

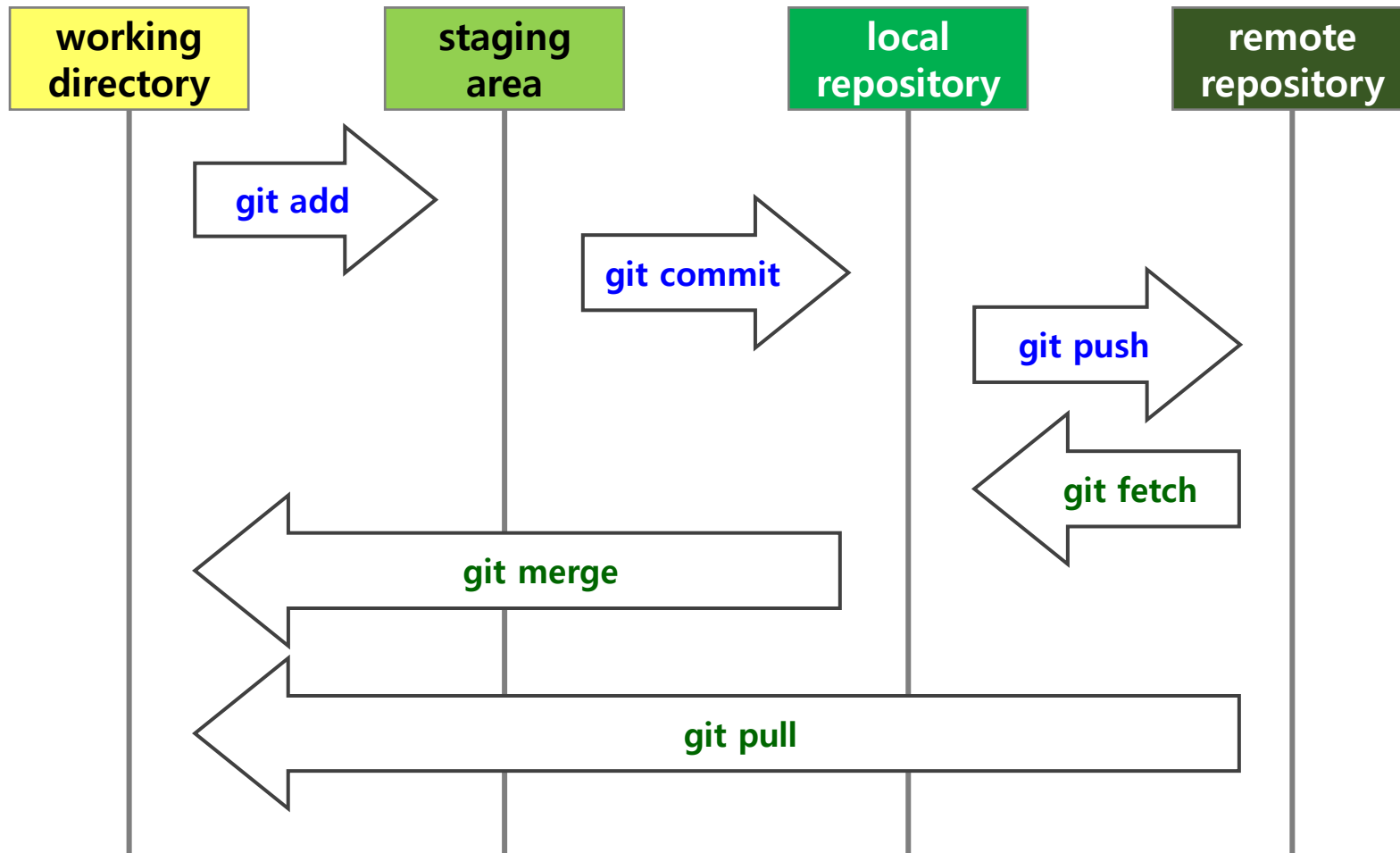
# github 협업

**fork, pull request**

동의과학대학교 컴퓨터정보과

# 개인이 혼자 git과 github 사용 시 절차

- 자신의 local repo에서 자신의 remote repo로 동기화 할 때 절차



# windows 용 git 설치

- 다운로드 : <https://git-scm.com/>



The screenshot shows the Git website homepage. At the top, the Git logo and tagline "--fast-version-control" are visible. Below this, a search bar and a description of Git as a free and open source distributed version control system are present. A diagram of a commit history graph is shown on the right. The bottom section contains four links: About, Documentation, Downloads, and Community. On the right side, a monitor displays the latest source release (2.24.0) and a button to "Download 2.24.0 for Windows", which is highlighted with a red box.

git --fast-version-control

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

**About**  
The advantages of Git compared to other source control systems.

**Documentation**  
Command reference pages, Pro Git book content, videos and other material.

**Downloads**  
GUI clients and binary releases for all major platforms.

**Community**  
Get involved! Bug reporting, mailing list, chat, development and more.

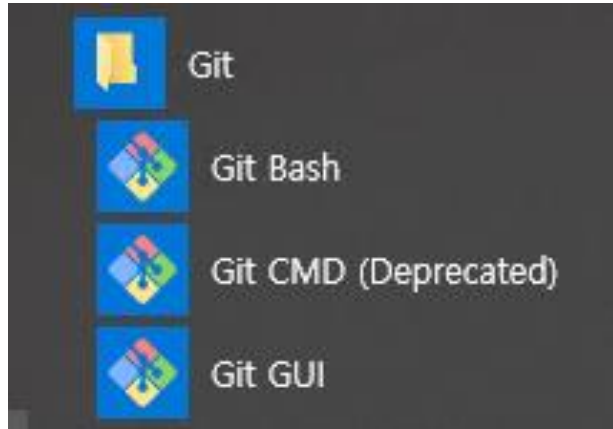
Latest source Release  
**2.24.0**  
Release Notes (2019-11-04)

Download 2.24.0 for Windows

다운로드

# git – CLI

- windows용 git 을 설치하면 아래의 프로그램을 사용할 수 있음



- 위의 협업 예제는 Git CMD를 사용했으나 이 프로그램이 deprecated(곧 사용되지 않을 예정)임으로 Git Bash를 사용할 것을 추천함

# git – GUI Clients

- 플랫폼에 따른 다양한 git 관리 클라이언트가 있음
- 차후에 자신에게 맞는 프로그램을 사용하도록 함

The screenshot shows the official git website. At the top, the git logo is followed by the tagline "--local-branching-on-the-cheap". A search bar is on the right. On the left, a navigation menu includes links for About, Documentation, Downloads, GUI Clients (highlighted), and Logos. Below this is a Community section. The main content area is titled "GUI Clients" and explains that git has built-in GUI tools (git-gui and gitk) but also lists several third-party tools. A row of buttons allows filtering by platform: All, Windows, Mac, Linux, Android, and iOS. Below this, six GUI clients are displayed in a grid:

- SourceTree**: Platforms: Mac, Windows; Price: Free; License: Proprietary. The image shows its interface with a file explorer on the left and a commit message editor on the right.
- GitHub Desktop**: Platforms: Mac, Windows; Price: Free; License: MIT. The image shows its clean, modern interface with a list of repositories on the left.
- TortoiseGit**: Platforms: Windows; Price: Free; License: GNU GPL. The image shows its integration into the Windows context menu, with options like "Commit", "Push", and "Pull".
- Git Extensions**: Platforms: Linux, Mac, Windows; Price: Free; License: GNU GPL. The image shows its complex interface with multiple panes for file explorer, commit history, and commit details.

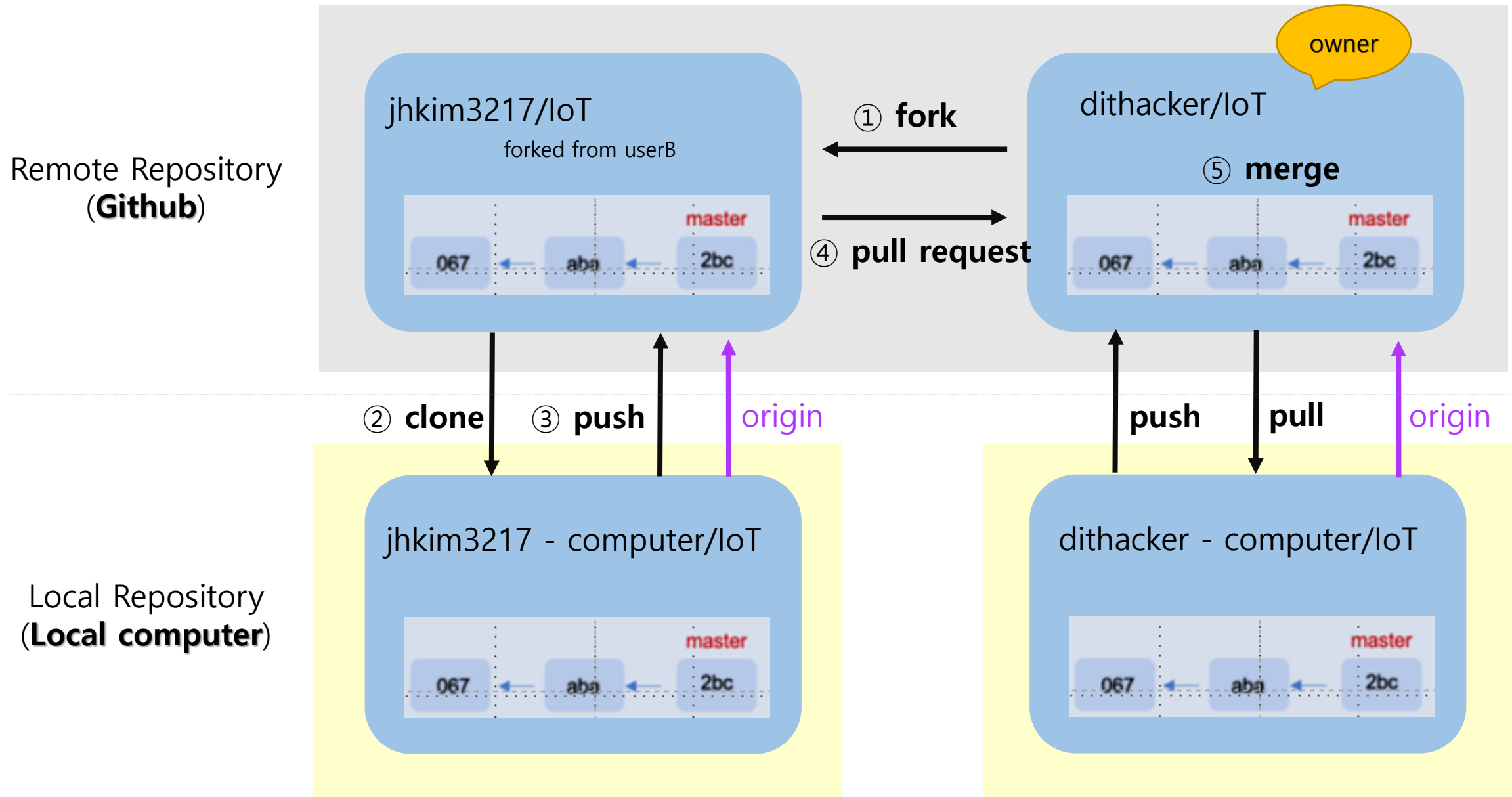
At the bottom, there are two more partial screenshots of other GUI clients.

# Github 협업 플로우

참조 : <https://git-scm.com/book/ko/v2>

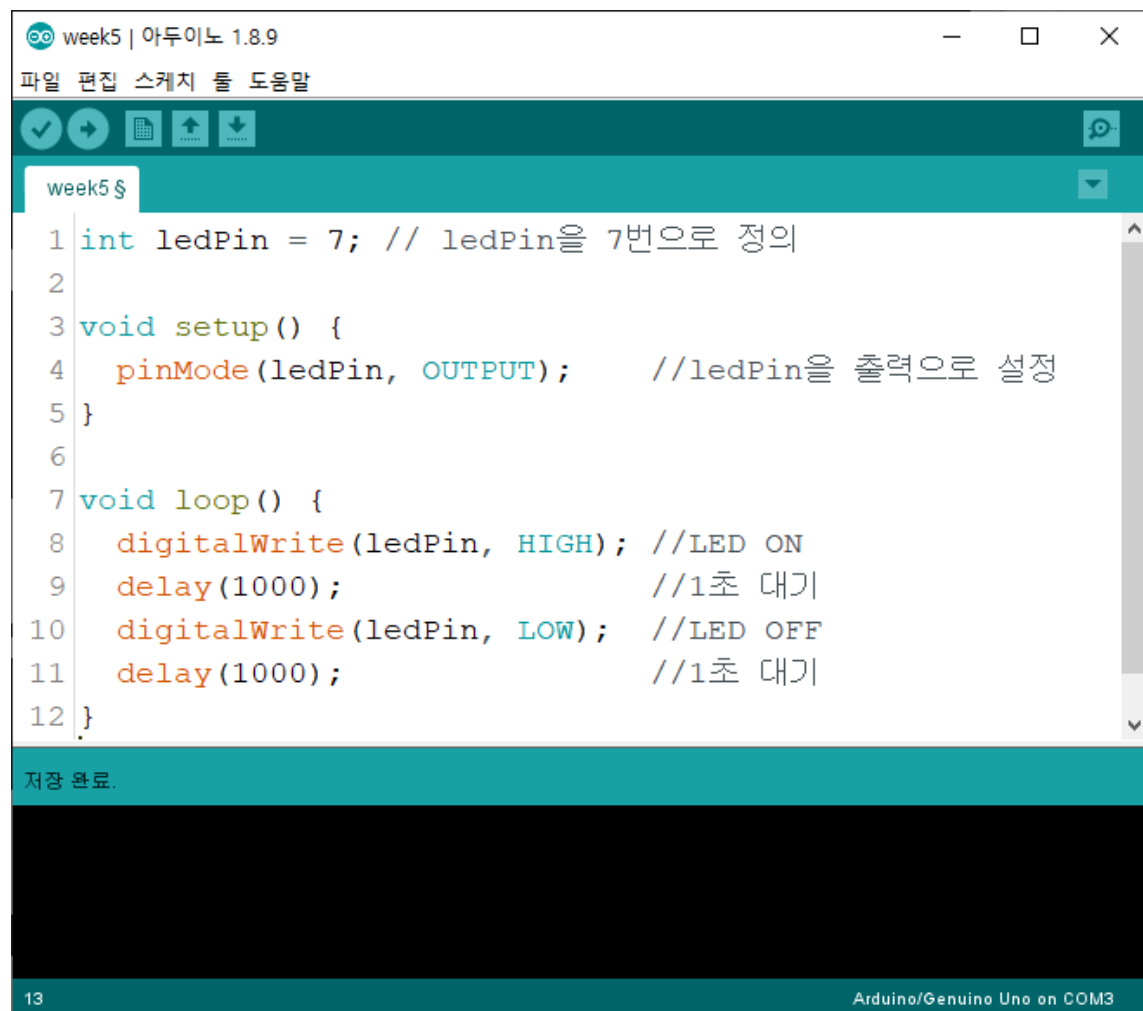
1. 프로젝트를 Fork 한다.
2. master 기반으로 토픽 브랜치를 만든다.
  - % git clone URL
  - % git branch ...
3. 뭔가 수정해서 커밋한다.
4. 자신의 GitHub 프로젝트에 브랜치를 Push 한다.
5. GitHub에 Pull Request를 생성한다.
6. 토론하면서 그에 따라 계속 커밋한다.
7. 프로젝트 소유자는 Pull Request를 Merge 하고 닫는다.

# 협업 실습 구성도



# 준비 사항

- 지난 학기 사물인터넷에서 코딩한 week5/week5.ino 파일 사용

A screenshot of the Arduino IDE interface. The title bar reads "week5 | 아두이노 1.8.9". The menu bar includes "파일", "편집", "스케치", "툴", and "도움말". The toolbar contains icons for checking, running, saving, and other functions. The code editor shows the following code:

```
1 int ledPin = 7; // ledPin을 7번으로 정의
2
3 void setup() {
4   pinMode(ledPin, OUTPUT); //ledPin을 출력으로 설정
5 }
6
7 void loop() {
8   digitalWrite(ledPin, HIGH); //LED ON
9   delay(1000); //1초 대기
10  digitalWrite(ledPin, LOW); //LED OFF
11  delay(1000); //1초 대기
12 }
```

The status bar at the bottom indicates "13" and "Arduino/Genuino Uno on COM3".



# dithacker - computer

1. URL(dithacker/IoT)  
fork
2. 워킹 디렉토리 작성
3. 로컬 repo 설정
4. staging area에 week5  
폴더 올리기
5. 로컬 repo에 commit  
(스냅샷 만들기)하기

```
C:\W>mkdir iot
```

```
C:\W>cd iot
```

```
C:\W\iot>dir
```

```
C 드라이브의 볼륨에는 이름이 없습니다.  
볼륨 일련 번호: EA70-82BF
```

```
C:\W\iot 디렉터리
```

```
2019-12-07 오후 07:36 <DIR> .
```

```
2019-12-07 오후 07:36 <DIR> ..
```

```
0개 파일 0 바이트  
2개 디렉터리 447,626,768,384 바이트 남음
```

```
C:\W\iot>git init
```

```
Initialized empty Git repository in C:/iot/.git/
```

```
C:\W\iot>git add week5
```

```
C:\W\iot>git commit -m "first commit"
```

```
[master (root-commit) 60bf03d] first commit  
1 file changed, 12 insertions(+)  
create mode 100644 week5/week5.ino
```

# dithacker - computer

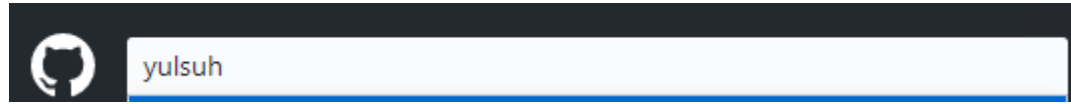
5. 리모트 repo에 연결 설정하기
6. 리모트 repo에 폴더 push 하기

```
C:\Wiot>git remote add origin https://github.com/dithacker/loT.git :
```

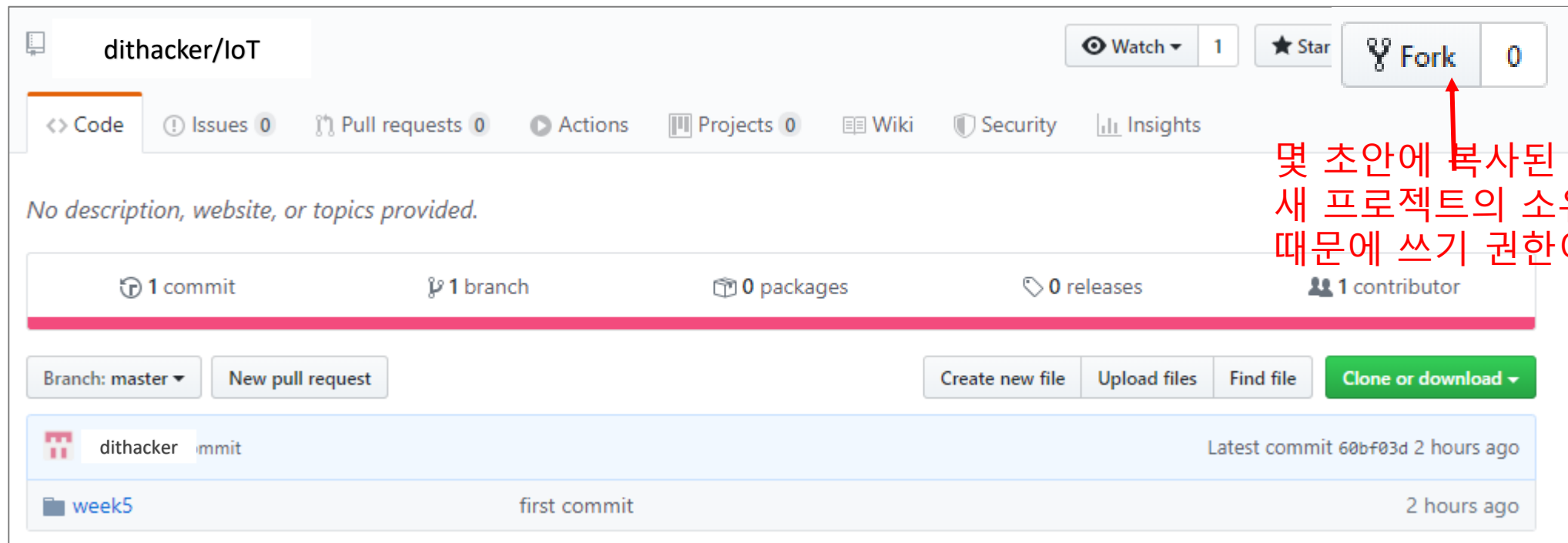
```
C:\Wiot>git push -u origin master
```

# jhkim3217 - github

1. dithacker/loT 검색하기



2. dithacker/loT 에서 fork 하기

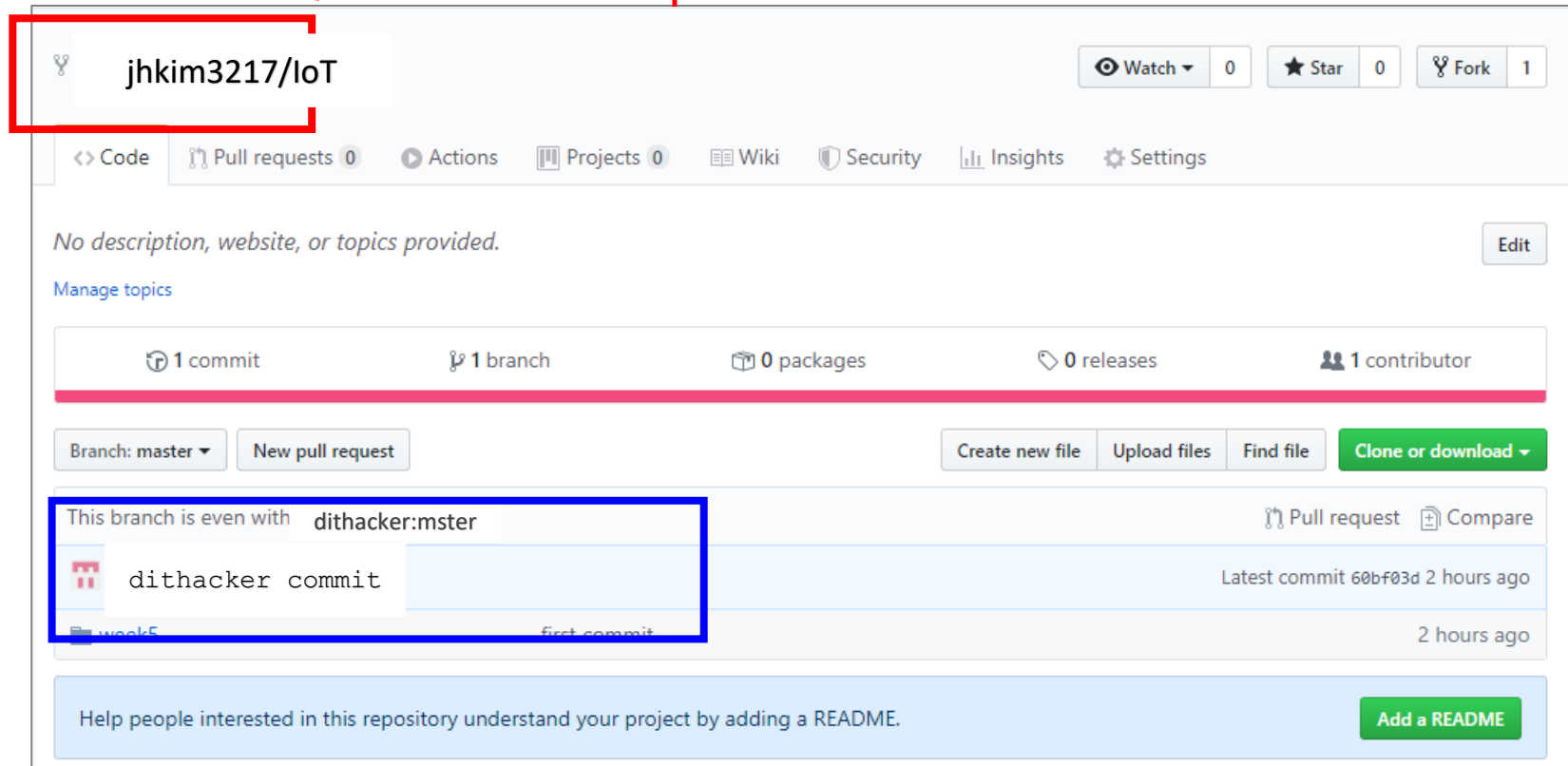


몇 초안에 복사된 프로젝트 페이지  
새 프로젝트의 소유자는 Fork 한  
때문에 쓰기 권한이 있음

# jhkim3217 - github

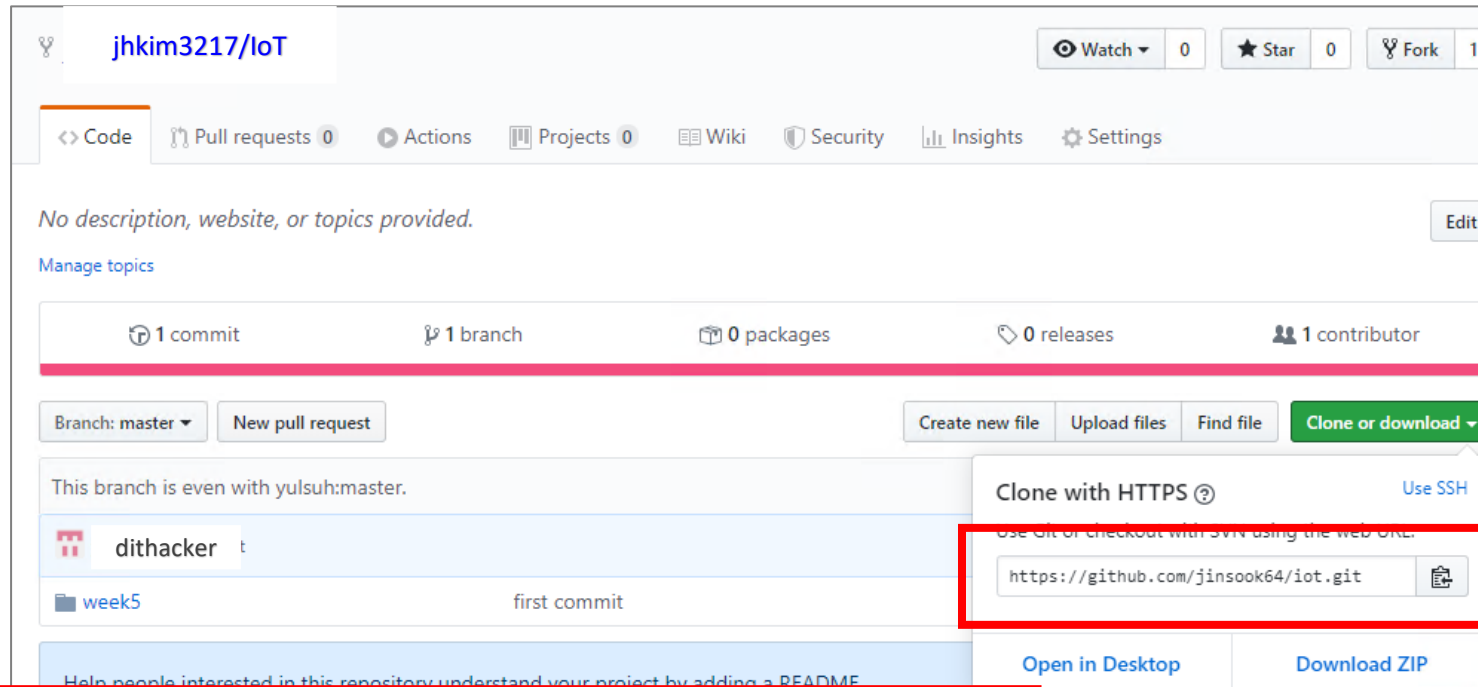
## 1. jhkim3217에 IoT repo 생성

jhkim3217의 github에 dithacker로부터 fork해온 IoT repo가 복사되어 생성



# jhkim3217 - computer

## 1. github의 jhkim3217/IoT 를 로컬 repo로 cloning 하기



주소 복사

```
D:\#>git clone https://github.com/jhkim3217/IoT.git
Cloning into 'IoT'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
```

```
$ git clone -b master --single-branch https://github.com/jhkim3217/IoT.git
```

```
Unpacking objects: 100% (4/4), done.
```

```
D:\#>
```

기본 브랜치가 아닌 master 브랜치에서

# jhkim3217 – computer

1. 로컬 repo 로 이동한다.

```
D:\>cd iot
D:\iot>git status
On branch master
Your branch is up to date with 'origin/master'.
nothing to commit, working tree clean
```

필요하다면 이 때 브랜치를 만들어서 사용할 수 있다.

2. week5폴더의 week5.ino 파일을 다음과 같이 변경한다.

```
1 int ledPin = 9; // ledPin을 9번으로 정의
2
3 void setup() {
4   pinMode(ledPin, OUTPUT); //ledPin을 출력으로 설정
5 }
6
7 void loop() {
8   digitalWrite(ledPin, HIGH); //LED ON
9   delay(3000); //3초 대기
10  digitalWrite(ledPin, LOW); //LED OFF
11  delay(3000); //3초 대기
12 }
```

7 → 9

1000 → 3000

# jikim3217 – computer

2. 변경사항을 저장하고 staging area에 올리고 commit 한다.

```
D:\#iot>git status
On branch master
Your branch is up to date with 'origin/master'.

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:   week5/week5.ino

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

D:\#iot>git add *

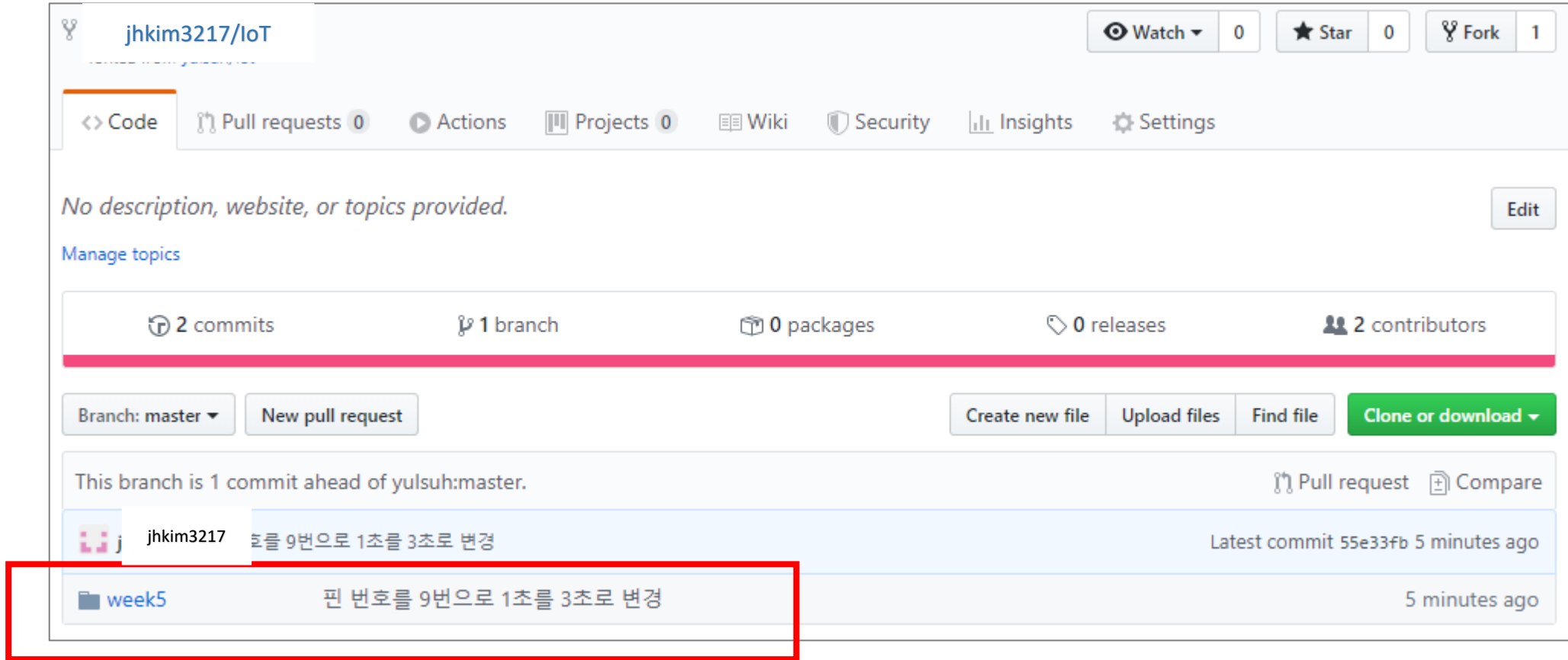
D:\#iot>git commit -m "핀 번호를 9번으로 1초를 3초로 변경"
[master 55e33fb] 핀 번호를 9번으로 1초를 3초로 변경
1 file changed, 3 insertions(+), 3 deletions(-)
```

2. github에 push 한다.

```
D:\#iot>git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 399 bytes | 399.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/jinsook64/iot.git
60bf03d..55e33fb master -> master
```

# jhkim3217 - github

## 1. push 된 사항 확인



jhkim3217/IoT

Watch 0 Star 0 Fork 1

Code Pull requests 0 Actions Projects 0 Wiki Security Insights Settings


No description, website, or topics provided. [Edit](#)


[Manage topics](#)

2 commits 1 branch 0 packages 0 releases 2 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

This branch is 1 commit ahead of yulsuh:master. [Pull request](#) [Compare](#)

 jhkim3217 호를 9번으로 1초를 3초로 변경 Latest commit 55e33fb 5 minutes ago

 week5 핀 번호를 9번으로 1초를 3초로 변경 5 minutes ago



# jhkim3217 - github

## 2. pull request 준비

New pull request

### Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base repository: yulsuh/iot base: master head repository: jinsook64/iot compare: master

✓ Able to merge. These branches can be automatically merged.

Create pull request

Discuss and review the changes in this comparison with others.

1 commit

1 file changed

0 commit comments

1 contributor

Commits on Dec 07, 2019

jhkim3217/iot 권 번호를 9번으로 1초를 3초로 변경

55e33fb

Showing 1 changed file with 3 additions and 3 deletions.

Unified

Split

6 week5/week5.ino

변경 사항 확인


```
... @@ -1,12 +1,12 @@  
1 - int ledPin = 7; // ledPin을 7번으로 정의  
1 + int ledPin = 9; // ledPin을 9번으로 정의  
2  
3 void setup() {
```

# jhkim3217 - github

## 3. 의견을 첨부하여 pull request 요청

### 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](#).

 base repository: dithacker/iot


base: master ▼

←

head repository: jinsook64/iot ▼

compare: master ▼

✓ **Able to merge.** These branches can be automatically merged.



핀 번호를 9번으로 1초를 3초로 변경

Write

Preview

AA B i “ <> ↺ ⋮ ≡ ↕ @ 📌 ↶

LED의 깜박임을 조금 늦추었습니다.

프로젝트 소유자가 판단을 내릴 수 있을 정도로 공을 들여 작성해야 한다. 왜 수정했는지 얼마나 가치 있는지 설명해서 관리자를 설득해야 한다.

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

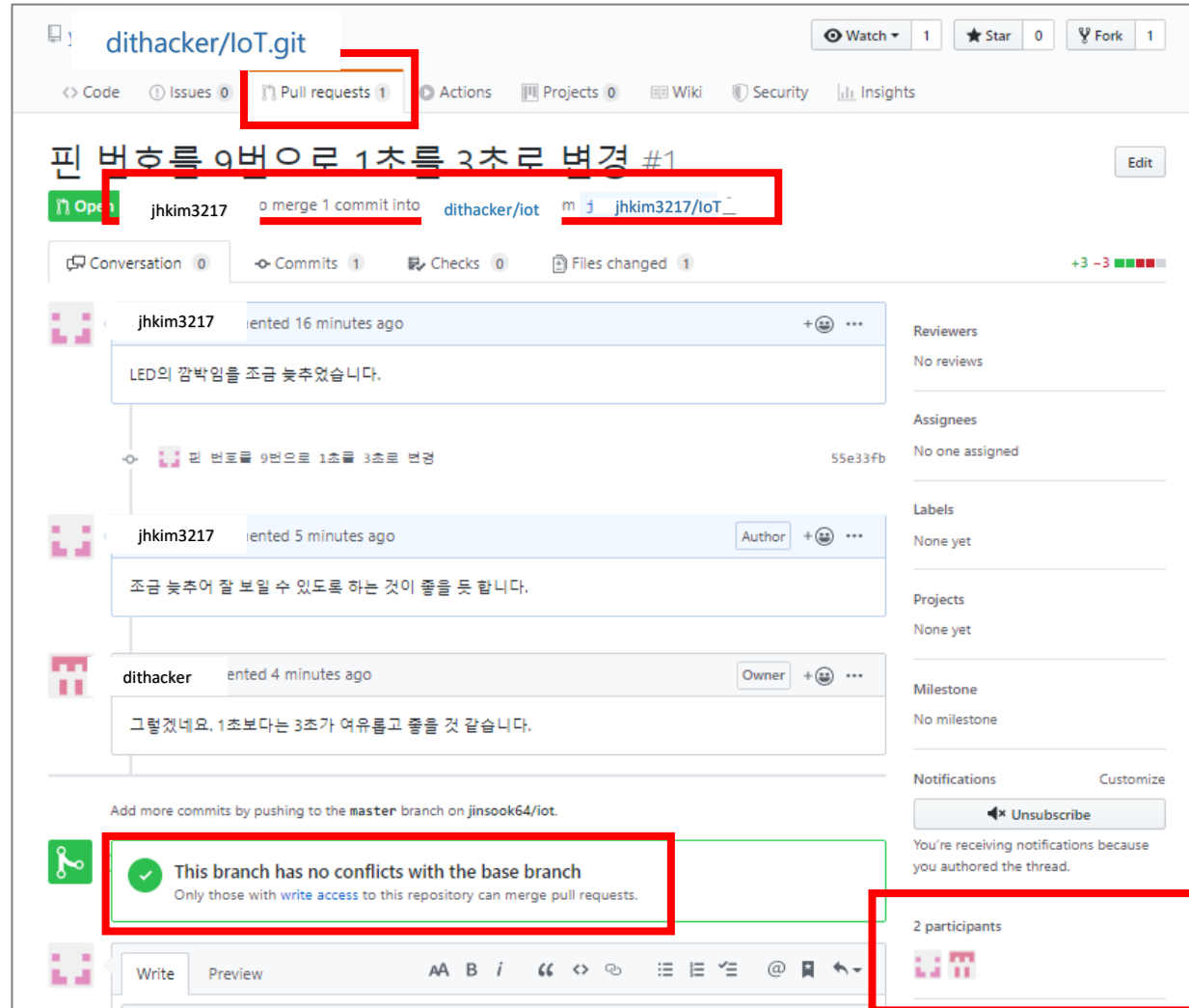
☒ Allow edits from maintainers. [Learn more](#)

Create pull request ▼

**DIT** 동의과학대학교 18

# jhkim3217 - github

3. pull request 요청이 마무리 되기 전까지 토론 페이지에서 논의 할 수 있다.

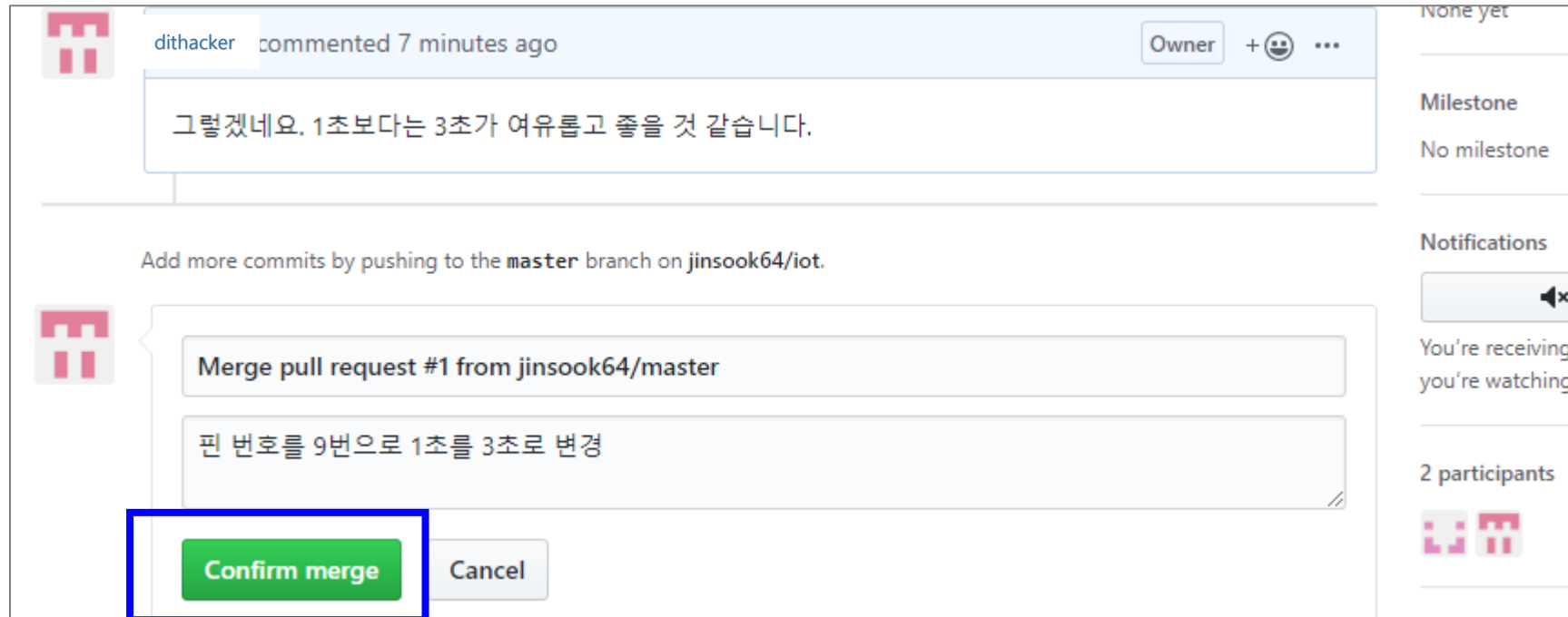


Pull Request는 초기부터 프로젝트 관리자와 소통할 수 있도록 해줌

토론자들

# dithacker - github

1. dithacker/loT 에서도 필요하다면 토론 페이지에서 논의한다.
2. 변경을 받아들이기로 결정을 하면 merge 한다.



# dithacker - github

3. code 탭을 선택하여 merge된 내용을 확인한다.

The image displays two screenshots of a GitHub repository page for 'dithacker/loT.git'.

The left screenshot shows the repository overview. It includes the repository name, navigation tabs (Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings), and a description. A red box highlights a pull request from 'dithacker' to 'jkhim3217/loT:master'.

The right screenshot shows the 'Code' tab. It displays the file 'week5/week5.ino' and its content, which is a C++ program for an LED. The code is as follows:

```
1 int ledPin = 9; // ledPin을 9번으로 정의
2
3 void setup() {
4   pinMode(ledPin, OUTPUT); //ledPin을 출력으로 설정
5 }
6
7 void loop() {
8   digitalWrite(ledPin, HIGH); //LED ON
9   delay(3000); //1초 대기
10  digitalWrite(ledPin, LOW); //LED OFF
11  delay(3000); //1초 대기
12 }
```

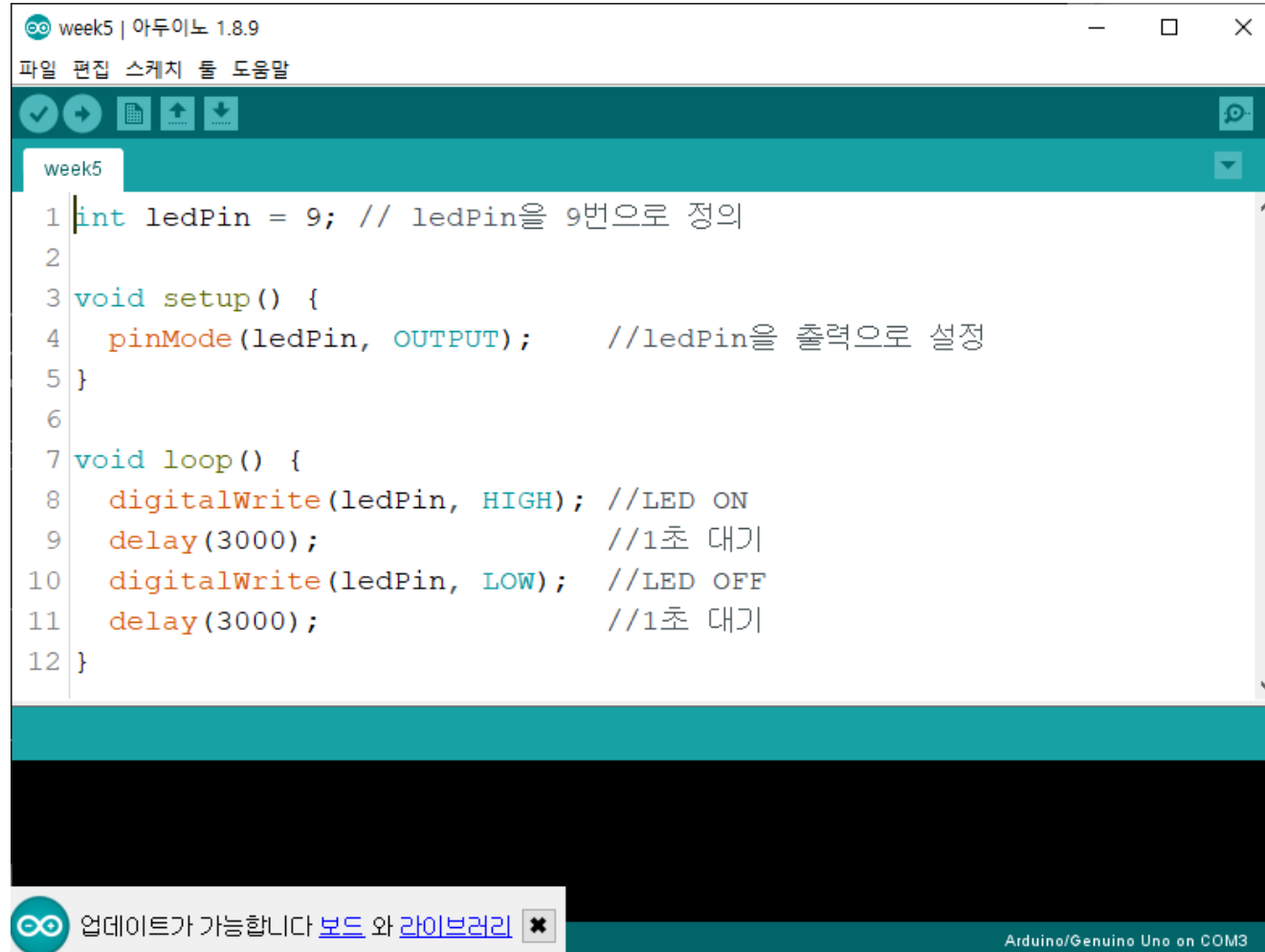
# dithacker - computer

1. dithacker/IoT를 로컬 repo로 가져온다.(동기화)

```
C:\#iot>git pull origin master
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 5 (delta 1), reused 4 (delta 1), pack-reused 0
Unpacking objects: 100% (5/5), done.
From https://github.com/yulsuh/iot
* branch                master      -> FETCH_HEAD
   60bf03d..52a6bb2      master      -> origin/master
Updating 60bf03d..52a6bb2
Fast-forward
 week5/week5.ino | 6 +++---
1 file changed, 3 insertions(+), 3 deletions(-)
```

# dithacker - computer

## 2. github로부터 가져온 파일 확인

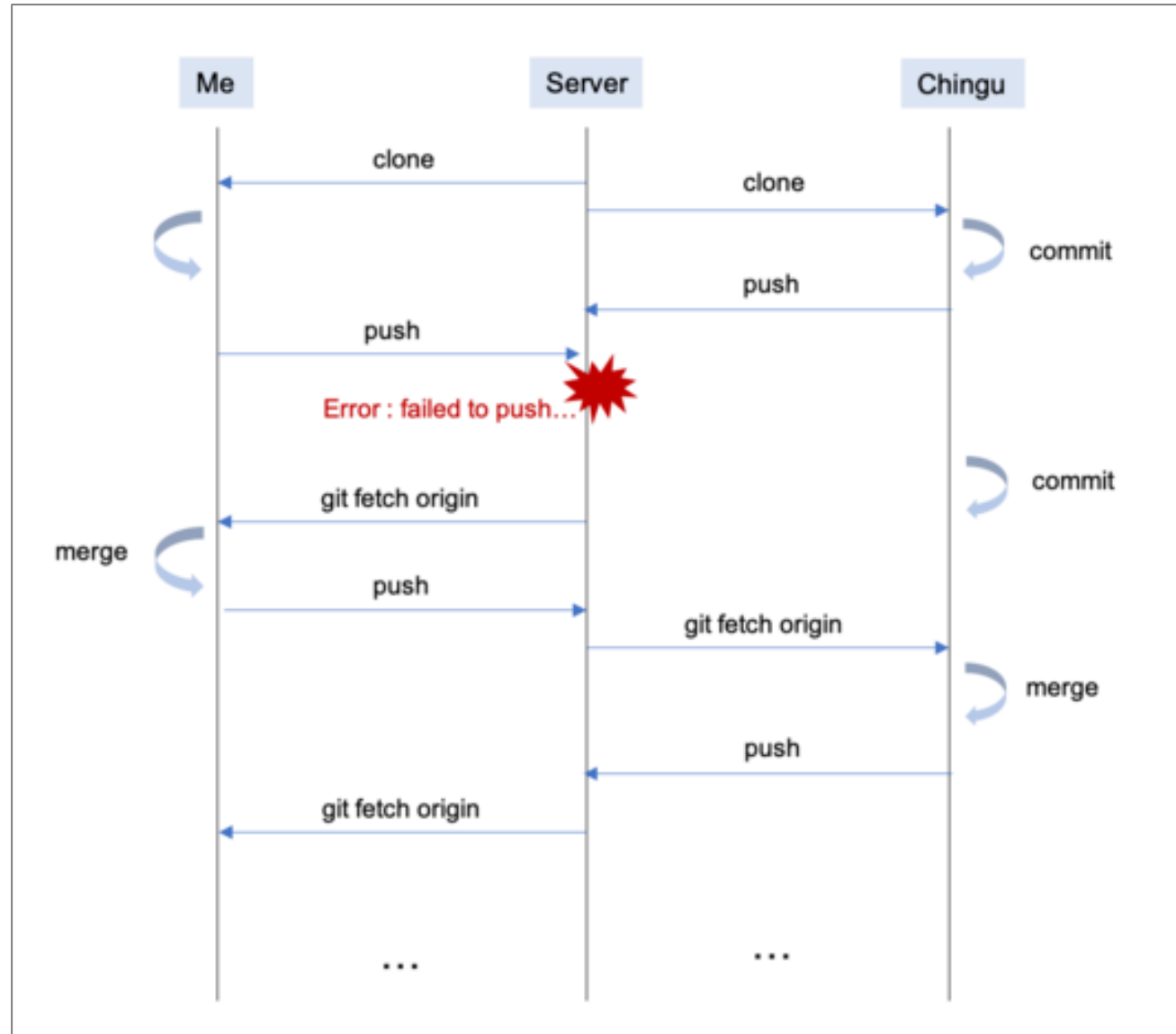


```
week5 | 아두이노 1.8.9
파일 편집 스케치 툴 도움말

week5
1 int ledPin = 9; // ledPin을 9번으로 정의
2
3 void setup() {
4   pinMode(ledPin, OUTPUT); //ledPin을 출력으로 설정
5 }
6
7 void loop() {
8   digitalWrite(ledPin, HIGH); //LED ON
9   delay(3000); //1초 대기
10  digitalWrite(ledPin, LOW); //LED OFF
11  delay(3000); //1초 대기
12 }

업데이트가 가능합니다 보드와 라이브러리
Arduino/Genuino Uno on COM3
```

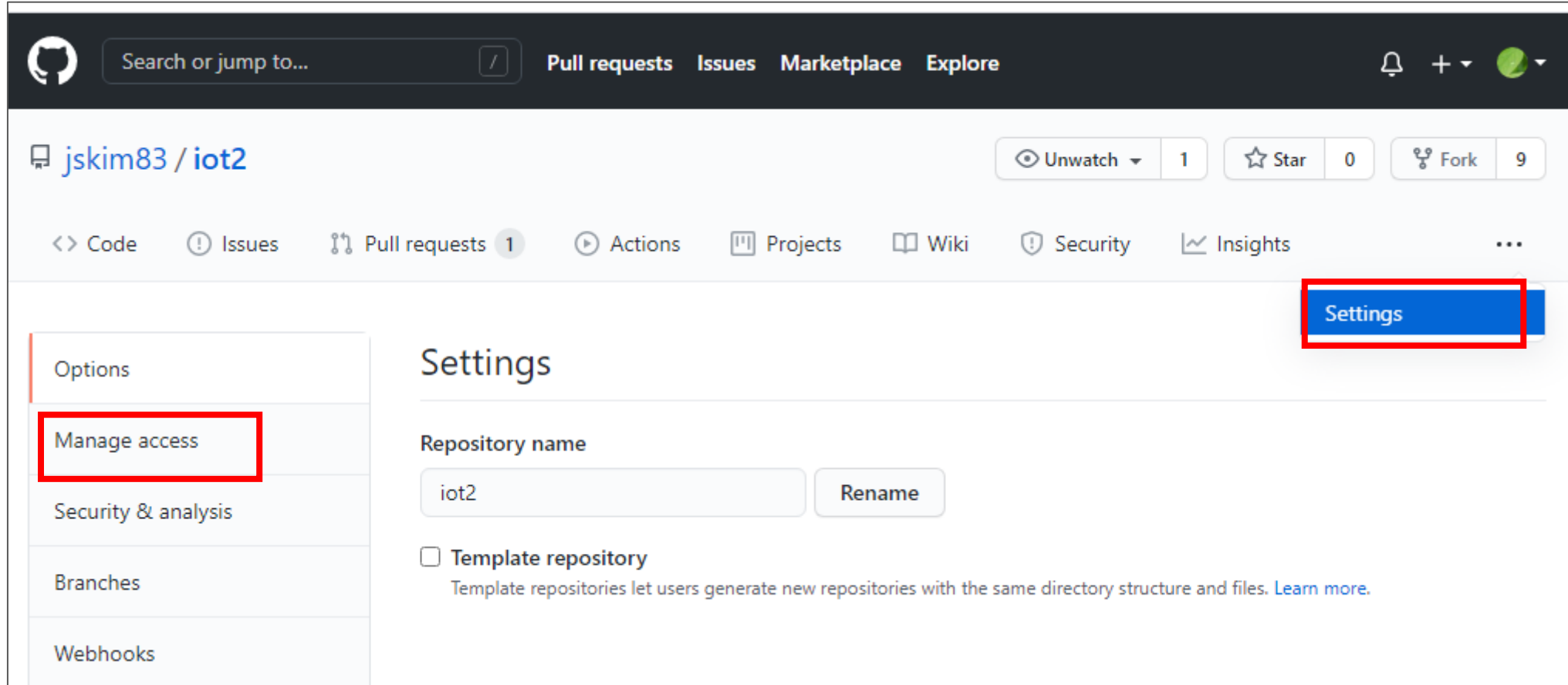
# 유의할 상황





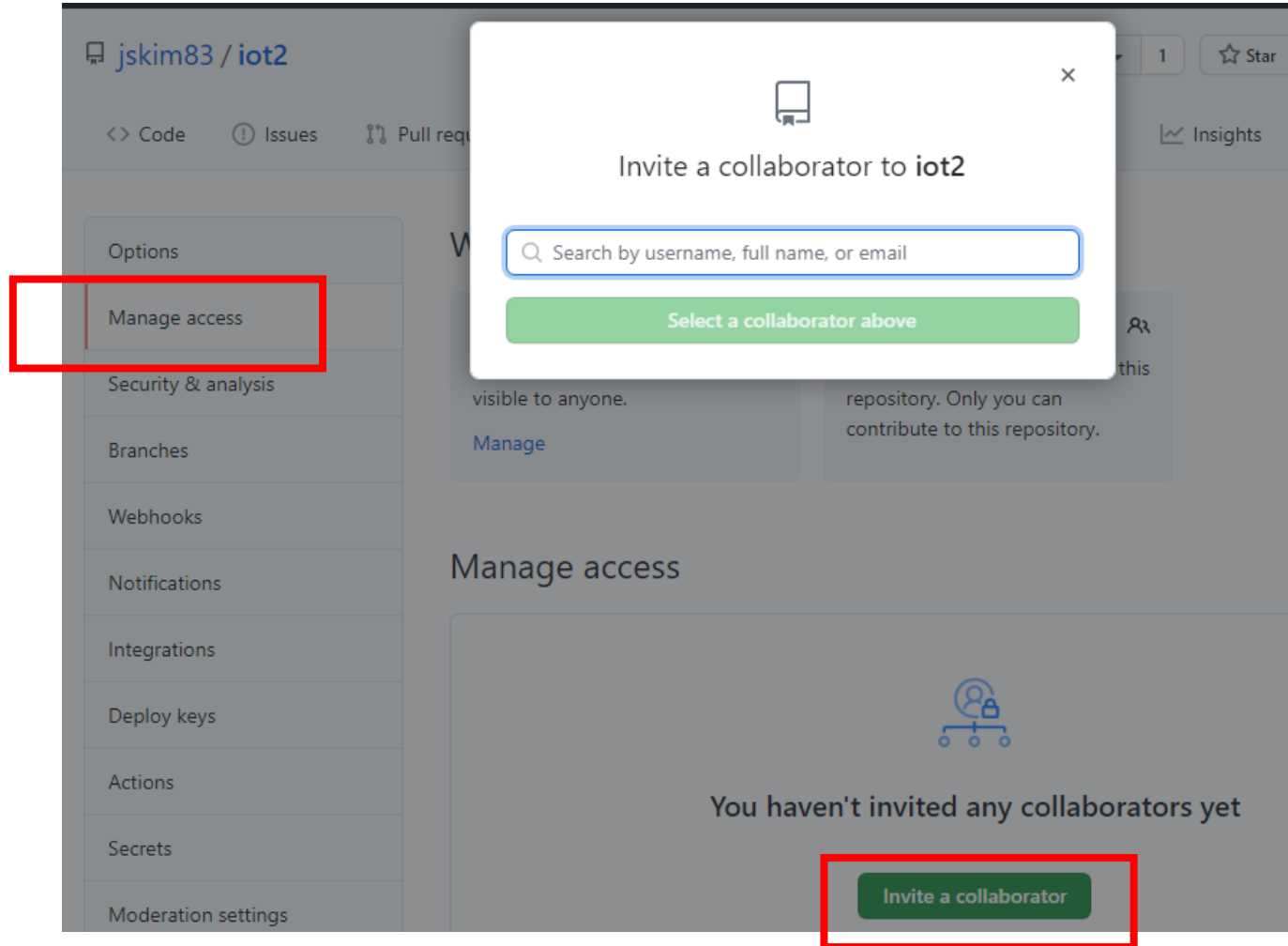
# github 프로젝트에 동료 추가하기

- 해당 저장소에 commit 권한을 주고 싶은 동료가 있으면 "collaborator"로 추가
- 해당 저장소 – settings – Collaborator – 사용자 id 입력 - Add



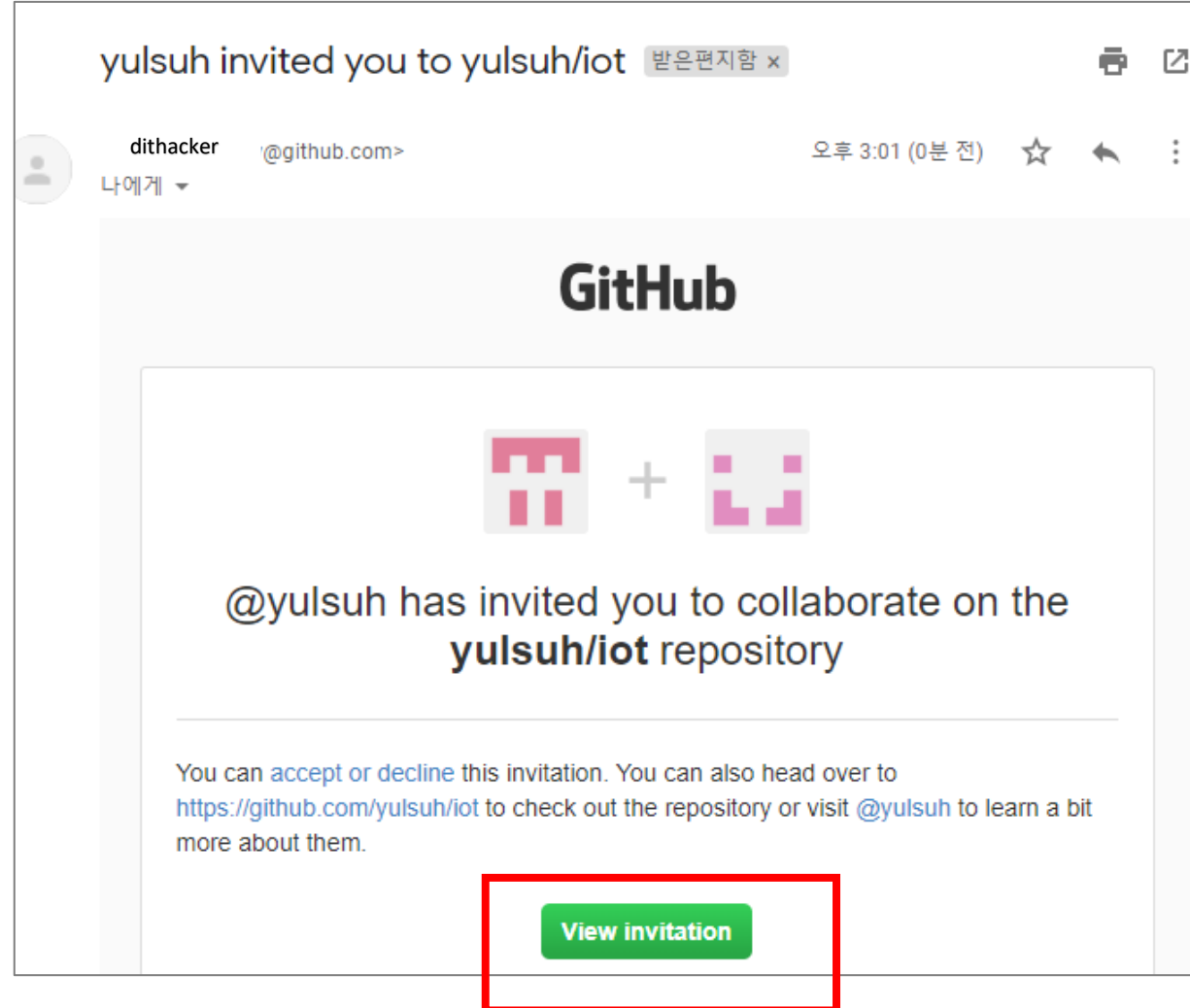
# github 프로젝트에 동료 추가하기

- Collaborator 를 추가하면 해당 동료에게 이메일이 발송된다.



# github 프로젝트에 동료 추가하기

- 이메일을 확인하고 “view Invitation” 버튼을 눌러 초대를 수락한다.



# 요약

- 참고 사이트 : <https://git-scm.com/book/ko/v2>

| 명령어                                       | 설명  |
|---|---|
| <b>git clone</b>                          | 클라이언트 상에 아무것도 없을 때 서버의 프로젝트를 내려받는 명령어   |
| <b>git remote</b><br><b>git remote -v</b> | 현재 프로젝트에 등록된 리모트 저장소를 확인할 수 있다. Clone 하면 `origin`이라는 리모트 저장소가 자동으로 등록 된다.   |
| <b>git remote add</b>                     | 기존 워킹 디렉토리에 새 리모트 저장소를 추가한다.  |
| <b>git remote show</b>                    | 리모트 저장소의 구체적인 정보를 확인한다.   |
| <b>git remote rename</b>                  | 리모트 저장소의 이름을 변경한다.  |
| <b>git remote remove</b>                  | 리모트 저장소를 삭제한다.  |
| <b>git push origin master</b>             | 원격 저장소 master branch에 commit을 저장한다. 리모트 저장소에 쓰기 권한이 있고, Clone 하고 난 이후 아무도 원격 저장소에 Push 하지 않았을 때만 사용할 수 있다. Clone 한 사람이 여러 명 있을 때, 다른 사람이 Push 한 후에 Push 하려고 하면 Push 할 수 없다. |
| <b>git fetch &lt;remote&gt;</b>           | 이 명령은 로컬에는 없지만, 리모트 저장소에는 있는 데이터를 모두 가져온다. 자동으로 merge 하지는 않는다.  |
| <b>git pull</b>                           | git서버에서 최신 코드 받아와 merge 한다.   |

# 실습

- 2사람씩 짝을 짓는다.
- 이번 학기 프로그래밍 실습 내용을 선택한다.
- 각자 클라이언트(fork)와 서버(pull request)의 역할을 맡아 진행한다.
- conversation 에서 5번 이상의 논의를 한다.
- 실습이 끝나면 역할을 바꾸어 진행한다.