

**Test Automation Basics**

---

# 테스트 자동화 기초

08 Git과 Jenkins를 활용한 CI/CD 파이프라인 구축

# 목차 08 Git과 Jenkins를 활용한 CI/CD 파이프라인 구축

---

1. Git 개념 및 활용
2. Git 브랜치 전략
3. Jenkins 개념 및 환경 설정
4. Jenkins를 활용한 테스트 자동화
5. CI/CD 파이프라인 구축



## 8주차 수업내용

- Git의 기본 개념과 기본 사용법을 이해합니다.
- Git을 통해 다른 사람들과 함께 테스트 스크립트를 작성하고, 공유합니다.
- Git branch 전략에 대해 알고, 효율적으로 협업하는 방법을 익힙니다.
- CI/CD의 기본 개념에 대해 이해합니다.
- Jenkins의 기본 개념과 기본 사용법을 알고, 기본적인 CI/CD 파이프라인을 구축해봅니다.
- Git과 Jenkins를 사용해 CI/CD 파이프라인을 구축하고 자동화 테스트 스크립트를 통합합니다.

# 1. Git 개념 및 활용

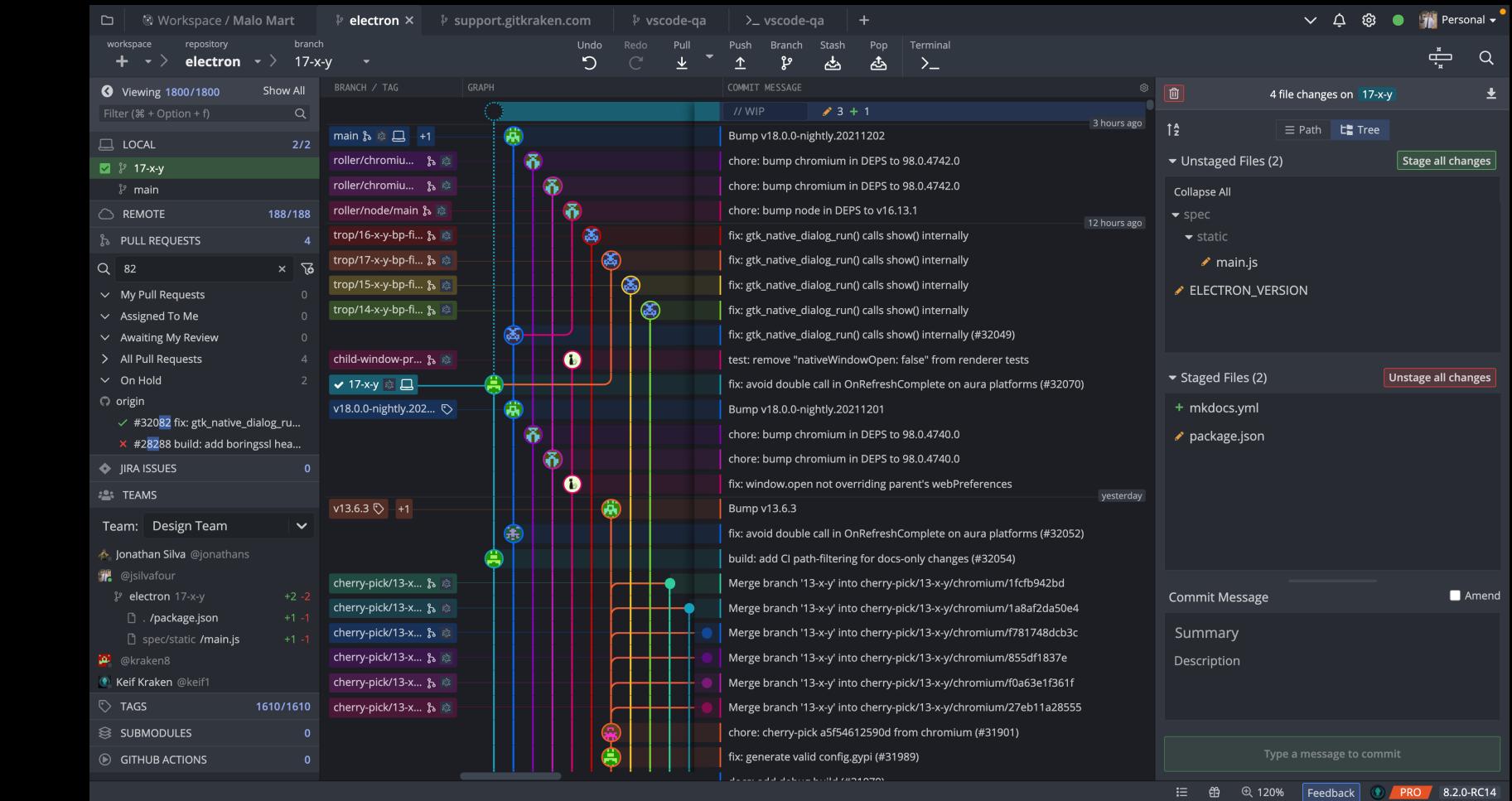
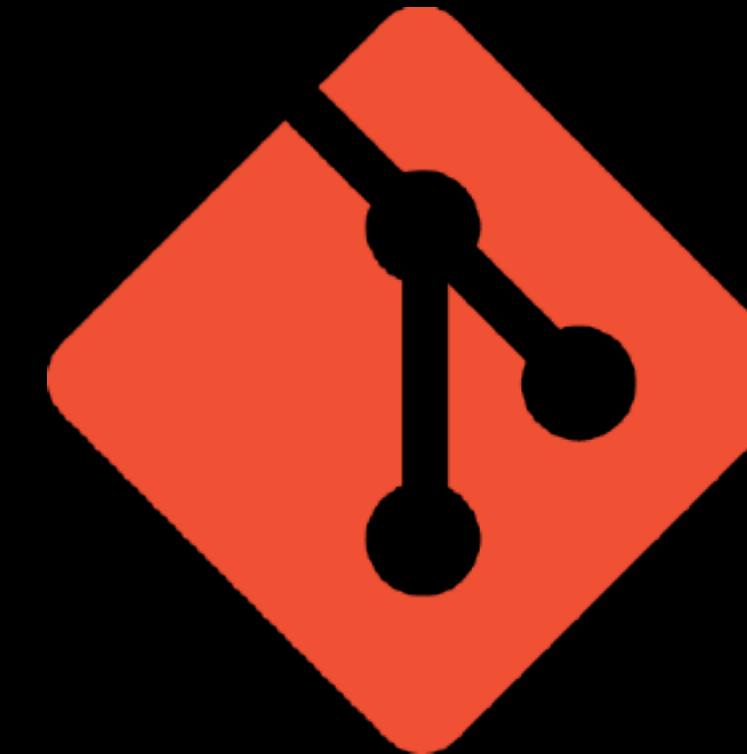


## 1일차 수업내용

- Git의 기본 개념과 기본 사용법을 이해합니다.
- 원격 저장소와 로컬 저장소의 차이점과 특징에 대해 이해합니다.
- Git을 설치하고 초기 설정을 완료할 수 있습니다.
- Git의 기본 명령어( add, commit, push, pull )를 익힙니다.
- Git을 통해 다른 사람들과 함께 테스트 스크립트를 작성하고, 공유합니다.
- 실습을 통해 Git의 기본 워크플로우를 반복 경험합니다.
- 협업 시 유용한 명령어에 대해 배웁니다.



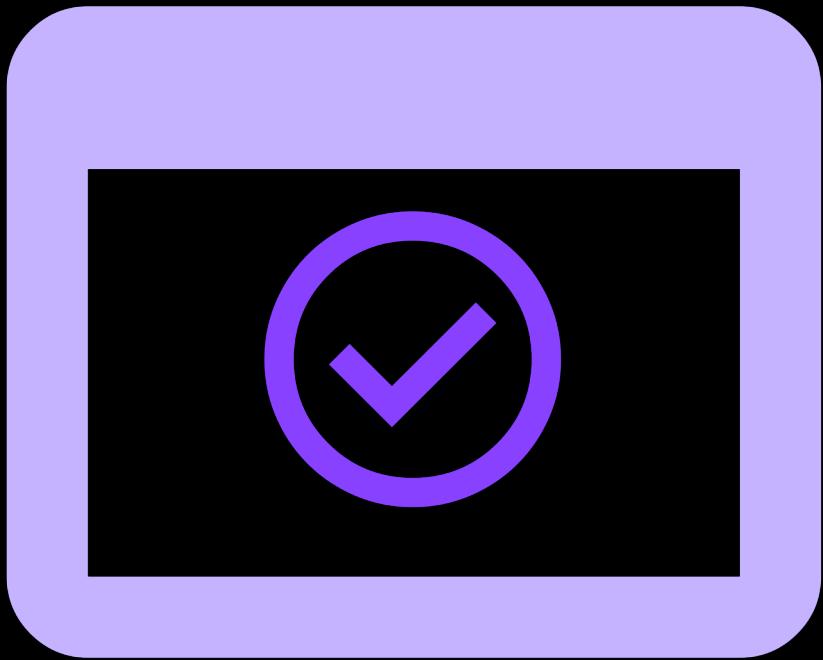
# What is Git?



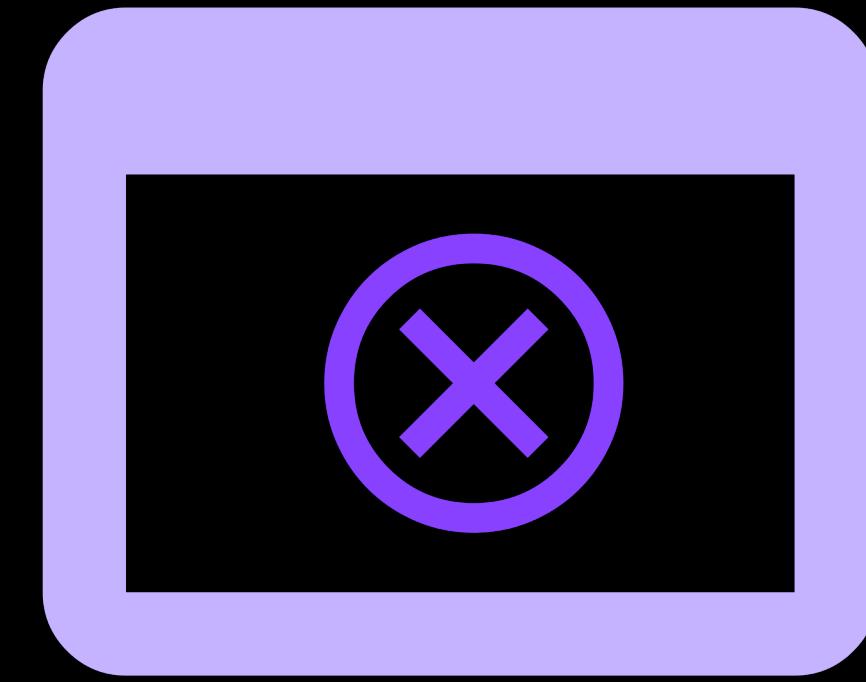
- 리눅스를 만든 '리누스 토르발스'가 개발한 시스템
- 분산 버전 관리 시스템
- 여러 명이 하나의 프로젝트를 개발할 때, 소스코드의 변경 내역을 추적하고 관리하여 버전을 관리할 수 있는 도구



## 1. 변경 이력 관리



Version 1.0 – 정상 실행



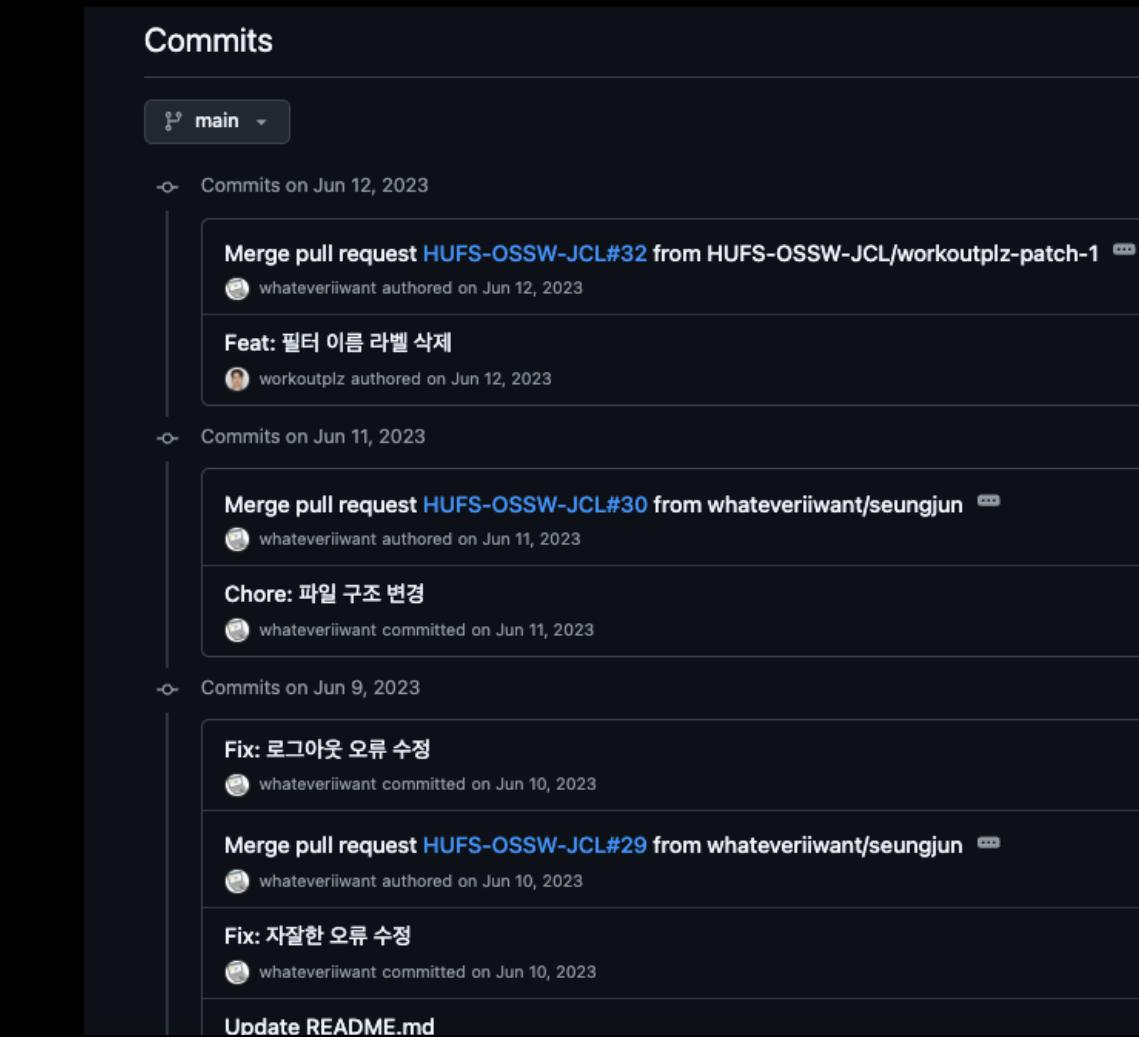
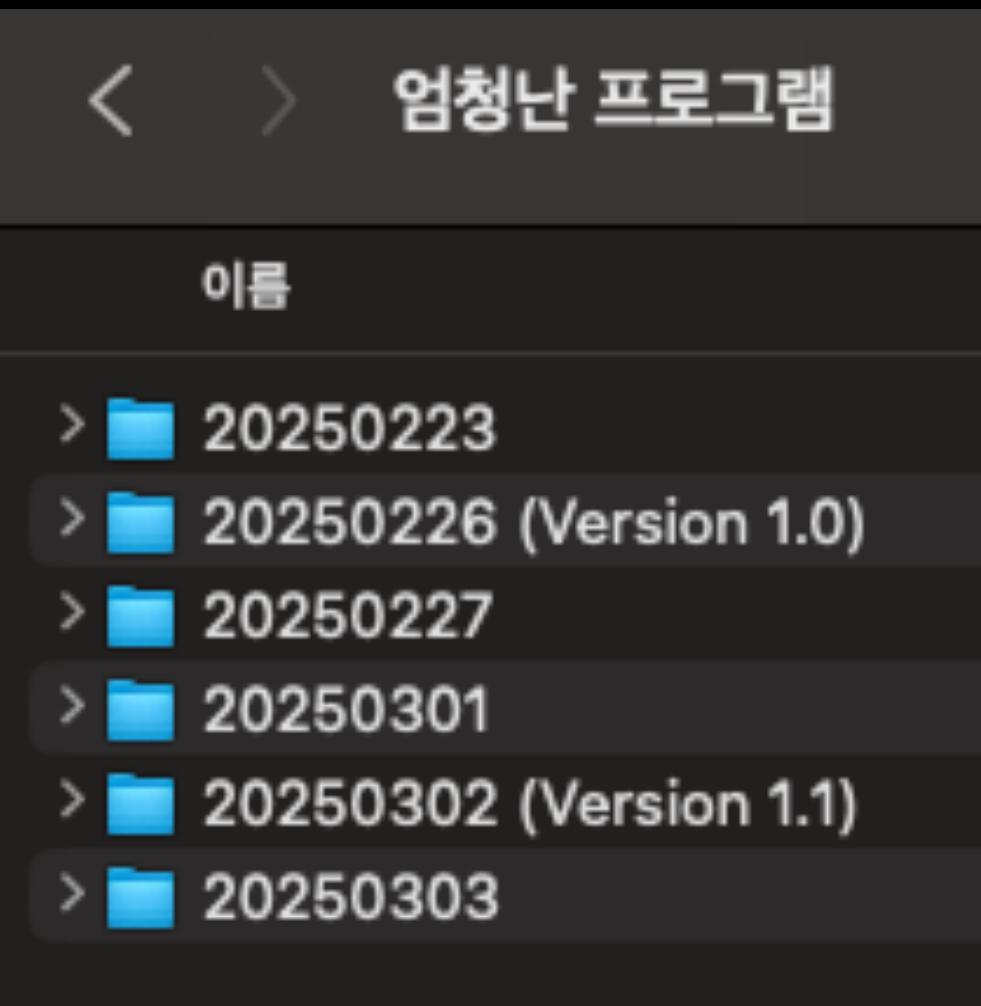
Version 1.1 – 오류

버전 업데이트 후 치명적 오류 발생 시,  
(1) 이전 버전으로 변경 필요 (2) 문제 추적 필요



# Why Git?

## 1. 변경 이력 관리



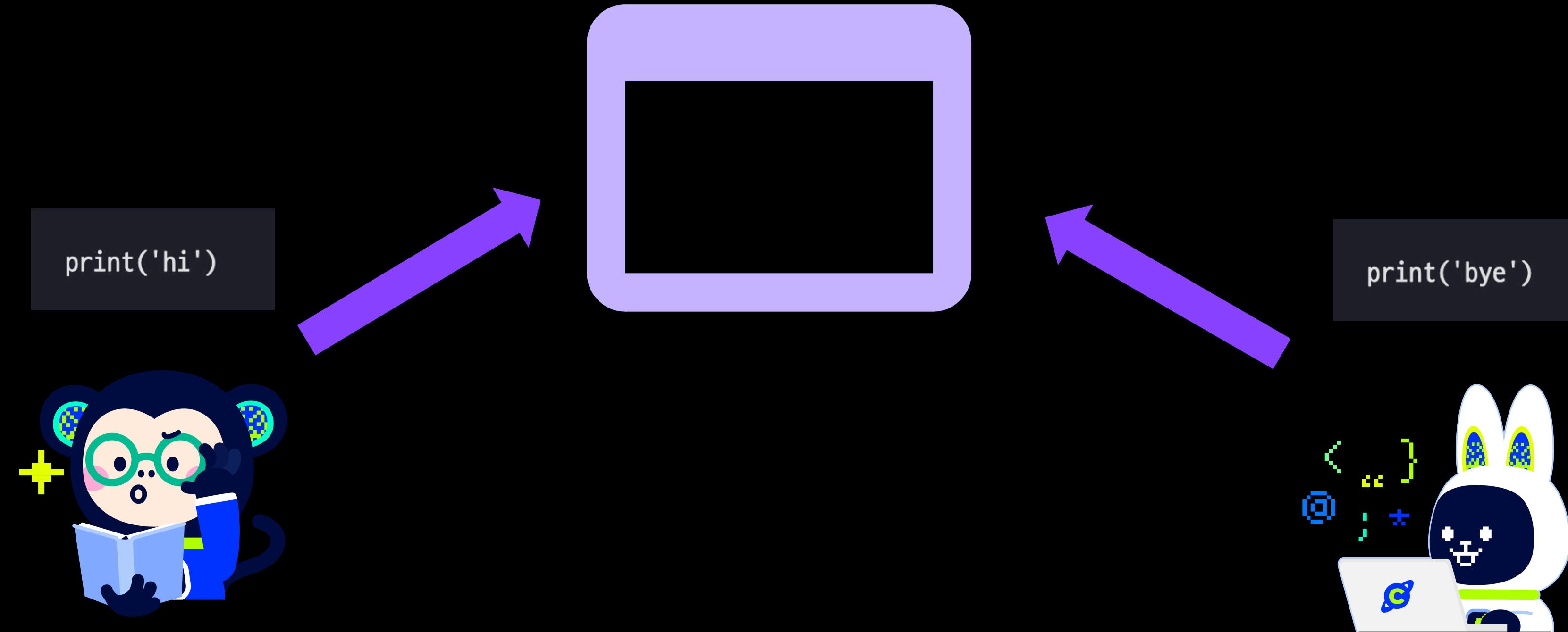
- 평소 작업하는 코드를 날짜 별로 폴더와 파일 생성 필요
- 이 자체만으로 굉장히 번거로운 일

- 일자 별로 변경 이력을 모두 확인 할 수 있음
- 이전 내역으로 손쉽게 복구 가능
- 문제가 발생한 버전과 직전 코드 비교 가능



# Why Git?

## 2. 협업에 용이



2025년 1월 1일 0시 0분 저장

2025년 1월 1일 0시 1분 저장



# Why Git?

## 2. 협업에 용이

```
print('hi')  
print('bye')
```

```
print('bye')
```

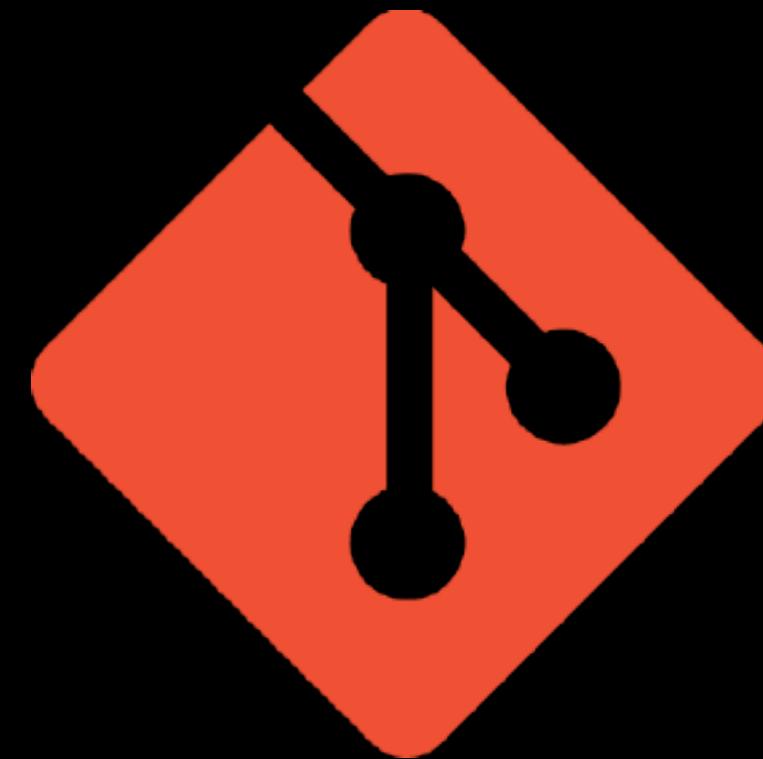
우리가 원했던 결과물  
(Git 사용 시)

실제 결과물  
(Git 미사용 시)

여러 명이 동시에 작업해도 작업 내역이 독립적으로 관리, 이후 코드 통합 가능



# What is GitHub?



!=



Git : 분산 관리 시스템 (오프라인 사용, 로컬에 직접 설치 필요)

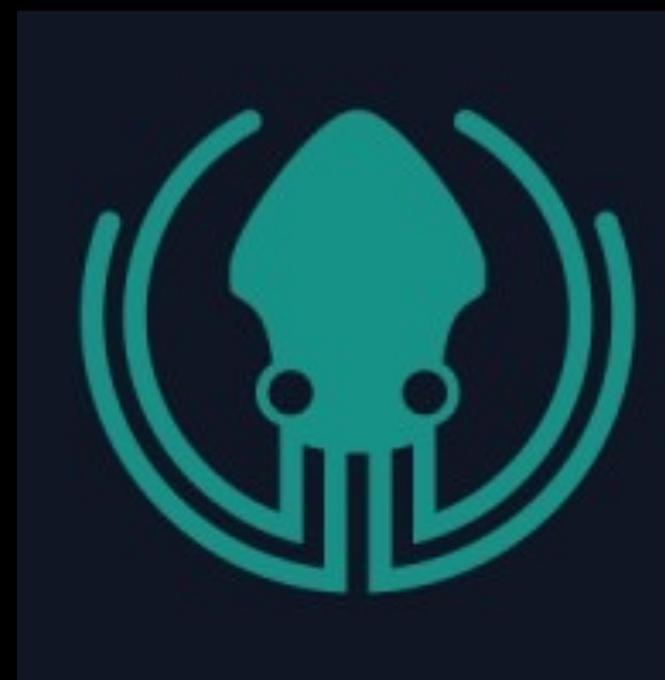
GitHub : Git을 기반으로 한 온라인 저장소, Git을 온라인으로 옮긴 것



# What is GitHub?



GitHub, GitLab, GitKraken, Bitbucket 등  
Git 기반의 다양한 서비스들이 존재.



그 중 개인이 가장 접하기 쉽고 많이 쓰는 것이  
**GitHub**



# What is GitHub?

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용

The screenshot shows the GitHub code editor interface. At the top, there's a navigation bar with links for Product, Solutions, Resources, Open Source, Enterprise, and Pricing. Below that, the repository path 'pytest-dev / pytest' is shown, along with a 'Code' tab that is currently selected. The main area displays the 'conftest.py' file under the 'testing' directory. The code in the file is as follows:

```
1 # mypy: allow-untyped-defs
2 from __future__ import annotations
3
4 from collections.abc import Generator
5 import dataclasses
6 import importlib.metadata
7 import re
8 import sys
9
10 from packaging.version import Version
11
12 from _pytest.monkeypatch import MonkeyPatch
13 from _pytest.pytester import Pytester
14 import pytest
15
16 if sys.gettrace():
17     @pytest.fixture(autouse=True)
18     def restore_tracing():
19         """Restore tracing function (when run with Coverage.py).
20
21             https://bugs.python.org/issue37011
22
23             """
24             orig_trace = sys.gettrace()
25             yield
26             if sys.gettrace() != orig_trace:
27                 sys.settrace(orig_trace)
28
29
30     @pytest.fixture(autouse=True)
31     def set_column_width(monkeypatch: pytest.MonkeyPatch) -> None:
32         monkeypatch.setattr("sys.stdout.isatty", lambda: True)
```

The screenshot shows the GitHub Trending page. At the top, there are tabs for Explore, Topics, Trending (which is currently selected), Collections, Events, and GitHub Sponsors. The main title 'Trending' is displayed, followed by the subtitle 'See what the GitHub community is most excited about today.' Below this, there are two tabs: 'Repositories' (selected) and 'Developers'. The page lists several popular repositories, each with a star icon, the repository name, the developer, a brief description, the programming language, the number of stars, and the date it was starred. The repositories listed are:

- viratt / ai-hedge-fund**: An AI Hedge Fund Team. Python. 13,897 stars. Built by 🐫 🐳 🐶 🐱 🐴. 2,126 stars today.
- ComposioHQ / composio**: Composio equip's your AI agents & LLMs with 100+ high-quality integrations via function calling. Python. 23,470 stars. Built by 🐫 🐳 🐶 🐱 🐴. 1,989 stars today.
- microsoft / generative-ai-for-beginners**: 21 Lessons. Get Started Building with Generative AI. Jupyter Notebook. 73,898 stars. Built by 🐫 🐳 🐶 🐱 🐴. 744 stars today.
- KRTirtho / spotube**: Open source Spotify client that doesn't require Premium nor uses Electron! Available for both desktop & mobile! Dart. 36,835 stars. Built by 🐫 🐳 🐶 🐱 🐴. 125 stars today.
- EFForg / rayhunter**: Rust tool to detect cell site simulators on an orbic mobile hotspot. Rust. 788 stars. Built by 🐫 🐳 🐶 🐱 🐴. 266 stars today.
- grpc-ecosystem / grpc-gateway**: gRPC to JSON proxy generator following the gRPC HTTP spec. Go. 18,892 stars. Built by 🐫 🐳 🐶 🐱 🐴. 128 stars today.
- unionlabs / union**: The trust-minimized, zero-knowledge bridging protocol, designed for censorship resistance, extremely high security, and usage in decentralized finance.

많은 사람들이 본인이 직접 개발한 결과물을 오픈소스로 공개

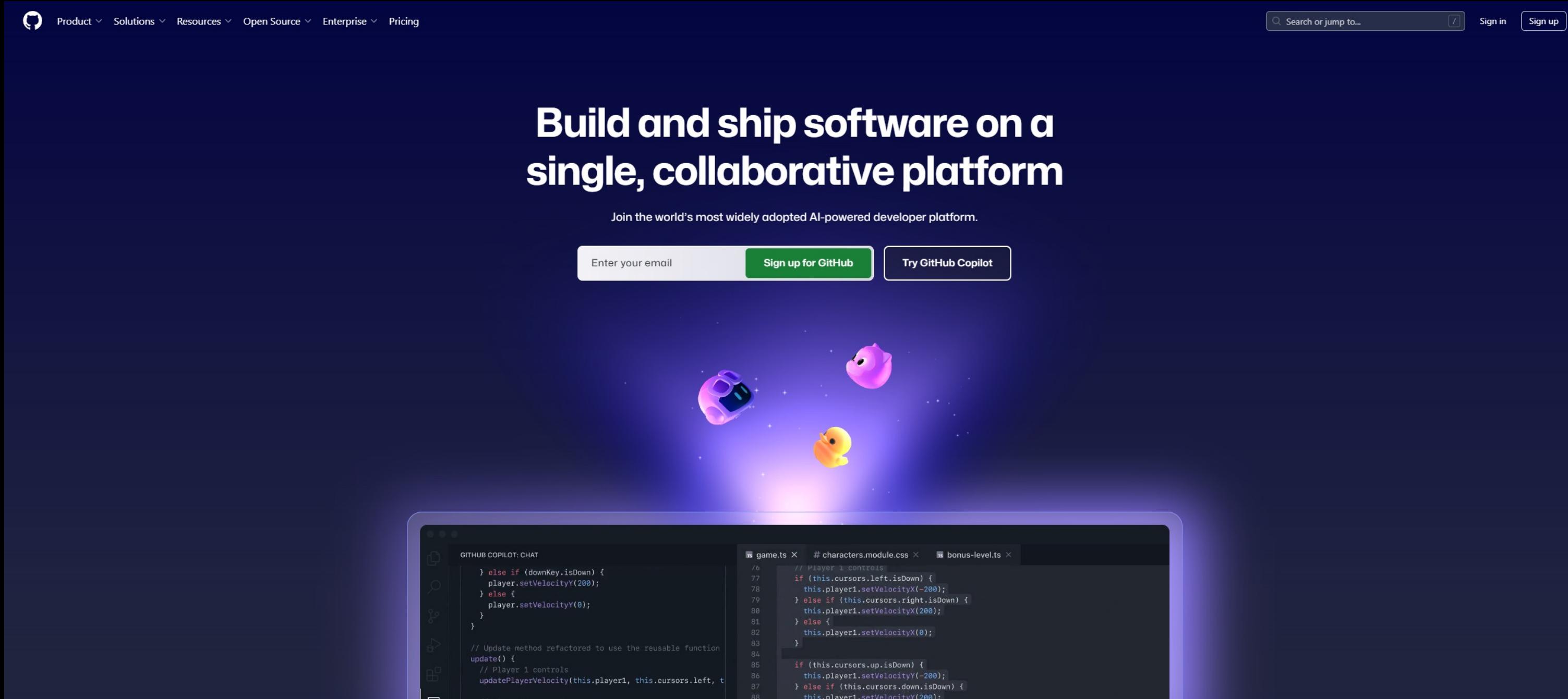


# [실습1] GitHub 사용 준비

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용



<https://github.com/> 접속 후 가입



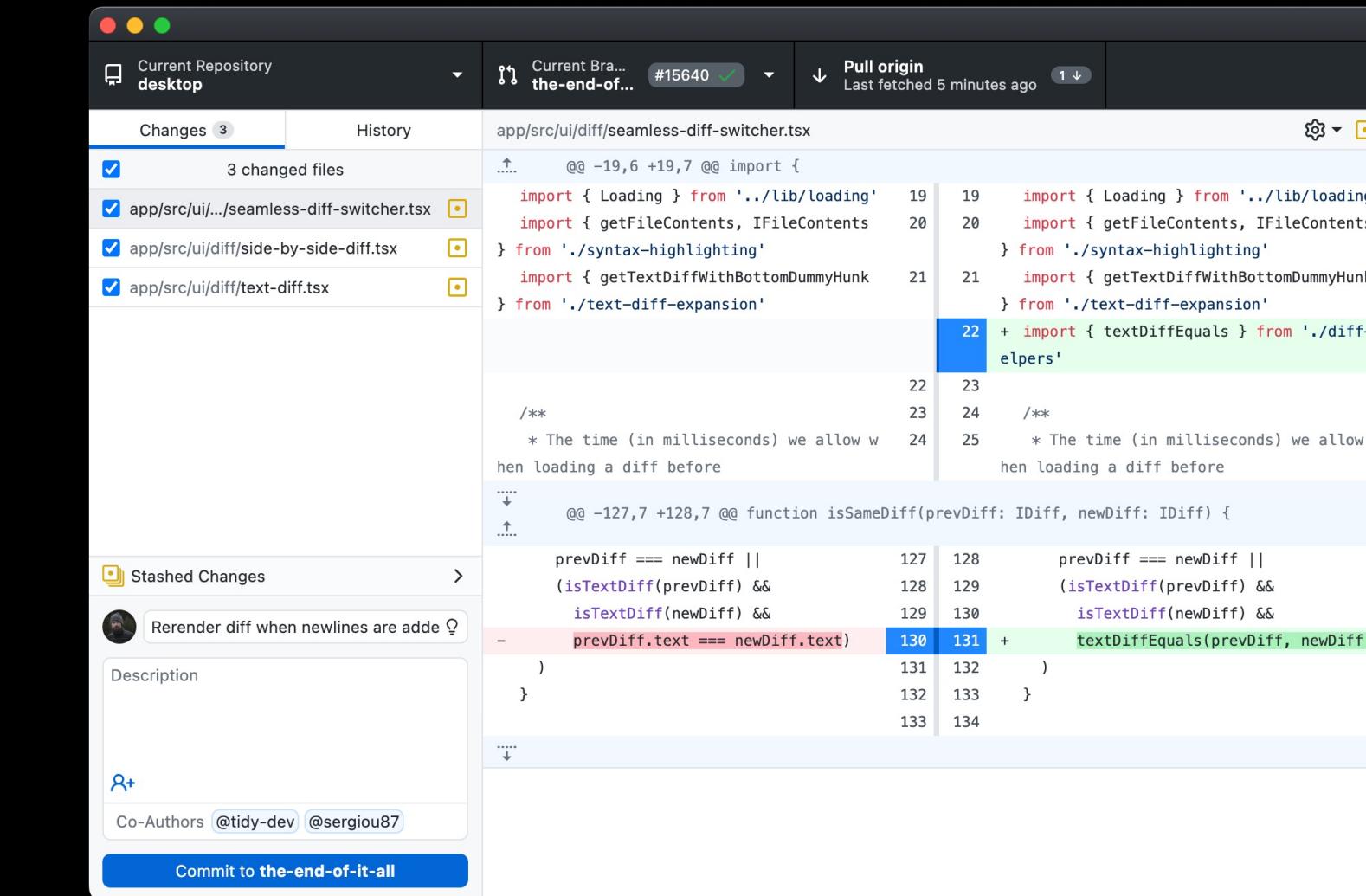
# [실습1] GitHub 사용 준비

```
$ gh pr status

Current branch
There is no pull request associated with [develop]

Created by you
#1011 Update readme [readme-fix]
- Checks pending - Review required

Requesting a code review from you
#1015 Improve error handling [better-error-handling]
✓ Checks passing + Changes requested
```



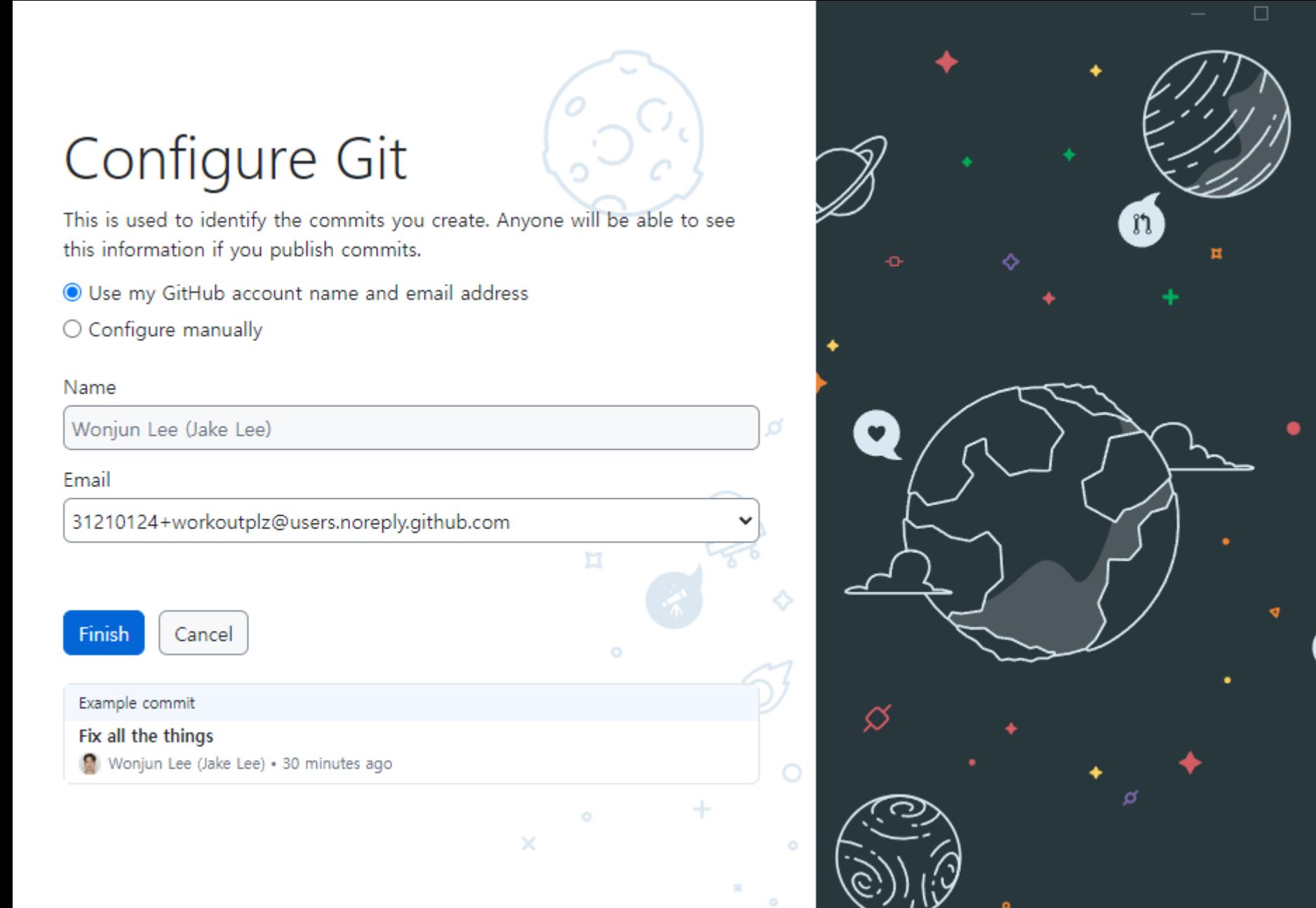
명령어 기반의 CLI

GUI 기반의 GitHub Desktop

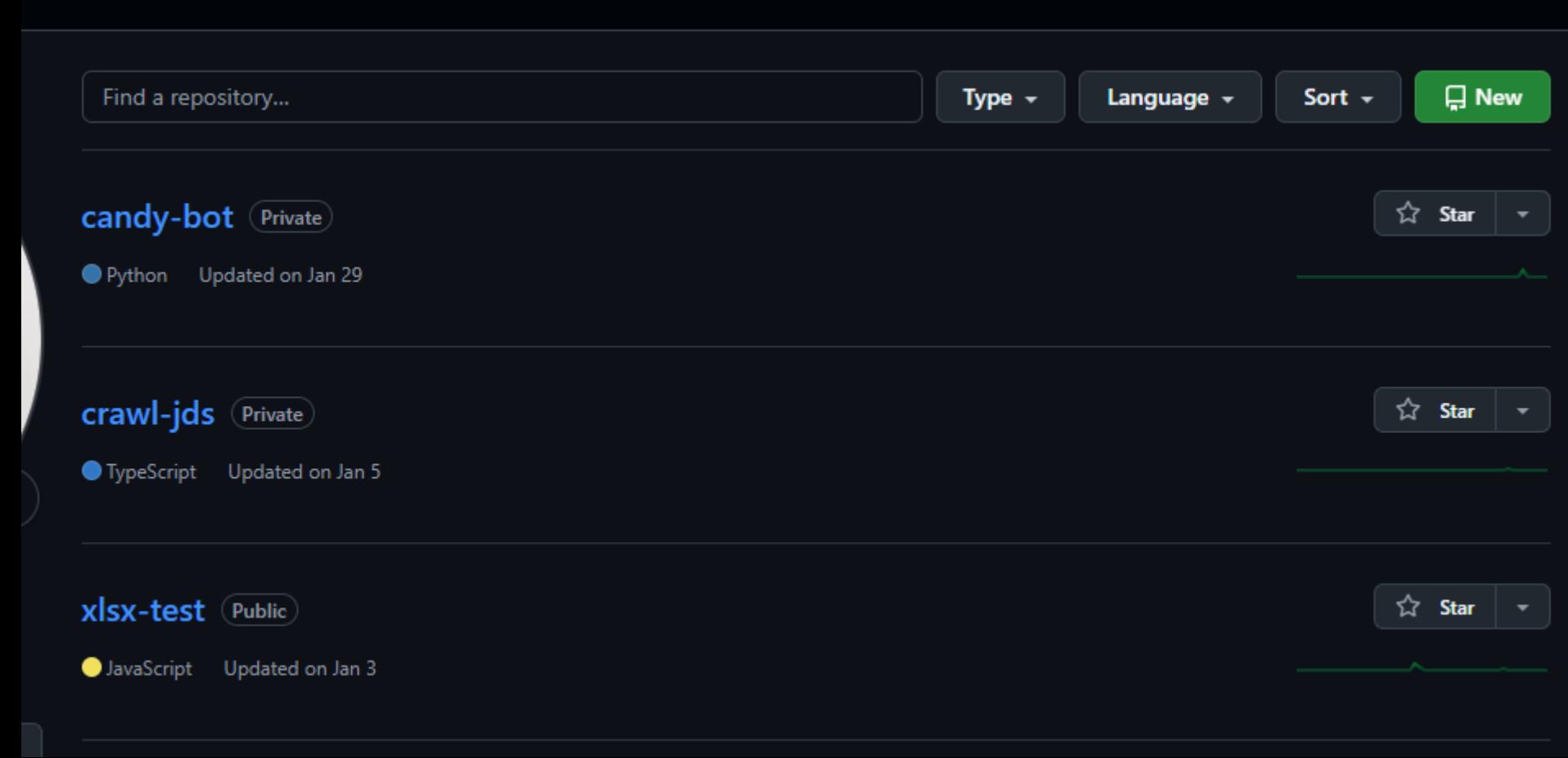
⇒ 본 수업에서는 비교적 다루기 쉬운 GitHub Desktop 위주로 진행



# [실습1] GitHub 사용 준비



- <https://desktop.github.com/download/>에서 다운로드
- GitHub Desktop 설치 시, 왼쪽 화면과 같이 설정



## 레포지토리(Repository)란?

- 하나의 코드 저장소
- 로컬 파일 기준, 하나의 폴더
- 프로젝트 하나 = Repository 하나
- 기술 스택 등에 따라 같은 프로젝트 내 여러 개의 Repository 생성
- 줄여서 '레포(repo)'라고도 표현



# 첫 Repository 만들기

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

**Repository template**

No template

Start your repository with a template repository's contents.

**Owner \***  /

Great repository names are short and memorable. Need inspiration? How about [miniature-spork](#) ?

**Description (optional)**

Public  
Anyone on the internet can see this repository. You choose who can commit.

Private  
You choose who can see and commit to this repository.

**Initialize this repository with:**

Add a README file  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

**Add .gitignore**

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

(i) You are creating a public repository in your personal account.

**Create repository**

## Repository 생성 시 설정해줘야 하는 항목

- Template
- Owner
- Repository name
- Description
- 공개 범위 (Public / Private)
- README file initialize
- .gitignore template
- license



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Repository template

No template

Start your repository with a template repository's contents.

Owner \*      Repository name \*

workoutplz /

Great repository names are short and memorable. Need inspiration? How about [miniature-spork](#) ?

Description (optional)

Public      Private

Anyone on the internet can see this repository. You choose who can commit.

Private      You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

You are creating a public repository in your personal account.

Create repository

## 공개 범위가 Public 일 경우

- Repository를 누구나 조회 가능
- 하나의 포트폴리오가 되기도 함
- 외부 서비스 연동 시 비교적 간편
- 민감 정보는 절대 업로드 X (패스워드 등)

## 공개 범위가 Private 일 경우

- Repository를 특정 사용자만 조회 가능
- 외부 노출이 안되어 비공개 프로젝트 가능
- 무료 사용자에게는 기능이 일부 제한
- 주로 사내 업무는 Private



# 첫 Repository 만들기

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere?  
[Import a repository.](#)

Required fields are marked with an asterisk (\*).

**Repository template**

No template

Start your repository with a template repository's contents.

**Owner \*** workoutplz /

**Repository name \***

Great repository names are short and memorable. Need inspiration? How about [miniature-spork](#)?

**Description (optional)**

Public  
Anyone on the internet can see this repository. You choose who can commit.

Private  
You choose who can see and commit to this repository.

**Initialize this repository with:**

Add a README file  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

**Add .gitignore**

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

ⓘ You are creating a public repository in your personal account.

**Create repository**

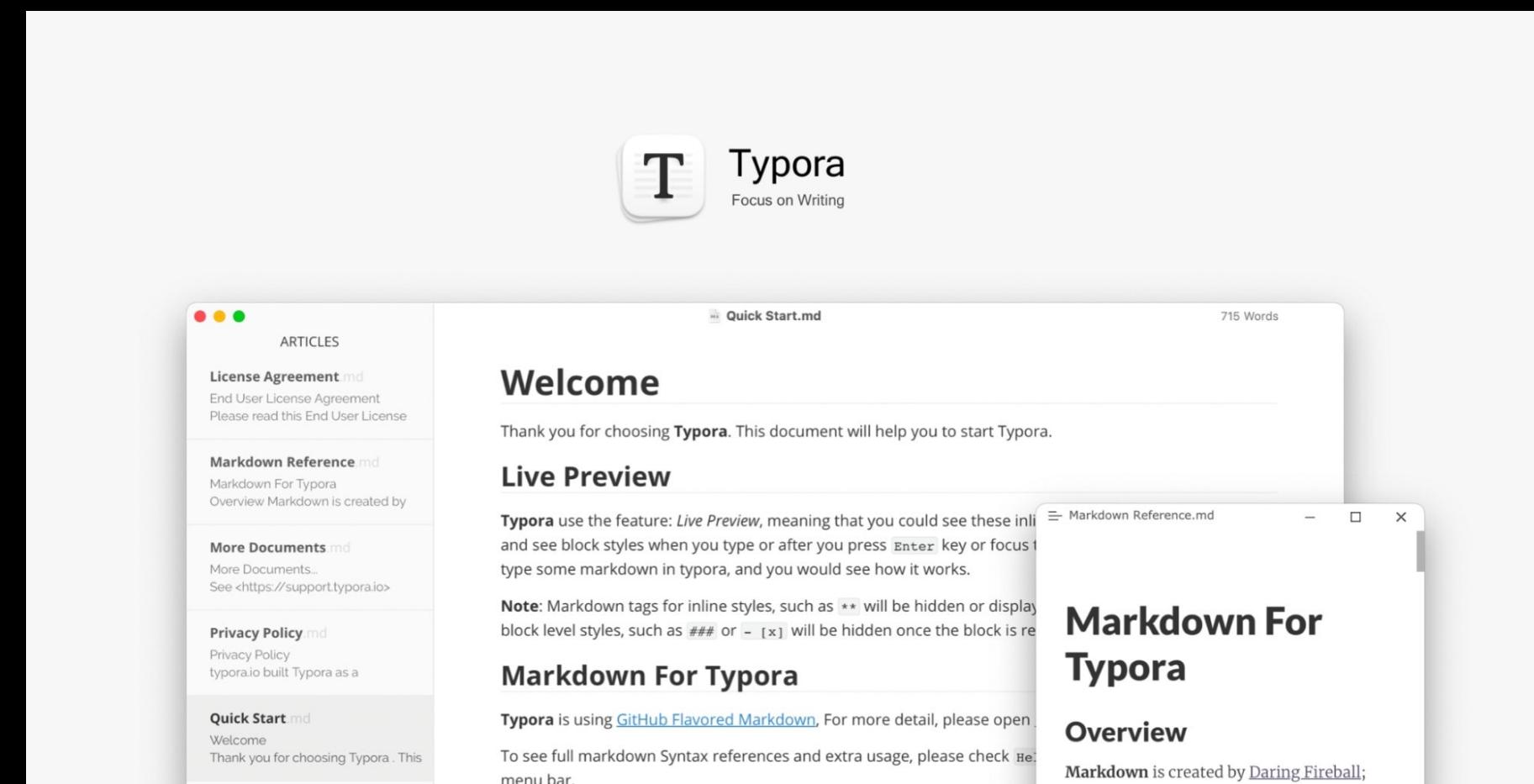
## README?

- Repository에 대해 설명하는 글
- 설치 및 실행 방법, 사용법, 기여하는 방법, License 및 저작권 정보 제공
- 사용자, 개발자가 프로젝트에 대해 이해하기 쉬워짐
- 프로젝트 신뢰도와 완성도가 올라가는 효과
- 원활한 커뮤니케이션을 위한 기본기
- 주로 Markdown 문법으로 작성
- 좋은 README에 대한 참고 링크  
<https://github.com/matiassingers/awesome-readme>



## Markdown?

- 일반 텍스트 문서 내에서 간단한 표기법을 사용해 서식을 지정할 수 있도록 한 마크업 언어
- 문법이 간결해 **텍스트를 손쉽게 서식화 할 수 있음**
- GitHub 뿐만 아니라 블로그, 메모 앱 등에서 꼭 넓게 사용
- README.md 와 같이 확장자가 .md



마크다운 파일(.md)를 하나 생성 후 VS Code로 편집

## 1. 제목

```
# 제목1
## 제목2
### 제목3
```

## 2. 텍스트 효과

~~취소선~~

**\*\*굵게\*\***

\*기울임\*

굵게

기울임

## 3. 항목

- 순서 없는 항목 1
- 순서 없는 항목 2

1. 첫 번째 항목
2. 두 번째 항목

## 4. 링크와 이미지

[Google](<https://www.google.com>)

![로고]([https://cdn-cms.elice.io/elice-strapi/CI\\_8fd881eb6a.webp](https://cdn-cms.elice.io/elice-strapi/CI_8fd881eb6a.webp))

## 5. 인용문

> 0/ 문장은 인용문입니다.

## 6. 취소선

```
---  
***  
---
```

## 7. 표

제목1   제목2
----- -----
내용1   내용2
내용3   내용4

## 8. 하이퍼 링크 적용

```
<https://www.example.com>
```

## 9. 코드

```
```python
print("Hello, Markdown!")
```

```

### 10. 이스케이프 문자

\* , - , # 등 서식화하지 않고 문자 그대로를 나타내고 싶을 때 앞에 \를 붙임

```
\*별표\*
```

본인을 소개하는 README.md를 작성해봅시다.



# 첫 Repository 만들기

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Repository template

No template

Start your repository with a template repository's contents.

Owner \*      Repository name \*

workoutplz /

Great repository names are short and memorable. Need inspiration? How about [miniature-spork](#) ?

Description (optional)

Public      Anyone on the internet can see this repository. You choose who can commit.

Private      You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

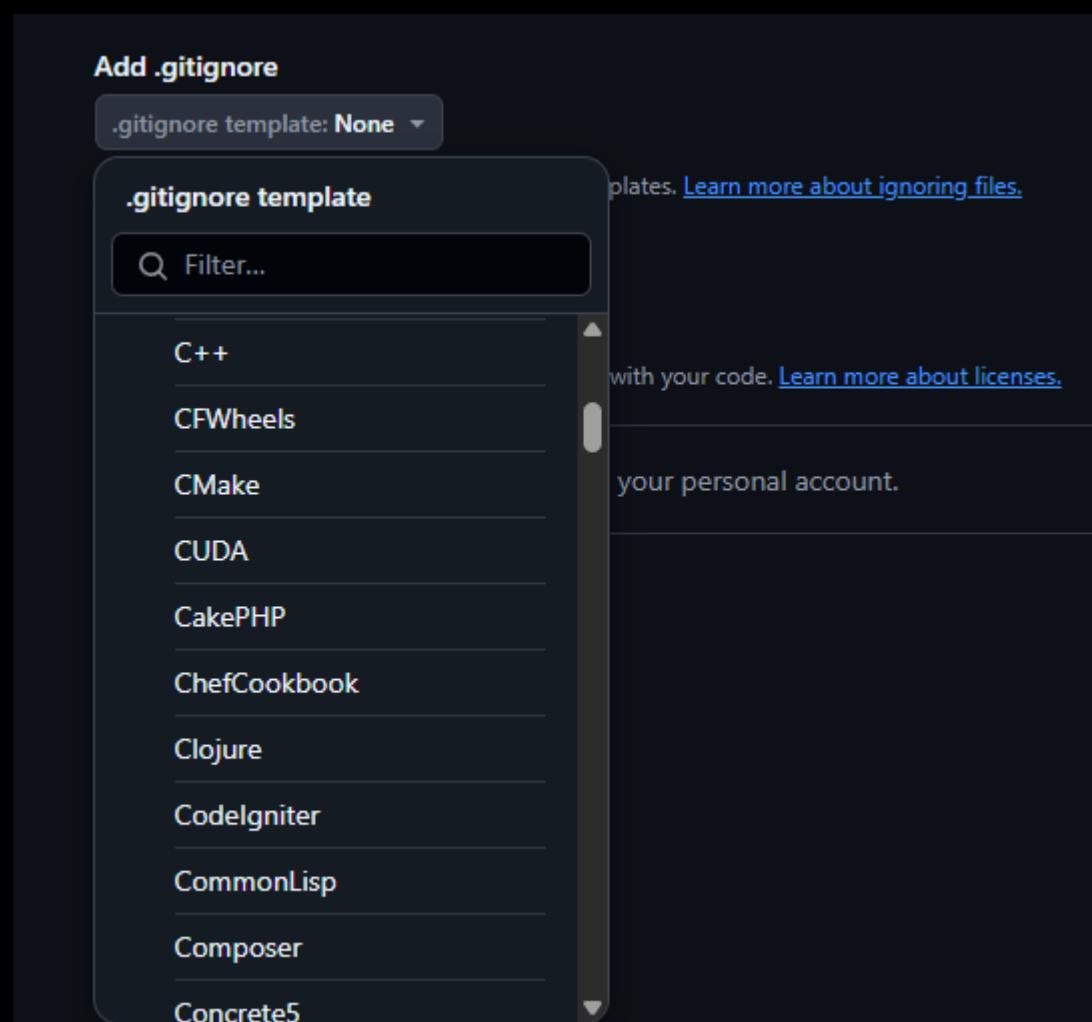
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

(i) You are creating a public repository in your personal account.

**Create repository**

## .gitignore?

- Git 이 무시해야하는 파일이나 폴더를 지정
- 지정되면 Repository에 업로드 되지 않음
- 불필요한 파일, 비밀번호 등 보안 파일을 지정하여 효율적이고 보안을 유지한 운용
- 다양한 언어와 프레임워크 별 템플릿 존재





# 첫 Repository 만들기

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

**Repository template**

No template

Start your repository with a template repository's contents.

**Owner \*** workoutplz **Repository name \*** /

Great repository names are short and memorable. Need inspiration? How about [miniature-spork](#) ?

**Description (optional)**

Public Anyone on the internet can see this repository. You choose who can commit.  
 Private You choose who can see and commit to this repository.

**Initialize this repository with:**

Add a README file This is where you can write a long description for your project. [Learn more about READMEs.](#)

**Add .gitignore**

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

(i) You are creating a public repository in your personal account.

**Create repository**

## license?

- SW 사용, 배포, 수정 등의 권한을 정의
- 사람들이 어떤 조건으로 코드를 사용하고 변경할 수 있는지를 명확히 규정
- 각 license마다 사용과 배포 조건이 다름
- 개인 공부나 간단한 프로젝트에는 지정하지 않아도 무방
- 타인의 코드 사용 시 반드시 확인해야 함
- 관련 사이트

<https://choosealicense.com/>



## MIT 라이선스

- 소스 코드의 수정, 복사, 배포, 상업적 이용 모두 허용
- 수정된 소스코드 공개 의무 없음
- 원본 저작권 및 라이선스 고지를 소스코드에 포함
- 매우 관대한 라이선스



## Apache License 2.0

- 기여자들의 특허권을 명시적으로 포함
- 특허 관련 소송 발생 시 사용자 보호
- 자유로운 소스코드 수정 및 배포
- 수정 내용에 대해 명시와 라이선스 고지 필수
- 수정한 코드를 공개 의무는 없으나,  
라이선스 조건을 준수



*Free as in Freedom*

## GNU GPL

- SW를 수정하거나 파생 작품을 만들 경우, 동일한 GPL 라이선스로 배포
- SW의 사용, 수정, 재배포에 관한 자유 보장
- 코드 공개 의무화
- 다른 라이선스와 호환성 문제가 발생할 수 있어 사용 전에 라이선스 조건 검토 필요



# 첫 Repository 만들기

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용

생성 완료 후  
'Set up in Desktop' 클릭

Repository를 연동할  
로컬 경로를 지정하여 Clone

The screenshot shows a GitHub repository page for a repository named 'first'. At the top, there are buttons for 'Pin', 'Unwatch', 'Fork', and 'Star'. Below the repository name, there are sections for 'Start coding with Codespaces' and 'Add collaborators to this repository'. The main section is titled 'Quick setup... if you've done this kind of thing before'. It features a 'Set up in Desktop' button, which is highlighted with a purple rectangle. Other options shown are 'HTTPS' and 'SSH' with the URL 'https://github.com/workoutplz/first.git'. Below this, there's a command-line setup guide and a 'ProTip!' at the bottom.

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

**Set up in Desktop** or **HTTPS** **SSH** <https://github.com/workoutplz/first.git>

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

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

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

💡 **ProTip!** Use the URL for this page when adding GitHub as a remote.

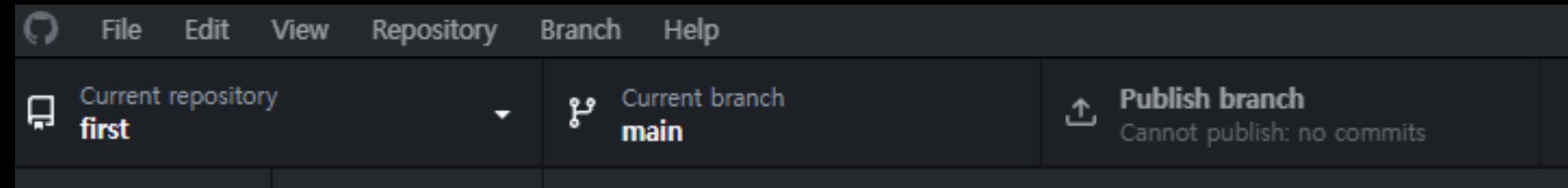


# 첫 Repository 만들기

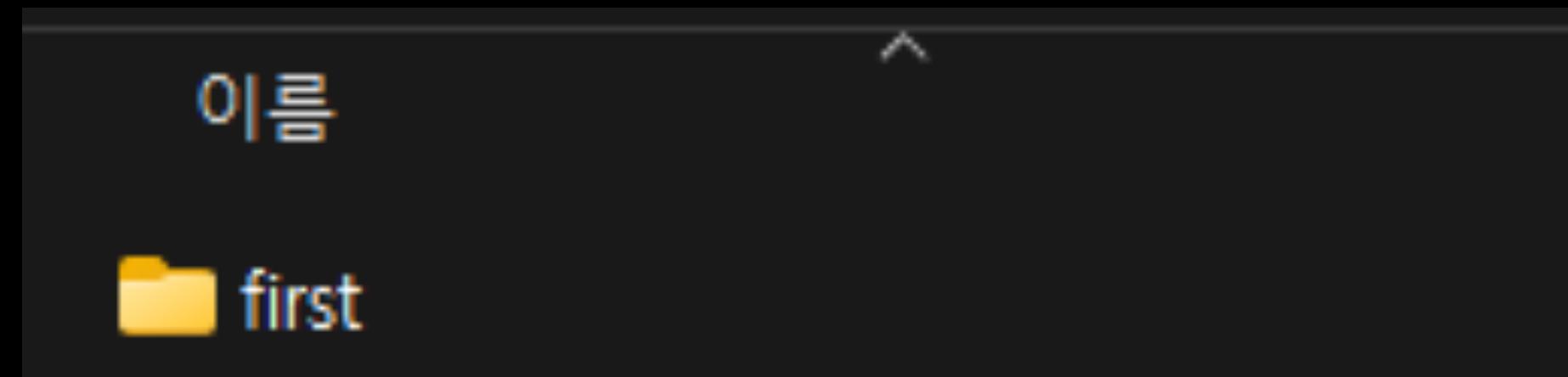
테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용



Github Desktop에 Clone한 repository 이름이 뜨고



내가 지정한 로컬 경로에 Clone한 repository 이름으로 폴더가 생성



## [실습3] Repository Clone

아래 조건에 맞는 Repository를 생성 후, Local 환경으로 Clone을 받아보세요.

1. Owner는 본인의 계정으로 설정해주세요.
2. 누구나 Repository 내 내용을 확인, 수정할 수 있도록 설정해주세요.
3. 현재는 README, LICENSE, .gitignore은 따로 설정하지 않습니다.
4. 그 외 Repository name 등은 자유롭게 설정해주세요.
5. 로컬 경로는 본인이 편한 곳으로 설정하되, 폴더 명은 Repository name과 일치해야합니다.



로컬 작업 폴더  
(Working Directory)



Staging Area



Local Repository



Repository  
(Remote Repository)



hello.txt를 Github repository에 업로드 하기 위해서 위 4개의 영역을 거쳐야 함



로컬 작업 폴더  
(Working Directory)



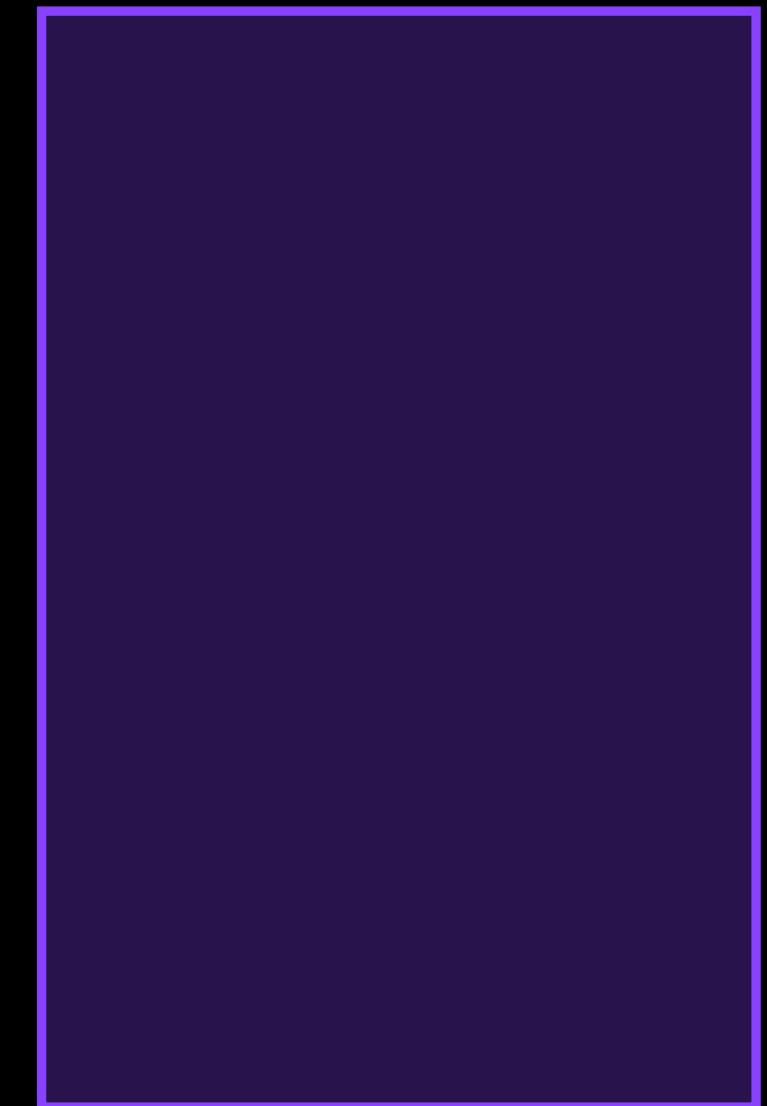
Staging Area



Local Repository



Repository  
(Remote Repository)



최초 파일 생성 시에는 Working Directory에서만 변경 사항을 인지



# Staging Area

로컬 작업 폴더  
(Working Directory)



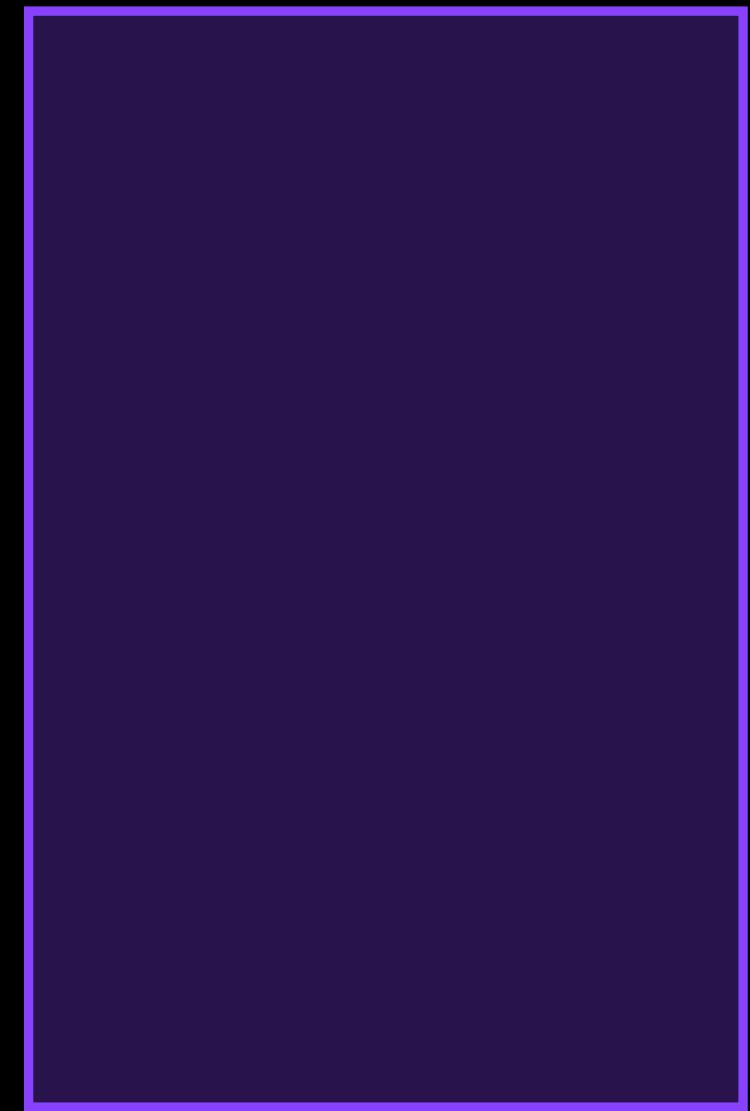
Staging Area



Local Repository



Repository  
(Remote Repository)



hello.txt 를 Git이 추적할 수 있도록 Staging Area로 추가  
Staging Area 는 변경 사항을 추적하고 잠깐 모아두는 공간



로컬 작업 폴더  
(Working Directory)



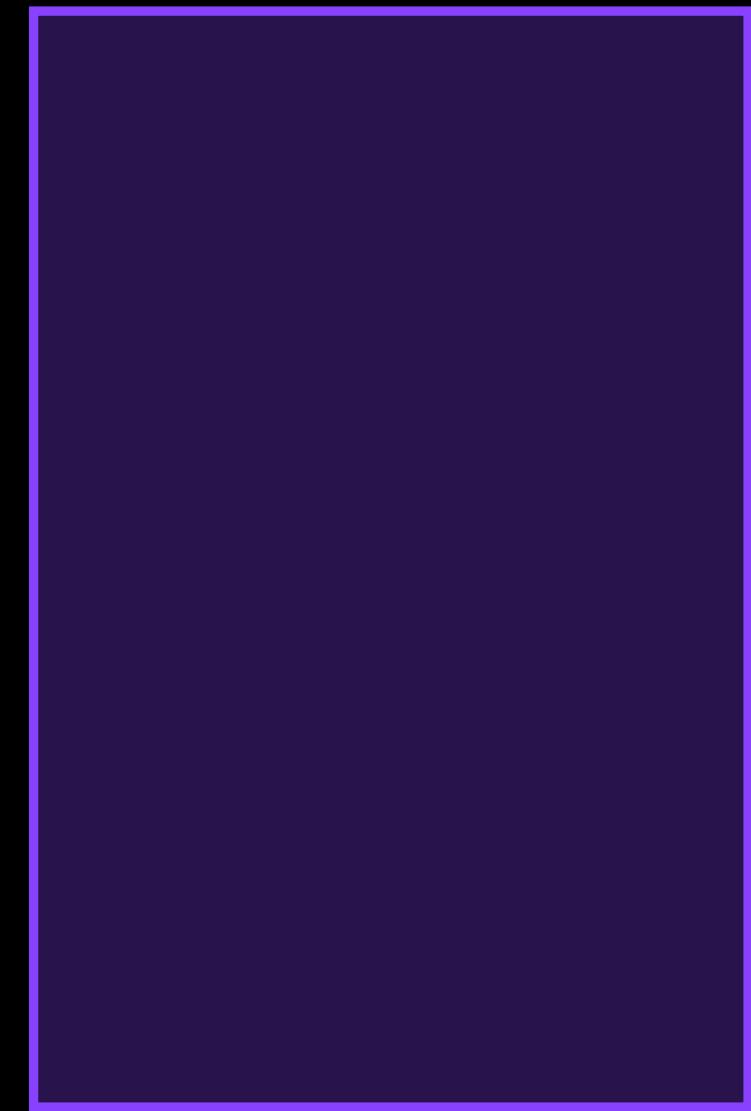
Staging Area



Local Repository



Repository  
(Remote Repository)



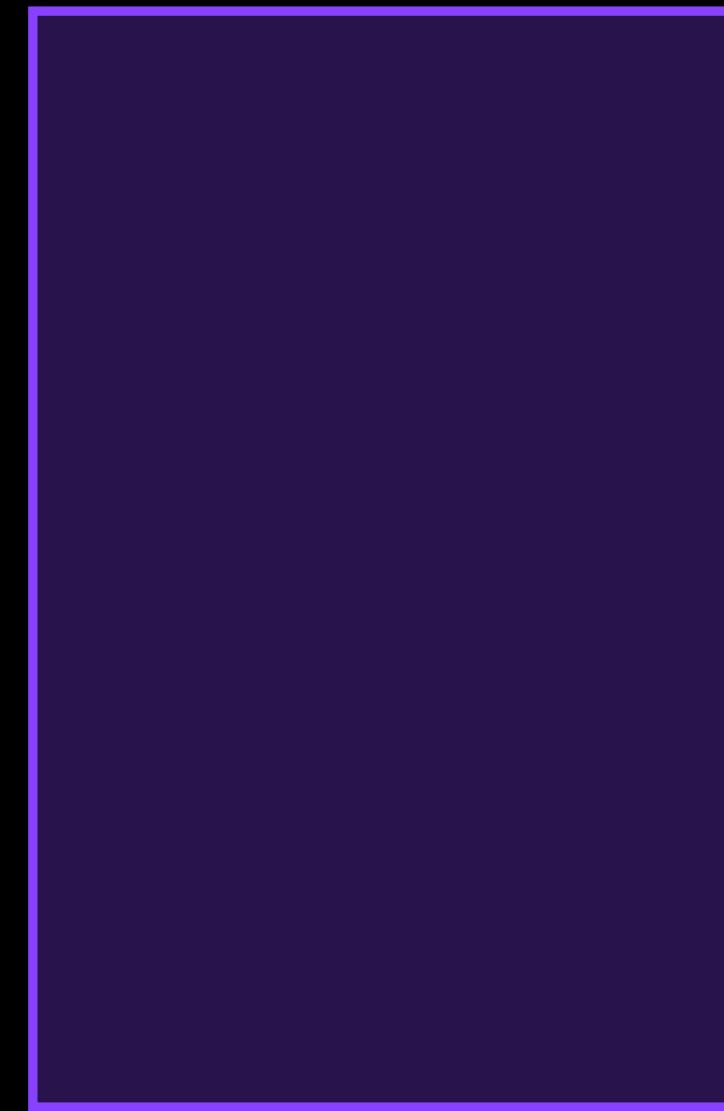
commit  
→

hello.txt의 변경 사항을 Github 상에 올리기 위해 로컬에서 변경 이력을 잠깐 관리  
실제 Remote Repository에는 변경 사항이 반영되지 않음



# Remote Repository

로컬 작업 폴더  
(Working Directory)



Staging Area



Local Repository



Repository  
(Remote Repository)



push  
→

Local Repository 영역에 모아진 변경 사항(commit)을 Remote Repository에 업로드



로컬 작업 폴더  
(Working Directory)



Staging Area



Local Repository



Repository  
(Remote Repository)



add  
→

commit  
→

push  
→

Git add, commit, push 의 절차를 거쳐야  
비로소 로컬의 변경 사항이 Remote Repository에 적용



# Commit 과 Push

The screenshot shows a GitHub repository page for 'first' and a GitHub Desktop application window.

**Github Repository Page:**

- Shows a 'Code' tab.
- A 'hello.txt' file is listed with the content "hello world".
- File details: 이름 (Name): hello, 상태 (Status): 정상 (Normal), 수정한 날짜 (Last modified): 2025-03-08 오후 10:51, 유형 (Type): 텍스트 문서 (Text Document), 크기 (Size): 1KB.

**Github Desktop Application:**

- Shows the 'first' repository with the 'main' branch selected.
- A 'Changes' tab shows one change: 'hello.txt' has been modified.
- The commit message is "hello world".

**Bottom Left:** A 'Create hello.txt' dialog box is open, showing a 'Description' field and a 'Commit to main' button.

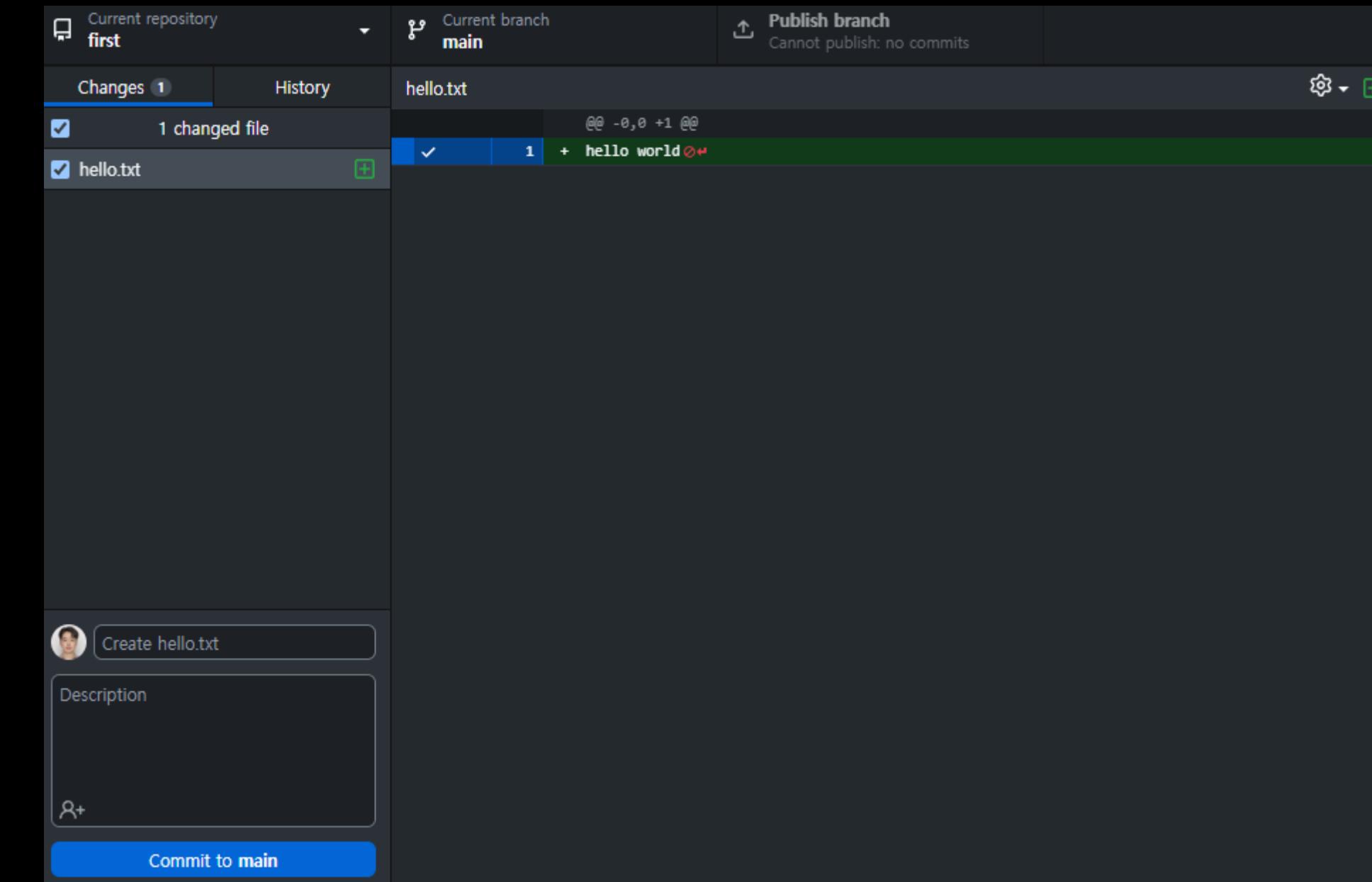
- Clone 받았던 로컬 경로에 hello.txt를 저장해도 Github 내에는 미반영
- 하지만 Github Desktop은 인식  
⇒ Staging Area에 있는 상태

현 상황에서 Github에 반영하려면?  
= Remote Repository에 올리려면?



# Commit

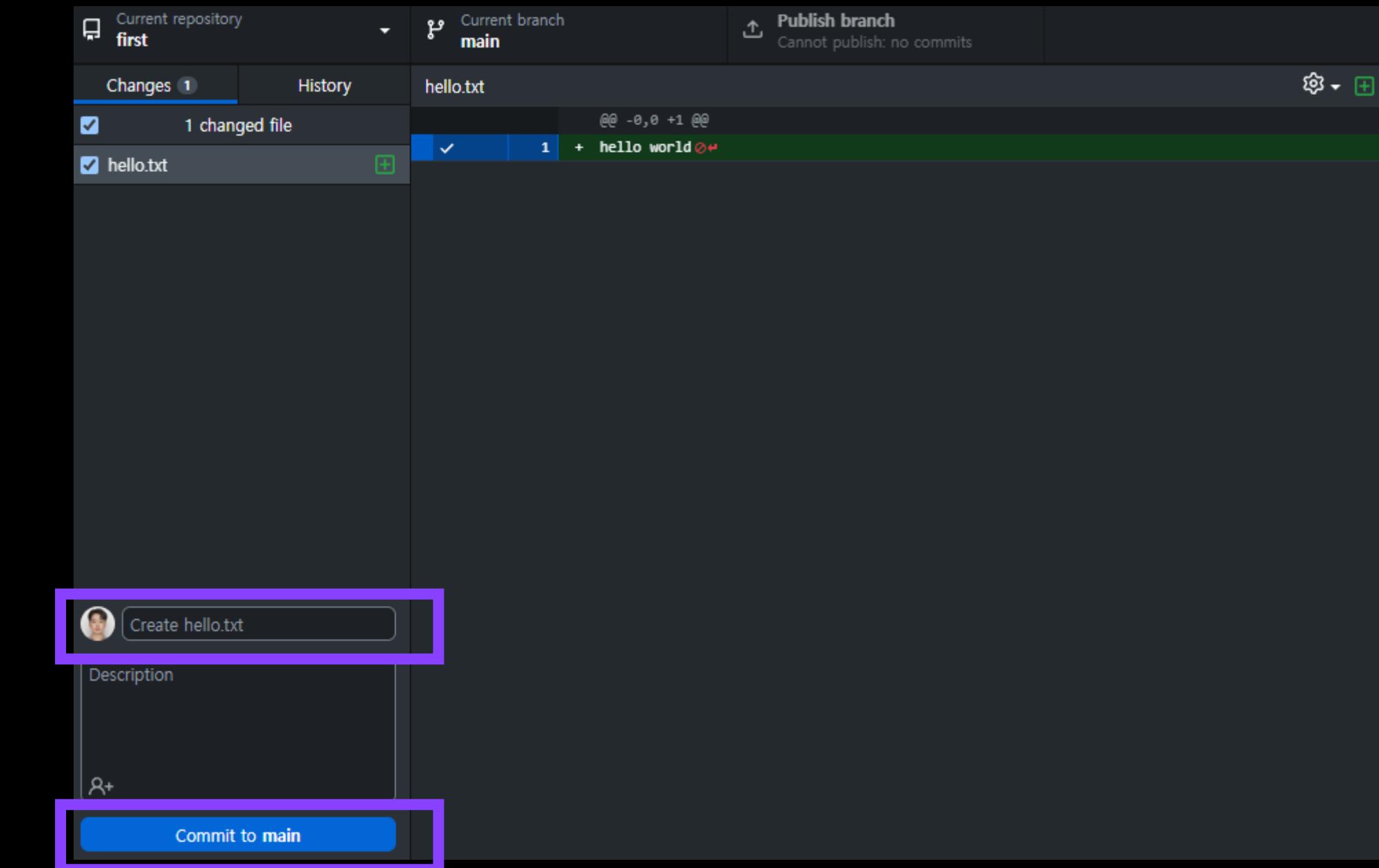
Staging에 있는 변경 사항을  
Local Repository 영역으로 이동  
⇒ Commit 필요





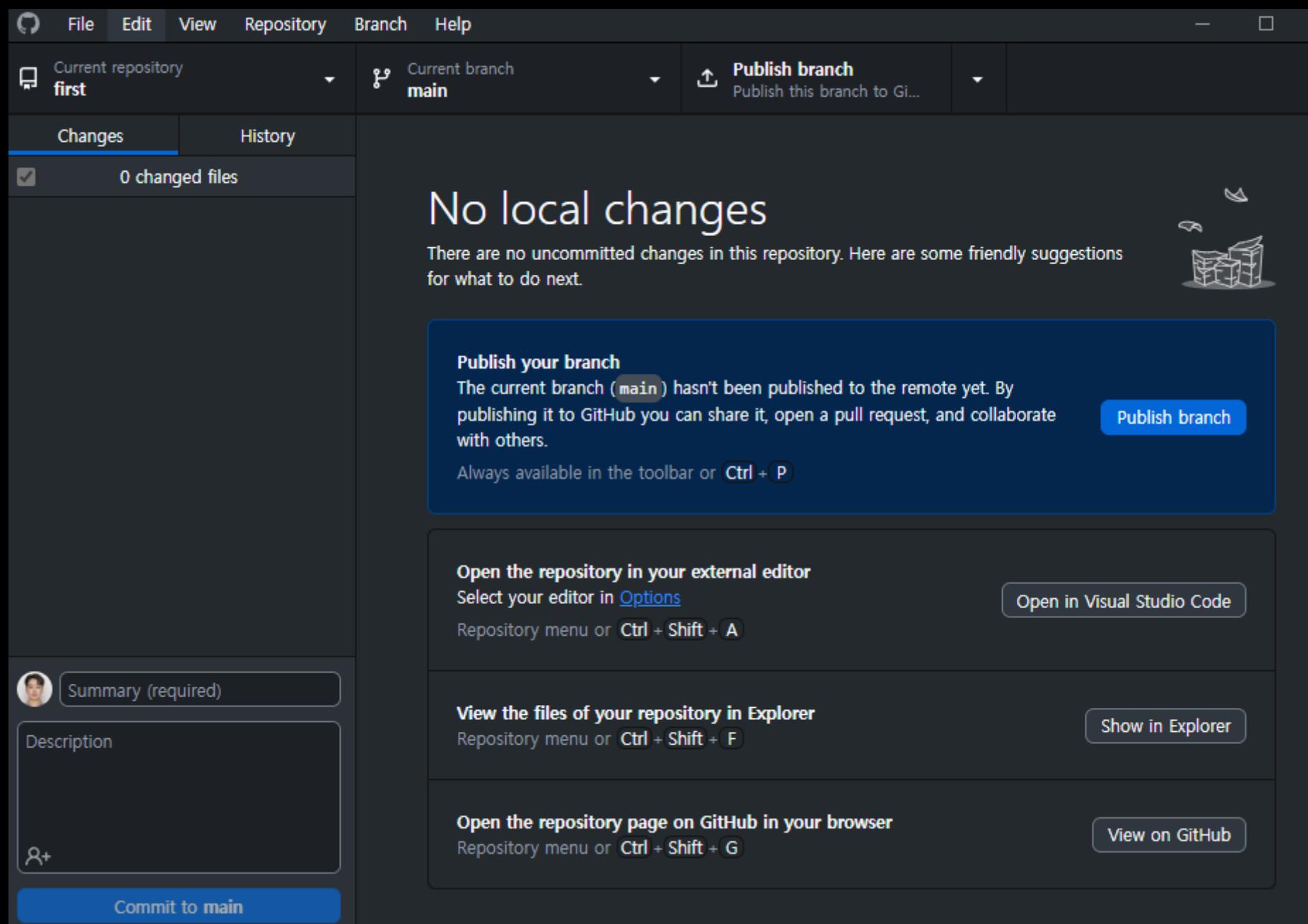
# Commit

Commit 시에는 변경 내용 요약 메시지,  
'commit message' 작성 후 commit





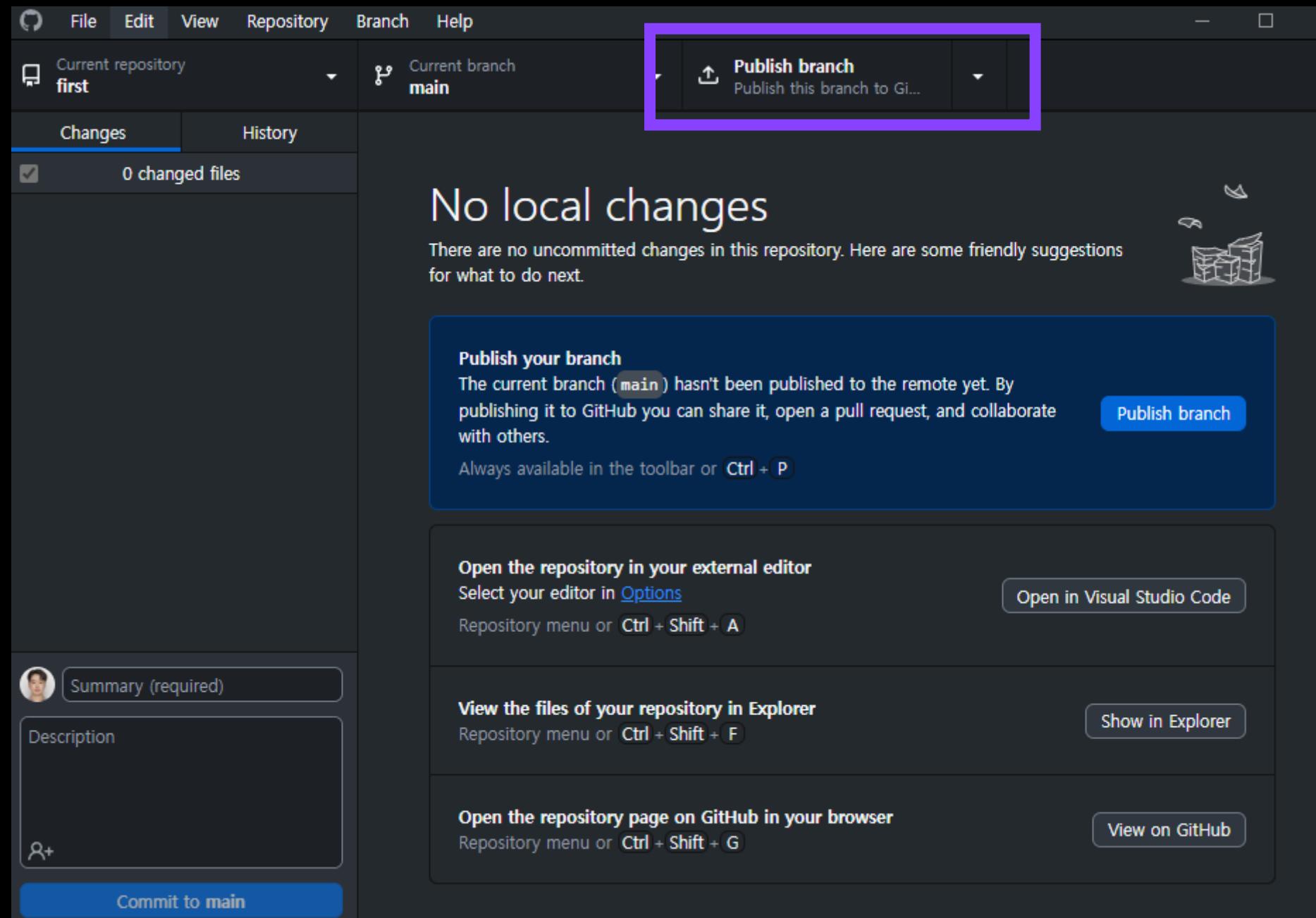
# Push



Local Repository에 옮겨 간 상황이므로 Staging Area에는 보이지 않으나,  
Remote Repository에도 존재하지 않음  
⇒ Remote Repository에 Push 필요



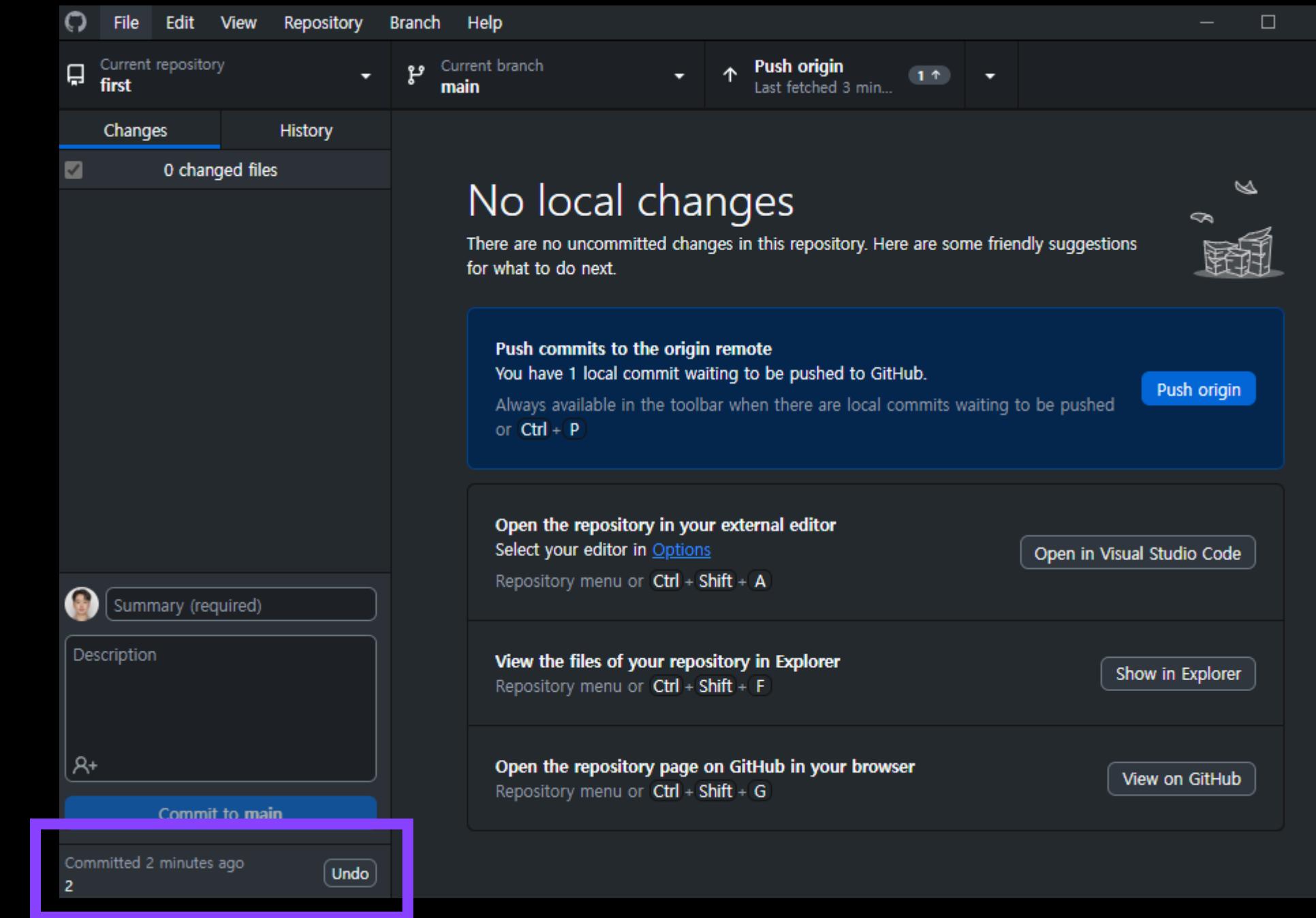
# Push



- 최초 push 시,  
Publish branch 버튼 클릭
- 이후 push 시에는  
'Push origin' 버튼으로 변경 됨
- Push 완료 시에는  
Remote Repository 도 변경



# Undo



만약 잘못 commit을 했을 경우,  
push 이전에 Undo 가능



# Commit 과 Push

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용

workoutplz / first

Type to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

first Public

Pin Unwatch 1 Fork 0 Star 0

main 1 Branch 0 Tags Go to file + <> Code

workoutplz 첫번째 커밋 8aa5c3c · 8 minutes ago 1 Commit

hello.txt 첫번째 커밋 8 minutes ago

README

Add a README

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

Add a README

No releases published Create a new release

No packages published Publish your first package

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

## Commits

main Commits on Mar 8, 2025

첫번째 커밋

workoutplz committed 7 minutes ago 8aa5c3c

Current repository: first Current branch: main Fetch origin: Last fetched 6 minutes ago

Changes History No branches to compare

첫번째 커밋 Wonjun Lee (Jake Lee) 8aa5c3c 1 changed file hello.txt

첫번째 커밋 Wonjun Lee (Jake Lee) 9 minutes ago 1 changed file hello.txt

| hello.txt | @@ -0,0 +1 @@ | 1 + hello world |
|-----------|---------------|-----------------|
|           |               |                 |

Remote Repository와 Local Repository 모두  
변경 내용이 정상적으로 기록, 적용 된 모습을 볼 수 있음



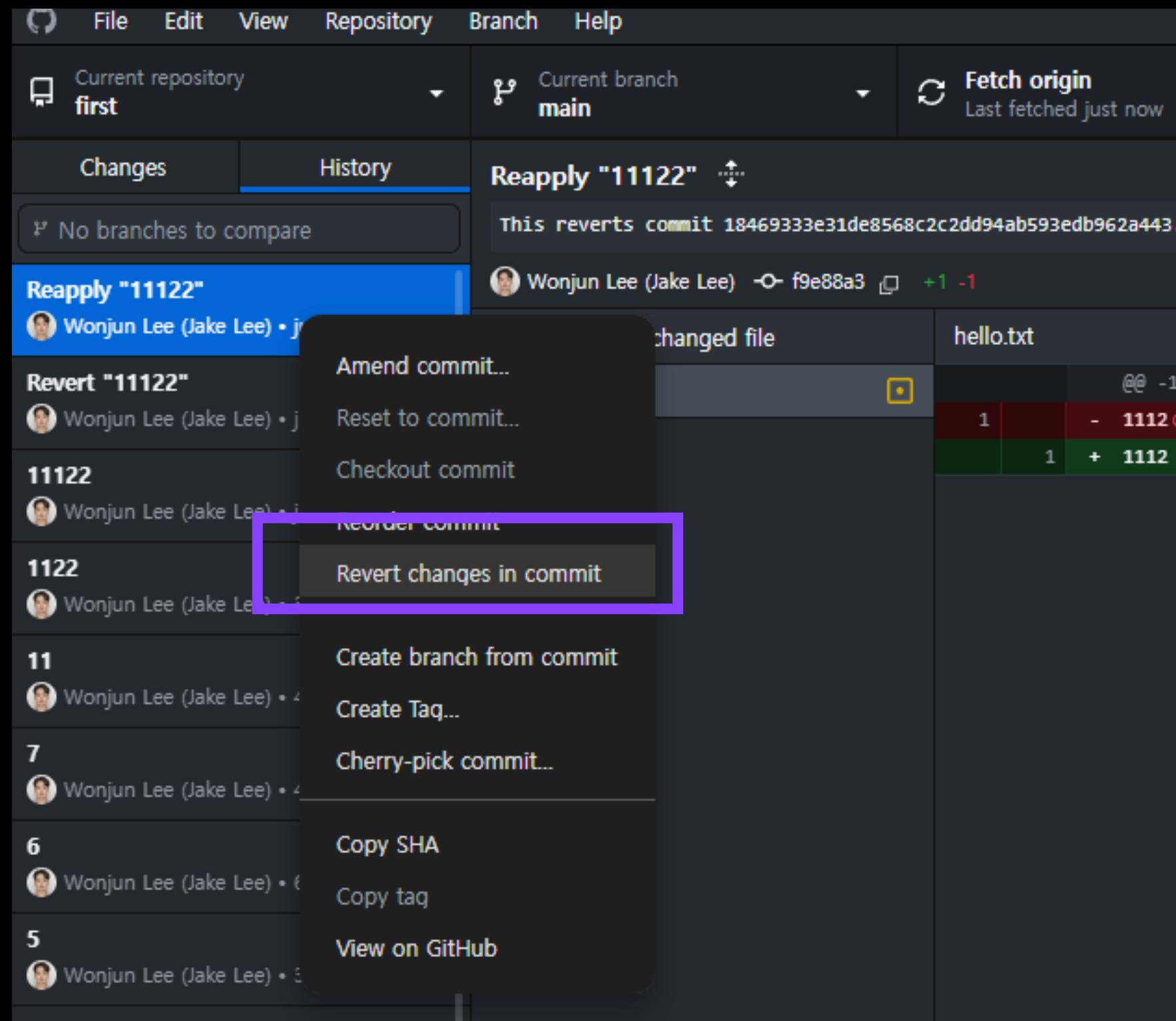
## [실습4] Commit & Push 5번

- Clone 받은 Repository를 대상으로 README.md를 push 해보세요.
- 추가로 최소 5번 이상의 commit과 push를 해보세요.
- 이전에 작성했던 코드들을 올려보세요.

| Commit ID | Author     | Time Ago       |
|-----------|------------|----------------|
| 9cf956a   | workoutplz | 45 minutes ago |
| 48a5d6c   | workoutplz | 48 minutes ago |
| e2b8fc6   | workoutplz | 48 minutes ago |
| 3032bd4   | workoutplz | 49 minutes ago |
| 12c2471   | workoutplz | 50 minutes ago |
| c037b0c   | workoutplz | 1 hour ago     |
| e31e9b1   | workoutplz | 1 hour ago     |
| 3a7acf6   | workoutplz | 1 hour ago     |
| a00b418   | workoutplz | 1 hour ago     |
| 8aa5c3c   | workoutplz | 1 hour ago     |
| 9cf956a   | workoutplz | 1 hour ago     |
| 48a5d6c   | workoutplz | 1 hour ago     |
| e2b8fc6   | workoutplz | 1 hour ago     |
| 3032bd4   | workoutplz | 1 hour ago     |
| 12c2471   | workoutplz | 1 hour ago     |
| c037b0c   | workoutplz | 1 hour ago     |
| e31e9b1   | workoutplz | 1 hour ago     |
| 3a7acf6   | workoutplz | 1 hour ago     |
| a00b418   | workoutplz | 1 hour ago     |
| 8aa5c3c   | workoutplz | 1 hour ago     |



# Revert



- Github Desktop 내 History 탭에서 이전 커밋 기록 확인 가능
- 'Revert changes in commit'을 통해 이전 커밋으로 복구 가능  
(단, 직전의 커밋으로만 가능)

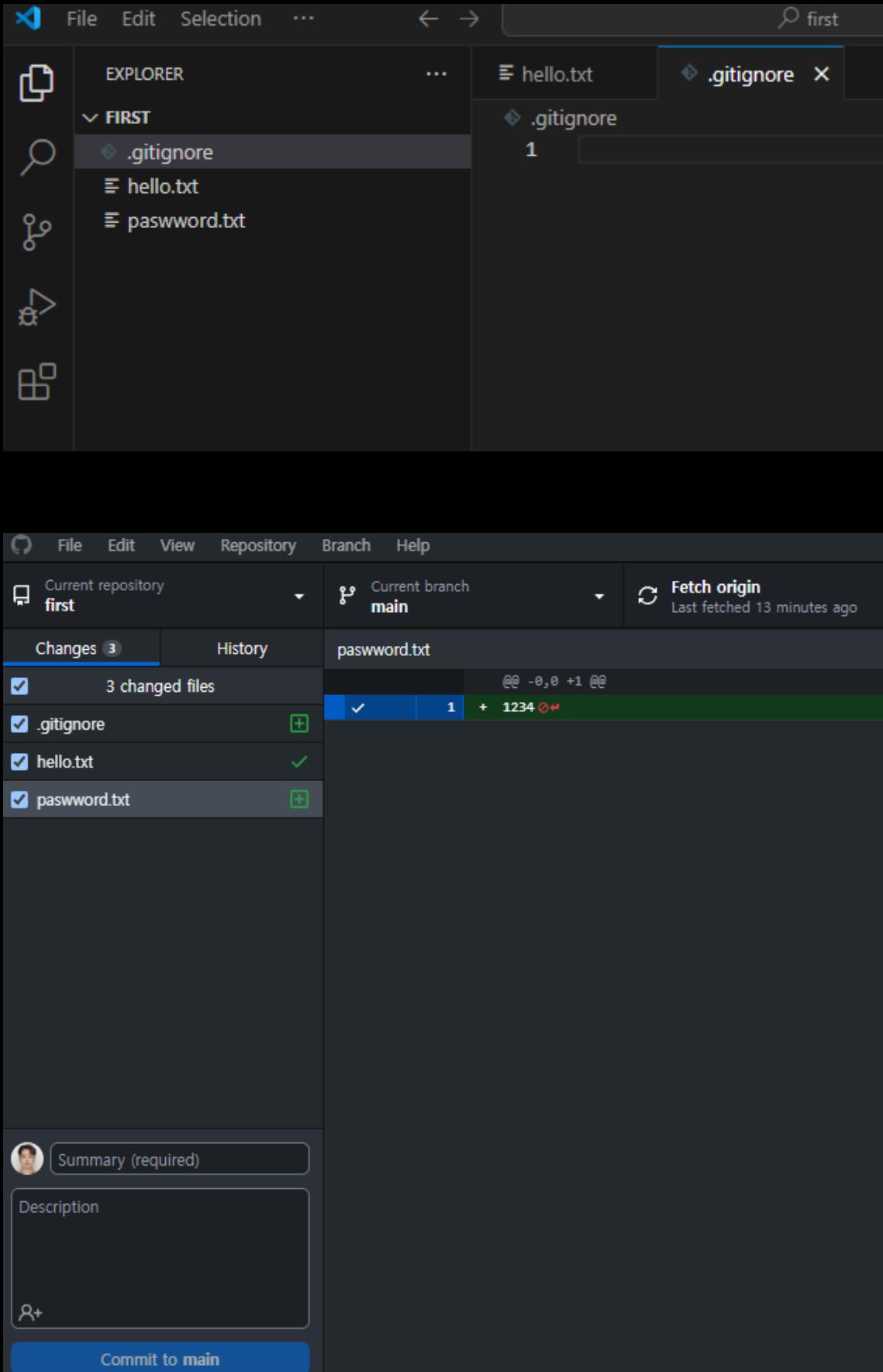


The screenshot shows the official Visual Studio Code website. At the top, there's a navigation bar with links to 'Visual Studio Code', 'Docs', 'Updates', 'Blog', 'API', 'Extensions', 'FAQ', and 'GitHub Copilot'. To the right of the navigation is a search bar labeled 'Search Docs' and a blue 'Download' button. Below the navigation, a call-to-action button says 'Get GitHub Copilot Free in VS Code!'. The main headline reads 'Your code editor. Redefined with AI.' with two buttons below it: 'Download for Windows' (in blue) and 'Get Copilot Free' (in grey). A small note below the buttons says 'Web, Insiders edition, or other platforms'. At the bottom of the screenshot is a large image of the VS Code interface showing a file named 'App.tsx' with some code and a sidebar with AI-generated suggestions.

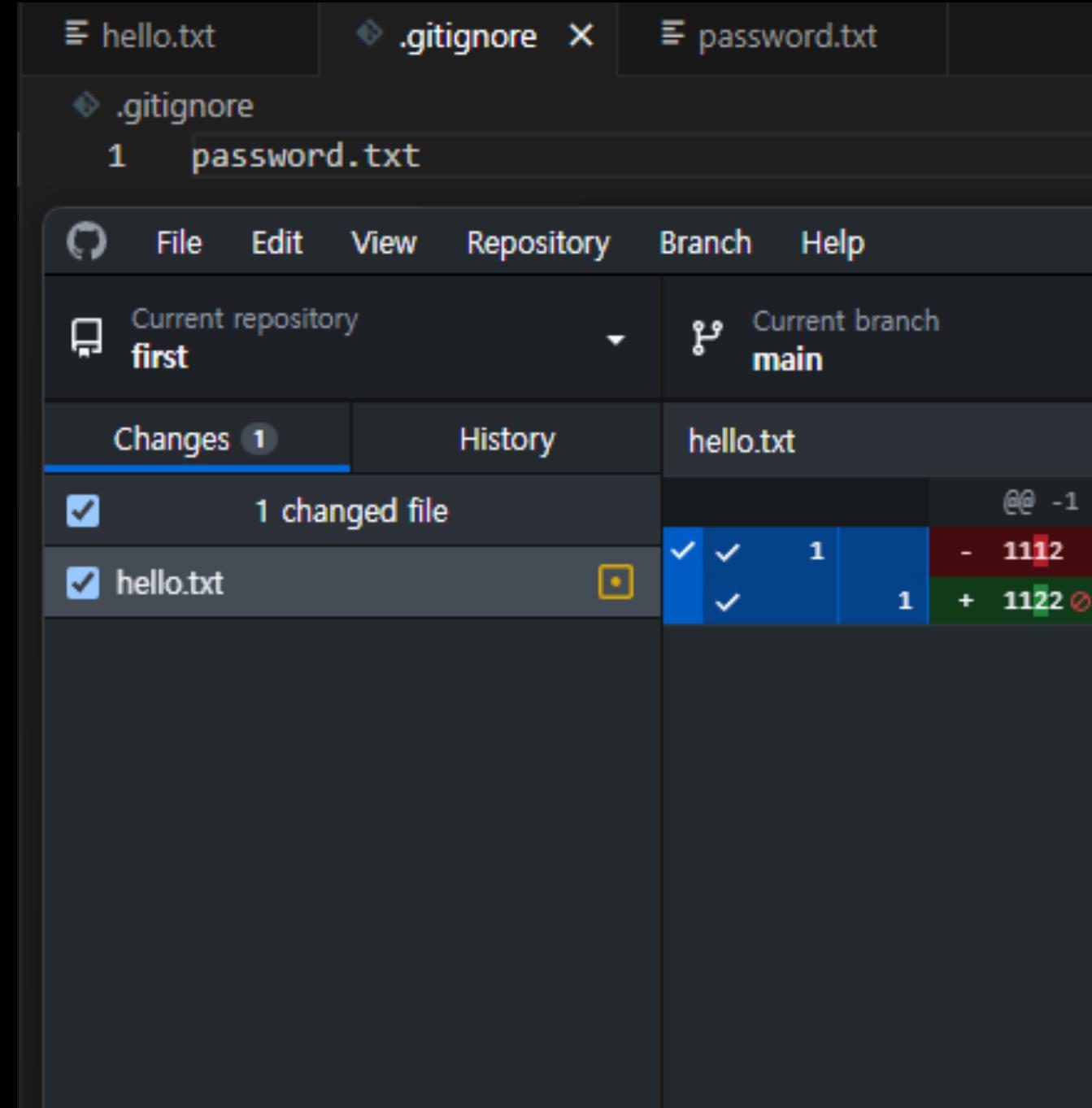
<https://code.visualstudio.com/>에서 다운 후 설치  
개인적으로 사용하는 IDE가 있다면 해당 IDE 사용 가능



# .gitignore



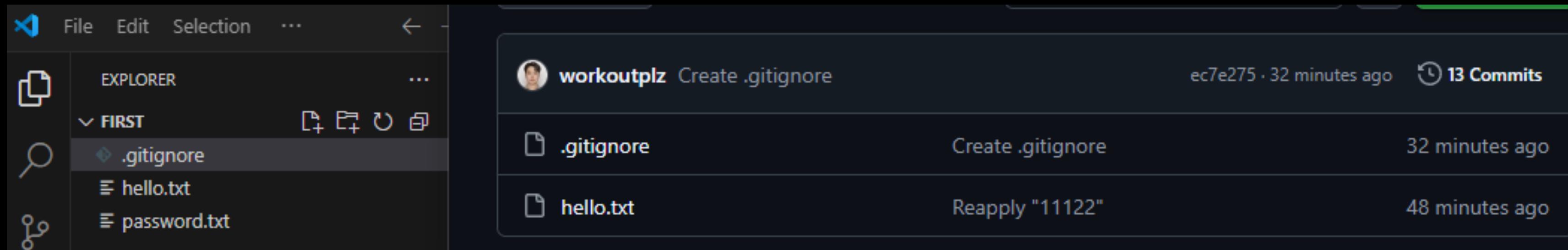
- 왼쪽 상단 사진과 같이 폴더가 구성되어 있다고 할 때, 보안 상 password.txt는 Git에서 추적하면 안됨
- 이 때 사용 하는 것이 .gitignore 파일
- 보안 관련 파일 뿐만 아니라 불필요한 빌드 파일, 라이브러리 관련 폴더를 주로 지정



- 다음과 같이 .gitignore 파일에 업로드가 되면 안되는 파일명 혹은 폴더명을 입력하면 더 이상 git이 추적하지 않음
- 계정에 대한 정보, 내부 IP 정보 등 보안 상 민감한 정보가 담겨 있는 파일은 반드시 .gitignore 파일에 명시



## [실습5] .gitignore 활용



로컬 환경에 password.txt 파일을 하나 생성하고,  
리모트 환경에는 계속해서 password.txt 파일이 업로드 되지 않도록 만들어 보세요.



The screenshot shows the GitHub repository settings page for 'workoutplz / first'. The 'Settings' tab is selected. A notification at the top states 'plz-workout has been added as a collaborator on the repository.' On the left, the 'Access' section is expanded, showing 'Collaborators' is selected. The main area displays 'Who has access' information: 'Public repository' (Manage button) and 'DIRECT ACCESS' (Manage button) which shows '1 user has access to this repository. 0 collaborators, 1 invitation.' Below this is the 'Manage access' section with a search bar 'Find a collaborator...' and a list containing 'plz-workout' (Pending Invite, trash bin icon). Navigation buttons 'Previous' and 'Next' are at the bottom.

## Collaborator(팀원) 등록방법

- Repository -> Settings  
-> Collaborators에서 타 수강생 초대
- 초대 받은 사람이 수락을 해야 함

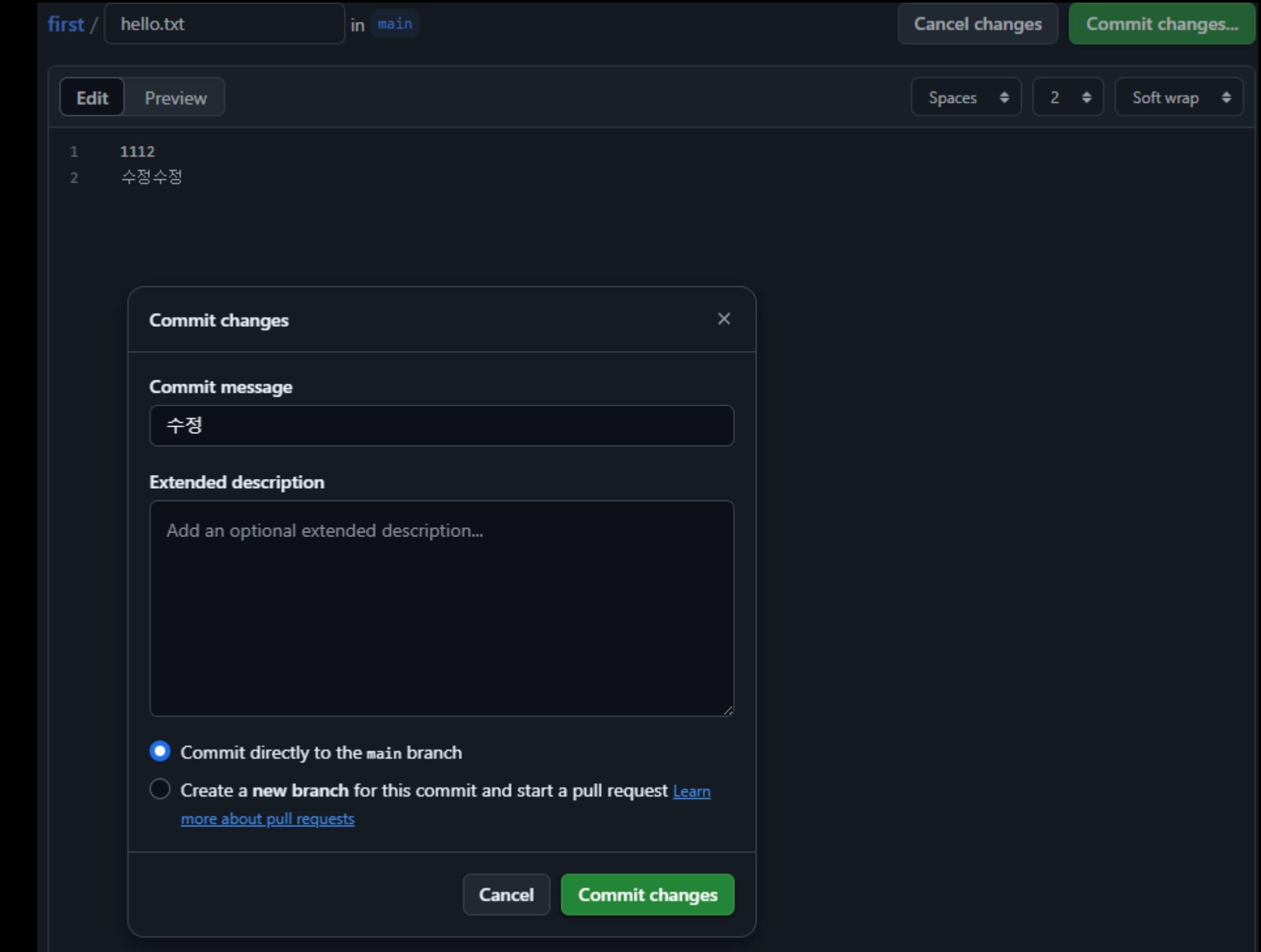
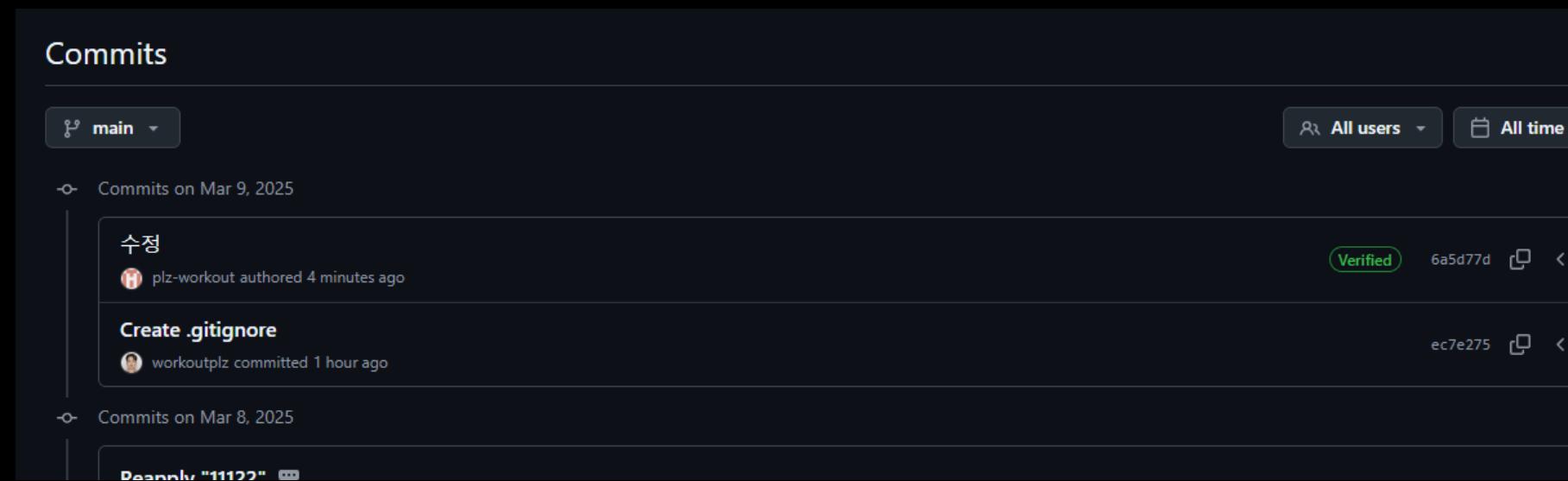
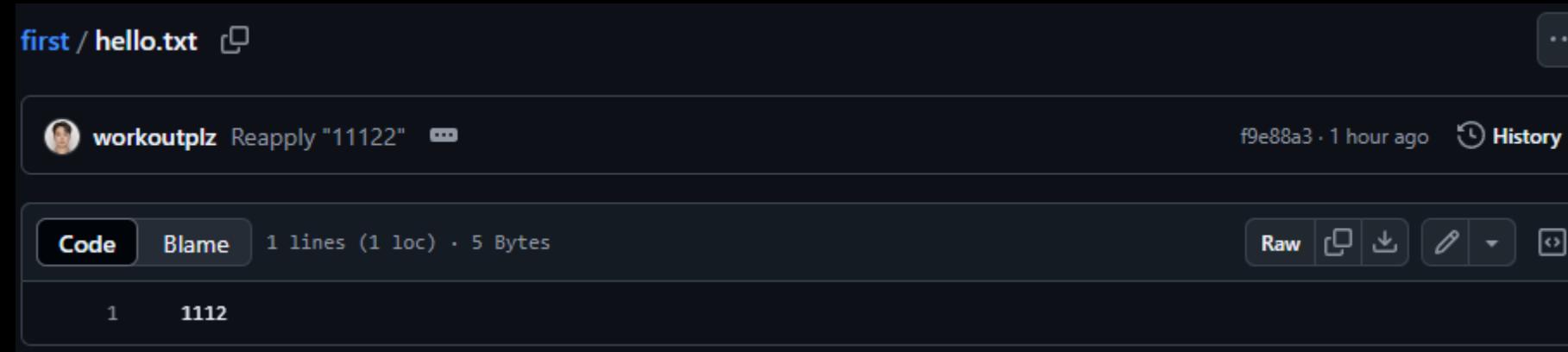


# 다른 사람과 협업하기(1) - Collaborator

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용



## Collaborator 등록 시

- Repository를 직접 수정 가능  
(연필 모양 버튼)
- 웹 상에서 바로 commit, push 가능

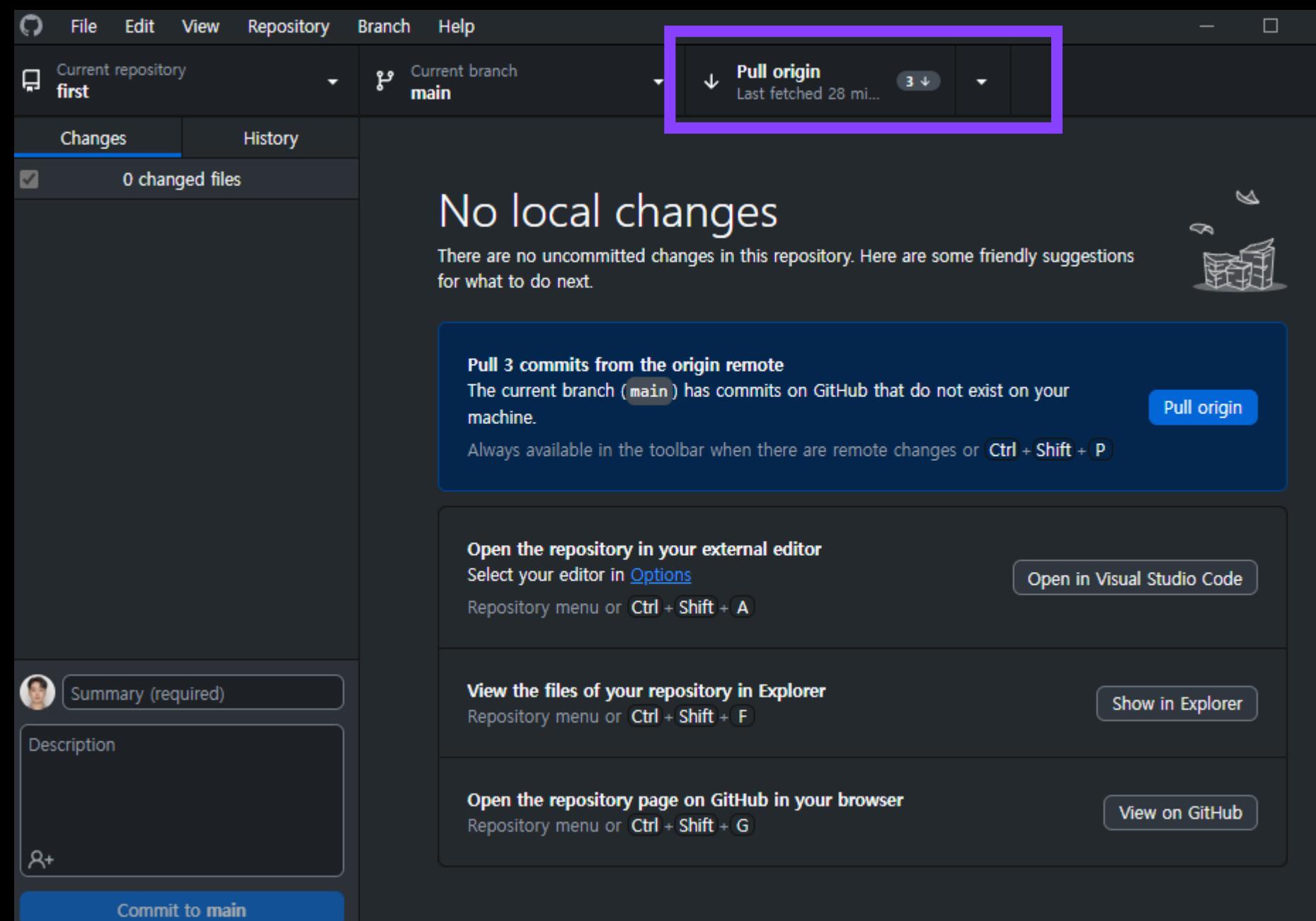


# 다른 사람과 협업하기(1) - Collaborator

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용



최신 버전 유지를 위해서,  
Remote Repository의 변경 사항 로컬 적용 필수  
→ Pull 필수

최신 버전이 유지되지 않을 때,  
충돌이 발생하여 원활한 협업 불가능

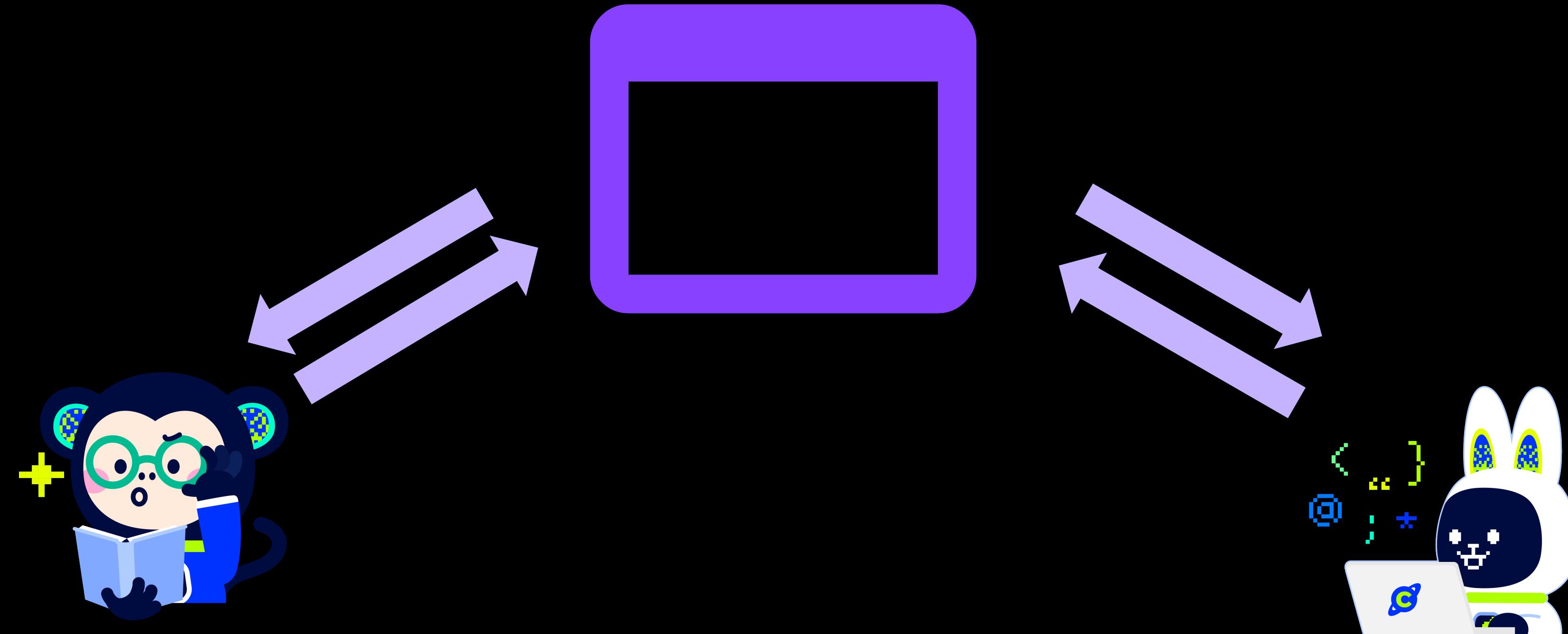


# Collaborator의 협업 개념

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용



같은 Remote Repository를 직접 수정

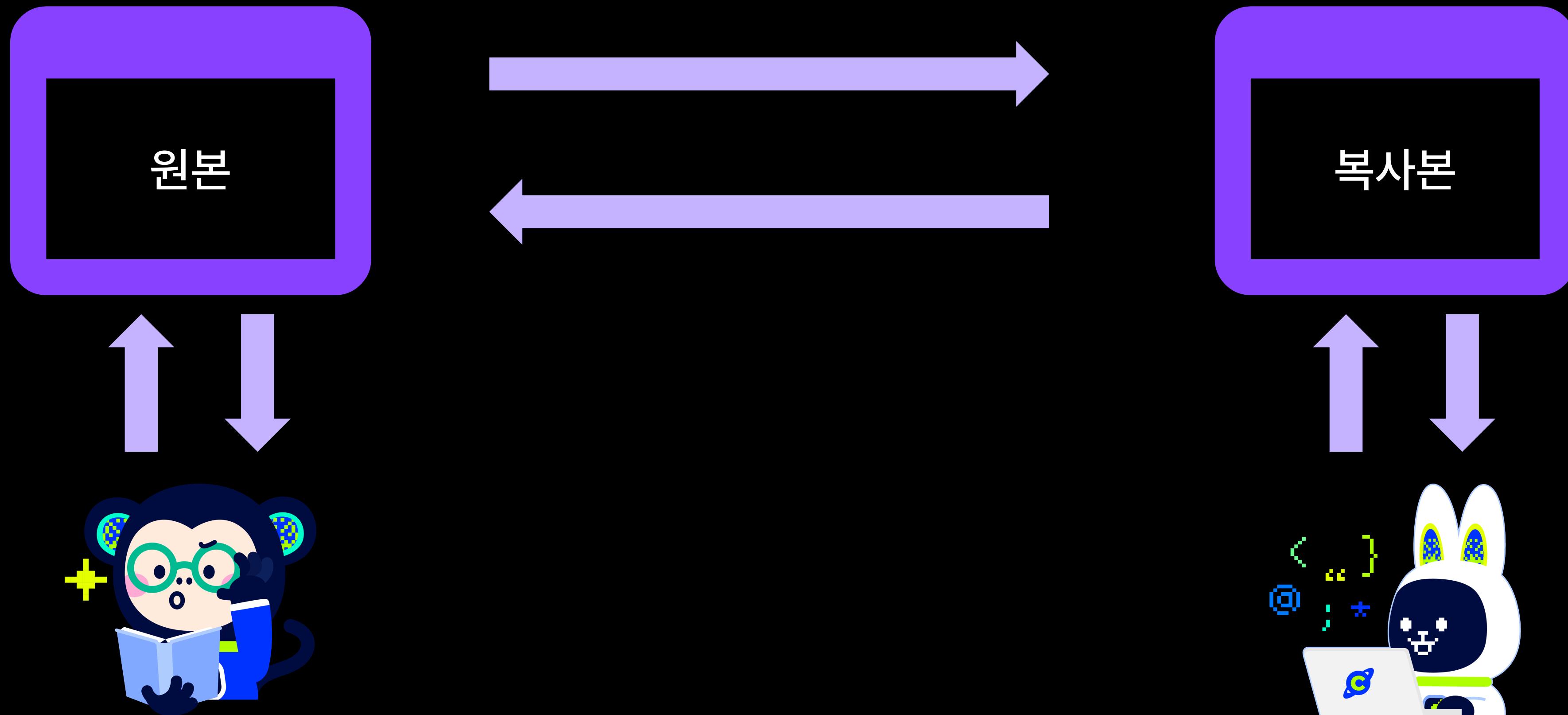


## 다른 사람과 협업하기(2) – Pull Request

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용



내용은 같으나 따로 존재하는 **Remote Repository**를 각자 직접 수정, 통합



# 다른 사람과 협업하기(2) – Pull Request

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용

The screenshot shows the GitHub fork creation interface. At the top, there are buttons for 'Pin', 'Unwatch', 'Fork' (which is highlighted with a purple box), and 'Star'. Below this, the repository details show 'workoutplz / first'. The main area is titled 'Create a new fork' and explains what a fork is. It has fields for 'Owner' (set to 'plz-workout') and 'Repository name' (set to 'first'). There's also a 'Description (optional)' field and a checkbox for 'Copy the main branch only'. A note at the bottom says 'You are creating a fork in your personal account.' A green 'Create fork' button is at the bottom right.

The screenshot shows the GitHub repository page for 'plz-workout / first'. The 'Code' tab is selected. The repository was forked from 'workoutplz/first'. The code view shows a single commit by 'plz-workout' that creates a '.gitignore' file. The README section is empty, with a green 'Add a README' button. The right sidebar shows repository statistics: 0 stars, 0 watching, and 0 forks. It also includes sections for 'About', 'Activity', 'Releases', and 'Packages'.

Fork = 내용이 동일한 Repository를 내 계정으로 복사



# 다른 사람과 협업하기(2) – Pull Request

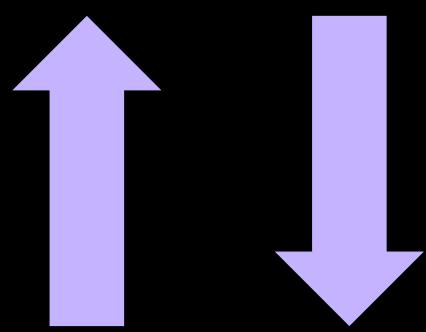
테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용

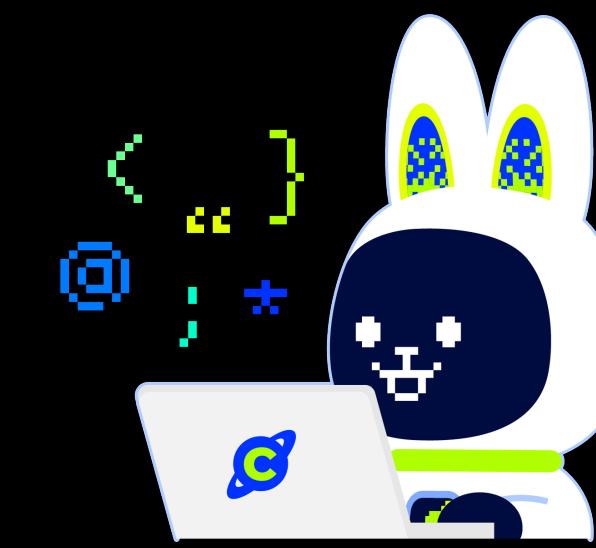
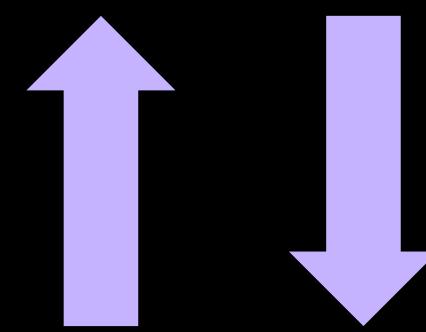
원본

```
print('hi')
```



복사본

```
print('hi')  
print('bye')
```



각각의 개발자는 본인의 Repository를 수정하므로 서로의 수정 내용이 반영되지 않음



# 다른 사람과 협업하기(2) – Pull Request

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용

Commits

main · All users · All time

- o Commits on Mar 9, 2025
  - 수정 plz-workout authored 4 minutes ago Verified 6a5d77d ⌂ ↗
  - Create .gitignore workoutplz committed 1 hour ago ec7e275 ⌂ ↗
- o Commits on Mar 8, 2025

People "11122" ⚙

기존 Repository

Commits

main · All users · All time

- o Commits on Mar 9, 2025
  - Update hello.txt plz-workout authored 1 minute ago Verified 057313a ⌂ ↗
  - 수정 plz-workout authored 4 minutes ago 6a5d77d ⌂ ↗
  - Create .gitignore workoutplz committed 1 hour ago ec7e275 ⌂ ↗

Forked Repository

Fork 한 Repository에서 수정을 해도 원래의 Repository에는 반영이 안되는 모습

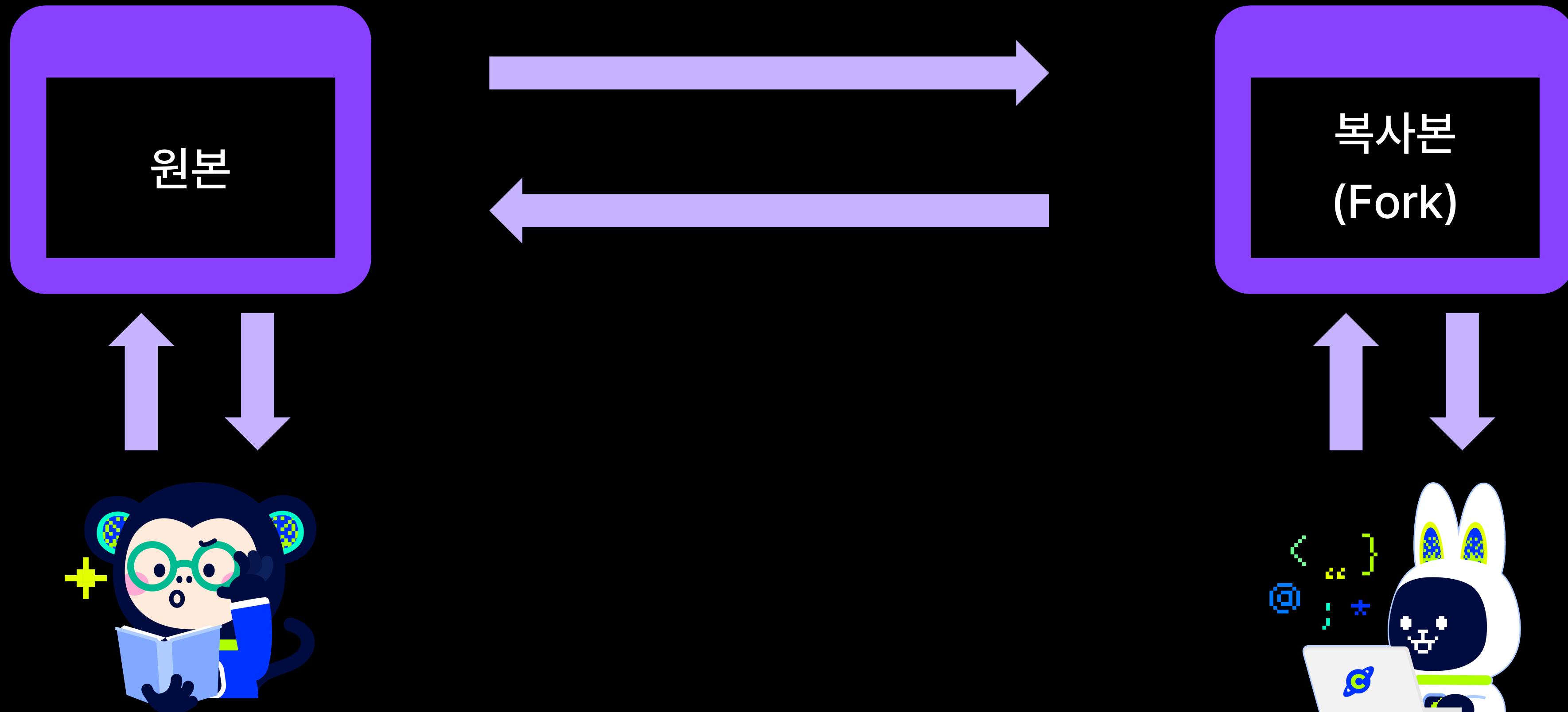


# 다른 사람과 협업하기(2) – Pull Request

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용



그리고 이 둘을 동기화 시켜 주는 것이 Pull Request와 Merge

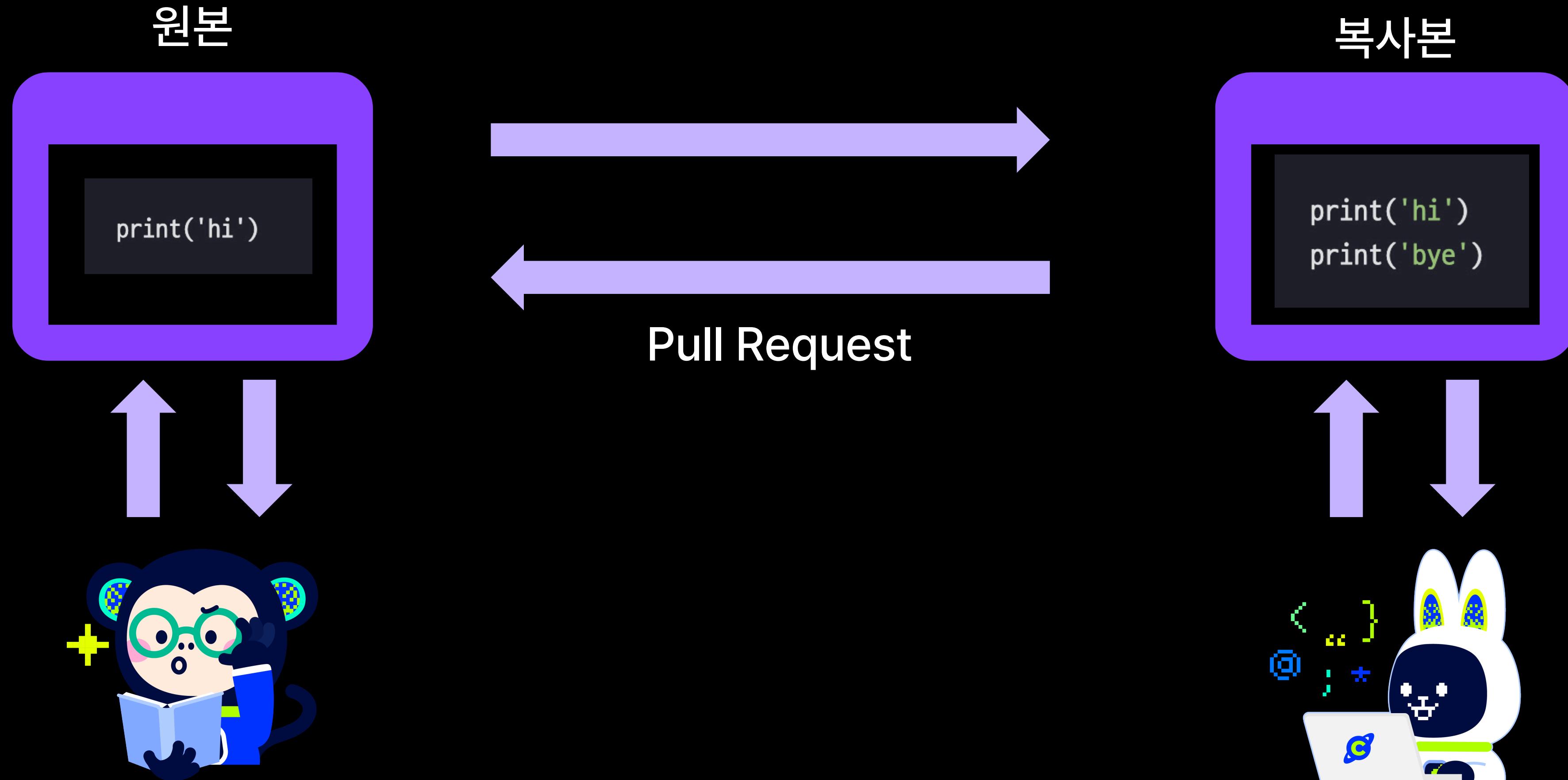


# 다른 사람과 협업하기(2) – Pull Request

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용



복사본에서 통합 요청을 하는 것 = Pull Request (PR)

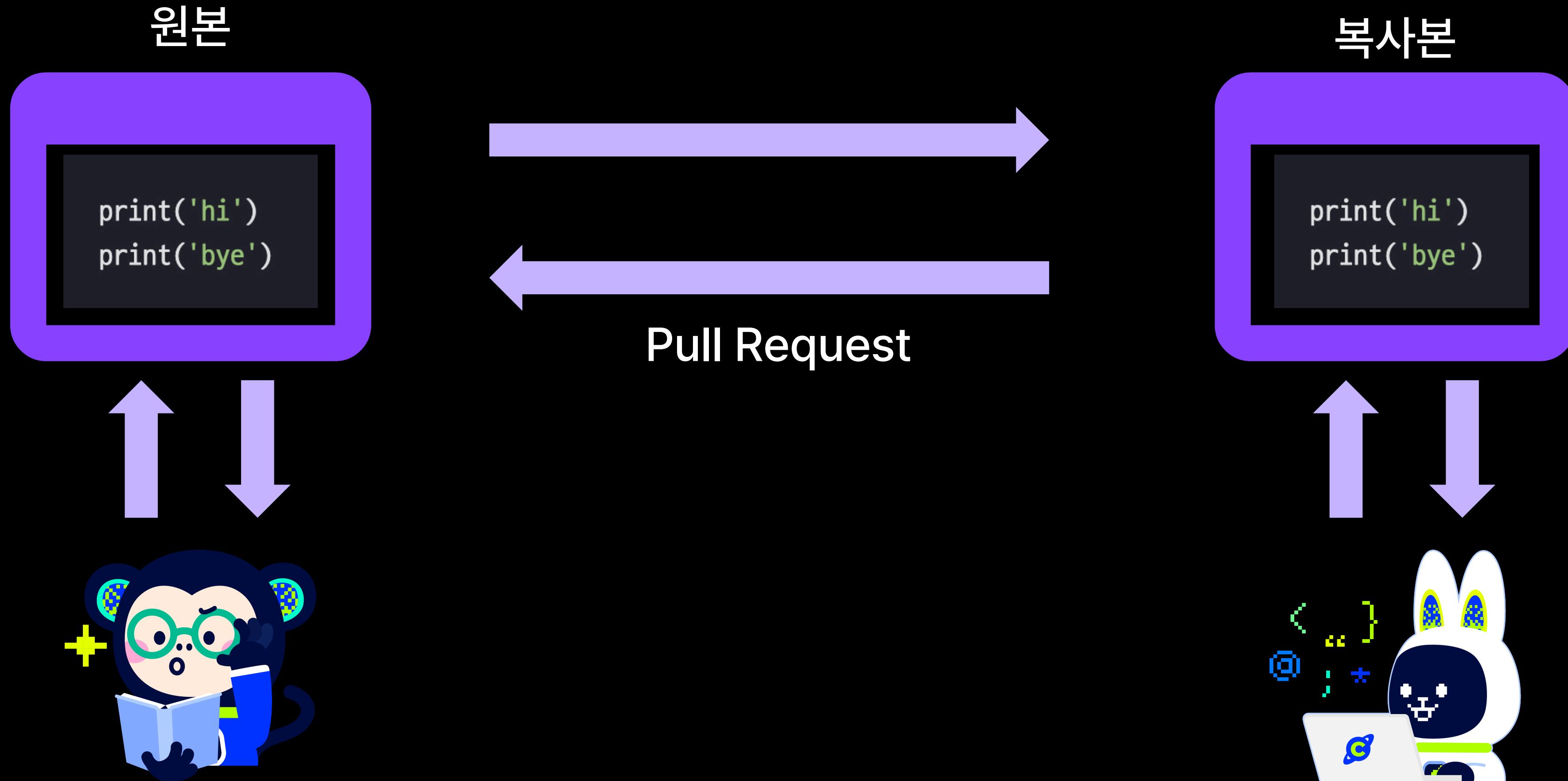


# 다른 사람과 협업하기(2) – Pull Request

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용



Pull Request를 승인 시 Merge 되어 원본 또한 바뀜



# 다른 사람과 협업하기(2) – Pull Request

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용

The screenshot shows a GitHub interface for comparing changes between two branches. At the top, it says "Comparing changes" and "base repository: workoutplz/first" and "head repository: plz-workout/first". It indicates that the branches can be automatically merged. Below this, there's a "Create pull request" button. The main area shows a single commit from "plz-workout" on March 9, 2025, which updated the file "hello.txt". The commit message is "Update hello.txt" and it was authored by "plz-workout". The commit has 1 addition and 0 deletions. At the bottom, there's a diff view for the "hello.txt" file, showing the change: "1 1 1112" and "2 2 수정수정".

commit을 기준으로  
Pull Request를 생성할 수 있음



# 다른 사람과 협업하기(2) – Pull Request

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks. [Learn more about diff comparisons here.](#)

base repository: workoutplz/first base: main head repository: plz-workout/first compare: main Able to merge. These branches can be automatically merged.

Add a title  
Update hello.txt

Add a description

Write Preview

hello.txt에 고칠 것이 있어서 조금 수정했습니다.

Markdown is supported Paste, drop, or click to add files

Allow edits by maintainers [Create pull request](#)

① Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

-o 1 commit 1 file changed 1 contributor

Commits on Mar 9, 2025

Update hello.txt plz-workout authored 1 minute ago

Showing 1 changed file with 1 addition and 0 deletions.

1 file changed

diff --git a/hello.txt b/hello.txt  
@@ -1,2 +1,3 @@  
1 1 1112  
2 2 수정수정  
3 + fork 이후 수정

- 어떤 변경이 있는지에 대한 설명을 작성
- 리뷰어가 필요한 경우 별도 지정



# 다른 사람과 협업하기(2) – Pull Request

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용

The screenshot shows a GitHub pull request page for a repository named 'first'. The pull request is titled 'Update hello.txt #1' and is currently open. A comment from 'plz-workout' is visible, stating 'hello.txt에 고칠 것이 있어서 조금 수정했습니다.' (I made some changes to hello.txt because there was something to fix). The pull request has one commit and one file changed. The status bar indicates it is 'Verified' and has a timestamp of '057313a'. The right sidebar contains sections for Reviewers (workoutplz), Assignees (None yet), Labels (None yet), Projects (None yet), Milestone (No milestone), Development (Successfully merging this pull request may close these issues. None yet), Notifications (Customize, Unsubscribe), and a lock conversation button.

The screenshot shows the GitHub pull requests index for the same repository. It lists one open pull request: '#1 opened now by plz-workout' with the title 'Update hello.txt'. The interface includes filters for 'is:pr is:open', a 'New pull request' button, and a 'ProTip!' for filtering by default branch with 'base:main'. The right sidebar shows general repository statistics: 1 Open pull request and 0 Closed pull requests.

Pull Request가 오면 원작자는 해당 내용을 리뷰 후  
Commit 내용을 Merge 할 수 있음



# 다른 사람과 협업하기(2) – Pull Request

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용

The screenshot shows a GitHub commit history for a repository named 'plz-workout/main'. The commits are listed in chronological order from top to bottom:

- Merge pull request #1 from plz-workout/main** (Verified, 7dda59a) - authored by 'workoutplz' now.
- Update hello.txt** (Verified, 057313a) - authored by 'plz-workout' 4 minutes ago.
- 수정** (Verified, 6a5d77d) - authored by 'plz-workout' 8 minutes ago.
- Create .gitignore** (ec7e275) - committed by 'workoutplz' 1 hour ago.

Below this, there is a section for commits on March 8, 2025:

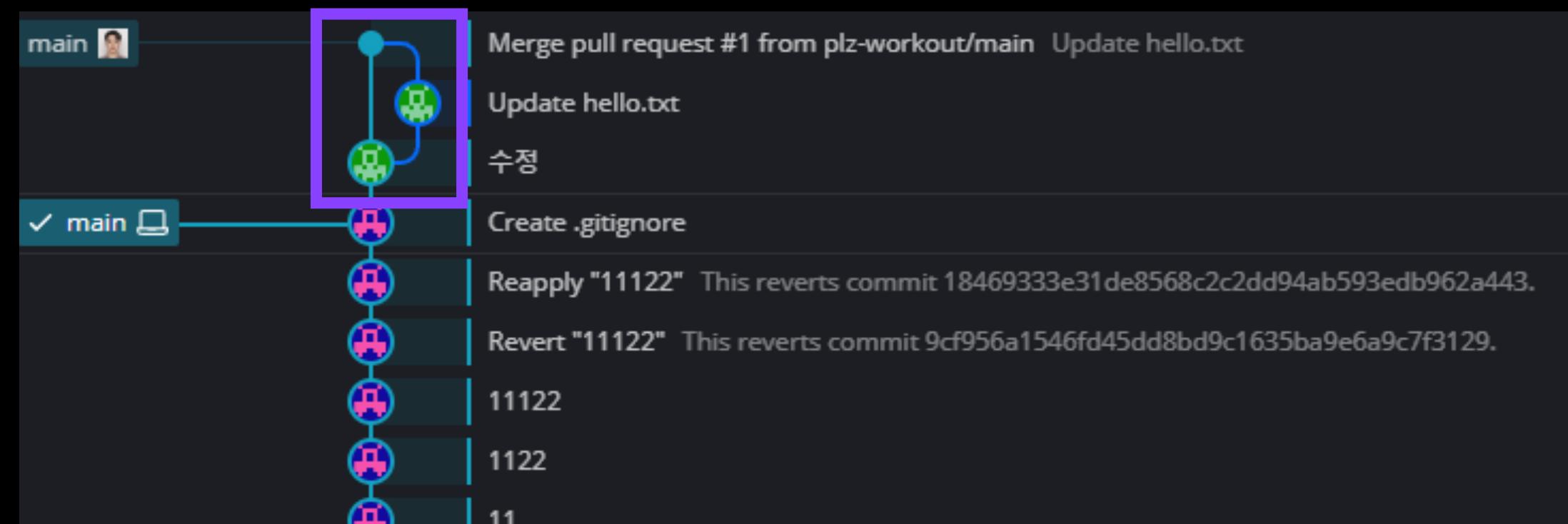
- Reapply "11122"** (f9e88a3) - committed by 'workoutplz' 1 hour ago.

A purple rectangular box highlights the first three commits (the merge commit and two regular commits), indicating that they were part of a single pull request merge.

Merge가 되면 정상적으로 Commit 내용들이 모두 정상적으로 반영 됨



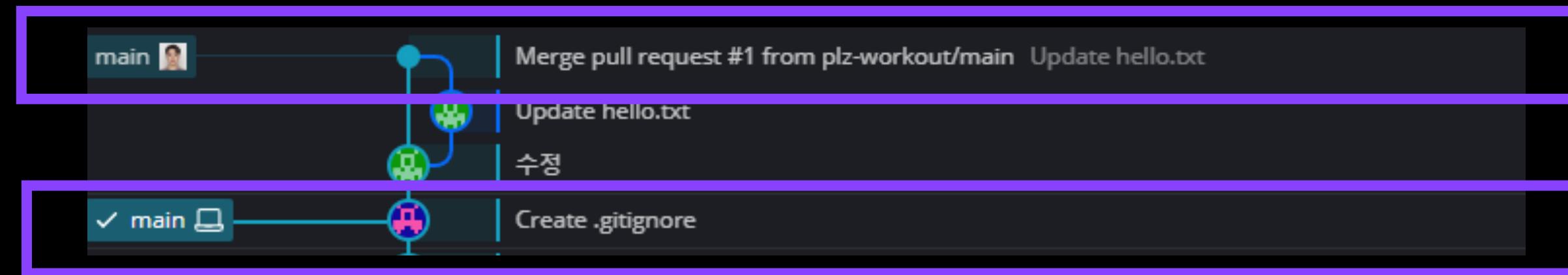
## Pull Request & Merge 시점



변경 내용을 시각화 하면 위와 같이 나타낼 수 있음  
(위 프로그램은 GitKraken)



## 현재 Remote Repository의 버전



## 현재 Local Repository의 버전

Merge 이후 Remote Repository와 Local Repository의 버전이 차이가 나면,  
이는 곧 버전 충돌로 이어질 수 있으므로 반드시 동기화(Pull) 필요

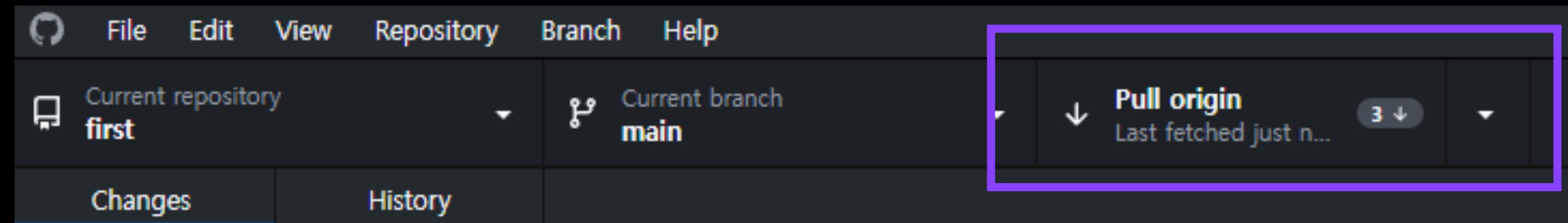


# 다른 사람과 협업하기(2) – Pull Request

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용



코드 작업시 Github Desktop에서 동기화(Pull) 할 것이 있는지  
반드시 먼저 확인하고, Pull 이후 작업을 해야 함.

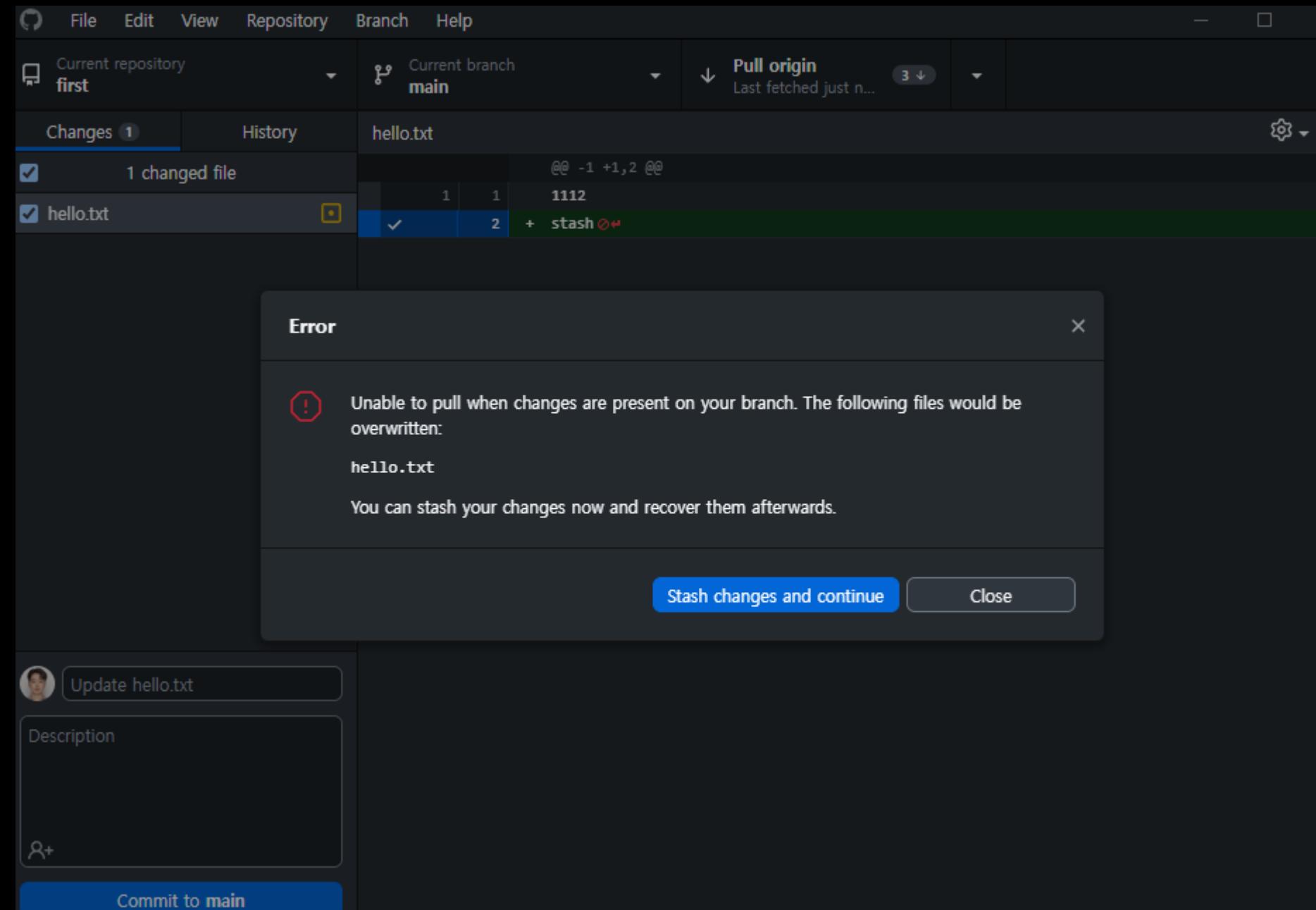


# 다른 사람과 협업하기(2) – Pull Request

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용



로컬의 버전과 리모트의 버전 차이가 있을 경우,  
**Conflict**가 발생

이 때 로컬의 수정 내용을  
임시 저장하는 기능이 '**stash**'

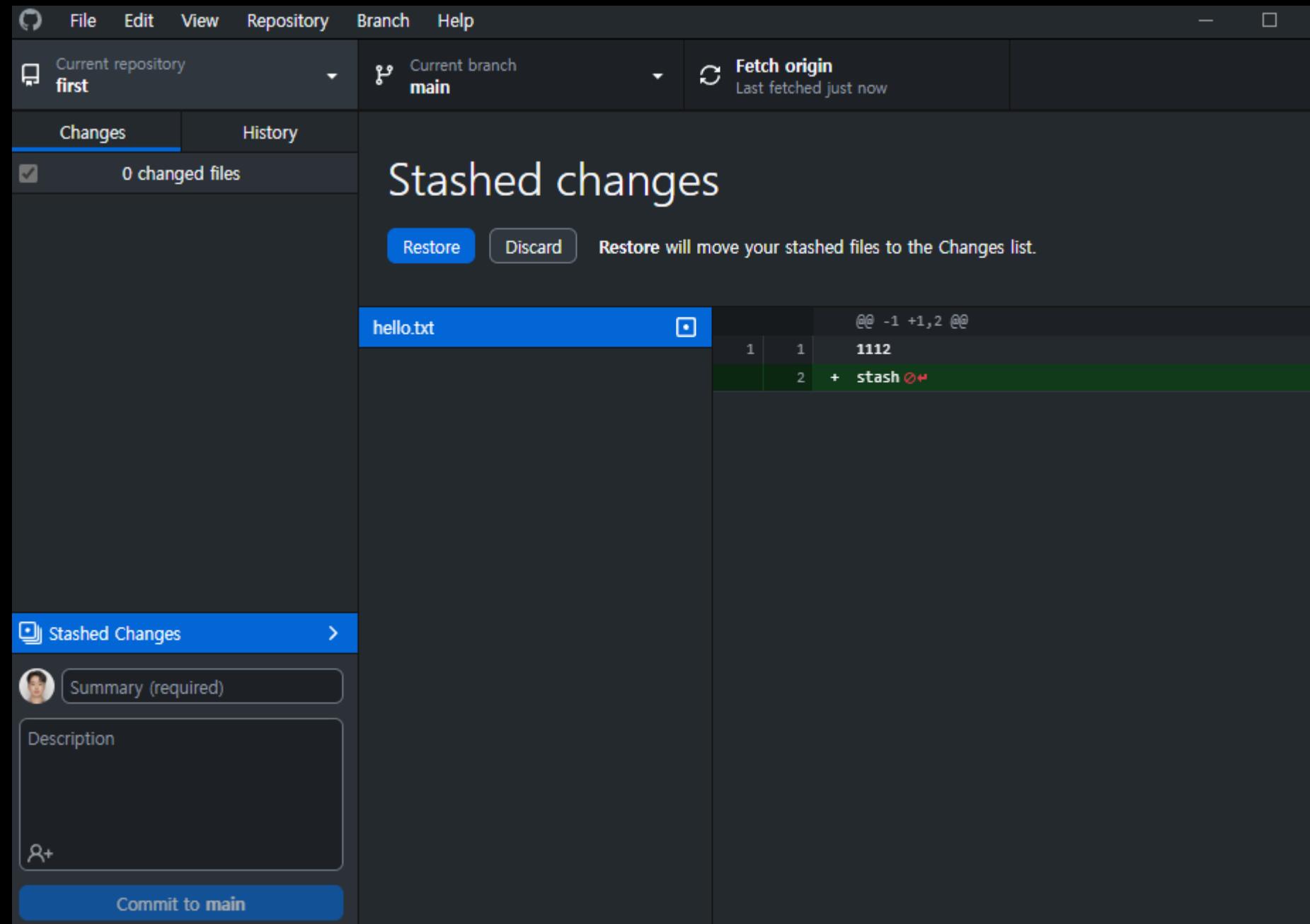


# 다른 사람과 협업하기(2) – Pull Request

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용



- 로컬의 변경 내용을 임시 저장(stash)하고, Remote와 동기화
- 동기화 이후 Stashed changes를 확인하여 이어서 작업 가능

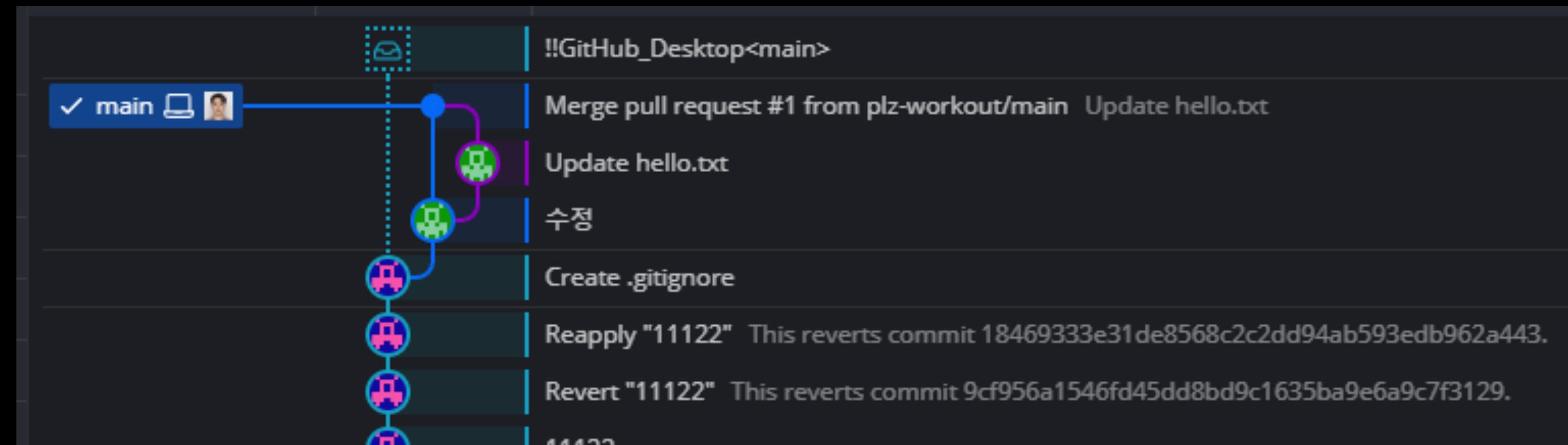


# 다른 사람과 협업하기(2) – Pull Request

테스트 자동화 기초

Git과 Jenkins를 활용한  
CI/CD 파이프라인 구축

1. Git 개념 및 활용



Remote Repository와 Local Repository가 동기화되고,  
작업 내용 중 일부가 Stash 된 모습



## Pull Request(PR) & Merge 시 주의점

- PR 작성 시 내용을 명확히 전달하는 것이 중요
- 적절한 코드 리뷰를 통해 무분별한 수정과 Merge를 방지

## Stash 사용시 주의점

- 편리하다고 남발하면 작업 내용이 꼬일 수 있으니, 남발해서는 안됨
- Stash는 어디까지나 임시 저장용이므로 중요한 변경 사항은 반드시 commit
- Stash가 목적은 아님. Stash 이후에도 충돌이 날 수 있으므로 Pull을 습관화
- 현재 branch의 변경 사항만 stash 됨 (branch 개념은 이후 학습 예정)



# [실습6] PR & Merge

Update hello.txt #1

Merged workoutplz merged 1 commit into `workoutplz:main` from `plz-workout:main` yesterday

Conversation 1 Commits 1 Checks 0 Files changed 1

plz-workout commented yesterday  
hello.txt에 고칠 것이 있어서 조금 수정했습니다.

Update hello.txt Verified 057313a

plz-workout requested a review from workoutplz yesterday

workoutplz commented yesterday  
좋네요

workoutplz merged commit `7dda59a` into `workoutplz:main` yesterday Revert

Pull request successfully merged and closed  
You're all set — the branch has been merged.

Add a comment Write Preview Add your comment here... Markdown is supported Paste, drop, or click to add files Comment

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

ProTip! Add comments to specific lines under [Files changed](#).

- 다른 수강생의 Repository에 Pull Request를 날려보세요.
- Pull Request를 받은 수강생 분들은 Comment를 남기고 Merge 해보세요.
- 최소 2번 이상의 Pull Request와 Merge를 해보세요.