

Coming  
soon













수업 준비중입니다  
잠시만 기다려주세요...

2주차 수업

# Git, GitHub

2주차

## 이런 상황 겪어보셨나요? 😅

 [경영자료분석]중간보고서_4조(수정완).pdf	
 [경영자료분석]중간보고서_4조(오타수정).pdf	
 [경영자료분석]중간보고서_4조(찐완).pdf	
 [경영자료분석]중간보고서_4조(찐찐완).pdf	
 [경영자료분석]중간보고서_4조(피드백반영).pdf	
 [경영자료분석]중간보고서_4조.pdf	

## 이런 상황 겪어보셨나요? 😓

- 모든 변경사항의 히스토리가 남아 언제든지 이전 버전으로 돌아갈 수 있음

 [경영자료분석]중간보고서\_4조(수정완).pdf 

- 여러 팀원이 동시에 작업해도 변경사항을 깔끔하게 합칠 수 있음

 [경영자료분석]중간보고서\_4조(오타수정).pdf 

- 누가, 언제, 어떤 부분을 수정했는지 명확하게 알 수 있음

 [경영자료분석]중간보고서\_4조(피드백완).pdf 

- 실수로 파일을 잘못 수정하거나 삭제해도 복구 가능

 [경영자료분석]중간보고서\_4조(피드백반영).pdf 

- ...

 [경영자료분석]중간보고서\_4조.pdf 

# Git

컴퓨터 파일의 변경사항을 추적하고,  
여러 명의 사용자들간에 해당 파일들의 작업을 조율하기 위한  
분산 버전 관리 시스템

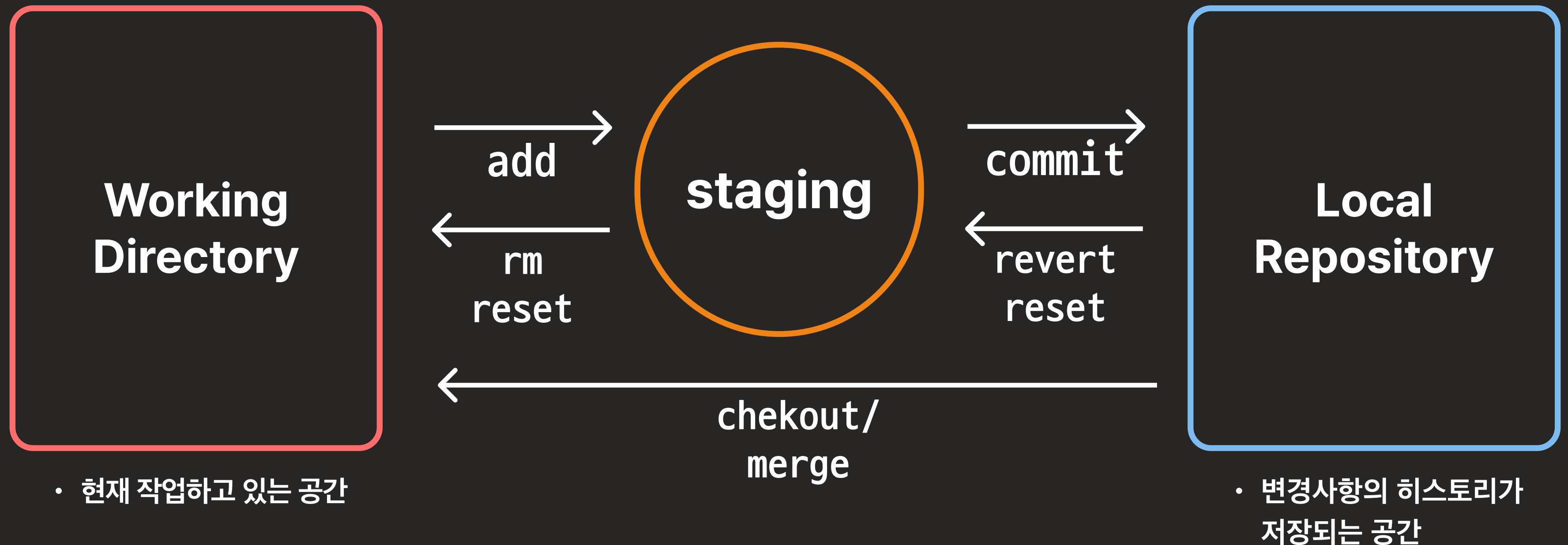


# GitHub

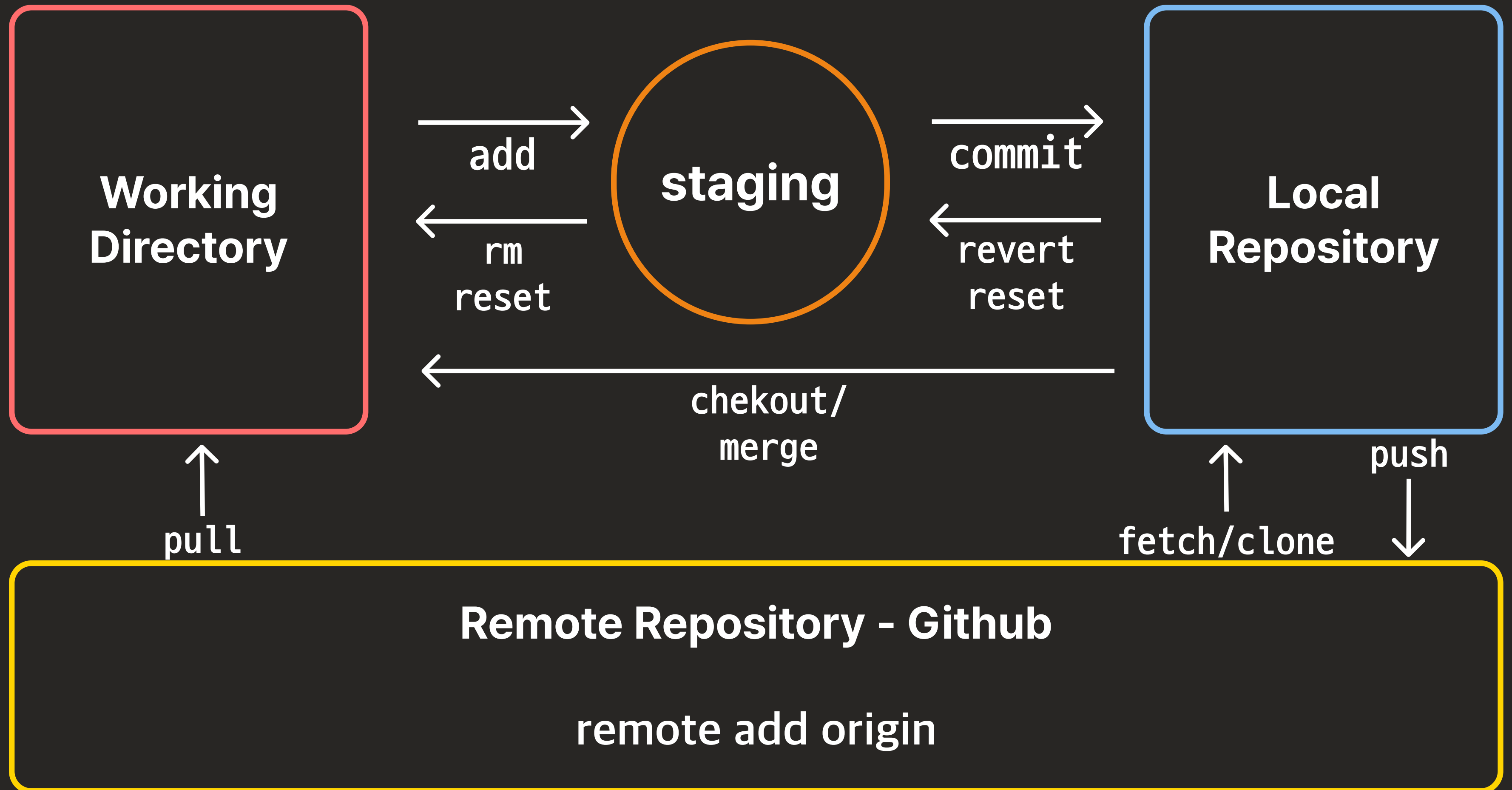
본인 컴퓨터내에서 보관중인 파일(소스코드)과,  
그 추적기록(깃)을 인터넷 서버 상에서 보관하게 해 주는 웹 호스팅 서비스



# Git의 기본 구조



# Git의 기본 구조





# 깃 초기설정하기(init)

- 터미널 프로그램 (Git Bash) 에서 아래 명령어 실행



- `git config --global user.name "(본인 이름)"`
- `git config --global user.email "(본인 이메일)"`

- 아래 명령어로 확인



- `git config --global user.name`
- `git config --global user.email`

```
thdrk@gaeun MINGW64
• $ git config --global user.name
gn-ioeo

thdrk@gaeun MINGW64
• $ git config --global user.email
thdrkdms1030@hanyang.ac.kr
```

# Git 관리 시작

- VS Code 터미널에서 아래 명령어를 입력



```
git init
```

```
thdrk@gaeun MINGW64 ~/OneDrive/바탕 화면/likelion13 (master)
$ git init
Reinitialized existing Git repository in C:/Users/thdrk/OneDrive/바탕 화면/likelion13/.git/
```

- 아래 명령어로 커밋



```
git add . (현 디렉토리에 있는 모든 파일 담기)
```

```
git add 1.html (특정 파일 담기)
```

```
git commit -m "first commit"
```

```
git status (변경사항 확인)
```

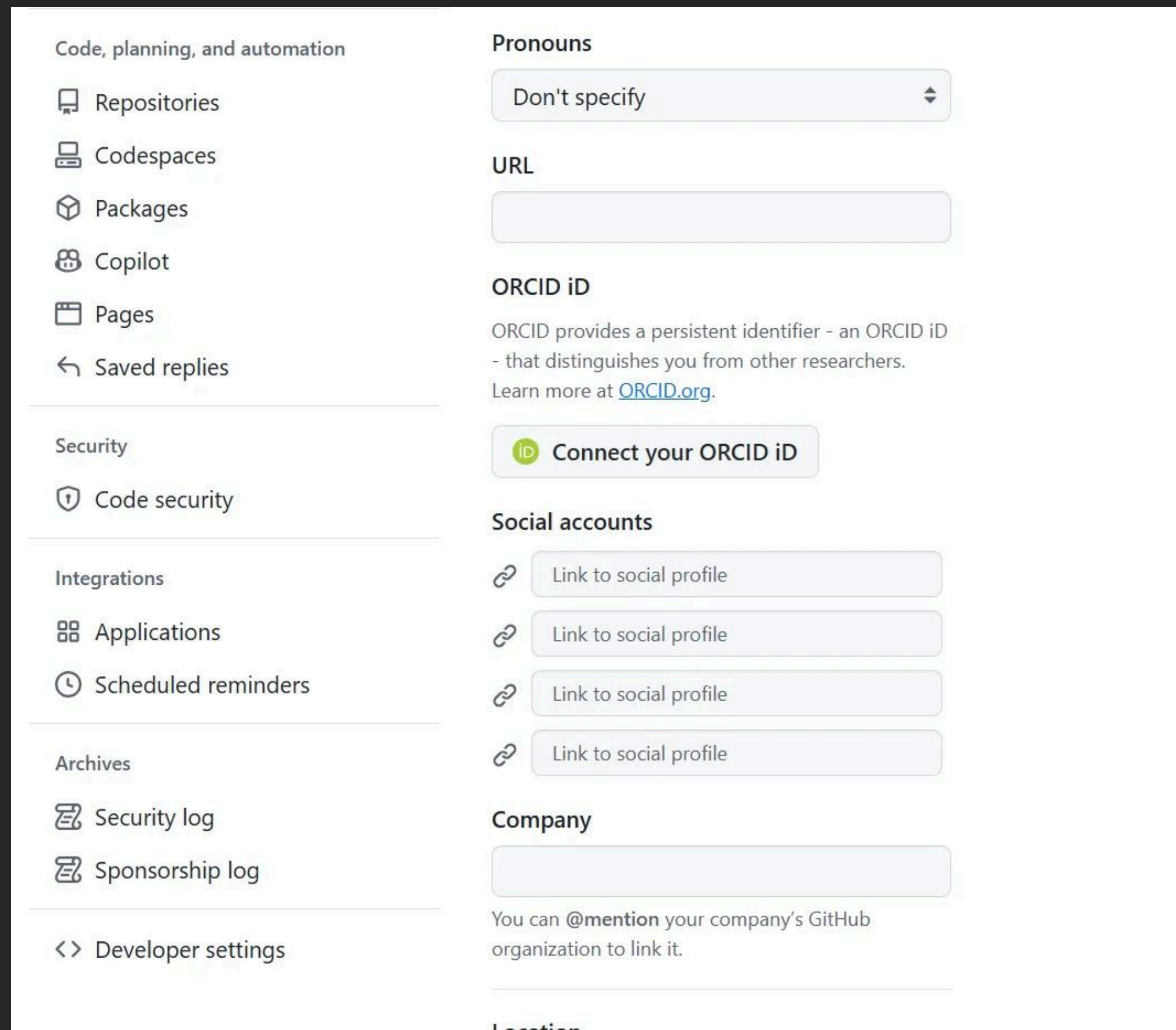
# Github 사용해보기 : 접속용 토큰 만들기

- 본인 프로필을 눌러 Settings 에 들어갑니다.

The screenshot shows the GitHub Dashboard for user 'gn-ioeo'. The left sidebar lists 'Top repositories' including 'meomeoknyang/client', 'HYU-ERICA-Likelion/Front', 'Stagecontainer/Client', 'sorongosdev/apple\_kiosk', 'gn-ioeo/2024STB\_songgaeun', 'moyeothon/UniVerse\_Front', and 'gn-ioeo/study-nextjs'. The main content area features a 'Join GitHub Education!' banner with a 'Join GitHub Education' button. Below the banner is an 'Ask Copilot' input field and two suggested queries: 'What are Python decorators?' and 'Pull requests in microsoft/vscode'. The right sidebar shows the user's profile 'gn-ioeo' with a dropdown menu containing options like 'Set status', 'Your profile', 'Your repositories', 'Your Copilot', 'Your projects', 'Your stars', 'Your gists', 'Your organizations', 'Your enterprises', 'Your sponsors', 'Try Enterprise' (marked 'Free'), 'Feature preview', 'Settings' (highlighted), 'GitHub Website', 'GitHub Docs', 'GitHub Support', 'GitHub Community', and 'Sign out'.

# Github 사용해보기 : 접속용 토큰 만들기

- 왼쪽 아래 Developer settings 클릭



# Github 사용해보기 : 접속용 토큰 만들기

- Personal access token > Tokens(classic)

 GitHub Apps

 OAuth Apps

 Personal access tokens ^

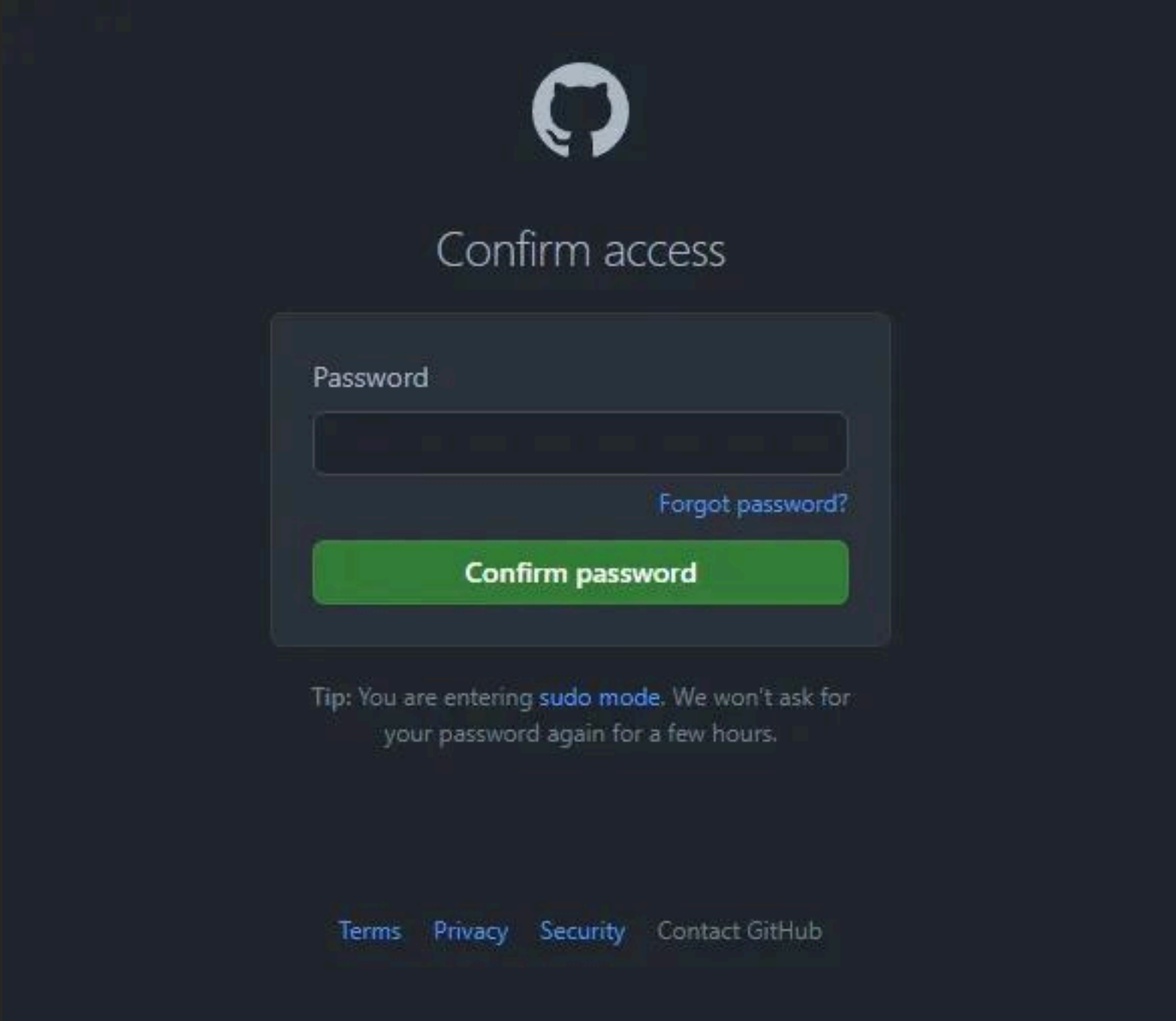
Fine-grained tokens

Preview

 Tokens (classic)

# Github 사용해보기 : 접속용 토큰 만들기

- 비밀번호 입력

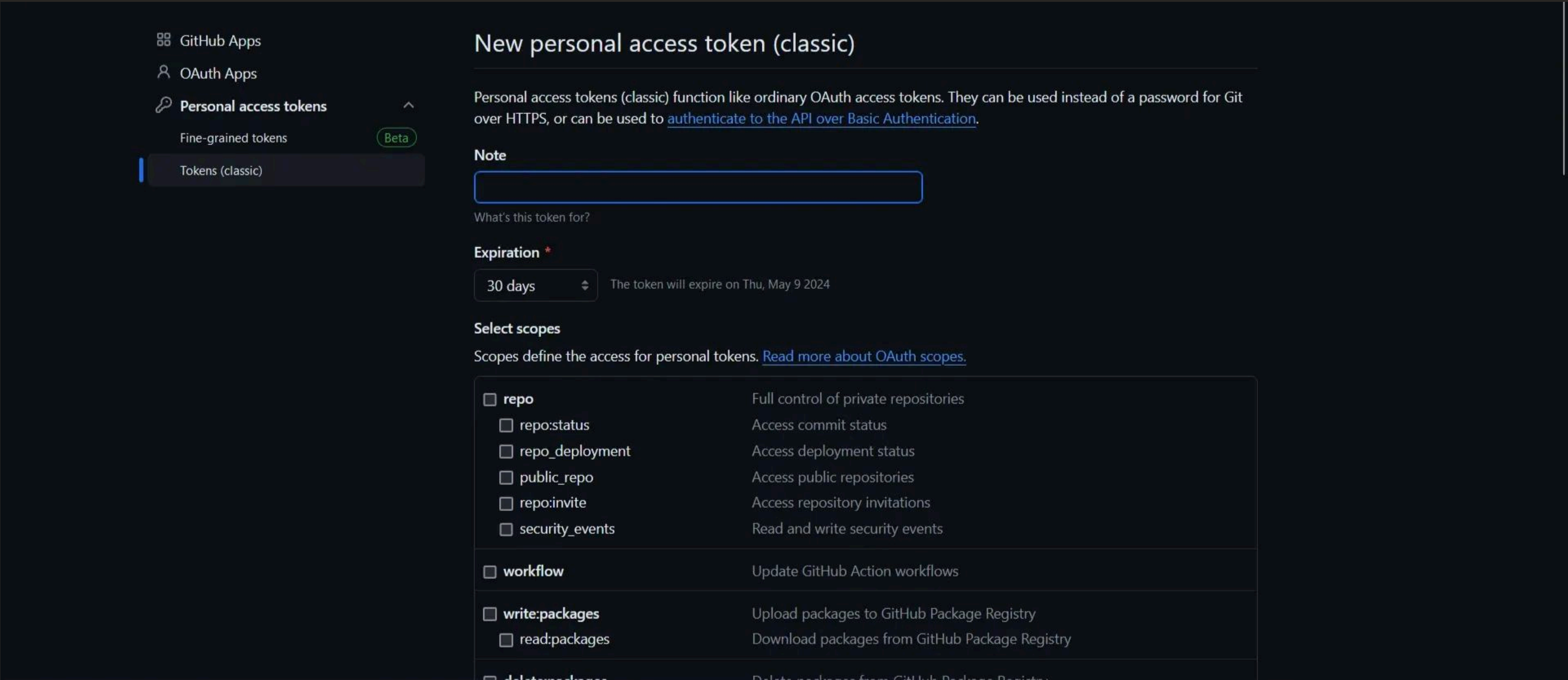


The screenshot shows the GitHub 'Confirm access' interface. At the top is the GitHub logo. Below it, the text 'Confirm access' is centered. A form box contains a 'Password' label, a password input field with a strength indicator (showing 100% strength), a 'Forgot password?' link, and a green 'Confirm password' button. Below the form, a tip message states: 'Tip: You are entering `sudo mode`. We won't ask for your password again for a few hours.' At the bottom, there are links for 'Terms', 'Privacy', 'Security', and 'Contact GitHub'.



# Github 사용해보기 : 접속용 토큰 만들기

- 이름은 아무거나 적어주시고, Expiration은 제일 길게 설정해주세요.



# Github 사용해보기 : 접속용 토큰 만들기

- repo 체크를 합니다. (꼭!!!)

## New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

### Note

What's this token for?

### Expiration \*

30 days



The token will expire on Thu, May 9 2024

### Select scopes

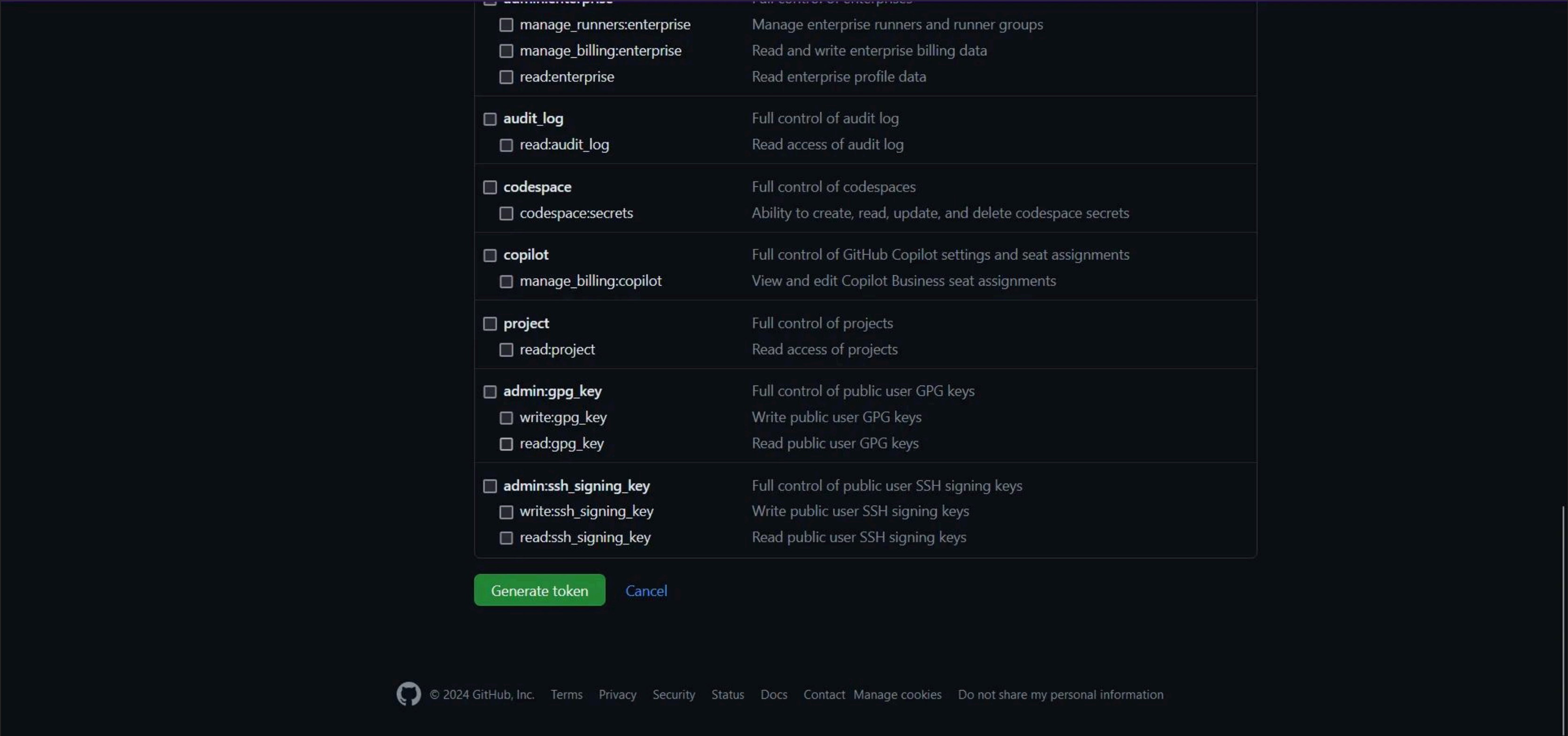
Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> <b>repo</b>	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> <b>workflow</b>	Update GitHub Action workflows
<input type="checkbox"/> <b>write:packages</b>	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> <b>admin:org</b>	Administrative actions on GitHub organization



# Github 사용해보기 : 접속용 토큰 만들기


- 초록 버튼인 Generate token 클릭



# Github 사용해보기 : 접속용 토큰 만들기

- ghp\_@@@@@ 이 key이므로 복사해서 노션이나 카톡에 보내놓기! (페이지 나가는 순간 다신 볼 수 없음)
- 절~~~~~대 잃어버리지마세요

laptop'.



가은 (gn-ioeo)

Your personal account [Switch settings context](#)

Public profile

Account

Appearance

Accessibility

Notifications

Access

Billing and plans


Emails

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication keys



SSH

MyLaptop

SHA256:99Qdvf+gWvANACuE4B+VHrLWMTicnqvCjGQqzqXutag

Added on Feb 10, 2025

Never used — Read/write

Delete

Check out our guide to [connecting to GitHub using SSH keys](#) or troubleshoot [common SSH problems](#).

# 리포지토리 만들기 : Organization을 통한...

- 이메일로 온 초대장을 받아주세요!

The screenshot shows the GitHub organization page for 'erica-likelion-fe'. The page layout includes a top navigation bar with the organization name, a search bar, and various icons. Below the navigation bar, there's a header section with the organization's profile picture (a black square with an orange logo) and the name 'erica-likelion-fe'. A 'Follow' button is visible on the right. The main content area displays the 'README.md' file, which contains the following information:

**ERICA LIKELION-FE**

13기 FE 운영진

김세현	송가은	이가은

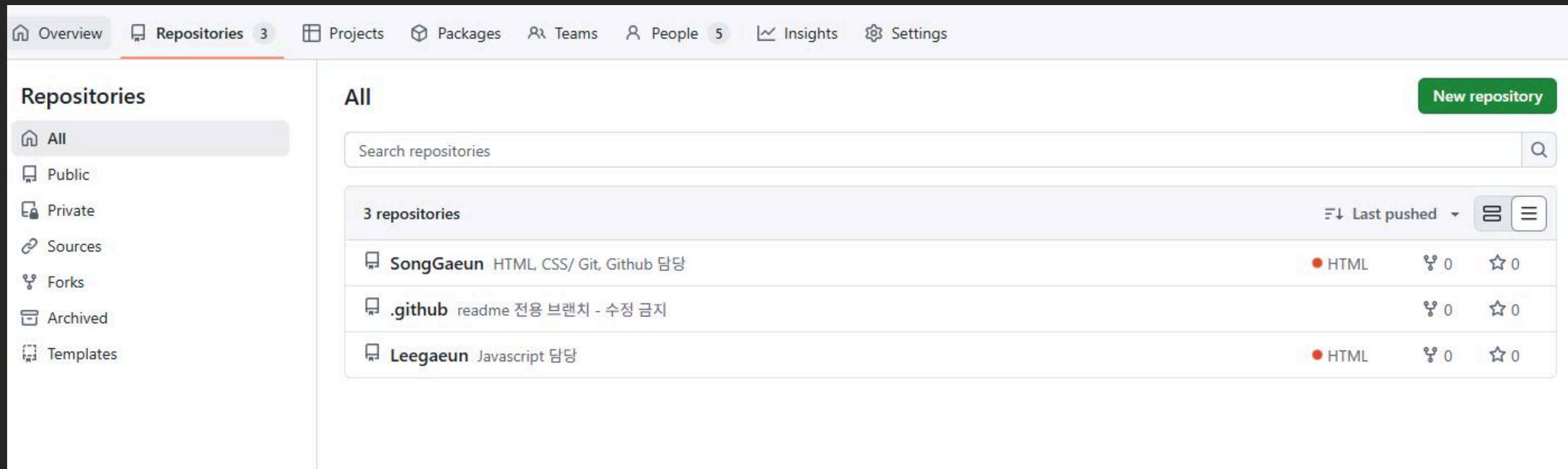
13기 아기사자

환영합니다! 🚀  
 이곳은 멋쟁이사자처럼 at 한양대학교 ERICA의 공식 GitHub FE Organization입니다.  
 동아리원들이 진행하는 스터디 자료/과제 등을 공유하는 공간입니다.

On the right side of the page, there are sections for 'View as: Public', 'Discussions', 'People', and 'Top languages'. The 'People' section shows five profile pictures of the organization members.

# 리포지토리 만들기

- 오른쪽에 초록색 버튼(new repository)을 눌러주세요





# 리포지토리 만들기

- 왼쪽에 초록색 버튼을 된 new 를 눌러주세요

The screenshot shows the GitHub Dashboard interface. On the left sidebar, under the user profile 'gn-ioeo', there is a 'Top repositories' section with a 'New' button (a green button with a computer icon and the word 'New'). Below this is a search bar 'Find a repository...' and a list of repositories: 'meomeoknyang/client', 'HYU-ERICA-Likelion/Front', 'Stagecontainer/Client', 'sorongosdev/apple\_kiosk', 'gn-ioeo/2024STB\_songgaeun', 'moyeothon/UniVerse\_Front', and 'gn-ioeo/study-nextjs'. A 'Show more' link is at the bottom of the list.

On the right side of the dashboard, a modal titled 'Join GitHub Education!' is displayed. The modal text says: 'GitHub Education opens doors to new skills, tools, and a collaborative community eager to drive innovation. Join us and build a foundation for your future in technology.' Below the text, it lists 'Free and discounted services for teachers and students.' with three service cards: 'Copilot' (Turn natural language prompts into coding suggestions), 'Heroku' (Build, run, and operate applications entirely in the cloud), and 'Microsoft Azure' (Access to Microsoft Azure cloud services and learning resources). At the bottom of the modal is a green button that says 'Join GitHub Education'.

# 리포지토리 만들기


- 원하는대로 이름을 짓고 나머지는 위 사진과 같이 맞춰주세요!
- 다 하셨으면 Create repository 버튼 누르기!

## 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 (\*).*

Owner \*


erica-likelion-fe

Repository name \*


SongGaeun2

✔ SongGaeun2 is available.

Great repository names are short and memorable. Need inspiration? How about **solid-fishstick** ?


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.](#)

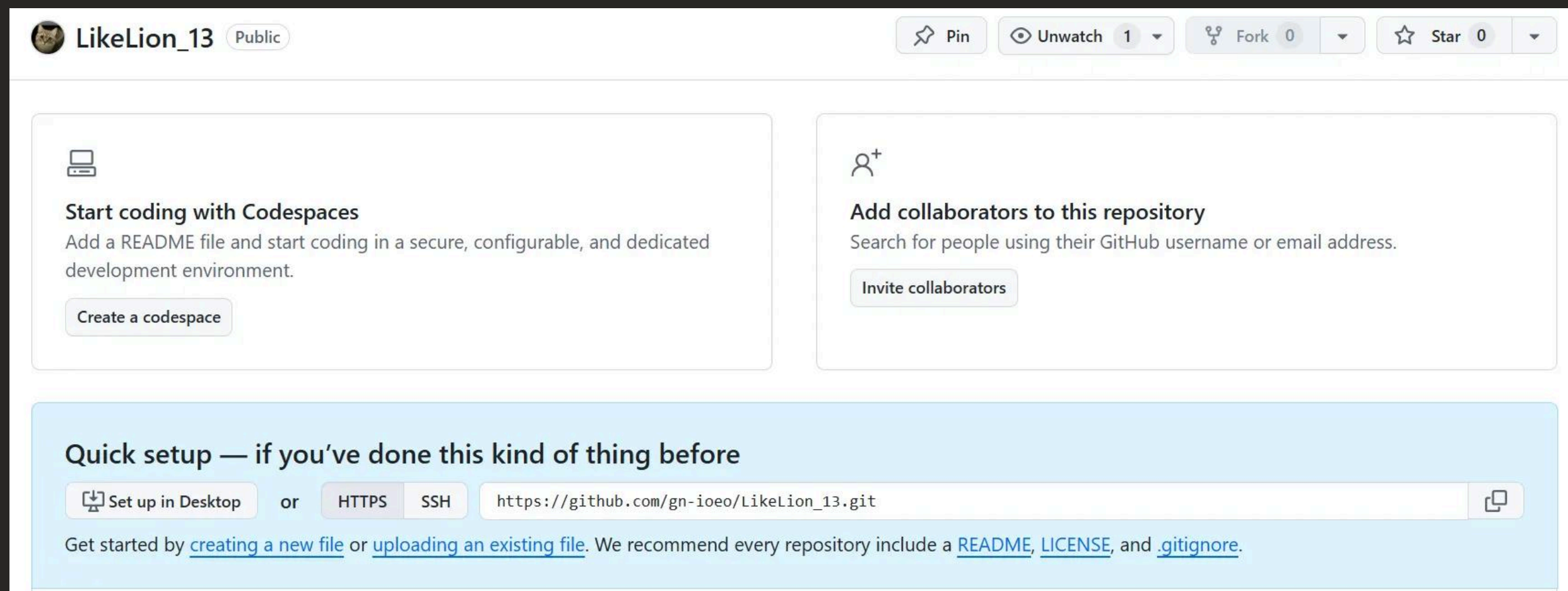
Choose a license

License: None

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

# 리포지토리 만들기

- 가운데 파란 부분에서 `https://어쩌구~.git` 옆 복사 버튼을 눌러주세요!
- 그리고 터미널로 다시 돌아가서 `git remote add origin [복사한 주소]` → 작업폴더와 github연결 완료



```
git remote add origin [복사한 주소]
```



# 리포지토리 만들기

- 컴퓨터에서(Git) 서버(Github)로 작업한 내역 업로드

```
thdrk@gaeun MINGW64 ~/OneDrive/바탕 화면/likelion13 (master)
● $ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 546 bytes | 546.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/gn-ioeo/LikeLion_13.git
 * [new branch]      master -> master

thdrk@gaeun MINGW64 ~/OneDrive/바탕 화면/likelion13 (master)
○ $
```



- `git push -u origin master` (or main)
- (깃 - 올려라 - origin(여러분 깃허브 주소를 다르게 부르는 말) - master(브랜치 - 중급개념))



# 리포지토리 만들기

- 컴퓨터에서(Git) 서버(Github)로 작업한 내역 업로드

```
thdrk@gaeun MINGW64 ~/OneDrive/바탕 화면/likelion13 (master)
● $ git push origin master
Enumerating objects: 3, done.
```

- Username for github:
- Password for ~~:

```
thdrk@gaeun MINGW64 ~/OneDrive/바탕 화면/likelion13 (master)
○ $ █
```



- git push origin master (or main)
- (깃 - 올려라 - origin(여러분 깃허브 주소를 다르게 부르는 말) - master(브랜치 - 중급개념))

# 리포지토리 만들기

- 컴퓨터에서(Git) 서버(Github)로 작업한 내역 업로드

master
1 Branch
0 Tags

t
+
<> Code

gn-ioeo
fist commit
2808962 · 4 hours ago
1 Commit

1.html

fist commit
4 hours ago

README

## Add a README

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

Add a README

### About

No description, website, or topics provided.

Activity
0 stars
1 watching
0 forks

### Releases

No releases published

[Create a new release](#)

### Packages

No packages published

[Publish your first package](#)

### Languages

HTML 100.0%

# 원격 저장소(깃허브)와 연결하기

- "Clone or Download" 버튼을 클릭하여 저장소 URL을 복사

- **원격 저장소를 직접 연결하는 방법**

- 로컬 저장소와 원격 저장소를 연결



```
git remote add origin <깃허브 저장소 URL>
```

- **저장소를 복제하는 방법**

- 깃허브 저장소를 가져와서 작업하고 싶을 때 사용(팀원)

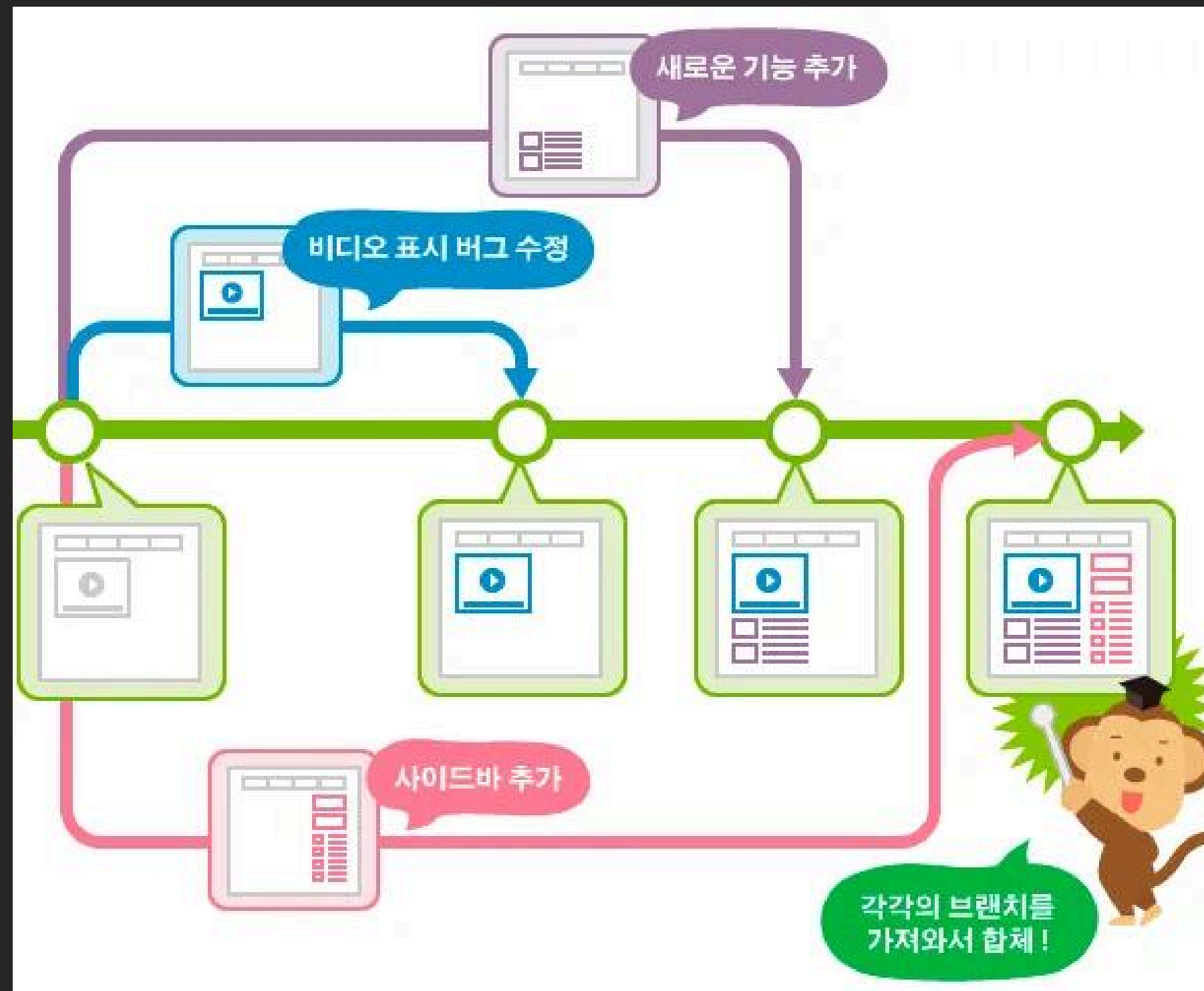


```
git clone <깃허브 저장소 URL>
```

시나리오	명령어
처음부터 새 프로젝트를 만들 때	git init → git remote add origin <URL> → git push
기존 프로젝트를 가져올 때 (팀원이 처음 받을 때)	git clone <URL> (자동으로 원격 저장소와 연결됨)

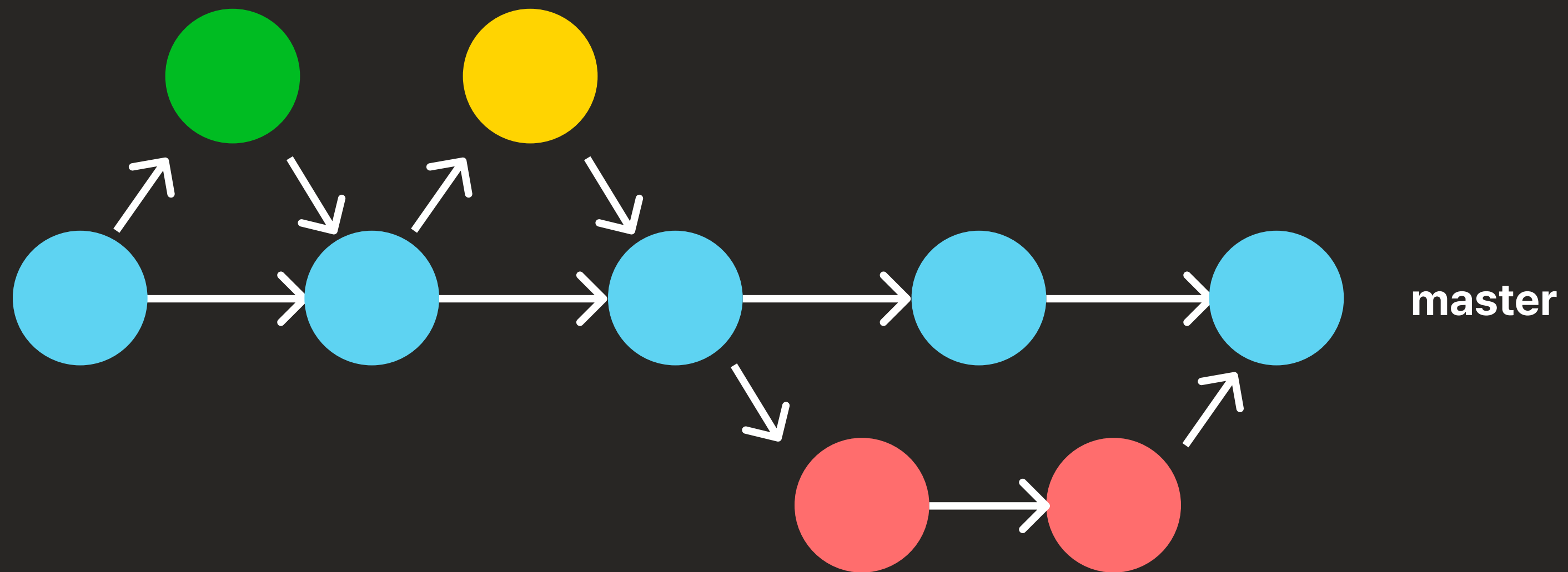
# Branch

- 개발자들이 동시에 독립적으로 작업할 수 있도록 각자의 작업 공간을 만들어주는 기능
- 효율적인 개발을 위해 각자가 브랜치를 생성하여 기능 개발, 버그 수정 등을 진행한 후 메인 브랜치에 통합 (Merge) 합니다.



# 브랜치를 이용한 전반적인 개발 과정

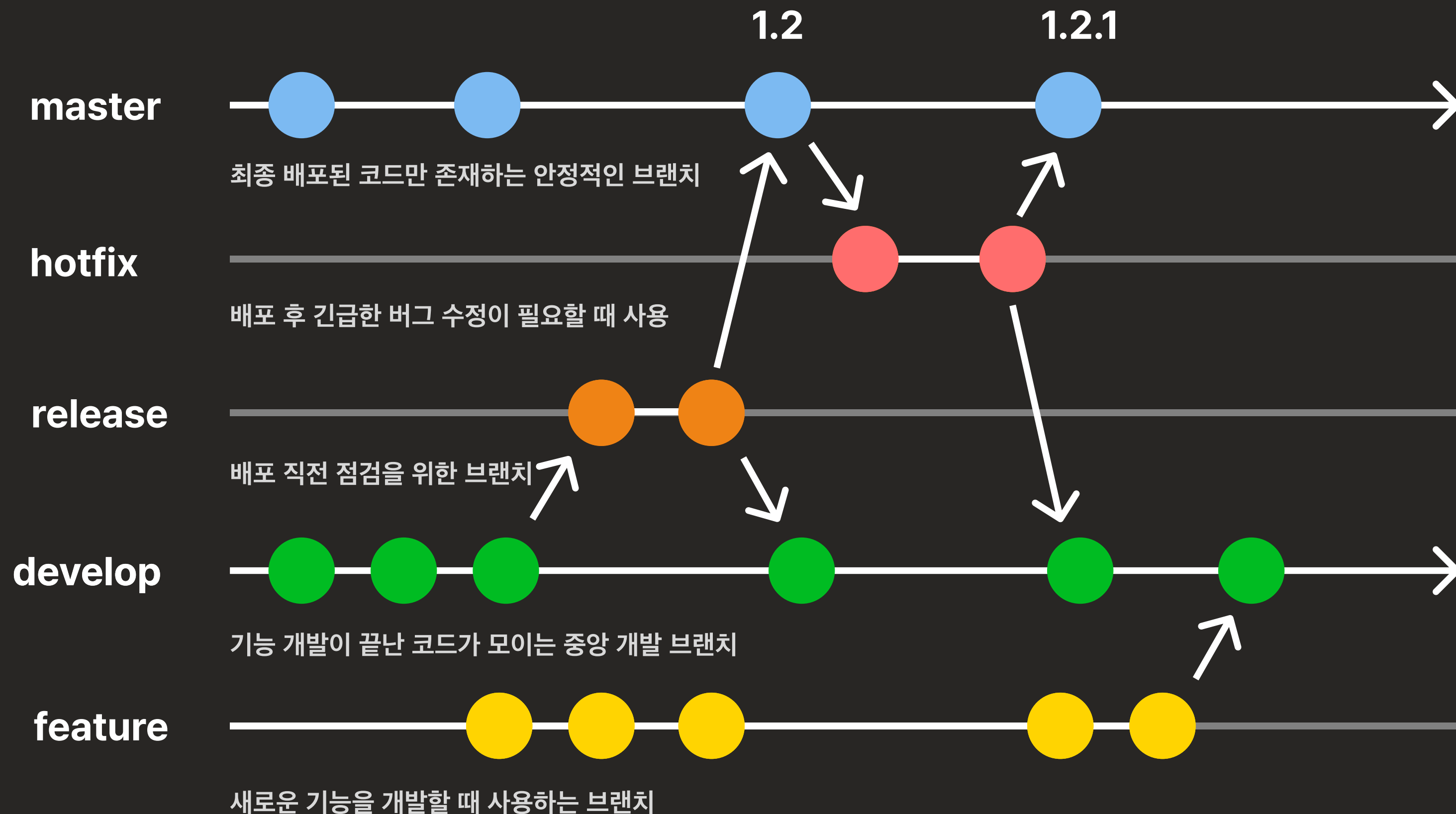
- 단순 브랜치 구조



여러 브랜치에서 작업한 후, master 브랜치로 병합(Merge)

# 브랜치를 이용한 전반적인 개발 과정

- Git Flow 전략



# 브랜치 생성 / 확인 / 전환

## • 브랜치 생성하기



```
git branch 브랜치 이름
```

## • 브랜치 확인하기



```
git branch
```

## • 브랜치 전환하기



```
git checkout 기능추가브랜치
```

## • 브랜치 생성 + 전환



```
git checkout -b 새로운브랜치
```

```
thdrk@gaeun MINGW64 ~/Documents/likelion13 (main)
● $ git branch
    feat-login
*   main
```

\*표시가 현재 내가 있는 브랜치

# Git Pull & Push

- 프로젝트를 깃허브(GitHub)에 올리기
- git pull**: 최신 상태로 가져오기
  - 👉 원격 저장소에서 최신 코드를 가져오는 명령어
- git push**: 내 작업을 원격 저장소에 올리기
  - 👉 내 컴퓨터에서 작업한 내용을 원격 저장소에 올리는 명령어



```
git pull origin 브랜치이름
```



```
git push origin 브랜치이름
```

구분	git pull	git push
목적	원격 저장소에서 최신 코드 가져오기	내 코드를 원격 저장소에 업로드하기
방향	☁️ → 🏠 (깃허브 → 내 컴퓨터)	🏠 → ☁️ (깃허브 → 내 컴퓨터)
사용 시기	작업 시작 전에 최신 코드 받아올 때	작업 끝나고 커밋한 후 공유할 때



# Merge

- 작업을 현재 브랜치에 합치는(merge) 과정



```
git merge 브랜치 이름
```

Fast-forward 머지	다른 작업이 없는 경우 그냥 그대로 브랜치를 앞으로 이동시키는 간단한 머지
3-way 머지	여러 브랜치에서 수정된 코드가 겹치는 경우 충돌(conflict)이 발생합니다. 이때는 사용자가 직접 코드를 보고 해결해야 합니다.

# Conflict

- Conflict(충돌) 해결 : <<<>>> 예시처럼 보이는 부분을 확인하고 원하는 코드만 남김.



<<<<<< HEAD

현재 브랜치 내용

=====

다른 브랜치 내용

>>>>>> 기능추가브랜치

```
<body>
  <div class="container" >
    Accept Current Change | Accept Incoming Change | Accept Both Changes | Comp
    <<<<<< HEAD (Current Change)
      <div class="item self">집sdfsdf</div>
    =====
      <div class="items">집sdfsdf</div>
    >>>>>> conflict (Incoming Change)
      <div class="item self">가</div>
      <div class="item">자</div>
    </div>
  </body>
```

```
thdrk@gaeun MINGW64 ~/Documents/likelion13 (main|MERGING)
$ git add .

thdrk@gaeun MINGW64 ~/Documents/likelion13 (main|MERGING)
$ git commit -m "conflict 해결"
[main 8f323d5] conflict 해결
```

# Pull Request(PR)

- 깃허브에서 PR을 통해 팀원들과 소통하며 개발을 진행, "내 코드 머지해도 될까요?" 라고 요청

## 1. 기능 개발 후 커밋



```
git commit -m "feat: 새로운 기능 추가"
```

## 2. 원격 저장소에 브랜치 푸시

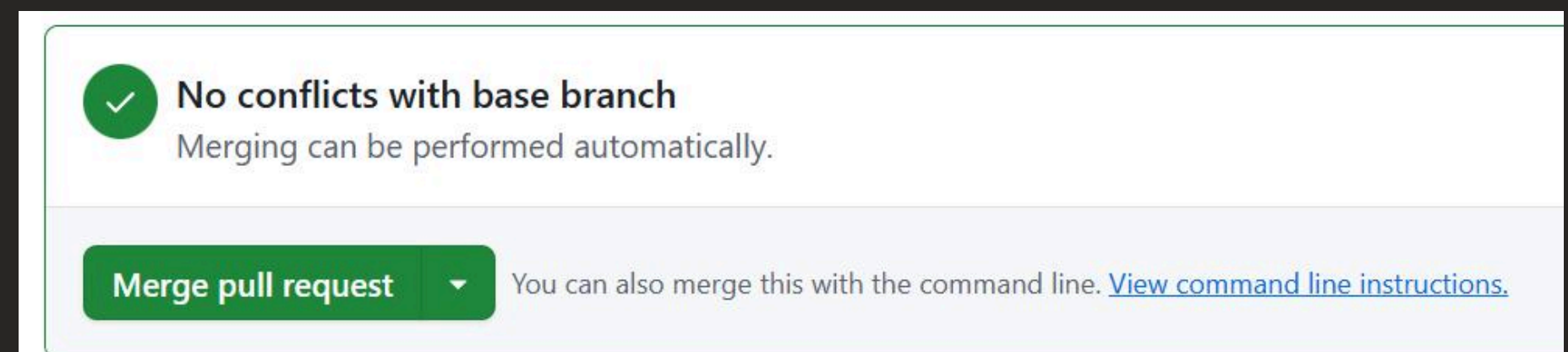
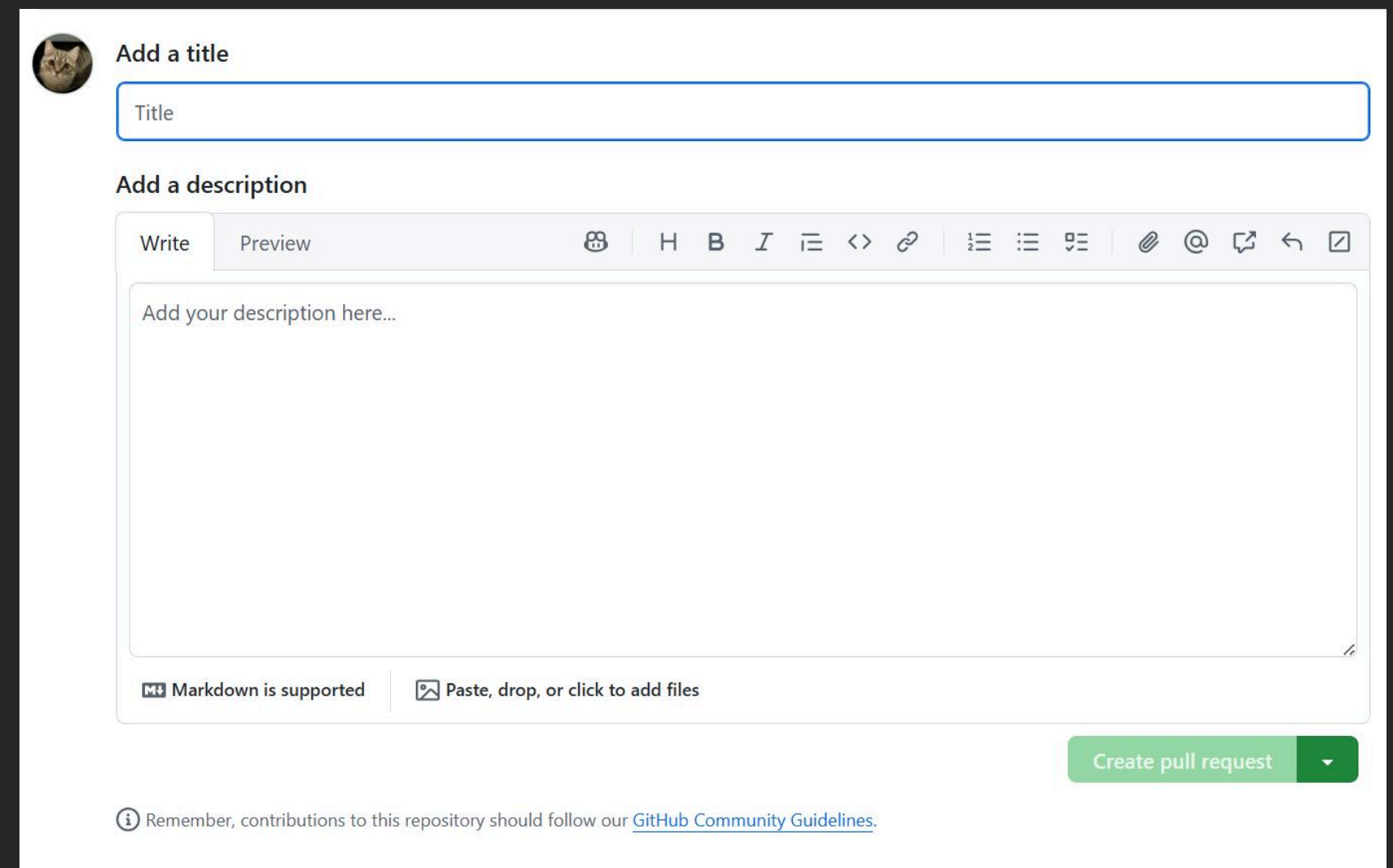


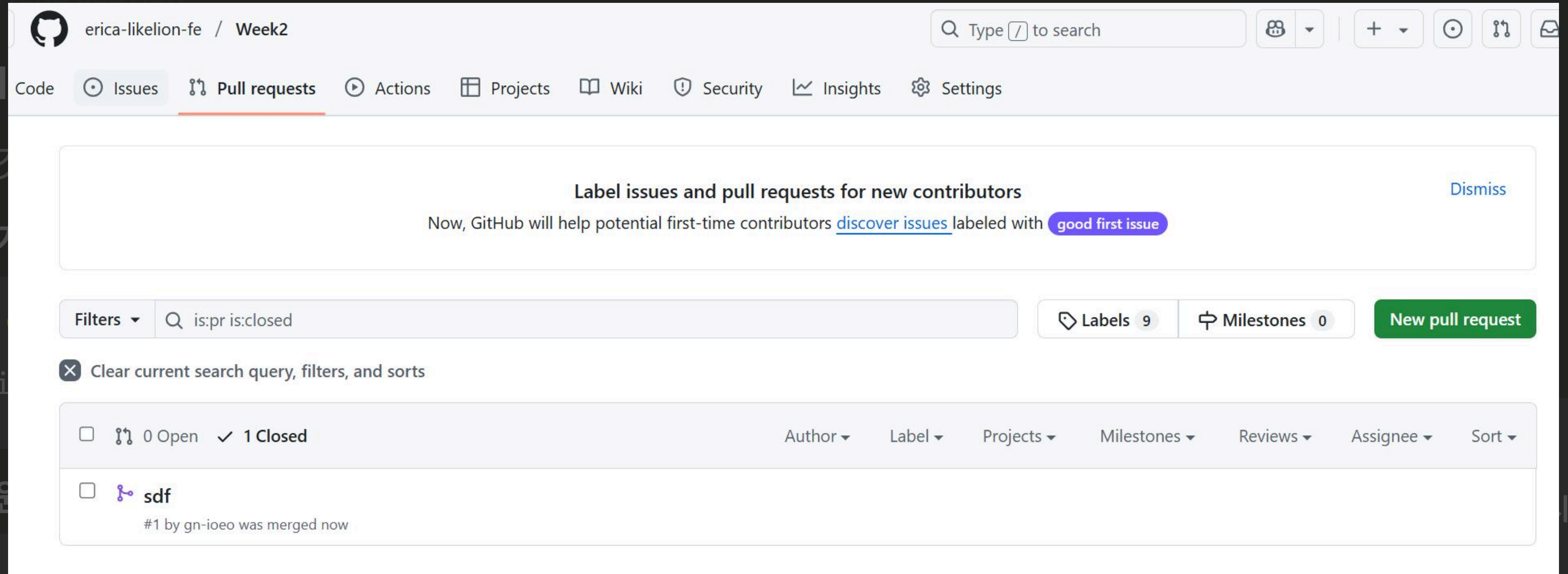
```
git push origin 내가_작업한_브랜치
```

## 3. 깃허브에 PR 생성



- 브라우저에서 깃허브에 접속하여 해당 저장소로 이동
- "Compare & Pull Request" 버튼 클릭
- PR 작성 화면에서 다음 내용을 입력
  - Title: PR 제목 (간결하게 작업 내용을 요약)
  - Description: 작업한 내용, 해결된 이슈, 코드 구조 설명





### ⚠ 자주 하는 실수와 주의사항

#### 1. PR 없이 바로 병합하지 않기

- PR 없이 병합하면 코드 리뷰가 생략되어 문제가 발생할 수 있어요.

#### 2. 커밋을 나누지 않고 한 번에 푸시

- 커밋을 너무 한 번에 몰아넣으면 코드 리뷰가 어려워져요.

#### 3. 설명 부족한 PR

- 작업 내용을 충분히 설명하지 않으면 리뷰어가 코드 이해에 어려움을 겪어요.

## 2주차 수업

# Git, GitHub

77

—

E



# 더배우기 : 커밋 취소

```
(END)... skipping...
commit 599948b4bcbf8da962b73530c9bf1323ad5baea7 (HEAD -> feat1)
Author: gn-ioeo <thdrkdms1030@hanyang.ac.kr>
Date: Mon Mar 17 00:59:43 2025 +0900

잘못된 수정

commit 019e94520c56f20b8f8c486e3dbfcf14a5f68451
Author: gn-ioeo <thdrkdms1030@hanyang.ac.kr>
Date: Mon Mar 17 00:19:08 2025 +0900

내용 수정
```

git log시 나오는 commit ID

## 1. revert

- 취소하고 싶은 특정 커밋의 내용을 되돌리는 새로운 커밋을 만듦
- 기존 커밋을 취소하는 **새로운 커밋**이 생성되므로 협업 환경에서 안전하게 사용 가능



```
git revert <되돌리고 싶은 커밋 ID>
```

## 2. reset

- 특정 커밋 이전 상태로 되돌립니다.
- 이미 원격 저장소에 푸시한 커밋을 삭제하면 문제가 발생할 수 있습니다.



```
git reset [옵션] [커밋ID]
```

- **--soft** : 커밋만 삭제하고 변경 내용은 그대로 둠
- **--mixed** (기본값) : 커밋과 스테이징 영역을 비우고 파일 변경 사항만 남김
- **--hard** : 커밋과 파일 변경 사항까지 모두 삭제