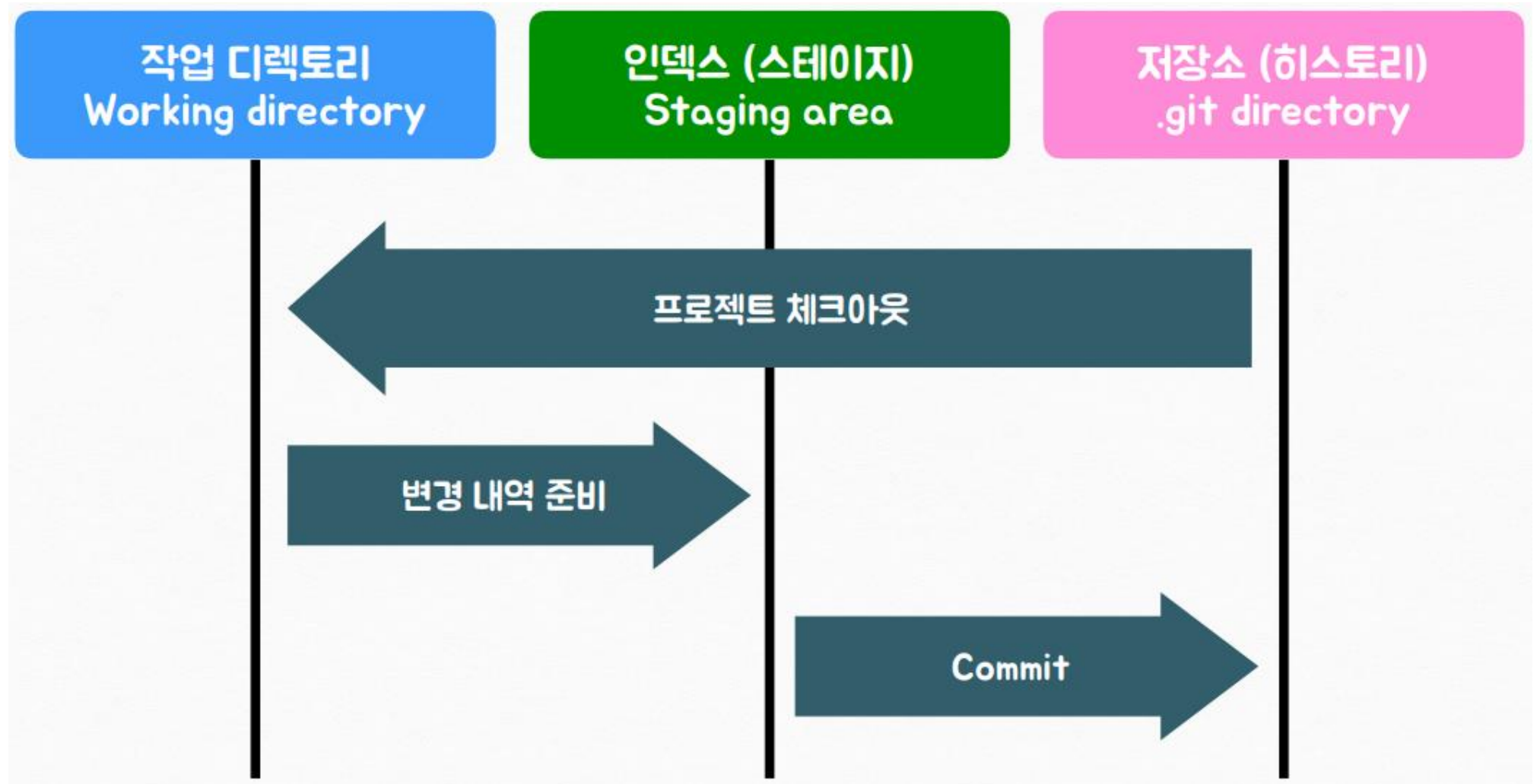


# Git 역사

---

- ✓ 초창기 **Linux Kernel Team**은 새로운 패치 버전마다 수동적으로 파일 버전을 관리
- ✓ BitKeeper 버전 관리 시스템 도입
  - BitKeeper 유료화
- ✓ BitKeeper 보다 더 빠르고 변경사항 적용 시 일관되고 안정적으로 적용할 수 있는 **Git 개발(2005년)**
  - 변경 내용만 히스토리로 관리하는 개념이 아닌 프로젝트 전체 내용을 스냅샷으로 관리
  - 버전 사이를 자유자재로 이동 가능하고, 브랜치들 사이에서 이동이 굉장히 빠르게 오류 없이 적용 가능

# Git 구조 (Working Directory, Staging Area, Git Repository)





# Git 설치 및 설정

✓ Git 공식사이트 : <https://git-scm.com/>

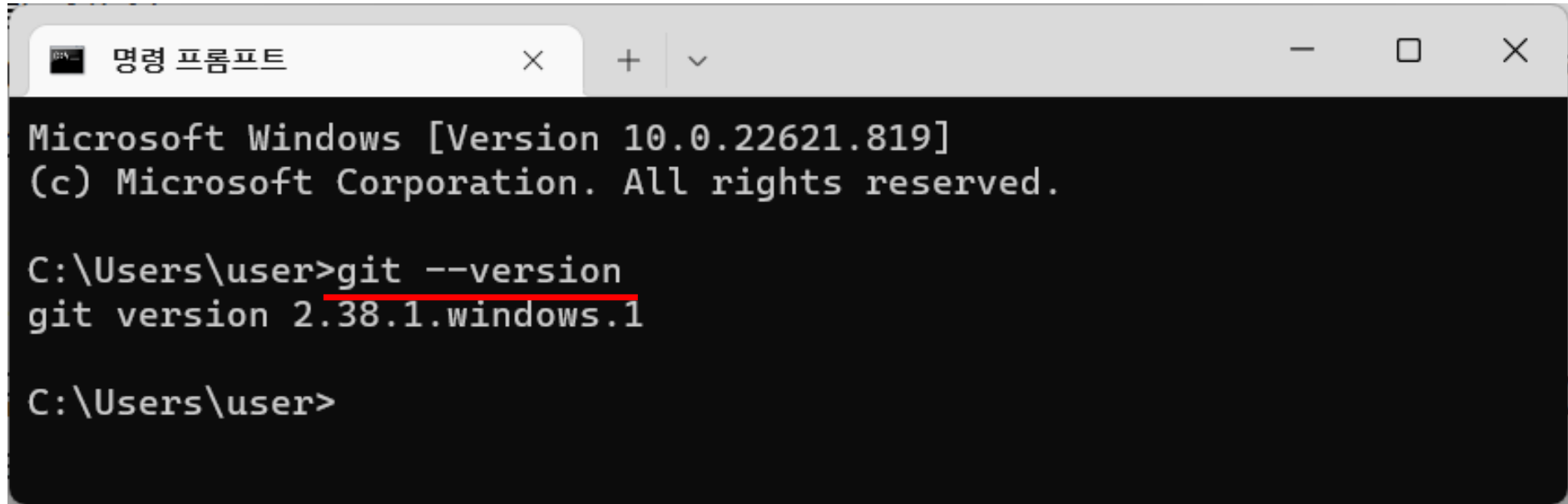
✓ Git은 **Terminal**을 이용한 명령어 기반 프로그램이다.

- **Terminal** : 로컬이나 원격 컴퓨터에 접속하여 **운영체제에 상관 없이 일관된 명령어를 이용**하여 상호작용 할 수 있도록 해주는 대표적인 CLI(Command Line Interface) 프로그램이다.
- Terminal에서 명령어로 실습하면 Git을 정확하게 사용하는 방법과 모든 기능을 사용할 수 있다.
  - Git을 처음 배울 때는 터미널을 이용하여 Git 명령어로 하나씩 공부해 나가는 걸 추천
- Git 설치 시 Gitbash(터미널 프로그램)가 기본 설치된다.

✓ 추후 GUI Client와 병행해서 사용하면 효과적이다.

- GUI Clients : <https://git-scm.com/downloads/guis>
  - Github Desktop (기능 미약 : 비 추천)
  - Source Tree (다양한 기능 지원 : 추천)
  - GitKraken (화려한 UI 좋아하는 개발자 추천)

## Git 실습 – 로컬 컴퓨터에 Git 설치 여부 및 버전 확인



```
명령 프롬프트
Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>git --version
git version 2.38.1.windows.1

C:\Users\user>
```

# Git 실습 - 명령어

## ✓ Git 명령어 형식

**git 명령어 - 옵션**

(config)

(add)

(commit)

## ✓ 명령어 목록

- <https://git-scm.com/docs>
- git 명령어만 사용할 수 있지만 Sourcetree를 이용하면 시각적(직관적)으로 확인 가능하다.

# Git 실습 – Git 환경 설정 (config)

- ✓ Git 사용을 위한 사용자 정보 (이름, 이메일) 전역 설정

```
MINGW64:/c/Users/user

user@DESKTOP-13TU8MC MINGW64 ~
$ git config --global user.name "bangry313"

user@DESKTOP-13TU8MC MINGW64 ~
$ git config --global user.email "bangry313@gmail.com"

user@DESKTOP-13TU8MC MINGW64 ~
$
```

- ✓ 설정 확인

```
MINGW64:/c/Users/user

user@DESKTOP-13TU8MC MINGW64 ~
$ git config user.name
bangry313
```

- ✓ 모든 설정 정보 확인

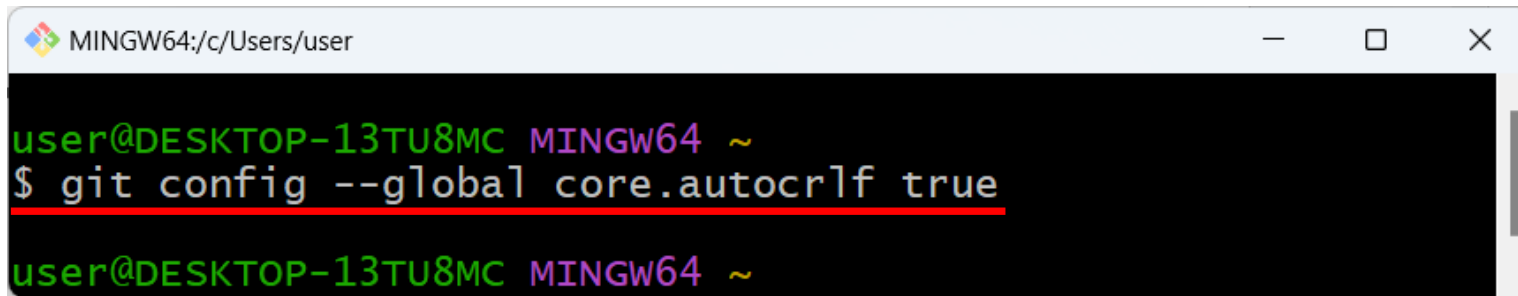
```
MINGW64:/c/Users/user

user@DESKTOP-13TU8MC MINGW64 ~
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
```

# Git 실습 – Git 환경 설정 (config)

## ✓ Git 줄 바꿈 자동 설정

- 윈도우 줄 바꿈 : \r\n
- 맥, 리눅스 줄 바꿈 : \n



```
MINGW64:/c/Users/user  
user@DESKTOP-13TU8MC MINGW64 ~  
$ git config --global core.autocrlf true  
user@DESKTOP-13TU8MC MINGW64 ~
```

- 참고) Mac : input

# Git 실습 – Git 환경 설정 (config)

---

- ✓ 로컬 Repository 기본 브랜치명 변경 (mater -> main)
  - `git config --global init.defaultBranch main`

# Git 실습 – 프로젝트 폴더 생성 (mkdir)

✓ 워크스페이스로 이동한 후 프로젝트 폴더 생성

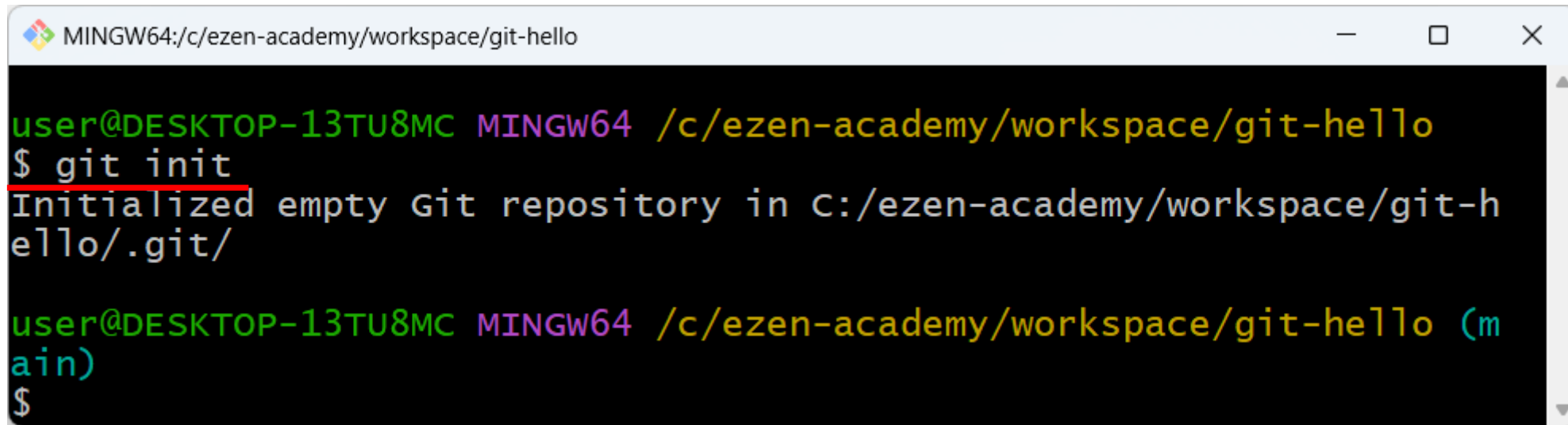


```
MINGW64:/c/ezen-academy/workspace/git-hello
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace
$ mkdir git-hello
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace
$ cd git-hello/
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello
$
```

- 생성된 프로젝트 폴더로 이동

# Git 실습 – Git 저장소 생성 및 초기화 (init)

- ✓ Git 히스토리 관리를 위한 Git 저장소(폴더) 생성 및 초기화



```
MINGW64:/c/ezen-academy/workspace/git-hello
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello
$ git init
Initialized empty Git repository in C:/ezen-academy/workspace/git-hello/.git/
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (main)
$
```

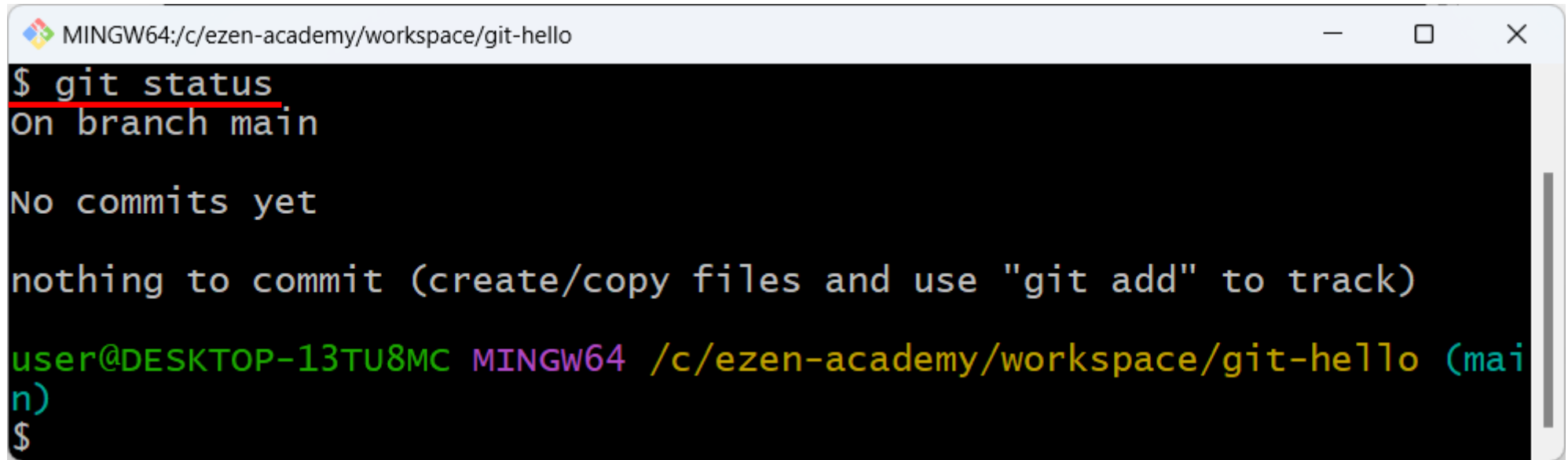
- ✓ `ls -al` 터미널 명령어로 `.git` 디렉터리(저장소) 생성 확인
  - 윈도우에서 숨김 파일로 설정됨
  - 소스 파일들을 git이 관리하게 함
- ✓ `rm -rf .git` 터미널 명령어로 git 저장소를 삭제할 수 있다.



# Git 실습 – Git 저장소 상태 확인 (status)

## ✓ Git 저장소 상태 확인

- 현재 main 디폴트 브랜치의 파일 상태 확인 - 아직까지 Git은 파일들을 관리하지 않음(untracking 상태)

A screenshot of a Windows terminal window titled 'MINGW64:/c/ezen-academy/workspace/git-hello'. The terminal shows the command '\$ git status' being entered and executed. The output is: 'on branch main', 'No commits yet', and 'nothing to commit (create/copy files and use "git add" to track)'. The prompt at the bottom shows the user is on the 'main' branch.

```
MINGW64:/c/ezen-academy/workspace/git-hello
$ git status
on branch main

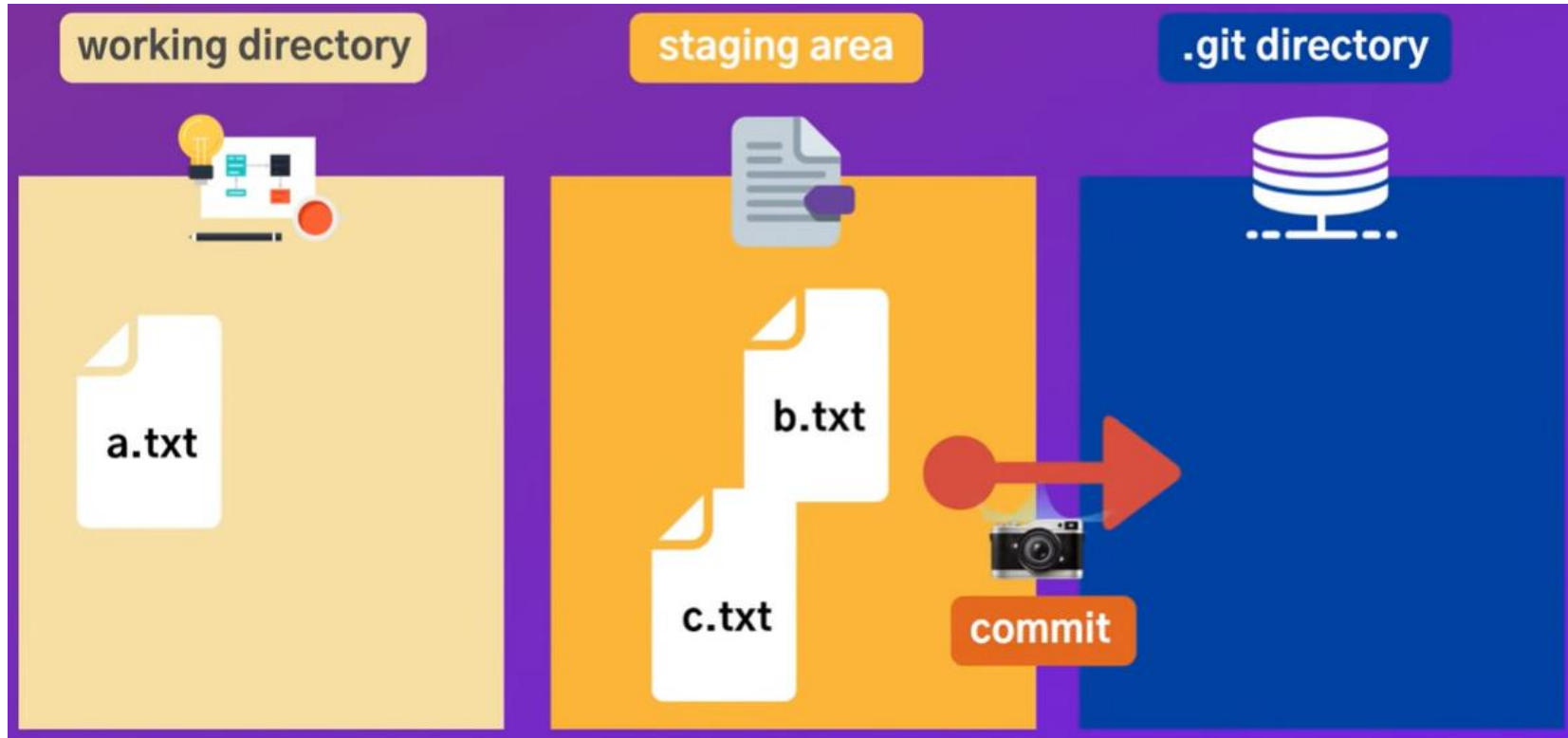
No commits yet

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

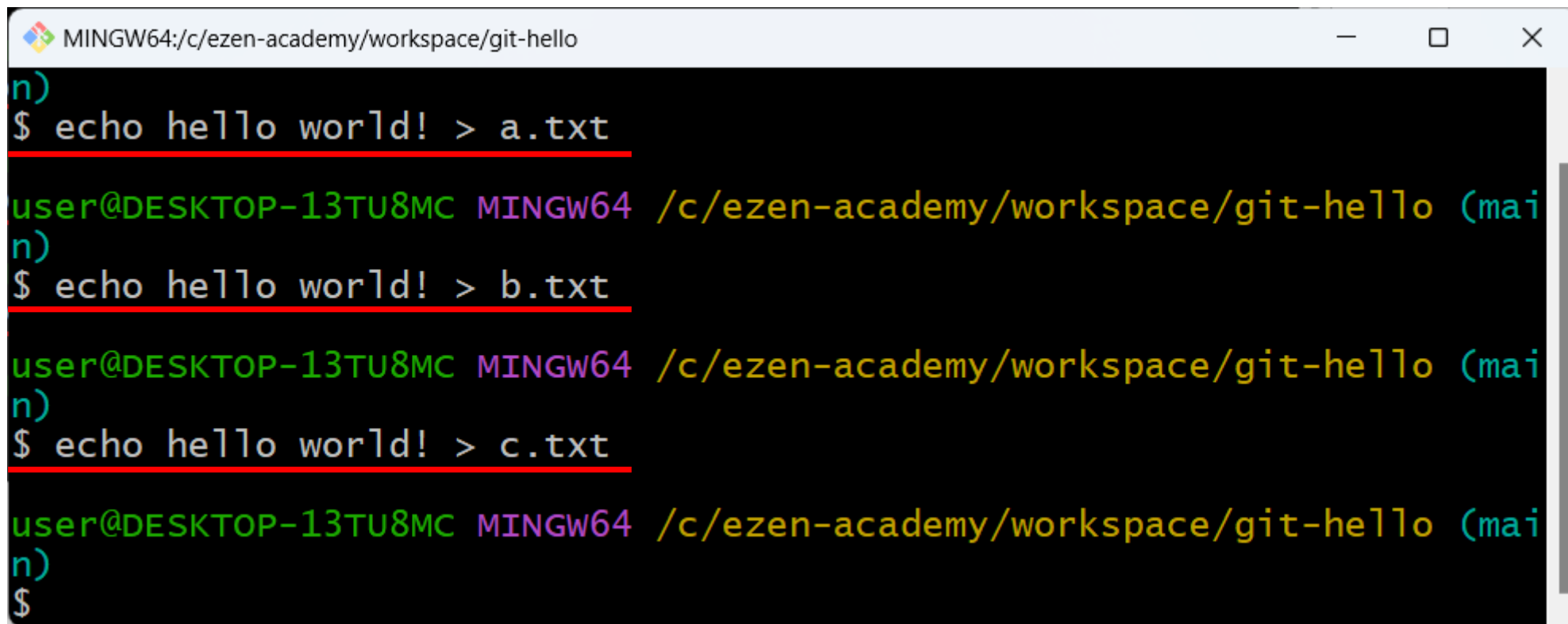
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (mai
n)
$
```

# Git 실습 – Git Workflow

- ✓ Working directory : 실제 작업 폴더 (프로젝트 폴더)
- ✓ Staging area : 버전 히스토리에 저장할 준비가 된 파일들을 옮겨 놓는 영역
- ✓ .git repository(directory) : 버전 히스토리 저장소



# Git 실습 – 파일 생성



A terminal window titled "MINGW64:/c/ezen-academy/workspace/git-hello" with standard window controls. The terminal has a black background with white and colored text. It shows three commands being entered and executed, each on a new line and underlined with a red line. The prompt is "\$". The first command is "echo hello world! > a.txt". The second is "echo hello world! > b.txt". The third is "echo hello world! > c.txt". The prompt "\$" is shown again at the bottom.

```
MINGW64:/c/ezen-academy/workspace/git-hello
n)
$ echo hello world! > a.txt
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (mai
n)
$ echo hello world! > b.txt
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (mai
n)
$ echo hello world! > c.txt
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (mai
n)
$
```

# Git 실습 – 파일 상태 확인

```
MINGW64:/c/ezen-academy/workspace/git-hello
$ git status
On branch main

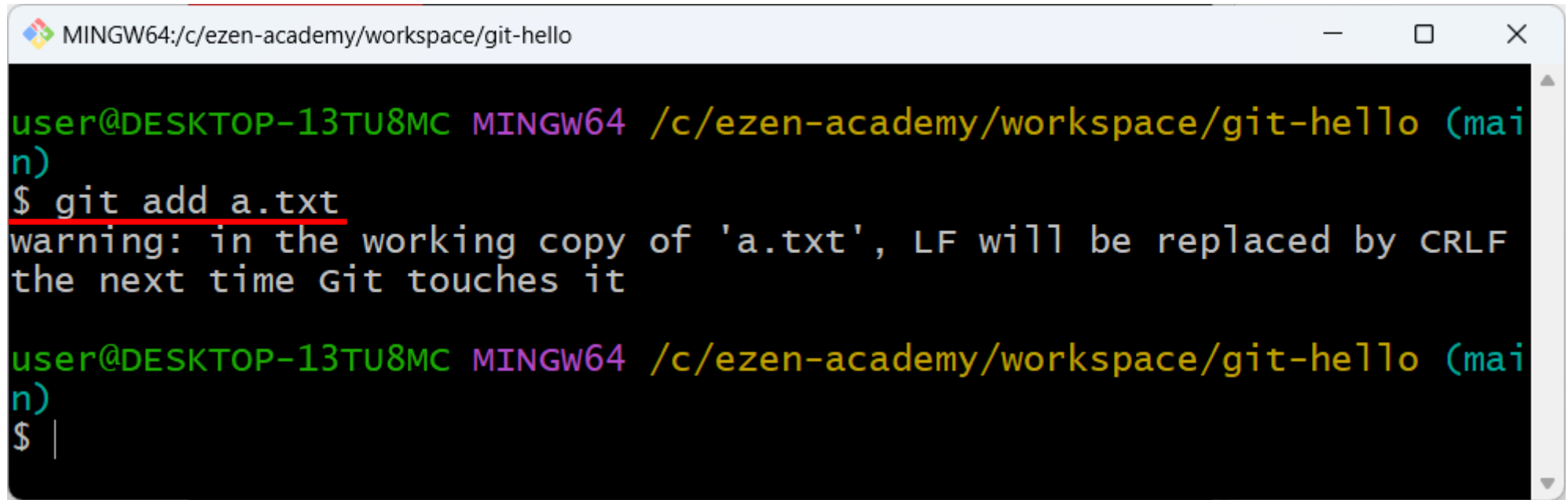
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        a.txt
        b.txt
        c.txt

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

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (mai
```

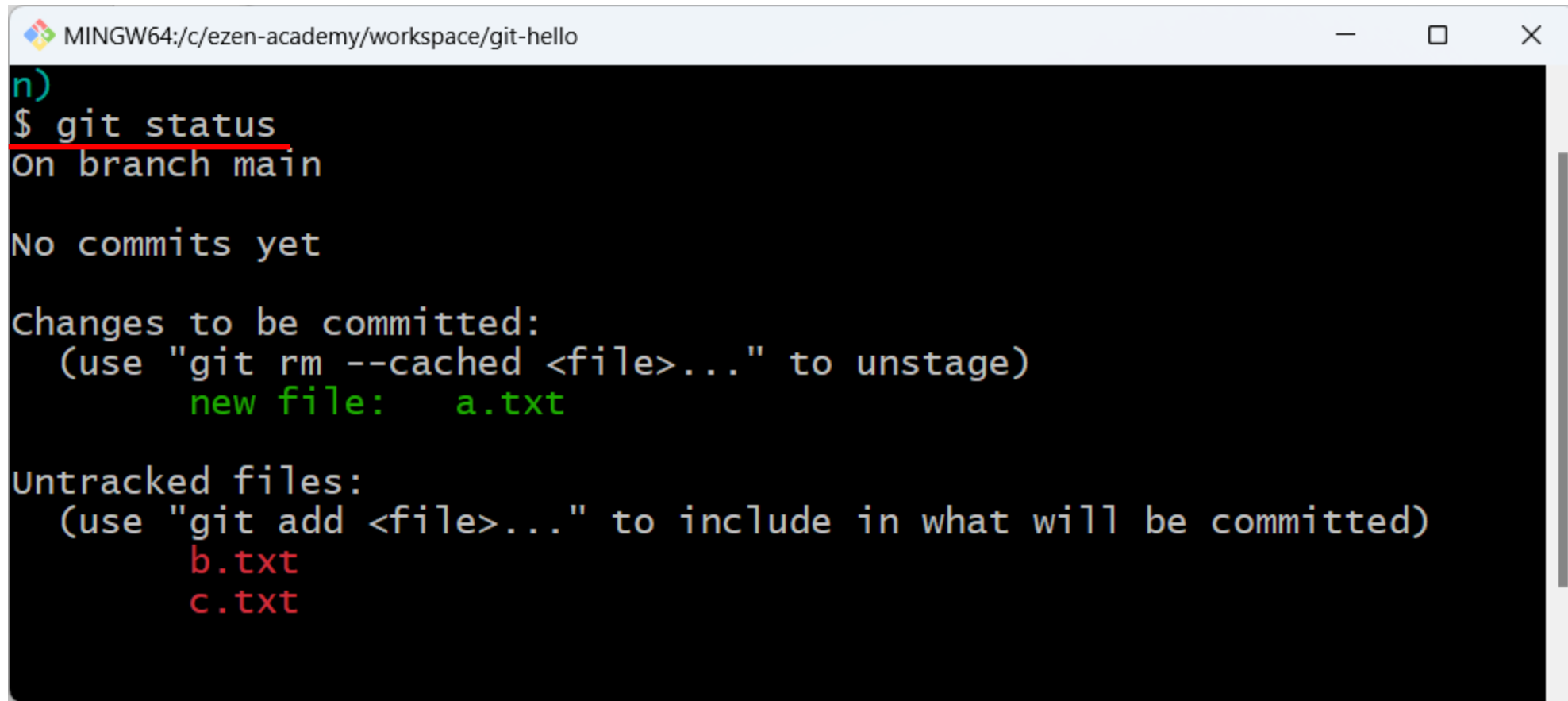
## Git 실습 – 파일들을 tracking 하기 위해 staging area에 추가 (add)

A screenshot of a Windows terminal window titled 'MINGW64:/c/ezen-academy/workspace/git-hello'. The terminal shows a user at the 'main' branch of a repository. They run the command 'git add a.txt'. A warning message appears: 'warning: in the working copy of 'a.txt', LF will be replaced by CRLF the next time Git touches it'. The prompt '\$' is shown again, indicating the command has completed.

```
MINGW64:/c/ezen-academy/workspace/git-hello (mai
n)
$ git add a.txt
warning: in the working copy of 'a.txt', LF will be replaced by CRLF
the next time Git touches it
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (mai
n)
$ |
```

# Git 실습 – 파일 상태 확인

## ✓ 파일 상태 확인

A screenshot of a Windows terminal window with a black background and white text. The title bar at the top shows the path 'MINGW64:/c:/ezen-academy/workspace/git-hello'. The terminal content shows the command '\$ git status' being entered, with the word 'git' underlined in red. The output indicates the current branch is 'main' and that there are no commits yet. It lists 'Changes to be committed' with a new file 'a.txt' in green. It also lists 'Untracked files' 'b.txt' and 'c.txt' in red.

```
MINGW64:/c:/ezen-academy/workspace/git-hello
n)
$ git status
On branch main

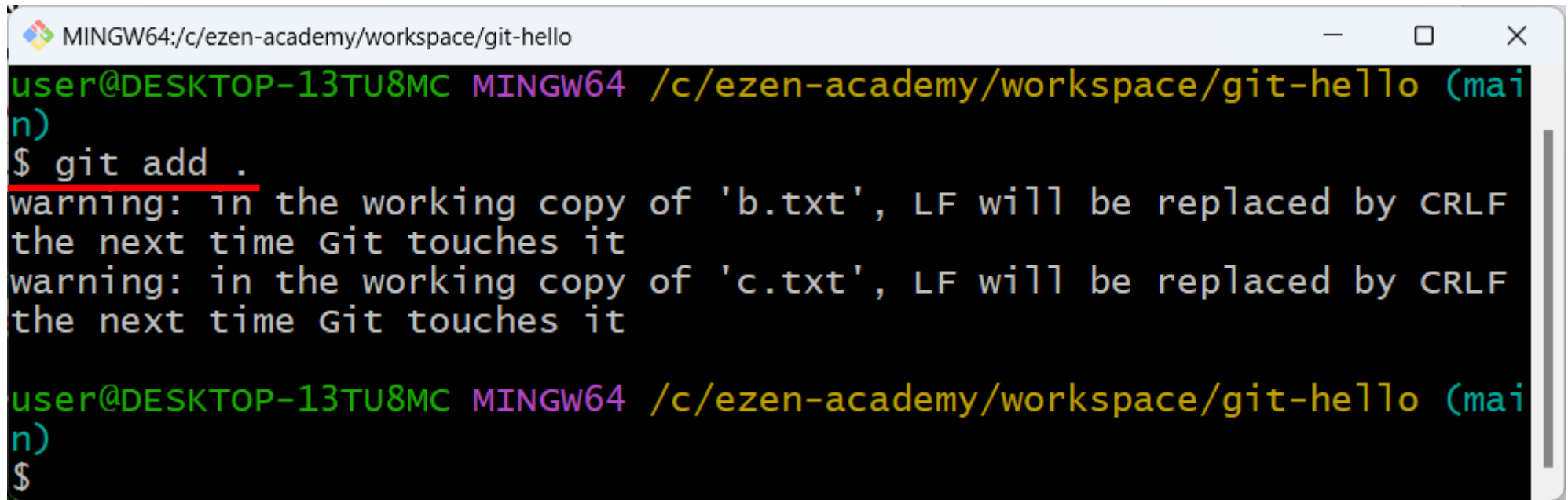
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   a.txt

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

# Git 실습 – 모든 파일들 staging area에 추가

- ✓ `git add b.txt c.txt`
- ✓ `git add *.txt`
- ✓ `git add *` (working directory의 모든 파일 staging area에 추가)
- ✓ `git add .` (working directory의 삭제된 모든 파일 포함 staging area에 추가)



```
MINGW64:/c/ezen-academy/workspace/git-hello
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (mai
n)
$ git add .
warning: in the working copy of 'b.txt', LF will be replaced by CRLF
the next time Git touches it
warning: in the working copy of 'c.txt', LF will be replaced by CRLF
the next time Git touches it

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (mai
n)
$
```

- ✓ 파일 상태 확인
  - `git status`

# Git 실습 – 테스트를 위한 a.txt 파일 변경 (새로운 내용 추가)

```
MINGW64:/c/ezen-academy/workspace/git-hello
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (main)
$ echo 안녕하세요 . 방그리 ! >> a.txt

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   a.txt
        new file:   b.txt
        new file:   c.txt

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:   a.txt
```



## Git 실습 – 변경된 a.txt 파일도 staging area에 추가

```
MINGW64:/c/ezen-academy/workspace/git-hello
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (mai
n)
$ git add a.txt
warning: in the working copy of 'a.txt', LF will be replaced by CRLF
the next time Git touches it

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (mai
n)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   a.txt
    new file:   b.txt
    new file:   c.txt
```

# Git 실습 – staging area에서 파일 제거

```
MINGW64:/c/ezen-academy/workspace/git-hello
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (main)
$ git rm --cached *
rm 'a.txt'
rm 'b.txt'
rm 'c.txt'

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        a.txt
        b.txt
        c.txt

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

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (main)
$ git add .
warning: in the working copy of 'a.txt', LF will be replaced by CRLF
the next time Git touches it
warning: in the working copy of 'b.txt', LF will be replaced by CRLF
the next time Git touches it
```

# Git 실습 – Git에서 관리(tracking) 하고 싶지 않은 파일 설정 (1/2)

## ✓ Git 관리 제외 파일

- 예) 로그파일, 보안파일, 빌드 파일 등
- **.gitignore** 파일 작성 및 제외하고자 하는 목록 설정

## ✓ 테스트를 위한 로그파일 작성

- `echo *.log > .gitignore`
- `echo test log~~~ > sample.log`
- `git status`
  - sample.log 파일은 목록에 보이지 않음

```
MINGW64:/c/ezen-academy/workspace/git-hello

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (main)
$ echo *.log > .gitignore

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (main)
$ echo test log~~~ > sample.log

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   a.txt
        new file:   b.txt
        new file:   c.txt

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

## Git 실습 – Git에서 관리(tracking) 하고 싶지 않은 파일 설정 (2/2)

### ✓ .gitignore 파일 작성 예

- # 파일 제외
- sample.log
- # 현재 경로에 있는 파일 제외
- /파일명.txt
- # 특정 경로안에 있는 파일 제외
- 디렉토리/파일명.txt
- # 특정 디렉토리안의 모든 파일 제외
- 디렉토리/
- # 해당 확장자 파일 전체 제외
- \*.txt
- # 예외
- !제외할 파일명.txt

### ✓ .gitignore 파일 작성 방법 (구글 검색 참조)

- <https://www.gitignore.io>
- 검색어에 java gradle 등 입력

# Git 실습 – Git Repository에 버전(히스토리) 저장 (commit)

## ✓ Git Repository에 히스토리 저장 - 스냅샷

- `Git commit -m "메시지(타이틀과 상세내용 기록)"`

```
MINGW64:/c/ezen-academy/workspace/git-hello
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (main)
$ git commit -m "김기정 프로젝트 초기화"
[main (root-commit) ba36c23] 김기정 프로젝트 초기화
4 files changed, 5 insertions(+)
create mode 100644 .gitignore
create mode 100644 a.txt
create mode 100644 b.txt
create mode 100644 c.txt
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (main)
```

# Git 실습 – commit 후 상태 확인 및 로그 정보 확인

```
MINGW64:/c/ezen-academy/workspace/git-hello

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (main)
$ git status
On branch main
nothing to commit, working tree clean

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (main)
$ git log
commit ba36c23d37956e975e32837df36bfb2a71339527 (HEAD -> main)
Author: bangry313 <bangry313@gmail.com>
Date: Mon Dec 19 16:53:36 2022 +0900

    김 기 정   프 로젝트   초 기   화

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (main)
```

# Git 실습 – 테스트를 위해 b.txt 파일에 내용 추가하고 commit

```
MINGW64/c/ezen-academy/workspace/git-hello
$ echo 방 그 리 파 일 내 용 변 경 >> b.txt

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (mai
n)
$ 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 director
y)
        modified:   b.txt

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

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (mai
n)
$ git add .
warning: in the working copy of 'b.txt', LF will be replaced by CRLF
the next time Git touches it

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (mai
n)
$ git commit -m "김 기 정 b.txt 파 일 변 경 "
[main 7629de4] 김 기 정 b.txt 파 일 변 경
1 file changed, 1 insertion(+)

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace/git-hello (mai
n)
$ git log
commit 7629de43d84fffd60bceb5eb5718a34b880aaa28 (HEAD -> main)
Author: bangry313 <bangry313@gmail.com>
Date: Mon Dec 19 16:59:44 2022 +0900

    김 기 정 b.txt 파 일 변 경

commit ba36c23d37956e975e32837df36bfb2a71339527
Author: bangry313 <bangry313@gmail.com>
Date: Mon Dec 19 16:53:36 2022 +0900
```

## Git 실습 – 적절한 commit 단위

---

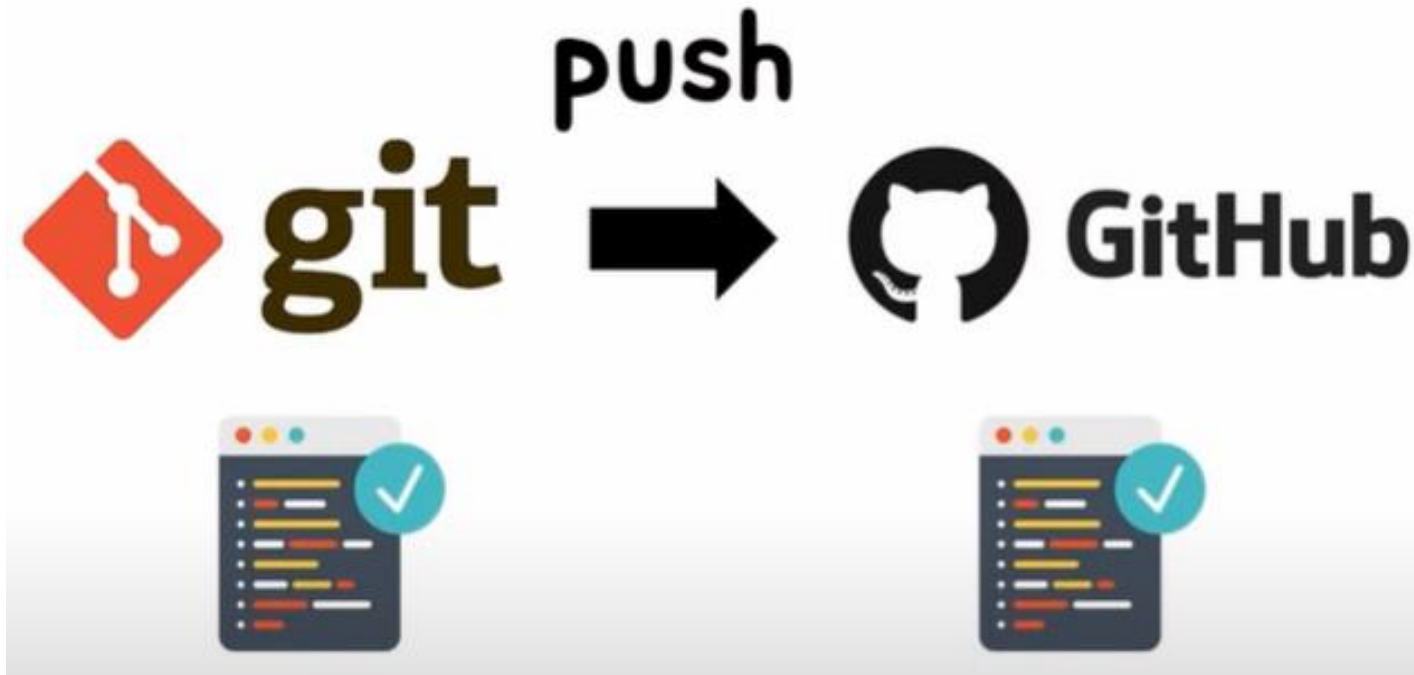
- ✓ 전체 애플리케이션 단위의 커밋은 의미가 없으므로 **기능별로 세분화**하여 의미 있게 커밋하여야 한다.
- ✓ 의미 있는 커밋 메시지를 사용하여야 한다.
  - 커밋 메시지는 init, add, fix 등과 같이 동사형으로 작성한다.
  - 예) 프로젝트 초기화, 로그인 서비스 모듈 커밋, 웰컴페이지 커밋, about 페이지 커밋 등
  - 커밋을 너무 방대하게 이것 저것 작성하지 말고 해당 기능만 간결한 메시지로 작성한다.



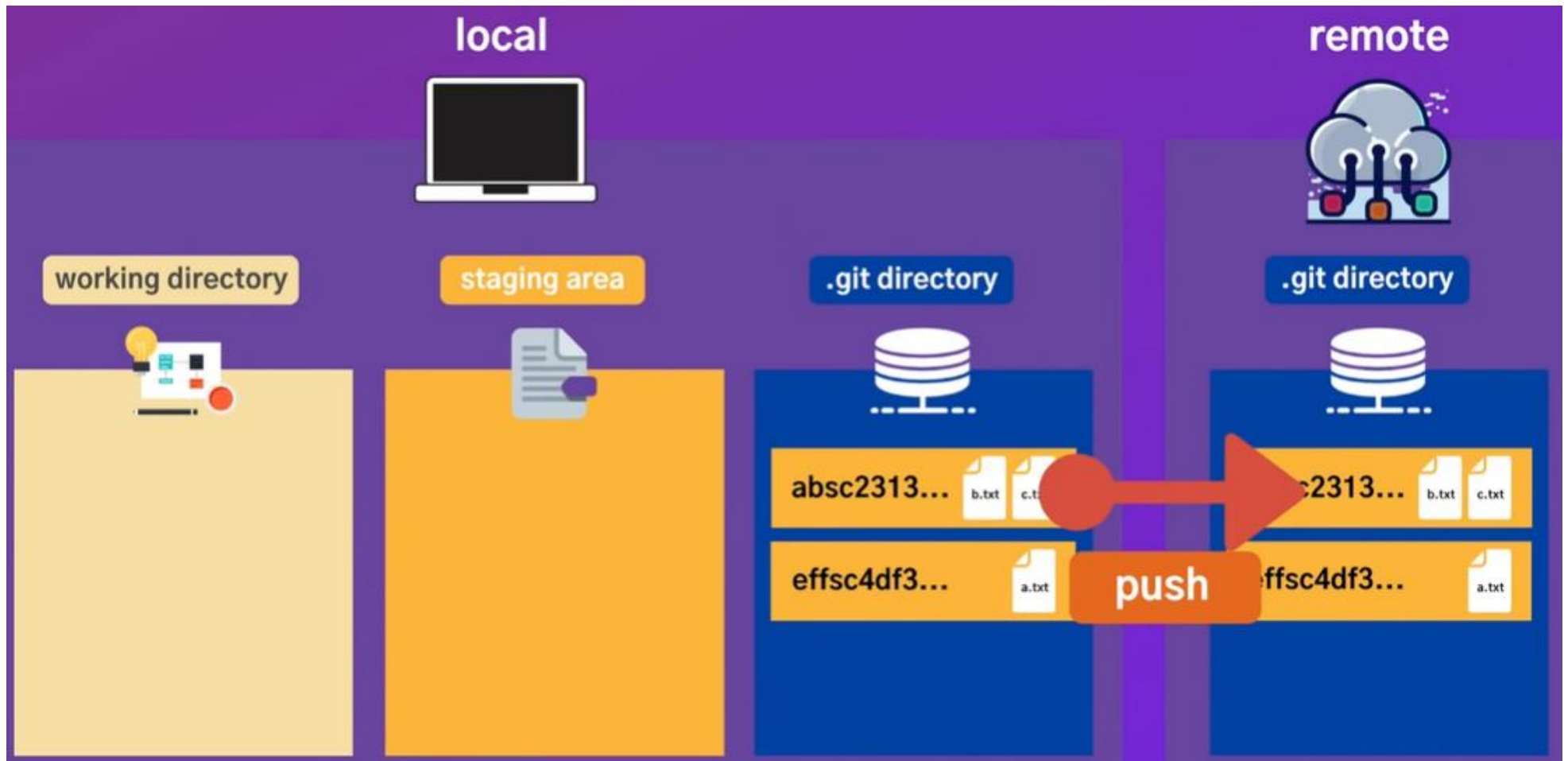


# GitHub 소개

- ✓ Git으로 관리된 프로젝트를 클라우드 방식으로 관리하고 공유할 수 있도록 지원하는 **웹 호스팅 서비스**이다.
  - 원격 Git 저장소(Repository) 제공
  - 프로젝트 작업물을 안전 하게 저장할 수 있으며, 많은 개발자들 간의 공유 및 협업이 가능하다.
  - Git으로 소스 코드 버전 관리하고, GitHub에 업로드(Push) 한다.

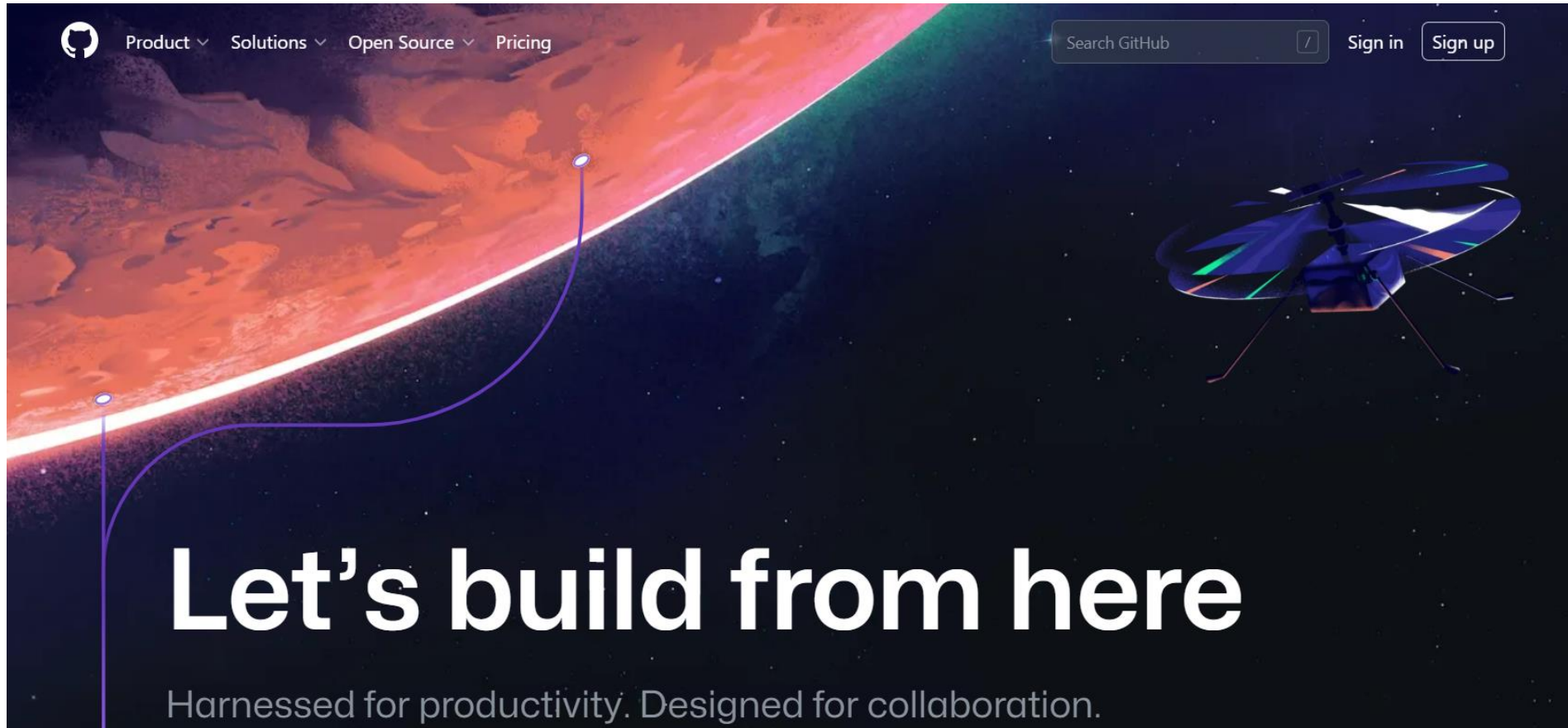


# GitHub Workflow



# GitHub 연동 – GitHub 회원 가입 및 로그인, 토큰(Token) 생성

✓ <https://github.com/>



# GitHub 연동 – 원격 Git Repository 생성

## ✓ New (New Repository)

- Repository name : first-project
- 공개여부 : Public

## ✓ Create repository

Quick setup — if you've done this kind of thing before

 Set up in Desktop

or

HTTPS

SSH



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# first-project" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/bangry313/first-project.git
git push -u origin main
```



...or push an existing repository from the command line

```
git remote add origin https://github.com/bangry313/first-project.git
git branch -M main
git push -u origin main
```



# GitHub 연동 – 실습을 위한 로컬 Repository 생성 및 커밋

```
i7A-56@i7C-66 MINGW64 /c/ezen-academy/workspace/first_project
$ git init
Initialized empty Git repository in C:/ezen-academy/workspace/first_project/.git/

i7A-56@i7C-66 MINGW64 /c/ezen-academy/workspace/first_project (main)
$ git add .

i7A-56@i7C-66 MINGW64 /c/ezen-academy/workspace/first_project (main)
$ git commit -m 'first commit'
[main (root-commit) 5bb1eb0] first commit
6 files changed, 48 insertions(+)
create mode 100644 .classpath
create mode 100644 .project
create mode 100644 .settings/org.eclipse.core.resources.prefs
create mode 100644 .settings/org.eclipse.jdt.core.prefs
create mode 100644 bin/first_project/Test1.class
create mode 100644 src/first_project/Test1.java
```

# GitHub 연동 – 원격 Repository 연결

## ✓ 원격 저장소 URL을 git에서 별칭으로 만들어 연결하기

- `git remote add [별칭(alias)] [원격 저장소 URL]`
- `git remote add origin https://token@github.com/원격 저장소`

```
i7A-56@i7C-66 MINGW64 /c/ezen-academy/workspace/first_project (main)
$ git remote add origin https://ghp_8S0JpfIjJSLD90cz9vfQ56Gsn05pkJ43XT8R@github.com/bangry313/first_project.git
```

## ✓ 추가한 원격 저장소 목록 가져오기

- `git remote` : 원격 저장소 별칭 목록
- `git remote -v` : 별칭과 URL 같이 표시

```
$ git remote -v
origin https://ghp_8S0JpfIjJSLD90cz9vfQ56Gsn05pkJ43XT8R@github.com/bangry313/first_project.git (fetch)
origin https://ghp_8S0JpfIjJSLD90cz9vfQ56Gsn05pkJ43XT8R@github.com/bangry313/first_project.git (push)
```

## ✓ 원격 저장소 삭제

- `git remote remove origin`

# GitHub 연동 – git push 사용해 원격 저장소에 커밋된 로컬 저장소 업로드 (1/2)

✓ `git push -u [원격 저장소 별칭] [로컬 저장소 브랜치명]`

```
i7A-56@i7C-66 MINGW64 /c/ezen-academy/workspace/first_project (main)
$ git push -u origin main
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (13/13), 1.64 KiB | 279.00 KiB/s, done.
Total 13 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/bangry313/first_project.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```



# GitHub 연동 – git push 사용해 원격 저장소에 커밋된 로컬 저장소 업로드 (2/2)

main ▾


1 branch

0 tags






Go to file

Add file ▾

<> Code ▾

 bangry313 first commit

5bb1eb0 12 minutes ago 1 commit

 .settings	first commit	12 minutes ago
 bin/first_project	first commit	12 minutes ago
 src/first_project	first commit	12 minutes ago
 .classpath	first commit	12 minutes ago
 .project	first commit	12 minutes ago

Help people interested in this repository understand your project by adding a README.

Add a README

## GitHub 연동 – 실습을 위한 소스 변경 및 히스토리(버전) 관리 (1/2)

```
$ 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:   bin/first_project/Test1.class
        modified:   src/first_project/Test1.java

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

i7A-56@i7C-66 MINGW64 /c/ezen-academy/workspace/first_project (main)
$ git add .

i7A-56@i7C-66 MINGW64 /c/ezen-academy/workspace/first_project (main)
$ git commit -m 'second commit'
[main 3c38158] second commit
 2 files changed, 4 insertions(+)
```

# GitHub 연동 – 테스트를 위한 소스 변경 및 히스토리(버전) 관리 (2/2)

```
i7A-56@i7C-66 MINGW64 /c/ezen-academy/workspace/first_project (main)
$ git push -u origin main
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (8/8), 1.01 KiB | 258.00 KiB/s, done.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/bangry313/first_project.git
   5bb1eb0..3c38158  main -> main
branch 'main' set up to track 'origin/main'.
```

main

1 branch

0 tags

Go to file

Add file

<> Code

bangry313 second commit

3c38158 3 minutes ago 2 commits

.settings	first commit	17 hours ago
bin/first_project	second commit	3 minutes ago
src/first_project	second commit	3 minutes ago
.classpath	first commit	17 hours ago
.project	first commit	17 hours ago

Help people interested in this repository understand your project by adding a README.

Add a README

# GitHub 연동 – 원격 저장소 파일 가져오기 – 3가지 방법

---

## ✓ zip 파일 가져오기

- .git 디렉토리가 없는 채로 파일만을 받을 수 있다.

## ✓ clone

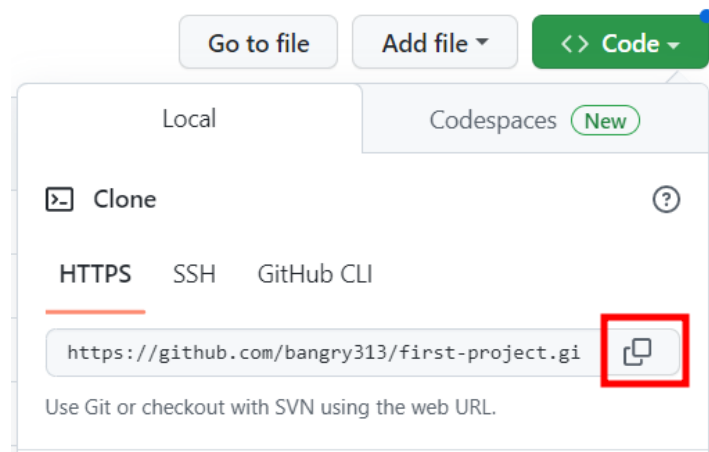
- .git 디렉토리까지 포함해서 모든 파일들을 받을 수 있다.

## ✓ pull

- 원격 저장소에 저장된 파일들을 가져와 로컬 저장소의 내용을 갱신한다.
- 병합(merge) 과정이 발생한다.

# GitHub 연동 – git clone 사용해 원격 저장소 복제

- ✓ `git clone {원격 저장소 URL} [저장소를 복제할 로컬 디렉토리]`
  - 디렉토리 생략하면 원격 저장소 이름과 동일한 이름으로 디렉토리 생성



```
i7A-56@i7C-66 MINGW64 /c/ezen-academy/workspace
$ git clone https://github.com/bangry313/first_project.git first_project_clone
Cloning into 'first_project_clone'...
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 21 (delta 3), reused 19 (delta 1), pack-reused 0
Receiving objects: 100% (21/21), done.
Resolving deltas: 100% (3/3), done.
```

# GitHub 연동 – git pull 사용해 원격 저장소 모든 파일들 가져오기

## ✓ git pull {원격 저장소 별칭} {원격 저장소 브랜치명}

- git clone은 원격 저장소의 모든 파일들을 가져오기만 하지만
- git pull은 로컬 저장소와 비교하여 병합하고, 로컬 저장소에 add, commit 명령까지 수행한다.

## ✓ git pull은 팀 협업 과정에서 프로젝트의 최신 코드들을 로컬 저장소로 가져오는 역할로 많이 사용한다.

- 현재 내가 작업중인 로컬 저장소와 팀 동료가 작성한(push) 최신 코드가 비교 및 병합되어, 최신 버전 파일들이 나의 로컬 저장소에 적용된다.

```
$ git pull origin main
```

```
9b56f14..37abed1  main      -> origin/main
Updating 9b56f14..37abed1
Fast-forward
 bin/first_project/Test1.class | Bin 663 -> 702 bytes
 src/first_project/Test1.java  | 2 ++
 2 files changed, 2 insertions(+)
```

# GitHub 연동 – git pull 사용해 원격 저장소 모든 파일들 가져오기

- ✓ git log 로 확인

```
i7A-56@i7C-66 MINGW64 /c/ezen-academy/workspace/first_project_clone (main)
$ git log
commit 37abed13611283934bd14311c807a3868c204c3d (HEAD -> main, origin/main, origin/HEAD)
Author: bangry313 <bangry313@gmail.com>
Date: Thu Feb 9 11:10:53 2023 +0900

    4th commit

commit 9b56f14a3c5a0586256f2de5111ca0f867bfd5c3
Author: bangry313 <bangry313@gmail.com>
Date: Thu Feb 9 10:55:51 2023 +0900

    third commit
```

- ✓ 팀 프로젝트가 진행되면 원격 저장소의 소스코드와 개발자들 로컬 저장소의 소스코드가 다를 수 있기 때문에, 특정 개발자의 소스코드가 변경되면 원격 원격 저장소에 Push 하기 전에 반드시 Pull 부터 진행하여야 한다.
  - PULL -> PUSH -> PULL -> PUSH ...

# GitHub 연동 – 팀 협업 시 branch 생성 및 push

```
user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace2/first-project (main)
$ git branch developer1

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace2/first-project (main)
$ git switch developer1
Switched to branch 'developer1'

user@DESKTOP-13TU8MC MINGW64 /c/ezen-academy/workspace2/first-project (developer1)
$ git push origin developer1
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'developer1' on GitHub by visiting:
remote:   https://github.com/bangry313/first-project/pull/new/developer1
remote:
To https://github.com/bangry313/first-project.git
 * [new branch]      developer1 -> developer1
```



# GitHub 연동 – 팀 협업 시 New pull request (끌어오기 요청)

🔑 developer1 had recent pushes less than a minute ago

Compare & pull request



개발자1 두번째 커밋

Write

Preview

H B I @

끌어오기 요청 메시지

Attach files by dragging & dropping, selecting or pasting them.




Create pull request





# GitHub 연동 – 팀 협업 시 Branch Merge


The screenshot shows the GitHub interface for a repository named 'first-project' by user 'bangry313'. The repository is marked as 'Public'. At the top right, there are buttons for 'Pin' and 'Unwatch' (with a count of 1). Below this is a navigation bar with tabs for 'Code', 'Issues', 'Pull requests' (highlighted with a red box and a count of 1), 'Actions', 'Projects', 'Wiki', and 'Settings'. Below the navigation bar, there are buttons for 'main' (with a dropdown arrow), '2 branches', '0 tags', 'Go to file', 'Add file' (with a dropdown arrow), and a green 'Code' button (with a dropdown arrow). Below these buttons is a warning box titled 'Your main branch isn't protected'. The warning text says: 'Protect this branch from force pushing or deletion, or require status checks before merging. [Learn more](#)'. To the right of the warning text is a button labeled 'Protect this branch' and a close icon (X).


# GitHub 연동 – 팀 협업 시 Branch Merge

 [bangry313 / first-project](#) Public


 Pin


 Unwatch 1


 Fork 0


 Star 0


<> Code


 Issues


 Pull requests 1


 Actions

 Projects

 Wiki

 Security


 Insights


 Settings


Label issues and pull requests for new contributors


Dismiss

Now, GitHub will help potential first-time contributors [discover issues](#) labeled with good first issue

Filters 



 is:pr is:open


 Labels 9


 Milestones 0


New pull request


☐


 1 Open  0 Closed


Author 


Label 

Projects 


Milestones 

Reviews 

Assignee 

Sort 


☐


 개발자1 두번째 커밋

#1 opened 3 minutes ago by bangry313

# GitHub 연동 – 팀 협업 시 Branch Merge

Add more commits by pushing to the **developer1** branch on **bangry313/first-project**.




**Require approval from specific reviewers before merging**


Add rule

×

[Branch protection rules](#) ensure specific people approve pull requests before they're merged.

**Continuous integration has not been set up**

[GitHub Actions](#) and [several other apps](#) can be used to automatically catch bugs and enforce style.

**This branch has no conflicts with the base branch**

Merging can be performed automatically.

Merge pull request

▼

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

# GitHub 연동 – 팀 협업 시 Branch Merge

src/ezen/git/HelloGitExample.java

↑	@@ -9,6 +9,8 @@ public static void main(String[] args) {	
9	9	
10	10	System.out.println("개발자1 코드 추가1");
11	11	System.out.println("개발자1 코드 추가2");
	12	+
	13	+
		System.out.println("개발자1 코드 추가3");
12	14	}
13	15	
14	16	}

# GitHub 연동 – 원격 저장소에 다른 사용자 초대

- ✓ [Settings 탭] > [Collaborators] > [Manage access] > [Add people]
- ✓ 초대된 사용자 메일에서 초대 수락

The screenshot shows the GitHub repository settings page. On the left is a sidebar with a 'General' section (containing 'Access' and 'Moderation options') and a 'Code and automation' section (containing 'Branches', 'Tags', 'Actions', 'Webhooks', 'Environments', 'Codespaces', and 'Pages'). Below these is a 'Security' section (containing 'Code security and analysis', 'Deploy keys', and 'Secrets'). The 'Collaborators' option under 'Access' is selected. The main content area is titled 'Who has access' and shows two panels: 'PUBLIC REPOSITORY' (stating the repository is public and visible to anyone, with a 'Manage' link) and 'DIRECT ACCESS' (stating 0 collaborators have access). Below this is a 'Manage access' section with a large box containing an icon of a person with a lock, the text 'You haven't invited any collaborators yet', and a green 'Add people' button.

General

Access

- Collaborators
- Moderation options

Code and automation

- Branches
- Tags
- Actions
- Webhooks
- Environments
- Codespaces
- Pages

Security

- Code security and analysis
- Deploy keys
- Secrets

### Who has access

**PUBLIC REPOSITORY**

This repository is public and visible to anyone.

[Manage](#)

**DIRECT ACCESS**

0 collaborators have access to this repository. Only you can contribute to this repository.

### Manage access

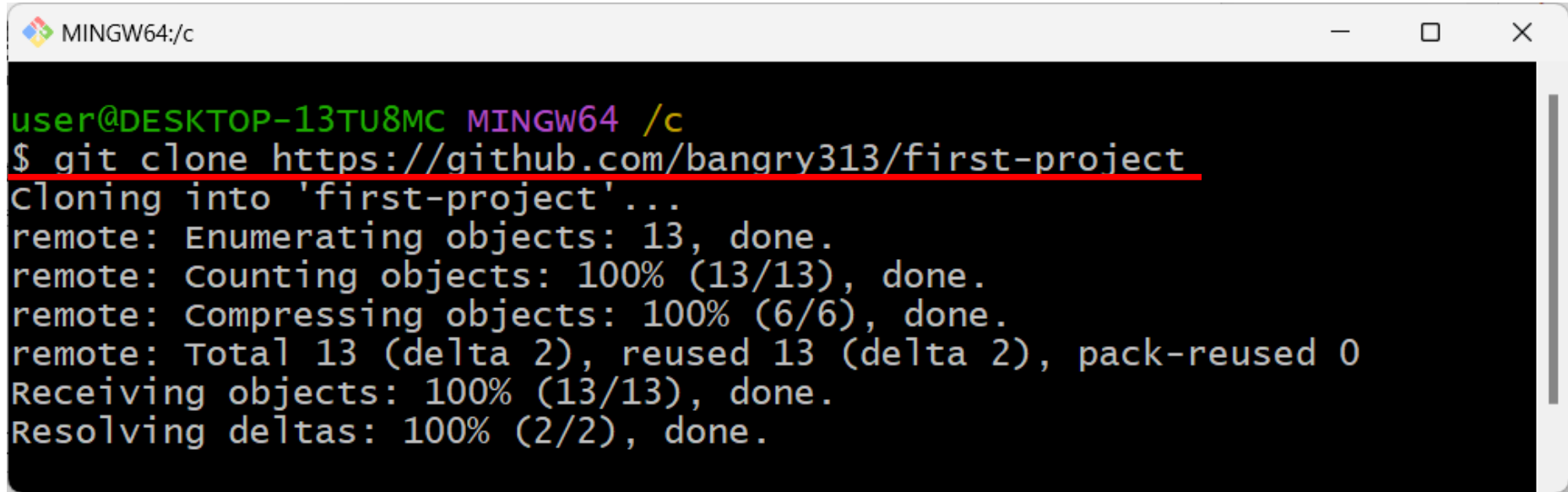
You haven't invited any collaborators yet

[Add people](#)

# GitHub 연동 – 초대 받은 사용자 원격 저장소 연결

## ✓ git clone

- git clone [remote repository 주소]
- 원격 설정을 자동으로 해주는 초기 다운로드에 사용한다.



```
MINGW64:/c
user@DESKTOP-13TU8MC MINGW64 /c
$ git clone https://github.com/bangry313/first-project
Cloning into 'first-project'...
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 13 (delta 2), reused 13 (delta 2), pack-reused 0
Receiving objects: 100% (13/13), done.
Resolving deltas: 100% (2/2), done.
```