

codingOn x posco

K-Digital Training 신재생에너지 활용 IoT 과정

Git과 협업

오늘 수업은?

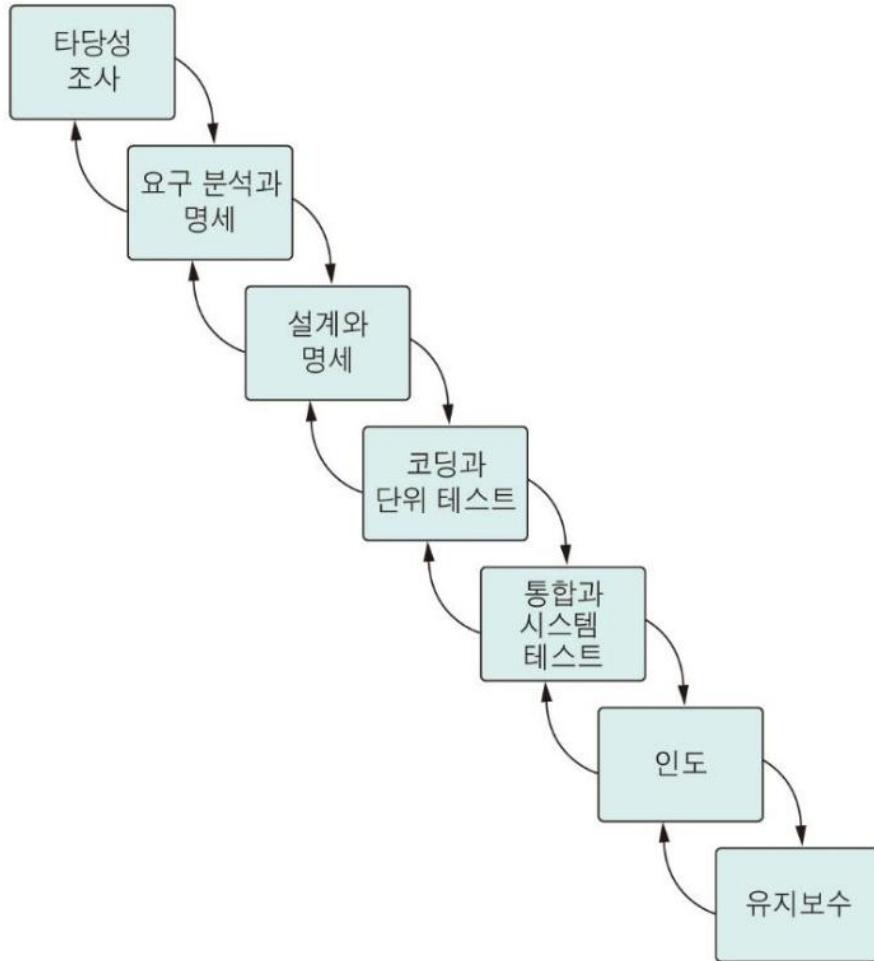
- 지난 시간에는 혼자 코드를 수정하여 원격저장소에 코드를 푸시하면서 Git을 왜 사용하고 어떻게 사용하는지에 대해서 알아보았습니다.
- 이번 시간에는 팀에서 협업을 진행할 때 Git을 어떻게 사용하는지 알아보겠습니다.

학습목표

- 협업 방법에 대해서 이해한다.
- Git의 협력자 등록과 복제 원격저장소를 생성할 수 있다.
- Pull request를 사용할 수 있다.

1. 협업 방법론 이해하기

Waterfall



Waterfall Model (폭포수 모델)

- 가장 익숙한 소프트웨어 개발 기법
- 고전적인 소프트웨어 생명 주기
- 병행 수행되지 않고 순차적으로 수행

Waterfall Model의 장단점

장점

- 단순한 모델이라 이해가 쉽다.
- 단계별로 정형화된 접근이 가능해 문서화가 가능하다.
- 프로젝트 진행 상황을 한눈에 명확하게 파악 가능하다.

단점

- 변경을 수용하기 어렵다.
- 시스템의 동작을 후반에 가야지만 확인이 가능하다.
- 대형 프로젝트에 적용하는 것이 부적합하고, 일정이 지연될 가능성이 크다.

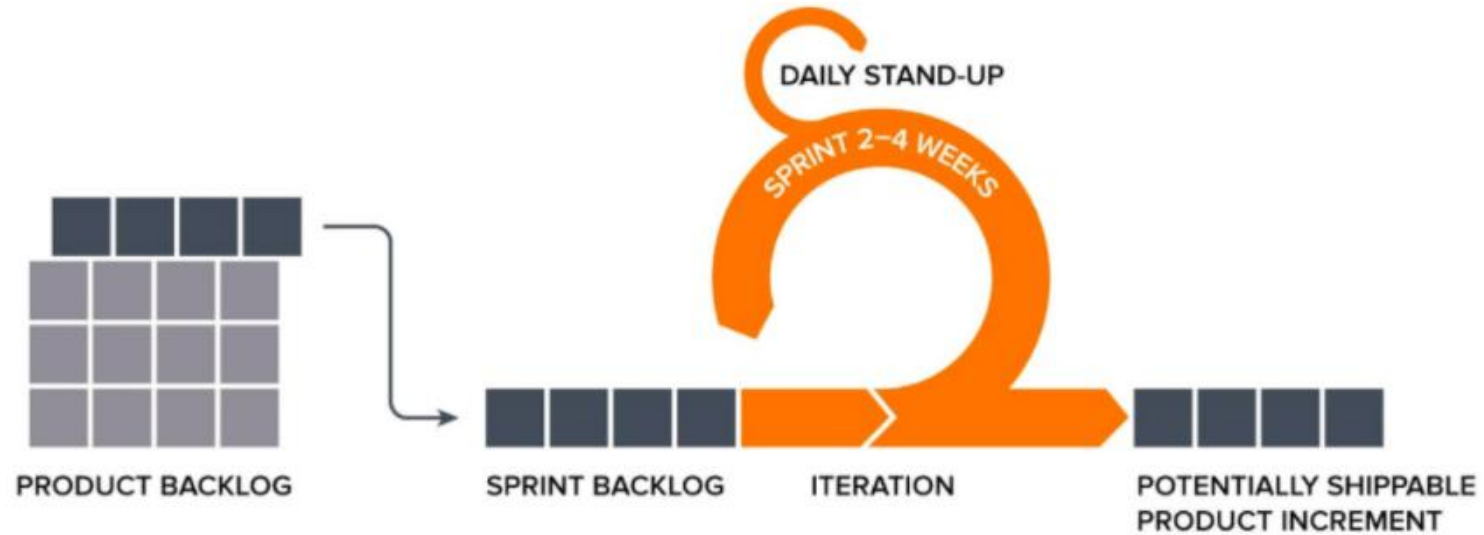
Agile

agile 미국식 ['ædʒɪl]  영국식 ['ædʒaɪl] 

1. [형용사] 날렵한, 민첩한 (=nimble)
2. [형용사] (생각이) 재빠른, 기민한

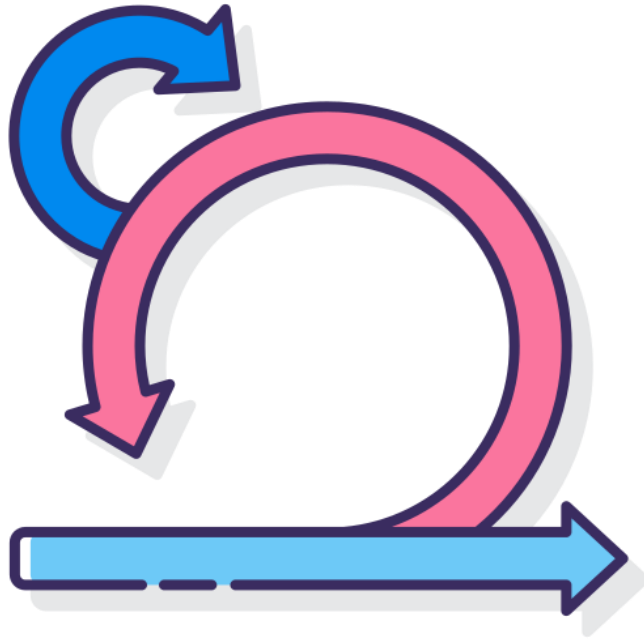
- 짧은 주기의 개발 단위를 반복해 하나의 큰 프로젝트를 완성해 나가는 것
- 🔑 협력과 피드백
- 🔑 유연한 일 진행 + 빠른 변화 대응

Agile



- 짧은 주기로 설계, 개발, 테스트, 배포 과정을 반복
- 요구 사항을 작은 단위로 쪼개 그에 대한 솔루션을 만들고, 빠르게 보여줌으로써 요구 사항에 대한 검증을 진행

Agile 방법론



Scrum(스크럼)

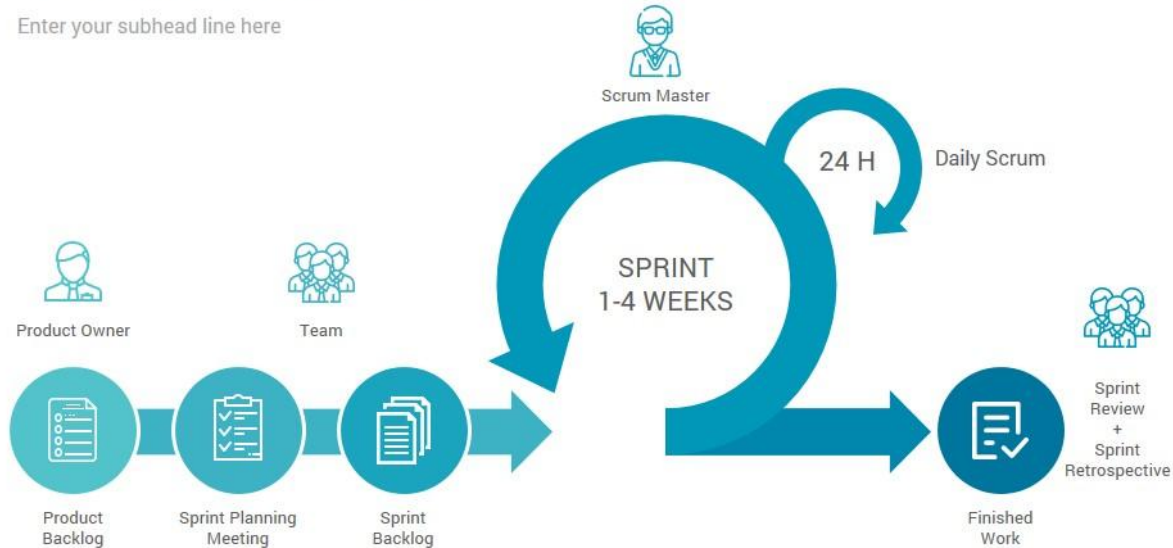


Kanban(칸반)

Scrum(스크럼)

Scrum Process

Enter your subhead line here

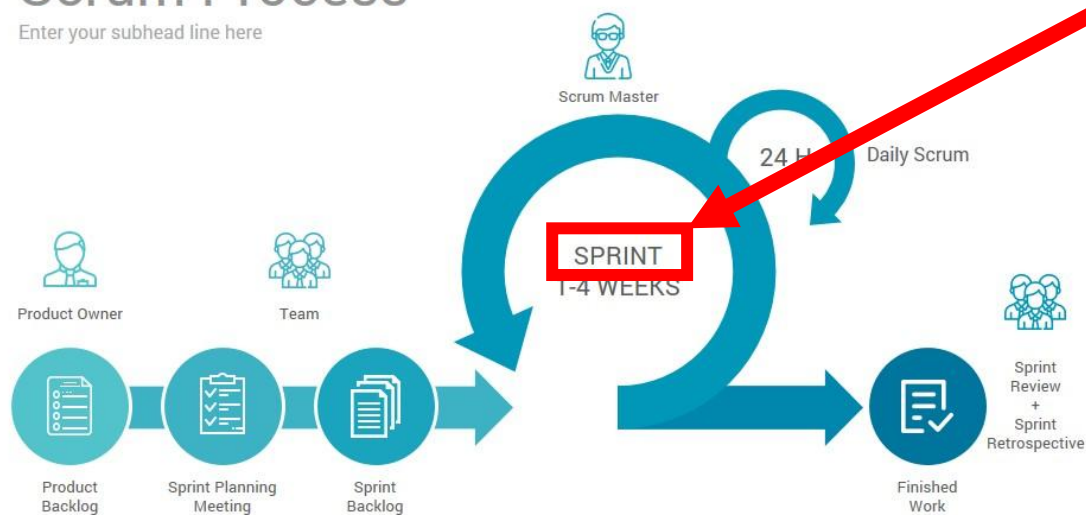


- 개발자와 고객 사이의 지속적인 커뮤니케이션을 통해 요구사항을 수용
- 고객이 결정한 사항을 가장 우선적으로 시행
- 팀원들과 주기적인 미팅을 통해 프로젝트를 점검
- 주기적으로 제품 시현을 하고 고객으로부터 피드백 수용

Scrum(스크럼)

Scrum Process

Enter your subhead line here



작은 기능에 대해

“계획, 개발, 테스트, 기능 완료”

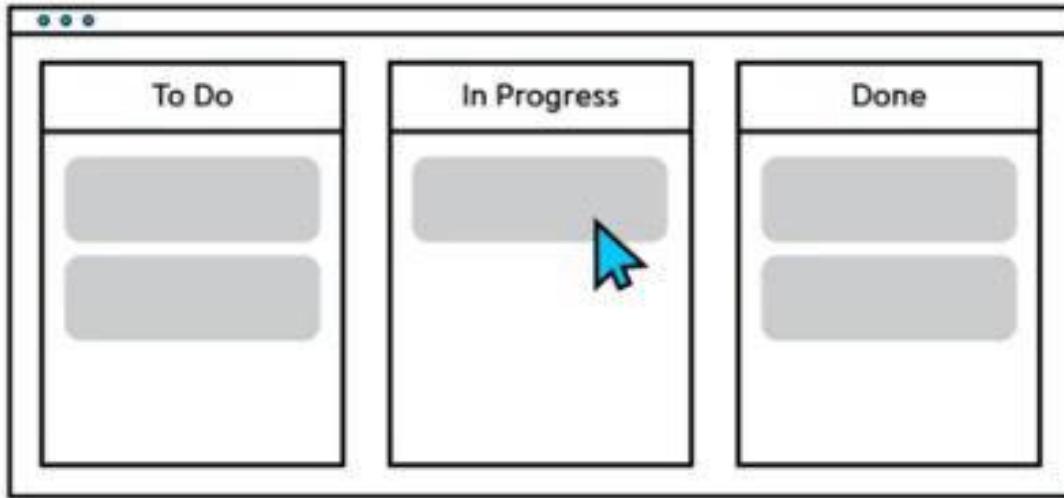
에 대해 주기적으로 시행하는 것

일반적인 스프린트 주기 : 1~2주

Kanban(칸반)

칸반(Kanban)이란?

칸반(Kanban)은 단계별 작업 현황을 열(column) 형식의 보드 형태로 시각화하는 프로젝트 관리 방법을 말합니다.



장점

- 업무 흐름의 시각화
- 진행 중 업무의 제한
- 명시적 프로세스 정책 수립
- 업무 흐름의 측정과 관리

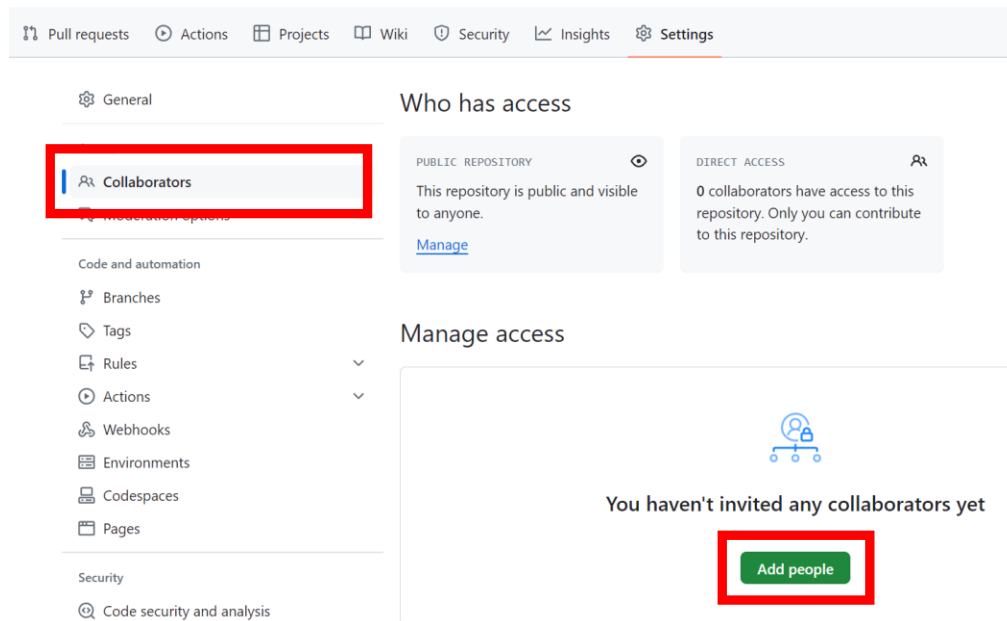
2. Git과 협업

Git 협업 이해하기

- 협업은 2명 이상의 개발자가 하나의 프로젝트에서 서로 다른 작업을 하는 것을 말함
- 브랜치는 하나의 main브랜치에서 여러가지 작업을 위한 브랜치를 생성하는 것으로 소규모 협업에 적당
- 하지만 개발자가 많아져서 100명 이상이 브랜치를 생성하고 올리는 작업을 한다면?
- 브랜치가 복잡해지고 관리하기 힘들게 될 것
- 이런 불편한 점을 해결 하기 위한 방법은 저장소를 통째로 복제(fork)하면 됨

소규모 협업하기

- Git 원격저장소에 코드를 직접 푸시할 수 있는 사람은 소유자만 가능
- 다른 사람들이 내 원격저장소에 브랜치를 만들어서 코드를 푸시하게 하려면 협력자로 등록
- 원격저장소에서 Settings -> Collaborators



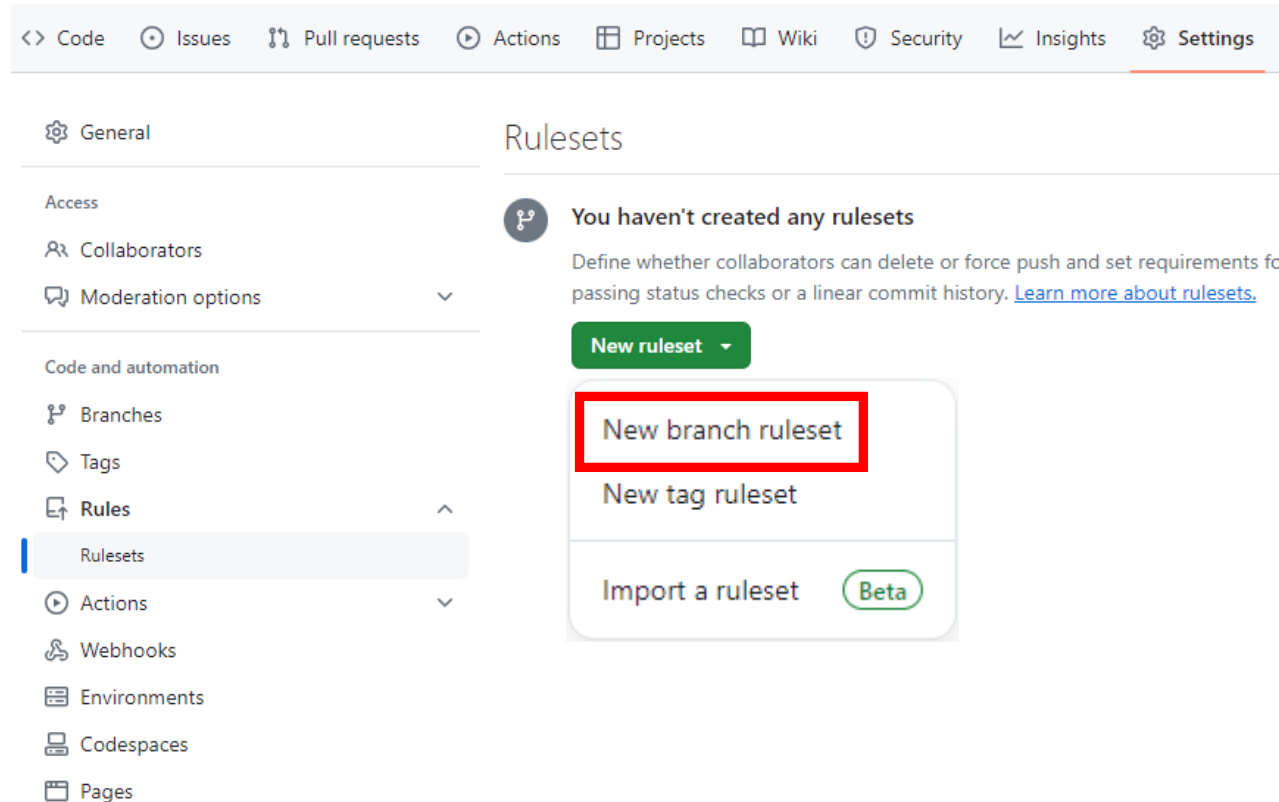
권한을 줄 작업자를 추가하게 되면 해당 작업자에게 승인메일이 보내짐. 작업자가 메일을 승인하게 되면 해당 원격저장소에 푸시할 수 있게 됨

대규모 협업하기

- 원격저장소를 복제하여 협업하기
- 원본의 원격저장소를 통째로 복제하게 되면 모든 커밋 이력도 복제되어 또 하나의 원격저장소가 생성됨
- 원본의 원격저장소에 영향을 끼치지 않으므로 복제된 원격저장소에서 코드를 수정해서 작업할 수 있음
- 소규모 협업에서도 협력자로 등록하지 않고 복제하여 사용해도 됨

원격저장소 브랜치 규칙 생성하기

- 규칙을 생성하여 내 원격저장소에 병합을 제한할 수 있음
- 이는 무작위로 병합하여 코드가 오류나는 것을 방지



원격저장소 브랜치 규칙 생성하기

Ruleset Name

rules

규칙의 이름

Enforcement status

⌚ Disabled

규칙의 사용 유/무

Bypass list

+ Add bypass

이 규칙에서 제외할 팀, 권한, 앱

Exempt roles, teams, or apps from this ruleset by adding them to the bypass list.

Bypass list is empty

Targets

Which branches do you want to make a ruleset for?

Target branches

Branch targeting determines which branches will be protected by this ruleset. Use inclusion patterns to expand the list of branches under this ruleset. Use exclusion patterns to exclude branches.

Branch targeting criteria

Add target

이 규칙을 적용할 브랜치

Branch targeting has not been configured

Applies to 0 targets.

원격저장소 브랜치 규칙 생성하기

- ☒ **Require a pull request before merging**
Require all commits be made to a non-target branch and submitted via a pull request before they can be merged.

병합전 pull request를 사용

Hide additional settings ^

Required approvals

리뷰어에 등록된 사용자들이
승인을 해줘야 병합가능

1

The number of approving reviews that are required before a pull request can be merged.

- ☒ **Dismiss stale pull request approvals when new commits are pushed**
New, reviewable commits pushed will dismiss previous pull request review approvals.

리뷰어가 승인 후 병합전 코드가
변경 되었다면 다시 승인을 요청

- ☐ **Require review from Code Owners**
Require an approving review in pull requests that modify files that have a designated code owner.

파일의 코드 소유자가 있는 경우 해당
파일은 코드 소유자의 승인 필요

- ☐ **Require approval of the most recent reviewable push**
Whether the most recent reviewable push must be approved by someone other than the person who pushed it.

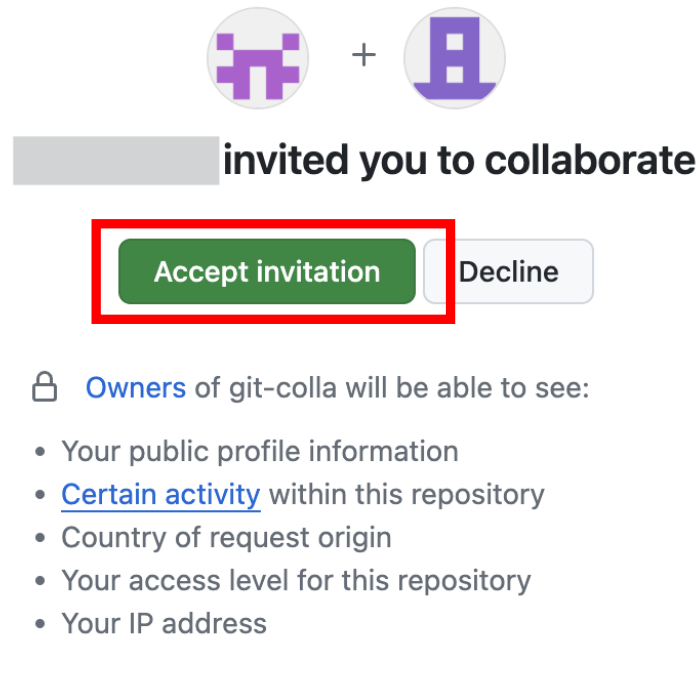
코드 작성자 외 다른 리뷰어가 승인

- ☐ **Require conversation resolution before merging**
All conversations on code must be resolved before a pull request can be merged.

모든 리뷰가 해결 되어야 병합 가능

실습. 다른사람을 협력자로 등록하기

- 옆 사람과 아이디를 공유하여 본인의 Git 저장소에 2명씩 다른사람을 추가하기
- 추가하는 방법 : 본인의 원격저장소에서 Settings -> Collaborators



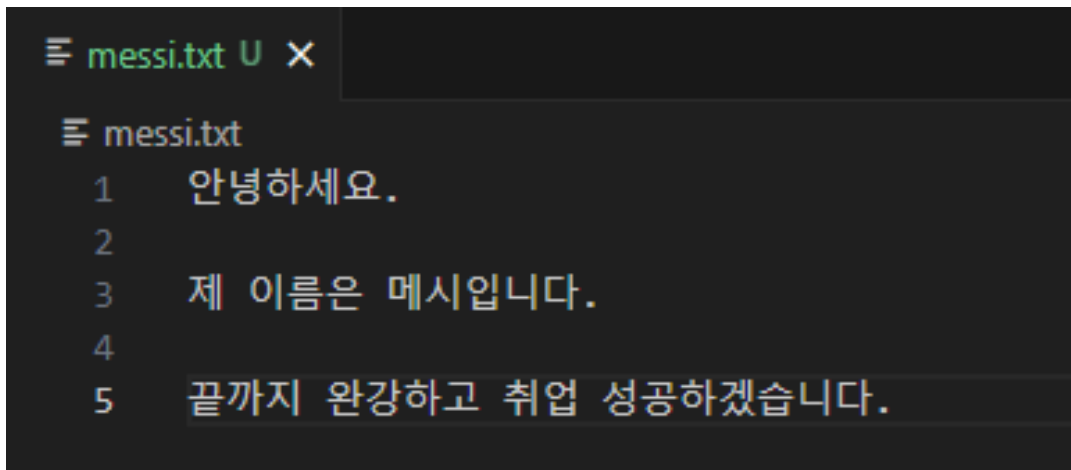
초대받으셨으면 본인의 메일함에 메일이 오게 됩니다.
accept invitation 누르셔서 초대한 사람의 Git 저장소에
협력자로 등록하세요

- 초대받은 저장소 확인은 github 내 프로필 -> Settings -> Repositories

실습. 초대받은 저장소 clone

- 초대받은 수 만큼 본인 컴퓨터에 각각 다른이름으로 폴더를 생성
- 각 폴더에 초대받은 Git 저장소를 clone
- 본인이름으로 브랜치와 텍스트 파일 생성
- 텍스트 파일에는 본인 소개글을 적고 푸쉬하기

예)

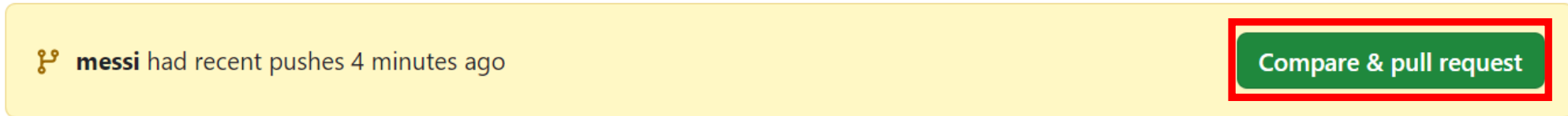


```
messi.txt U x
messi.txt
1  안녕하세요.
2
3  제 이름은 메시입니다.
4
5  끝까지 완강하고 취업 성공하겠습니다.
```

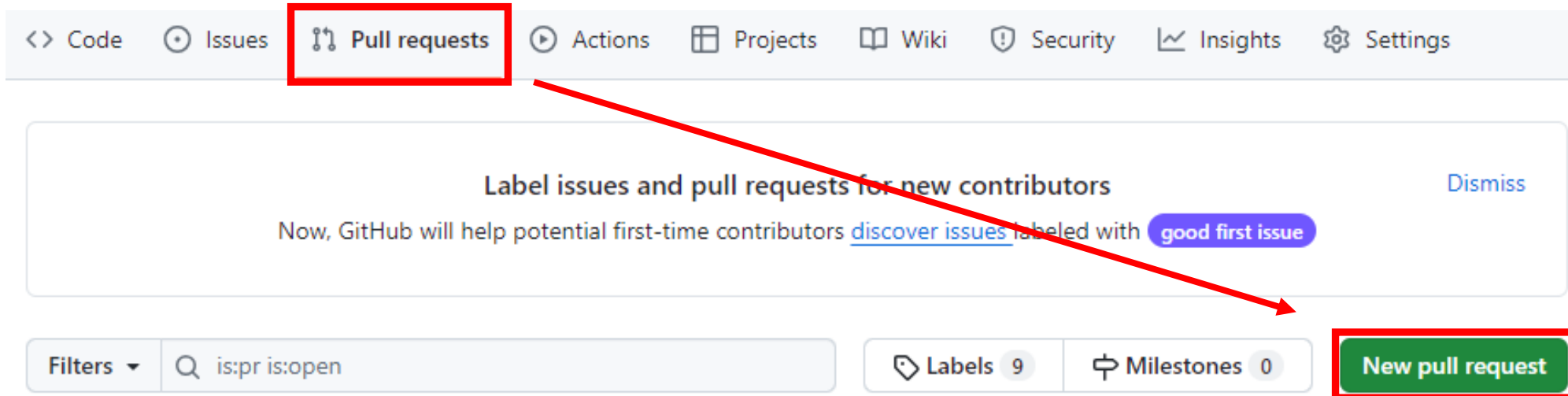
3. Pull request

Pull request 란?

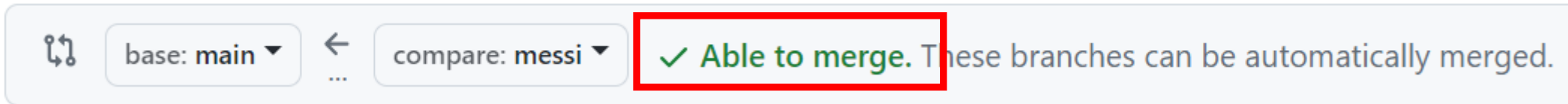
- github에서 코드를 병합하는 방법
- 브랜치를 최근에 푸시했을때 github에 아래와 같은 메시지가 출력됨



- 그 외에 풀 리퀘스트 하고 싶다면 New pull request 클릭



Pull request



- compare에 있는 브랜치가 base의 브랜치로 병합된다는 뜻
- 두개의 브랜치가 병합되기 전 코드 상태를 자동으로 비교 시켜 줌

메세지

- Able to merge : 바로 merge가 가능
- Can't automatically merge : 충돌 발생으로 충돌 해결 후 merge 가능

Pull request

Add a title

저는 메시입니다 풀 리퀘스트 제목

Add a description

Write

Preview

H B I ≡ <> 🔗 | ≡ ≡ ≡ | 📎 @ ↗ ↶ 🗑

제 소개를 추가하였습니다 협업하는 동료에게 어떤 내용을 작업하였는지 적는곳

확인 부탁드립니다.

Pull request

Reviewers

No reviews

Assignees

No one—[assign yourself](#)

Labels

None yet

Create pull request

Draft pull request

동료들에게 해당 풀 리퀘스트에 대해 검토해 달라고 요청할 때 여기에 동료들을 추가함(선택사항)

해당 풀 리퀘스트 담당자 - 보통 본인(선택사항)

해당 풀 리퀘스트에 라벨 달아줄 때 사용(선택사항)

풀 리퀘스트 생성(일반적)

풀 리퀘스트 생성하지만 merge는 되지않음
pull request는 하지만 아직 작업 중이라는 뜻
보통 코드에 대해 여러명이 토론을 진행할때 사용

Pull request 생성(요청자)

저는 메시입니다 #1

Edit <> Code

Open wants to merge 1 commit into main from messi

Conversation 0

Commits 1

Checks 0

Files changed 1

+5 -0



commented 44 minutes ago

Collaborator

제 소개를 추가하였습니다.

확인 부탁드립니다.



저는 메시입니다

449acec

added the help wanted label 44 minutes ago

requested a review from 44 minutes ago

self-assigned this 44 minutes ago

Reviewers



Still in progress? [Convert to draft](#)

Assignees



Labels

help wanted

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

Notifications

Customize

Unsubscribe

You're receiving notifications because you authored the thread.

pull request를
요청한 사용자의 화면

reviewer가 승인하지
않은 상태

review 등록(리뷰어)

저는 메시입니다 #1

Edit <> Code

reviewer 화면

Open wants to merge 1 commit into main from messi

Conversation 0 Commits 1 Checks 0 Files changed 1

+5 -0

Changes from all commits File filter Conversations Jump to

0 / 1 files viewed

Review in codespace

Review changes

Finish your review

Write

Preview

H

B

I

≡

<>

🔗

≡

≡

≡

📎

@

📁

←

네 코드 승인합니다.

Markdown is supported

Paste, drop, or click to add files

☐ Comment

Submit general feedback without explicit approval.

☒ Approve

Submit feedback and approve merging these changes.

☐ Request changes

Submit feedback that must be addressed before merging.

Submit review

- Comment : 단순히 코멘트만 남김
- Approve : 변경사항 적용 승인
- Request changes : 변경을 다시 요청(거부)

review 등록

✓ [User] approved these changes now [View reviewed changes](#)

[User] left a comment Owner ...

코드 승인합니다.

🗨️

✓ **Changes approved** [Show all reviewers](#)
1 approving review by reviewers with write access. [Learn more about pull request reviews.](#)

✓ 1 approval ▾

✓ **This branch has no conflicts with the base branch**
Merging can be performed automatically.

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

승인 완료

🔴 [User] requested changes now [View reviewed changes](#)

[User] left a comment • edited ▾ Owner ...

코드 수정해서 다시 올려주세요

🗨️

🔴 **Changes requested** [Show all reviewers](#)
1 review requesting changes by reviewers with write access. [Learn more about pull request reviews.](#)

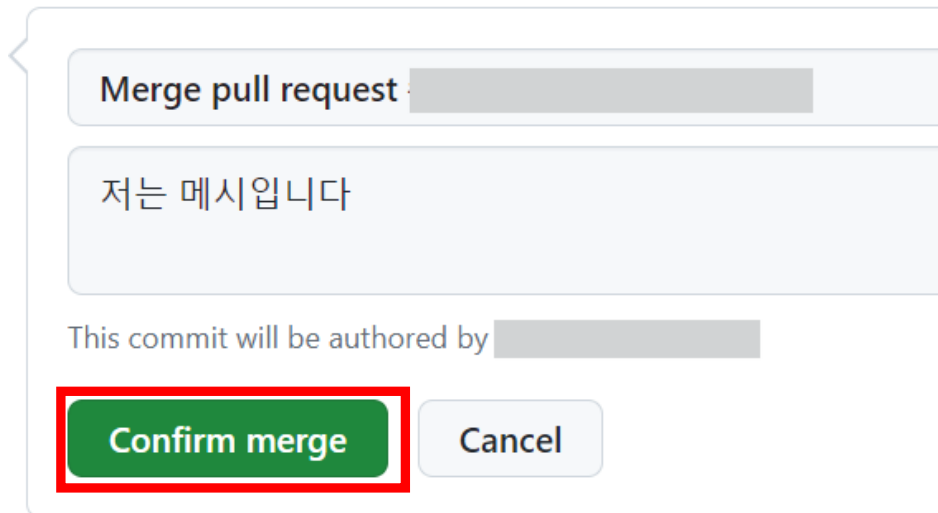
🔴 1 change requested ▾

🔴 **Merging is blocked** [View rules](#)
Merging can be performed automatically once the requested changes are addressed.

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

승인 거부

Pull request 완료



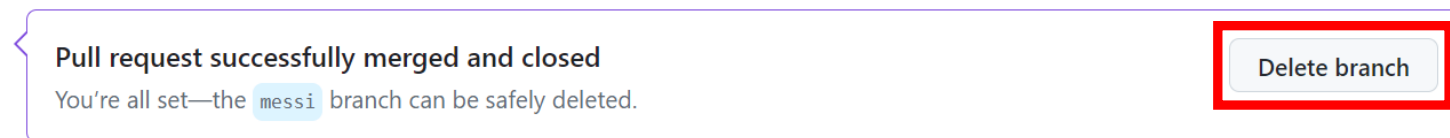
Merge pull request

저는 메시입니다

This commit will be authored by

Confirm merge Cancel

Merge pull request 버튼 클릭을 하면
Confirm merge 버튼이 나타남



Pull request successfully merged and closed

You're all set—the messi branch can be safely deleted.

Delete branch

병합한 브랜치는 삭제

Pull request 완료

- pull request는 github에서 작업을 하였기 때문에 로컬저장소의 main 브랜치에는 반영되지 않았음
- pull request가 완료 되면 로컬저장소 main 브랜치 이동 후 pull을 실행 해서 코드를 최신화 해야함(매.우.중.요)
- 팀원이 여러명이라면 병합을 모두 진행한 후 팀원들 모두 pull해야함

```
MINGW64 ~/Documents/git-colla (messi)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

MINGW64 ~/Documents/git-colla (main)
$ git pull origin main
```


Pull request 완료

- 현재 로컬저장소의 브랜치 목록을 조회하면 브랜치가 삭제가 안된 상태
- 로컬저장소는 `git branch -d 브랜치명`으로 삭제하면 됨
- 풀 리퀘스트 머지 후 삭제한 브랜치는 `git push origin -d 브랜치명`으로 삭제가 될 수 없음. 이미 삭제한 상태이기 때문

```
MINGW64 ~/Documents/git-colla (main)
$ git branch -a
* main
  messi
remotes/origin/HEAD -> origin/main
remotes/origin/main
remotes/origin/messi
```

- 이때는 `git fetch -p origin` 명령어로 원격저장소의 브랜치를 동기화 시켜 주면 됨.

```
MINGW64 ~/Documents/git-colla (main)
$ git fetch -p origin
From https://github.com/logdev413/git-colla
- [deleted]          (none)      -> origin/messi
```

4. Pull request 충돌



Pull request 충돌

- 충돌은 두명 이상의 개발자가 Git에 작업한 같은 파일의 같은 부분을 수정할 때 생김
- 이전시간에 실습했던 충돌 상태 다시 구현
- pull request로 병합하기
 - 현재 main브랜치에서 dog, cat브랜치 생성
 - dog브랜치 이동 후 my.txt 파일을 수정한뒤 git에 올리기
 - pull request를 이용하여 main브랜치에서 dog브랜치 병합(dog브랜치 삭제)
 - 병합 완료 후 cat브랜치로 이동
 - cat브랜치에서 my.txt파일을 수정한뒤 git에 올리기
 - pull request를 이용하여 main브랜치에서 cat브랜치 병합 시도


```
my.txt
1  안녕하세요.
2
3  저는 git을 학습중에 있습니다.
4
5  저는 강아지를 좋아합니다.
```

```
my.txt
1  안녕하세요.
2
3  저는 git을 학습중에 있습니다.
4
5  저는 고양이를 좋아합니다.
```


Pull request 충돌


 base: main  compare: cat ✖ Can't automatically merge. Don't worry, you can still create the pull request.

충돌때문에 병합을 자동으로 처리하지 못함

 **Changes approved** [Show all reviewers](#)

1 approving review by reviewers with write access. [Learn more about pull request reviews.](#)

 1 approval

 **This branch has conflicts that must be resolved**
Use the [web editor](#) or the [command line](#) to resolve conflicts.
Conflicting files
my.txt


Merge pull request

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

간단한 해결 같은 경우
github에서 해결이 가능

Pull request 충돌 해결하기

충돌개수 충돌이 여러 개 일경우 화살표를 이용하여 이동

| | | | | | | |
|--|---|--|--------|--------|----|------------------|
| 1 conflicting file | my.txt | 1 conflict | Prev ^ | Next v | ⚙️ | Mark as resolved |
|  my.txt my.txt | <pre>1 안녕하세요. 2 3 저는 git을 학습중에 있습니다. 4 5 <<<<<<< cat 6 저는 고양이를 좋아합니다. 7 ===== 8 저는 강아지를 좋아합니다. 9 >>>>>> main 10</pre> | <div>병합을 시도한 브랜치</div> <div>기준이 되는 브랜치</div> | | | | |

- >>>>>, <<<<<, ===== 을 지워서 코드수정
- 충돌을 모두 해결 후 Mark as resolved 버튼 클릭

Pull request 충돌 해결하기

Resolving conflicts between `cat` and `main` and committing changes → `cat`

Commit merge

| | | |
|--------------------|---|------------|
| 1 conflicting file | my.txt | ✓ Resolved |
| my.txt my.txt | <div>1 안녕하세요.</div> <div>2</div> <div>3 저는 git을 학습중에 있습니다.</div> <div>4</div> <div>5 저는 고양이를 좋아합니다.</div> <div>6 저는 강아지를 좋아합니다.</div> | |

- 충돌이 해결되면 Commit merge 버튼이 활성화됨
- 이후 작업은 pull request merge와 동일

협력자 삭제하기

Access

- Collaborators
- Moderation options

Code and automation

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

Security

- Code security and analysis
- Deploy keys

Public repository
This repository is public and visible to anyone

Manage

PUBLIC REPOSITORY
This repository is public and visible to anyone.
[Manage](#)

DIRECT ACCESS
1 user has access to this repository. [1 collaborator](#).

Manage access **Add people**

☐ Select all Type ▾

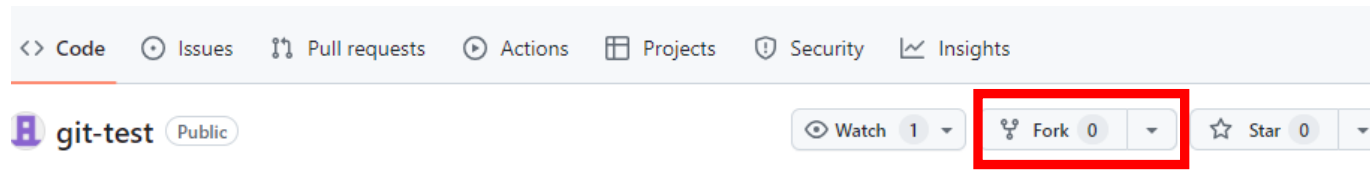
☐ Collaborator

Settings -> Collaborators 에서 권한 삭제

5. 원격저장소 복제

원본저장소 복제하기

- 복제를 하려는 원본 원격저장소 Git 주소로 이동



Fork클릭

Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Required fields are marked with an asterisk (*).

Owner * / Repository name *
git-test
git-test is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

☒ Copy the main branch only
Contribute back to ysdls/git-test by adding your own branch. [Learn more.](#)

You are creating a fork in your personal account.

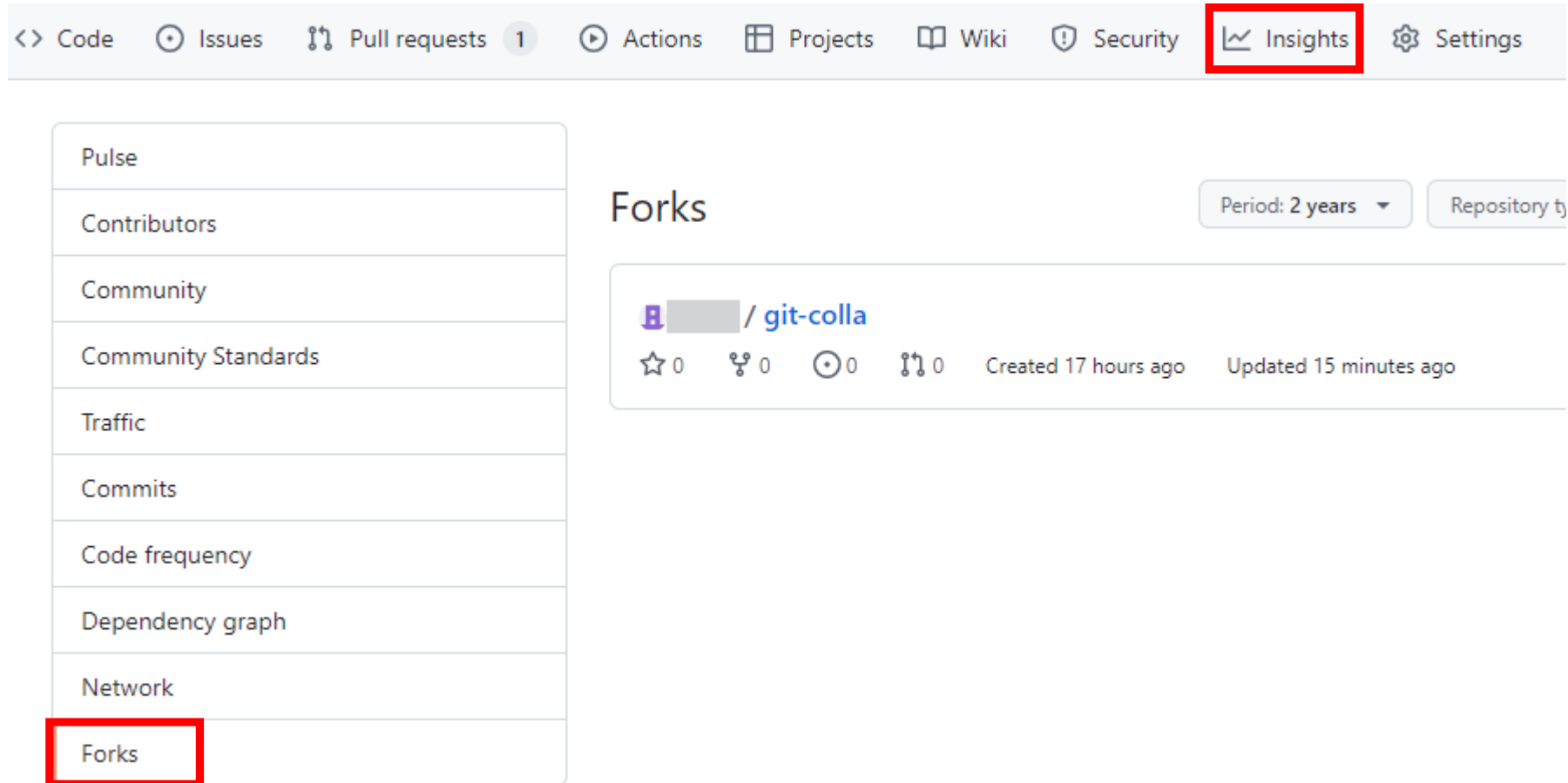
Create fork

git-test Public
forked from

복제가 완료된 복제 원격저장소에는
forked from 원본 git 저장소 이름이 적혀있음

복제 원격저장소 생성

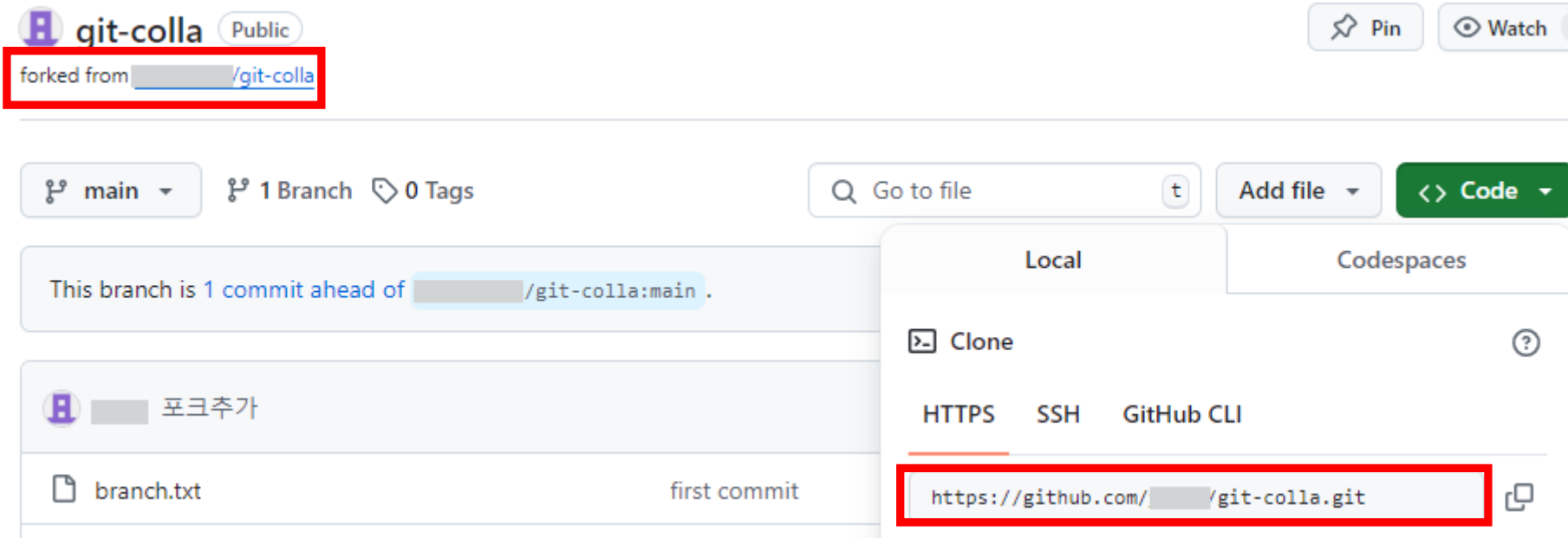
원본저장소에 fork확인하기



원본 원격저장소의 Insights -> Forks 에 접근하면 누가 fork를 했고 update를 진행했는지 확인 가능

복제저장소 clone

- 복제한 원격저장소를 git clone을 이용하여 로컬저장소로 코드를 가져옴



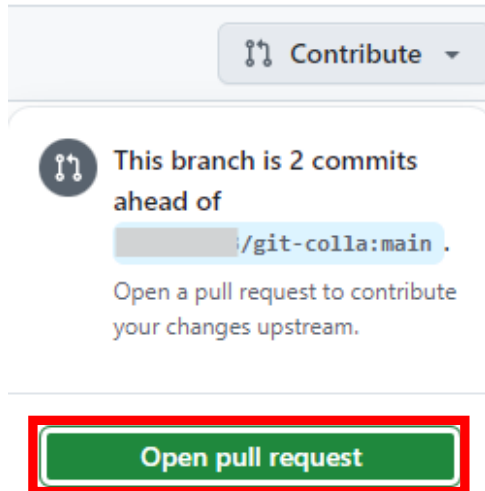
내 github로 가져온 복제 원격저장소

복제저장소 Pull request

- 복제한 원격저장소에 본인이름.txt파일 생성 후 푸쉬

```
messi.txt U x
messi.txt
1  안녕하세요.
2
3  제 이름은 메시입니다.
4
5  끝까지 완강하고 취업 성공하겠습니다.
```



- 복제 원격저장소의 Contribute 또는 메뉴의 Pull request 접속



복제저장소 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](#) or [learn more about diff comparisons](#).

 base repository: >git-colla ▾ base: main ▾  head repository: >git-colla ▾ compare: main ▾ ✓ Able to merge. These branches can be automatically merged.

원본 원격저장소와 복제한 원격저장소를 비교



Review required

At least 1 approving review is required by reviewers with write access. [Learn more about pull request reviews](#).



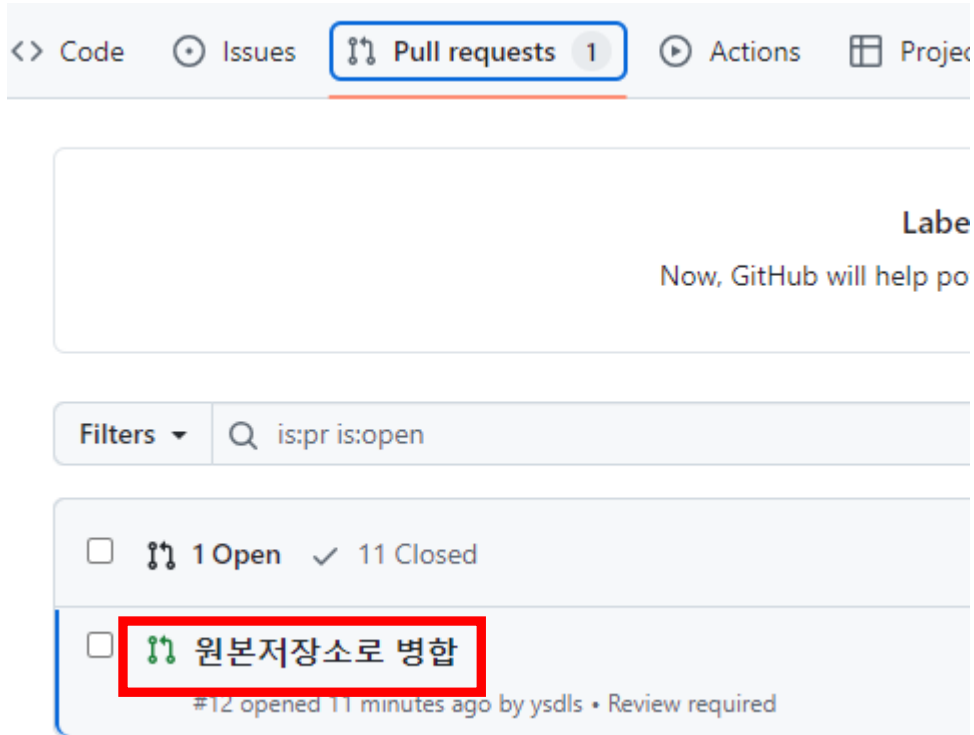
Merging is blocked

Merging can be performed automatically with 1 approving review.

[View rules](#)

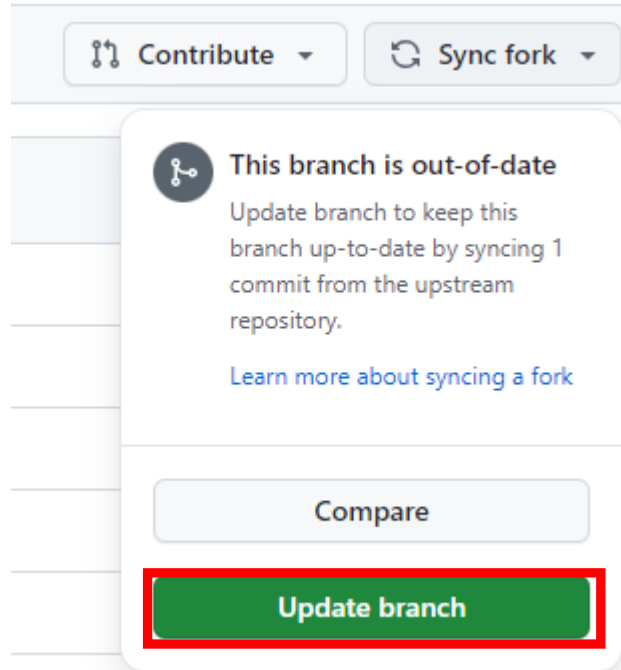
원본 원격저장소의 브랜치 규칙이 reviewer가 승인을 해야하므로 복제 원격저장소는 위와 같이 나옴

원본저장소에서 Pull request




- 원본 원격저장소의 Pull requests에 접근하여 요청한 pull request를 클릭
- 이후 작업은 pull request merge하는 것과 동일


복제저장소 동기화



원본 원격저장소에서 병합을 모두 완료하면 복제 원격저장소에 Sync fork에 Update branch가 활성화 됨

복제저장소 충돌

 **Review required**
At least 1 approving review is required by reviewers with write access. [Learn more about pull request reviews.](#)

 **This branch has conflicts that must be resolved**
Use the [web editor](#) or the [command line](#) to resolve conflicts.
Conflicting files
fork.txt

Resolve conflicts

복제 원격저장소에서 충돌이 날 경우에도 pull request에서 했었던것 처럼 충돌을 해결 후 진행하면 됨

6. Commit 되돌리기

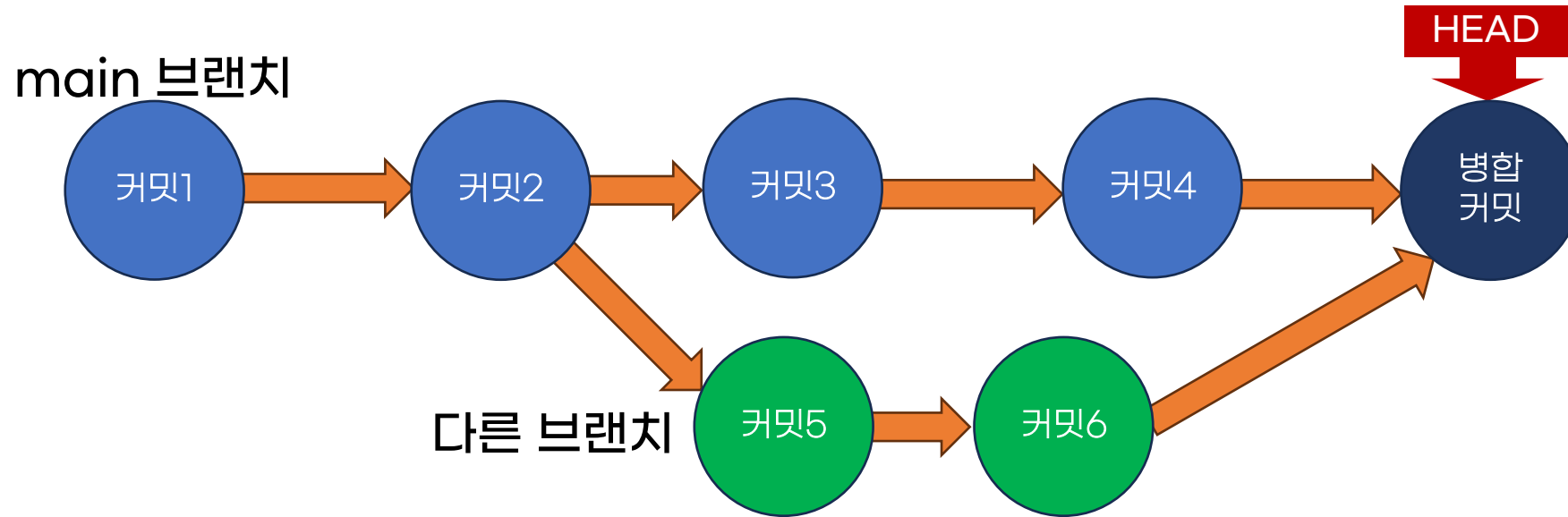
Commit 되돌리기

- Commit 되돌리기는 Git이 익숙해지기 전까지 사용안해도 무관
- 자주 사용 안할 수도 있는 명령어
- Git의 커밋을 이전상태로 되돌릴 때 사용
- 코드가 모두 삭제될 수도 있으므로 사용시 주의 해야함

reset

- git reset (옵션) <이동할 커밋>
 - 예) git reset --hard HEAD^2
- 이동할 커밋
 - HEAD~n : n번째 조상 커밋으로 이동
 - HEAD^n : 병합커밋에서 사용하며 n번째 부모로 이동
- 옵션값
 - --soft : 스테이징에는 올라가 있으며 커밋만 다시 하고 싶을때
 - --mixed(기본값) : 스테이징, 커밋을 모두 취소하고 다시 하고 싶을때
 - --hard : 최근 변경사항을 모두 삭제하고 이전 커밋으로 돌아가고싶을때
- 협업시에는 사용을 안하는게 좋음

reset



이동 명령어 예시

HEAD^1 : 병합커밋의 첫번째부모로 main 브랜치의 커밋4와 병합된 것이므로 커밋4

HEAD^2 : 병합커밋의 두번째 부모로 다른 브랜치의 커밋6과 병합된 것이므로 커밋6

HEAD~1 : 이전 커밋인 부모 커밋4

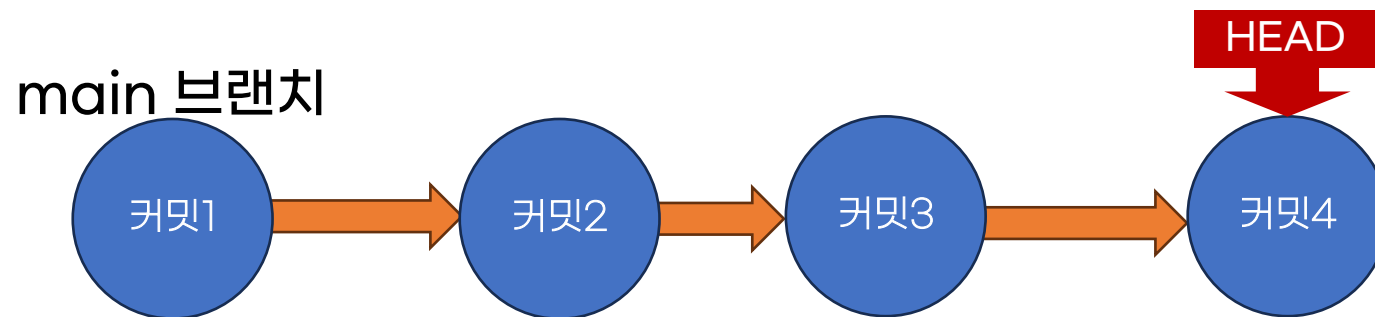
HEAD~2 : 두번째 조상 커밋인 커밋3

HEAD~3 : 세번째 조상 커밋인 커밋2

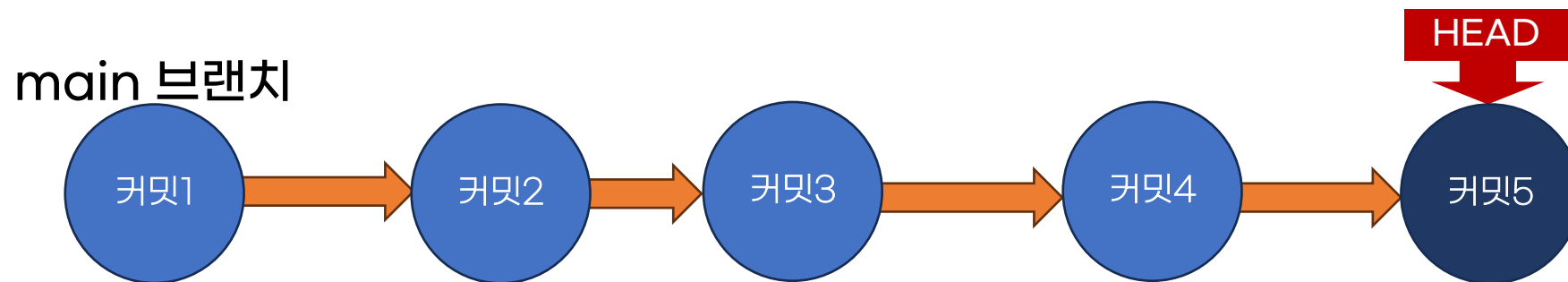
revert

- `git revert <이동할 커밋>`
- 특정 커밋을 취소하는 새로운 커밋 생성
- 기존 커밋 이력을 유지하면서 변경 사항을 되돌리는 새로운 커밋 생성
- 특정 커밋을 기준으로 되돌리며 필요시 여러커밋을 순차적으로 되돌리기
- 사용시에는 최근 커밋부터 되돌리기
- `reset`과의 차이점
 - `reset`은 커밋이력을 삭제하면서 이전으로 되돌림
 - `revert`는 기존 커밋이력을 남기고 변경된 새로운 커밋을 생성
- 협업에는 `reset`보다는 `revert`를 권장

revert



커밋3이 잘못되어 되돌릴다면
=> git revert 커밋3



커밋5는 커밋3을 되돌린 커밋

학습정리

- 최근 협업 방법은 애자일 방법론을 채택하여 계획, 개발, 테스트, 기능완료를 주기적으로 빠르게 하는 방법을 사용한다.
- Git 협업시에는 내 저장소에 협력자로 등록하는 방법과 저장소 복제를 이용하는 두가지 방법이 존재한다.
- 협업시 코드 병합을 할때는 Pull request를 이용하여 병합을 진행하는 것이 일반적이다.
- Pull request는 여러 개발자들이 코드에 대해서 토론을 진행할 수 있고 코드에 대한 리뷰를 볼 수 있는 원격 협력공간이다.

학습정리

- Pull request에서 병합시 충돌이 일어나게 되면 Pull request 상에서 해결이 가능하며 코드양이 많을 경우 VS Code에서 진행할 수 있다.
- reset과 revert는 Git 되돌리기에 사용하는 명령어로 Git이 익숙해지기 전까지는 사용안해도 무관하며 reset은 협업시에는 사용안하는게 좋다.
- 그 밖에 Git명령어도 여러분들이 Git에 익숙해지면 하나씩 더 공부해보시면 좋습니다.

다음 수업은?

- 이번시간까지 Git 사용에 대해서 공부하였습니다.
- 다음시간에는 VS Code 활용법과 Tip에 대해서 설명해 드리겠습니다.

복.습.철.저

수고하셨습니다