



Google Developer Groups  
On Campus • Sogang University

# Hurdlethon Web

# Git, Github

Web Core  
Heewon An

# What we will learn ...

## 1. **Git**

- Repository, commit ...

## 2. **GitHub**

- Branch, Merge ...

## 3. **GitHub for Cooperation**

- PR, Organization ...



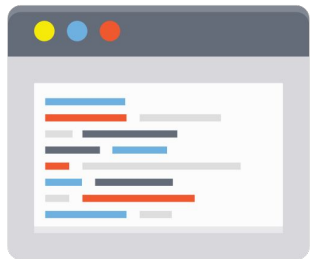
Google Developer Groups

On Campus • Sogang University

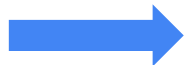
# What is Git?

```
lookup.KeyValue  
f.constant(['em  
=tf.constant([G  
lookup.StaticV  
_buckets=5)
```

# Why we use Git?

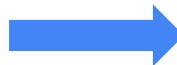


Original



New

다시 원래  
버전으로 되돌려  
주세요!



# Why we use Git?

기억에 의존하여 Original  
버전으로 되돌리기

```
original.js  
original_2.js  
original_final.js  
original_real_final.js  
⋮
```

코드를 바꿀 때마다 새로운  
파일로 관리하기

쌓여가는 파일들을 어떻게 다 관리할까?  
수정한 곳을 어떻게 빠르게 찾을 수 있을까?  
한 파일만 수정하는 것도 아닌데..



# We should use Git, now!



“코드 버전 관리 시스템”

- 여러 개발자가 병렬적으로 작업을 수행하고, 쉽게 합칠 수 있어 협업에도 용이해요.
- Git은 전 세계 많은 개발자가 사용하기 때문에 다른 프로젝트에 쉽게 기여하거나 참여할 수 있어요.
- Conflict가 발생했을 때 자세한 정보를 제공해 주는 것 역시 Git의 장점이에요.
- GitHub가 제공해주는 PR, Action 등 다양한 기능들을 통해 협업 및 개발을 효율적으로 관리할 수 있어요.
- 참고로 Subversion이나 Mercurial 등과 같이 Git과 비슷한 서비스도 있어요.

(자세한 비교는 [여기를 참고해 보세요.](#))



Google Developer Groups

On Campus • Sogang University

# What we need to know to use Git

```
lookup.KeyValue  
f.constant(['em  
=tf.constant([G  
lookup.StaticV  
_buckets=5)
```

# Repository, Staging Area, Working Directory

- Git은 총 3가지 작업 공간을 가지고 프로젝트 파일들을 관리해요.
- 3가지 공간 모두 일종의 '저장소'라고 이해하면 돼요.







# Command - Repository

Git이 설치되어 있는지  
확인해 주세요!

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

1. 가장 먼저 현재 위치에 Git Repository를 생성해요.

```
git init
```

.git 파일이 생성되었다면 성공입니다!

\* 파일은 숨겨진 파일이기 때문에 **파일 탐색기 - 보기 > 표시 > 숨긴 항목**을 체크해야 보여요.



.git



hello\_world.c

2. 변경한 파일들 중 commit할 파일들은 Staging Area에 올려요.

```
git add example.js //example.js 대신 원하는 파일명
```

```
git add . //지금까지 변경한 파일 모두 올리기
```

```
PS C:\Users\hw766\OneDrive\바탕 화면\Hurdlethon> git add .
PS C:\Users\hw766\OneDrive\바탕 화면\Hurdlethon> git status
On branch main
```

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: hello\_world.c

git status 명령어로 Staging Area에 올라간 파일들을  
확인할 수 있어요!



Google Developer Groups

On Campus • Sogang University

# Commit

A.js

B.js

C.js



A.js

B\_new.js

C\_new.js

지금 이 디렉토리의  
모습을 저장해놓고  
싶어!

- commit을 하면 Staging Area에 있는 변경 사항들이 Repository에 저장돼요.
- commit은 Staging Area에 있는 파일들을 올리기 때문에, **commit 하기 전 git add를 해야 한다는 것**에 주의하세요.
- 각 commit은 고유한 해시값을 가지기 때문에 언제든지 해당 commit의 directory 모습으로 돌아갈 수 있어요.

chore: lint 안맞는 부분 수정	8e48d10		
naya-h2 committed 3 weeks ago			
feat: 스터디, 세미나 디자인 변경 반영	75f28b8		
naya-h2 committed 3 weeks ago			
Merge pull request #9 from GDSC-Sogang-Univ/feat/landingpage_mk			
	5bf27a1		
naya-h2 authored 3 weeks ago			
chore: yarn file modified	2272c55		
moong23 committed 3 weeks ago			
fix: return 분기처리 및 내용 추가	5486a85		
moong23 committed 3 weeks ago			
feat: Animation 기능 구현	70e76a4		
moong23 committed 3 weeks ago			

실제 Web Core 팀의 commit 목록입니다!



# Command - Commit

1. Commit 하기 전, 사용자 정보를 설정해야 해요.

```
git config user.name "NAME"  
git config user.email "example@sogang.ac.kr"
```

2. Commit을 해볼게요.

```
git commit -m "initial Commit"
```

잠깐!  
git add file  
하셨죠?

변경 사항을 명확하게 기록하기 위해 commit message를 꼭 남겨주세요!

특히 여러 사람과 협업을 할 때,

어떤 목적을 가지고 이 commit에 포함된 파일들을 변경했는지 명확하게 작성하는 것이 좋아요.

👉 이러한 이유로 협업을 할 때에 commit 컨벤션을 정하곤 해요.



Google Developer Groups

On Campus • Sogang University



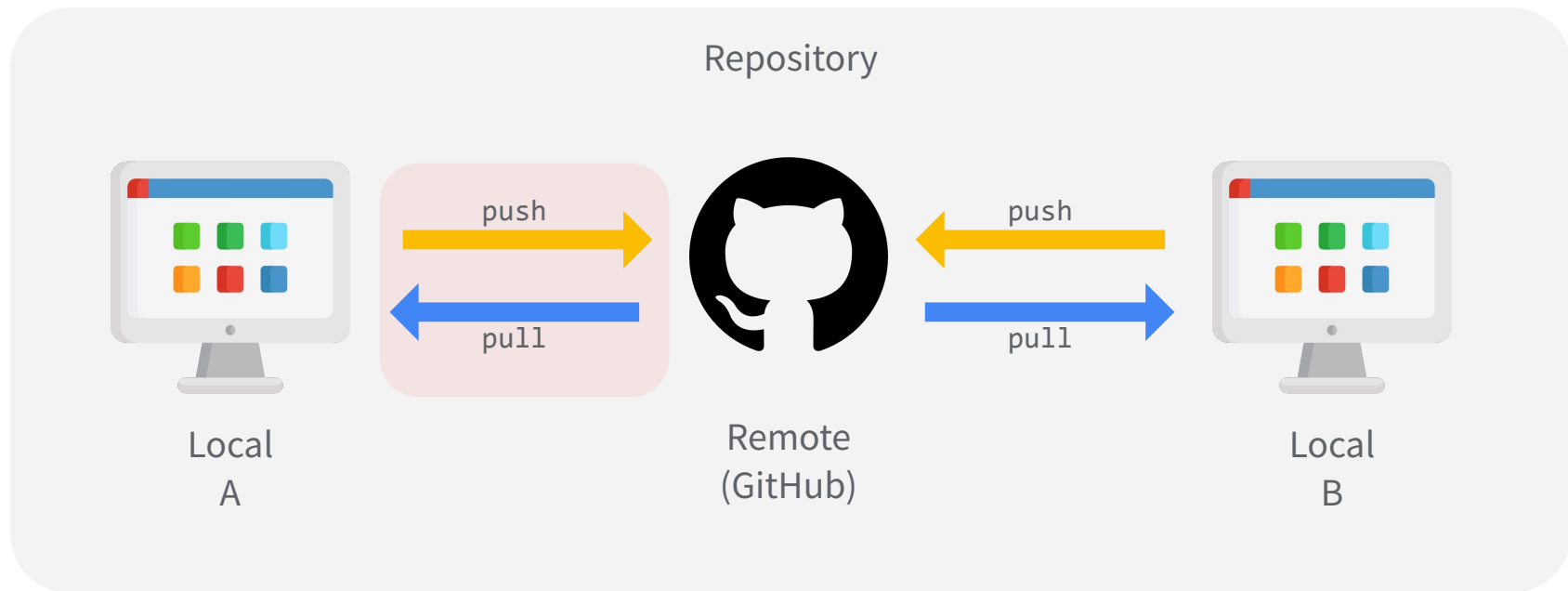
Google Developer Groups

On Campus • Sogang University

# What is GitHub?

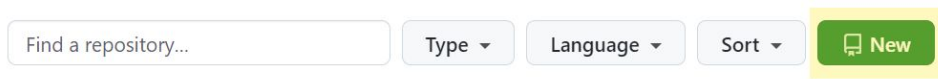
```
lookup.KeyValue  
f.constant(['em  
=tf.constant([G  
lookup.StaticV  
_buckets=5)
```

# GitHub vs Git



# Let's Create Remote Repository

1. 자신의 GitHub의 Repository 탭에서  
**New** 버튼을 클릭해요.



Find a repository... Type ▾ Language ▾ Sort ▾ **New**

2. Repository의 정보를 입력한 뒤,  
**Create repository** 버튼을 클릭하면 끝!

## 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 \*  naya-h2 ▾ / Repository name \*    
  Hurdlethon is available.

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

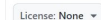
Add .gitignore



.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.](#)

① You are creating a public repository in your personal account.

**Create repository**

# Connect Remote Repository to Local Repository

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

 Set up in Desktop

or

**HTTPS**   **SSH**

<https://github.com/naya-h2/Hurdlethon.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 "# Hurdlethon" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/naya-h2/Hurdlethon.git
git push -u origin main
```

1. Local Repository를 새로 생성하고, 커밋 후에 Remote Repository를 연결하는 방법

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

```
git remote add origin https://github.com/naya-h2/Hurdlethon.git
git branch -M main
git push -u origin main
```

2. 이미 생성되어 있는 Local Repository를 새로 생성한 Remote Repository에 연결하는 방법



# Command - Remote Repository

1. 생성한 Local Repository와 Remote Repository를 연결해 볼게요.

```
git remote add origin <프로젝트 주소>
```

‘origin’이라는 별칭으로 <프로젝트 주소>에 해당하는 Remote Repository와 연결하겠다는 의미입니다.

```
git remote -v
```

현재 Local Repository에 연결된 Remote Repository 목록을 확인할 수 있어요.

```
PS C:\Users\hw766\OneDrive\바탕 화면\Hurdlethon> git remote -v
origin https://github.com/naya-h2/Hurdlethon.git (fetch)
origin https://github.com/naya-h2/Hurdlethon.git (push)
```

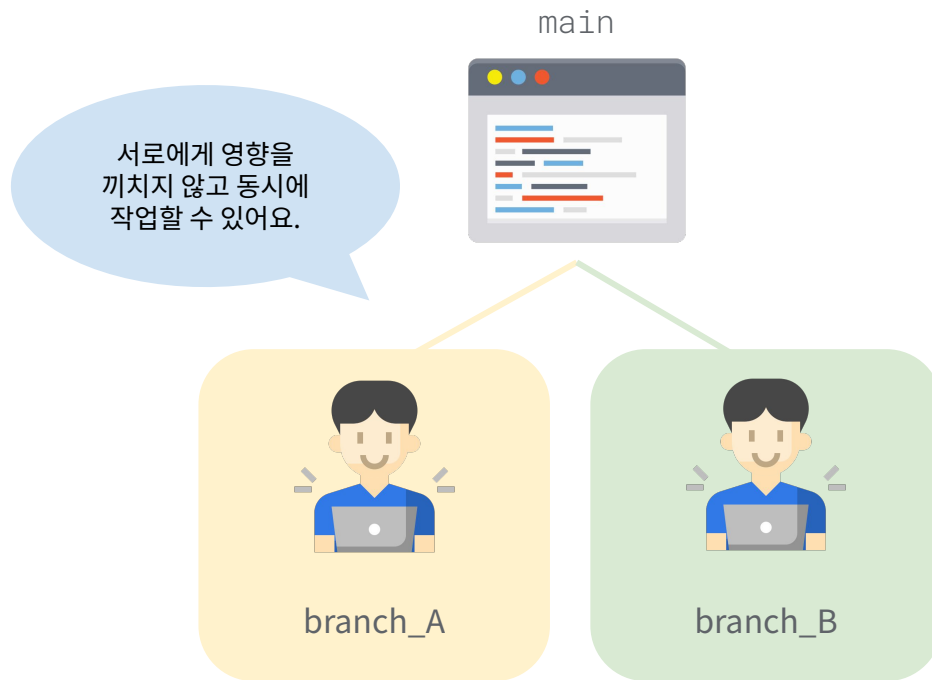
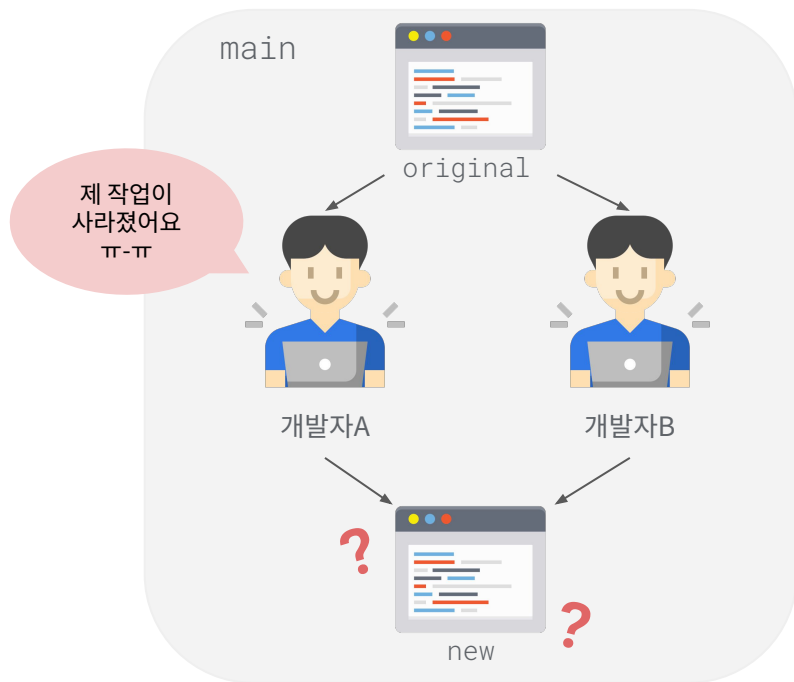
2. 반대로 Remote Repository를 가져와서 새로운 Local Repository를 만들 수도 있어요.

```
git clone <프로젝트 주소>
```





# Branch



각 branch는 독립적인 공간



# Command - Branch (1)

1. Branch를 생성해 볼게요.

```
git branch <브랜치 이름>
```

2. 생성한 Branch로 이동해 볼게요.

```
git switch <브랜치 이름>
```

```
git switch -c <브랜치 이름> //<브랜치 이름>이라는 Branch를 만들고, 이동까지 한 번에 할 수 있어요.
```

3. 생성한 Branch를 삭제해 볼게요.

```
git branch -d <브랜치 이름>
```





## Command - Branch (2) & Push / Pull

4. 현재 어떤 Branch에 있는 지 확인해 볼게요.

```
git branch
```

Repository에 main, edit이라는 Branch가 존재하고,  
현재 **edit** Branch에 위치한다는 의미입니다.

```
PS C:\Users\hw766\OneDrive\바탕 화면\Hurdlethon> git branch
* edit
main
```

5. 현재 Branch의 commit을 연결한 Remote Repository 중 'origin'에 push/pull해 볼게요.

```
git push origin <현재 브랜치 이름> //<local 브랜치명>:<remote 브랜치명> 으로 따로 올릴 이름을 설정할 수도 있어요.
git pull origin <작업 상황을 가져올 브랜치 이름>
```



# Merge

독립적으로 작업한 branch를 합치는 과정

feat/UI 브랜치를  
feat/auth 브랜치에  
merge한  
모습입니다.



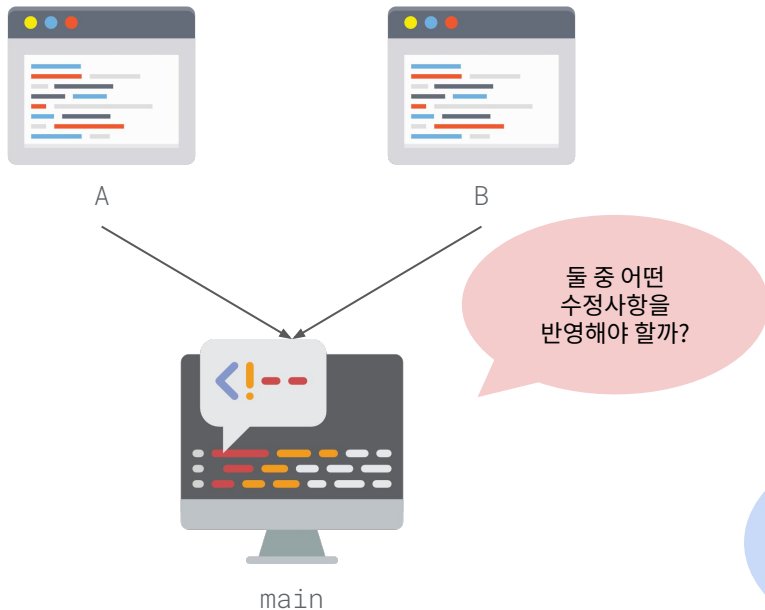
## Command - Merge

1. 현재 Branch에 'A'라는 Branch를 Merge해 볼게요.

```
git merge A //A 위치에 merge하고자 하는 Branch 이름을 쓰면 돼요.
```



# Conflict



- 둘 이상의 개발자가 **같은 파일의 같은 line**을 수정한 경우
- merge하고자 하는 브랜치(base branch)와 파일이 맞지 않는 경우  
(== 다른 개발자가 base branch에 새로운 작업 상황을 push했는데, **pull**을 받지 않은 경우)

=> merge하기 전, base branch를 먼저 pull 받아 최신화하는 것이 좋아요.

Conflict가 발생한  
파일에 표시된  
부분을 모두  
고쳐주세요.

```
<<<<<<< HEAD
print("Hello World")
=====
print("Hi there")
>>>>>>> feat/login
```

# Conflict Example

main

```
3 int main() {  
4     printf("cur branch is edit!\n");  
5  
6     return 0;  
7 }  
You, 9 minutes ago • initial commit
```

merge

edit

```
3 int main() {  
4     printf("Hello world_edited!\n");  
5  
6     return 0;  
7 }  
You, 10 minutes ago • initial commit
```

```
PS C:\Users\hw766\OneDrive\바탕 화면\Hurdlethon> git merge edit  
Auto-merging hello_world.c  
CONFLICT (content): Merge conflict in hello_world.c  
Automatic merge failed; fix conflicts and then commit the result.
```

Merge를 시도하면 conflict를 고친 후에 시도하라는 메시지가 나와요.



둘 중 어떤 버전을 반영해야  
할지 모르겠으니까  
개발자가 선택하게 하자!



# How to resolve Conflict (1)

1. 시도한 Merge를 취소할 수 있어요.

```
git merge --abort
```

2. 우선, conflict가 발생한 파일을 확인해 봐요.

```
git status
```

Unmerged paths 영역에서  
conflict가 발생한 파일들 목록을 확인할 수 있어요.

```
PS C:\Users\hw766\OneDrive\바탕 화면\Hurdlethon> git status
On branch main
Your branch is up to date with 'origin/main'.

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   hello_world.c

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



Google Developer Groups

On Campus • Sogang University



## How to resolve Conflict (2)

```
3  int main() {
   Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
4  <<<<<< HEAD (Current Change)
5      printf("Hello world_edited!\n");
6
7  ===== You, 21 seconds ago • Uncommitted changes
8      printf("cur branch is edit!\n");
9
10 >>>>>> edit (Incoming Change)
11     return 0;
12 }
```

Resolve in Merge Editor

3. Conflict가 발생한 파일을 보면, conflict가 발생한 코드 부분이 아래와 같이 표시되어 있어요.

4. 오른쪽 예시와 같이 표시된 부분을 원하는 branch의 코드를 선택해서 수정해요.

두 branch의 코드를 전부 선택해도 괜찮아요!

```
3  int main() {
   Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
4  <<<<<< HEAD (Current Change)
5      printf("Hello world_edited!\n");
6
7  ===== You, 6 minutes ago • Uncommitted changes
8      printf("cur branch is edit!\n");
9
10 >>>>>> edit (Incoming Change)
11     return 0;
12 }
```

Resolve in Merge Editor

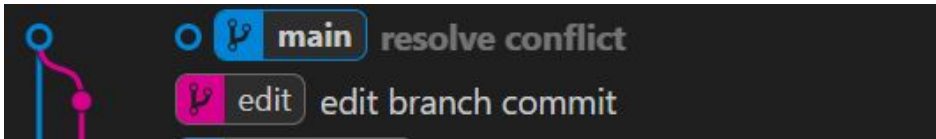




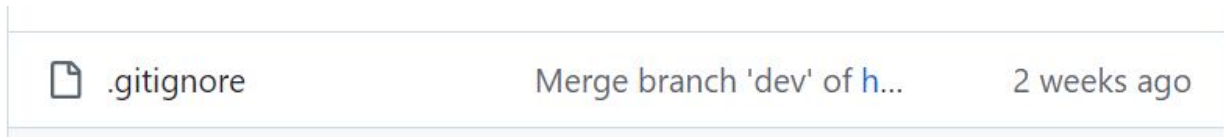
## How to resolve Conflict (3)

5. Conflict가 발생한 모든 파일을 수정하고 나면, 수정한 파일을 다시 커밋해요.

6. 커밋을 완료하면 아래와 같이 성공적으로 merge가 된 것을 확인할 수 있어요!



# .gitignore



- .gitignore 파일을 사용하면 Repository나 Staging Area에 추가되지 말아야 하는 폴더나 파일을 정의할 수 있어요.
- 한 마디로, **git이 무시할 파일을 정의**하는 파일입니다.
- 환경 변수 파일처럼 노출되면 안되는 **민감한 정보가 포함된 파일**은 반드시 추가하는 것이 좋습니다!



# Add file to .gitignore

1. <dir명> 하위 모든 파일을 무시해 볼게요.

```
<dir명>/
```

2. 특정 확장자를 가지는 모든 파일을 무시해 볼게요.

```
*.<확장자명>
```

3. 특정 파일 1개만 무시해 볼게요.

```
<dir명>/<파일명>  
<파일명> //root dir에 위치한 파일은 이렇게 추가할 수 있어요.
```





Google Developer Groups

On Campus • Sogang University

# GitHub for Cooperation

```
lookup.KeyValue  
f.constant(['em  
=tf.constant([G  
lookup.StaticV  
_buckets=5)
```

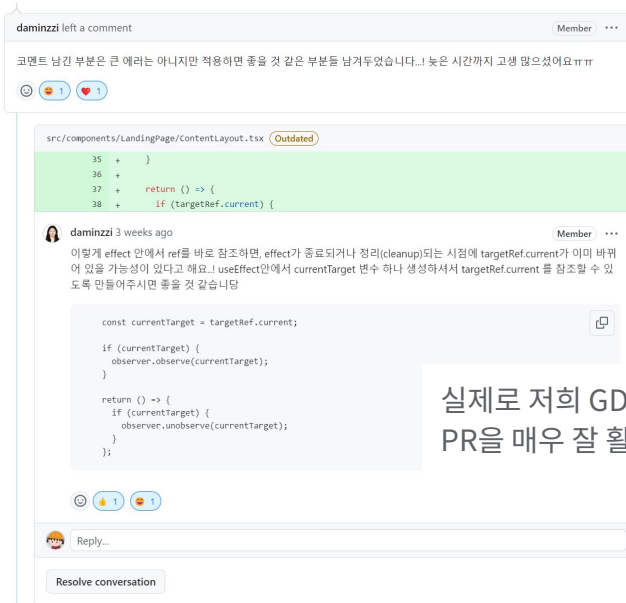
# PR(Pull Request)

변경사항을 merge하기 전, 팀원들에게 검토/코드 리뷰를 요청할 수 있어요.

이 과정에서 코드 실수나 문제를 수정할 수도 있고,  
누가 어떤 작업을 했는지 추적하기 쉬워 프로젝트 관리도 수월하게 할 수 있어요.



- ☐ feat: GFD 영역 description 문구 수정  
#18 by naya-h2 was merged 2 weeks ago
- ☐ [QA] 애니메이션 수정 반영  
#17 by moong23 was merged 2 weeks ago • Approved
- ☐ 랜딩페이지 수정(GDSC 텍스트, 애니메이션)  
#16 by naya-h2 was merged 2 weeks ago • Approved
- ☐ Feat: 배포 이미지 경량화  
#15 by lly2855 was merged 2 weeks ago



daminzzi left a comment

코멘트 남긴 부분은 큰 에러는 아니지만 적용하면 좋을 것 같은 부분들 남겨두었습니다.↓ 늦은 시간까지 고생 많으셨어요ㅠㅠ

src/components/LandingPage/ContentLayout.tsx Outdated

```
35 + }
36 +
37 + return () => {
38 +   if (targetRef.current) {
```

daminzzi 3 weeks ago

이렇게 effect 안에서 ref를 바로 참조하면, effect가 종료되거나 정리(cleanup)되는 시점에 targetRef.current가 이미 바뀌어 있을 가능성이 있다고 해요.↓ useEffect 안에서 currentTarget 변수 하나 생성해서 targetRef.current를 참조할 수 있도록 만들어주시면 좋을 것 같습니다

```
const currentTarget = targetRef.current;

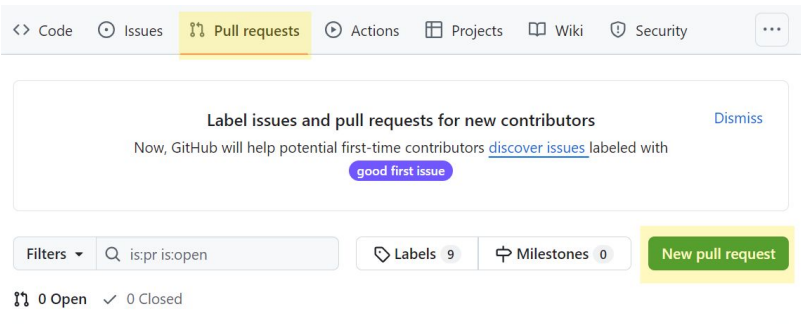
if (currentTarget) {
  observer.observe(currentTarget);
}

return () => {
  if (currentTarget) {
    observer.unobserve(currentTarget);
  }
};
```

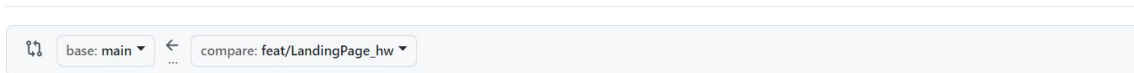
실제로 저희 GDG Sogang Web core 팀도 PR을 매우 잘 활용 중입니다!

# Let's Create PR

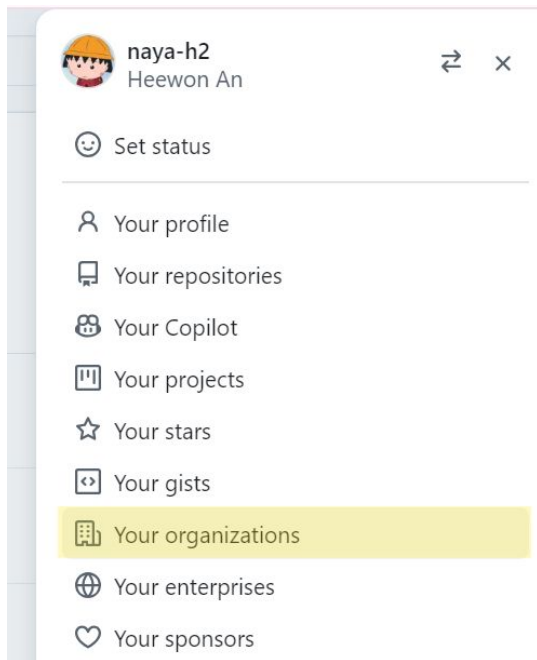
1. PR은 간단히 Repository의 Pull requests 탭의 **New pull request** 버튼을 통해 올릴 수 있어요.



2. PR 올리기 전, 병합하고자 하는 **base branch**를 **pull 받아 최신화**하는 것 잊지 마세요!  
base와 비교해 최신이 아닌 경우에는 아래 사진처럼 PR을 올릴 수 없어요.



# Organization



Team Project 진행 시, GitHub의 Organization 기능을 사용해 보세요!

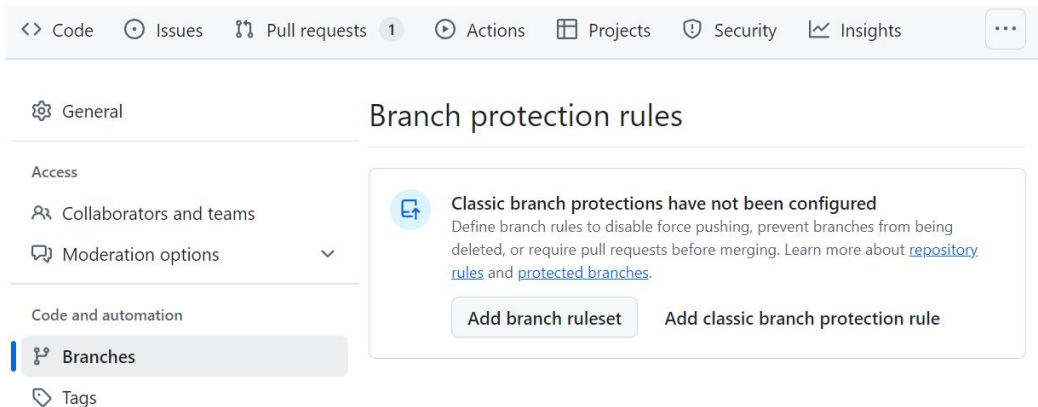
Organization은 말 그대로 협업을 할 수 있는 **단체 계정**을 의미해요.

- Project에 필요한 모든 Repository를 한 곳에서 관리할 수 있어요.
- 팀원 별로 역할과 권한을 설정할 수 있어요.
- Organization 대신 누군가의 계정에 원본 Repository를 만든다면, 해당 계정 주인이 실수를 할 경우 원본이 손상될 수 있어요.

👉 Organization에 원본 Repository를 만들고, fork해서 복사본을 만든 뒤 작업을 하는 방식을 채택할 수도 있어요!

# Branch Protection

Branch마다 규칙을 추가할 수 있어요.



The screenshot shows the GitHub interface for configuring branch protection rules. The top navigation bar includes links for Code, Issues, Pull requests (with a notification badge), Actions, Projects, Security, and Insights. The left sidebar contains a menu with 'General', 'Access' (including Collaborators and teams, and Moderation options), 'Code and automation', and 'Branches' (which is highlighted). The main content area is titled 'Branch protection rules' and displays a message: 'Classic branch protections have not been configured'. Below this message, it explains that branch rules can be used to disable force pushing, prevent branches from being deleted, or require pull requests before merging. At the bottom of the message box, there are two buttons: 'Add branch ruleset' and 'Add classic branch protection rule'.

- PR의 merge 조건을 추가하거나
- 특정 Branch에 push하는 것을 막는 등

협업 시,  
좀 더 안전하게 원본 Repository를 관리할 수 있어요.





Google Developer Groups

On Campus • Sogang University

# Assignment

```
lookup.KeyValue  
f.constant(['em  
=tf.constant([G  
lookup.StaticV  
_buckets=5)
```

# Assignment

1. [‘Git-GitHub’ Repository](#)를 fork하세요.
2. Organization의 Repository를 **upstream**, 내 계정의 Repository를 **origin**으로 연결하세요.
3. 자신의 Local repo에서 ‘git-1’ 이라는 branch를 새로 생성해서 upstream의 [‘hurdlethon-3’ branch](#)의 내용을 pull 받아요.
4. 해당 branch에서 README의 내용에 따라 알맞게 파일을 수정한 후, “**Git, GitHub 교육 이수**”라는 **commit**을 생성해 보세요.
5. 생성한 commit을 Remote Repository(origin)로 push한 후, ‘hurdlethon-3’ branch를 base branch로 PR을 올려주세요.
6. PR을 올릴 때, git remote -v 명령어의 결과 사진을 첨부해 주세요.
7. 마지막으로, 올린 PR 링크를 노션 과제 제출란에 추가하면 끝!

자세한 PR 방법 및 과제 제출 설명은 [링크](#)를 참고해 주세요.

```
children: [
  Icon(icon, color: color),
  Container(
    margin: const EdgeInsets,
    child: Text(
      label,
      style: TextStyle
```

**Fin.**

Thank you for watching!

