

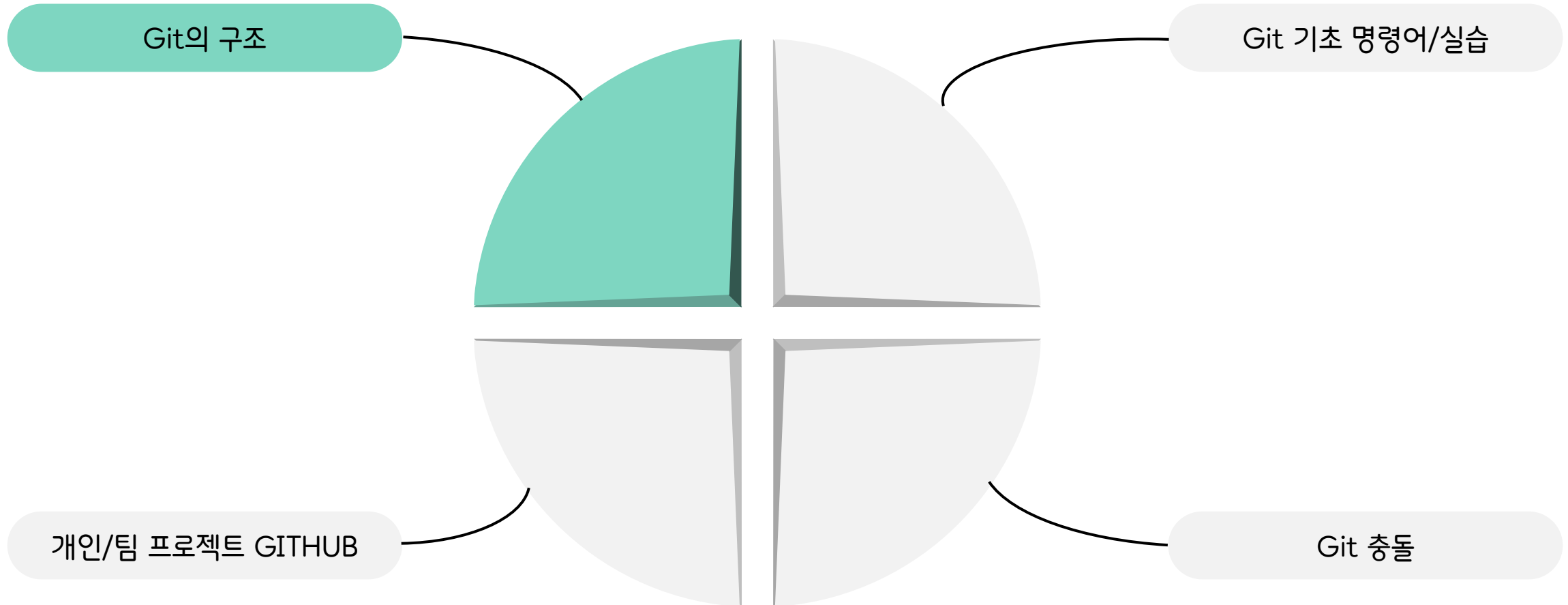
RATS 22년도 겨울 세미나

Git 기초부터 프로젝트까지



RATS 25기 컴퓨터공학과 황규도

Section1. Git의 구조



Git의 구조: 형상관리

구성 관리

文 18개 언어 ▾

위키백과, 우리 모두의 백과사전.



이 문서의 내용은 출처가 분명하지 않습니다.

이 문서를 편집하여, 신뢰할 수 있는 출처를 표기해 주세요. 검증되지 않은 내용은 삭제될 수도 있습니다. 내용에 대한 의견은 토론 문서에서 나누어 주세요. (2011년 1월)

소프트웨어 구성 관리(**영어**: Software Configuration Management) 또는 **형상 관리**는 소프트웨어의 변경사항을 체계적으로 추적하고 통제하는 것으로, 형상 관리는 일반적인 단순 버전관리 기반의 소프트웨어 운용을 좀 더 포괄적인 학술 분야의 형태로 넓히는 구간을 이야기한다.

일반적으로 형상 항목(**영어**: Configuration Item)이라는 형태로 작업 산출물을 선정하고, 형상 항목 간의 변경 사항 추적과 통제 정책을 수립하고 관리한다.

1. 개요

[편집]

Version Control System(VCS)

문서나 설계도, **소스 코드** 등의 변경점을 관리해주는 **소프트웨어**.

IT용어사전

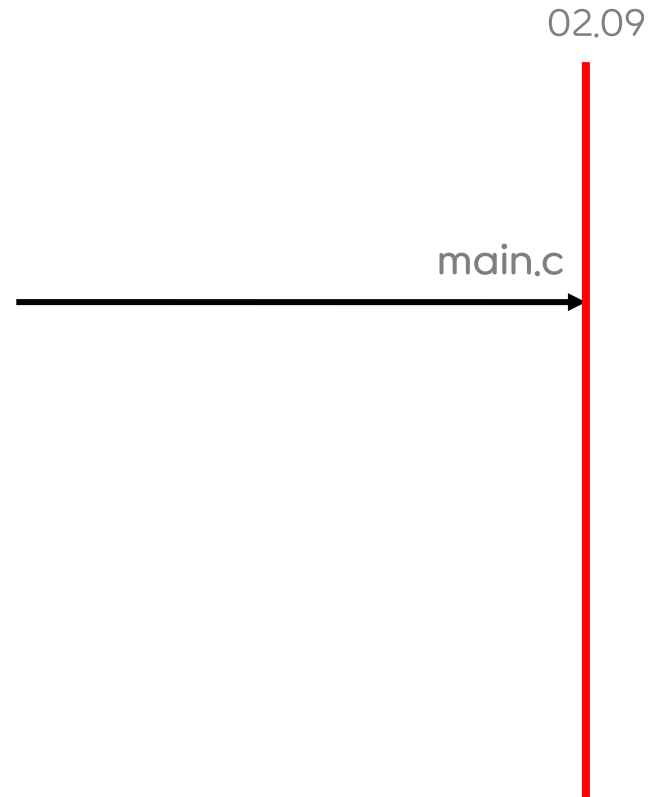
형상 관리

[configuration management , 形狀管理]

시스템 형상 요소의 기능적 특성이나 물리적 특성을 **문서화**하고 그들 특성의 변경을 관리하며, 변경의 과정이나 실현 상황을 기록·보고하여 지정된 요건이 충족되었다는 사실을 검증하는 것, 또는 그 과정.

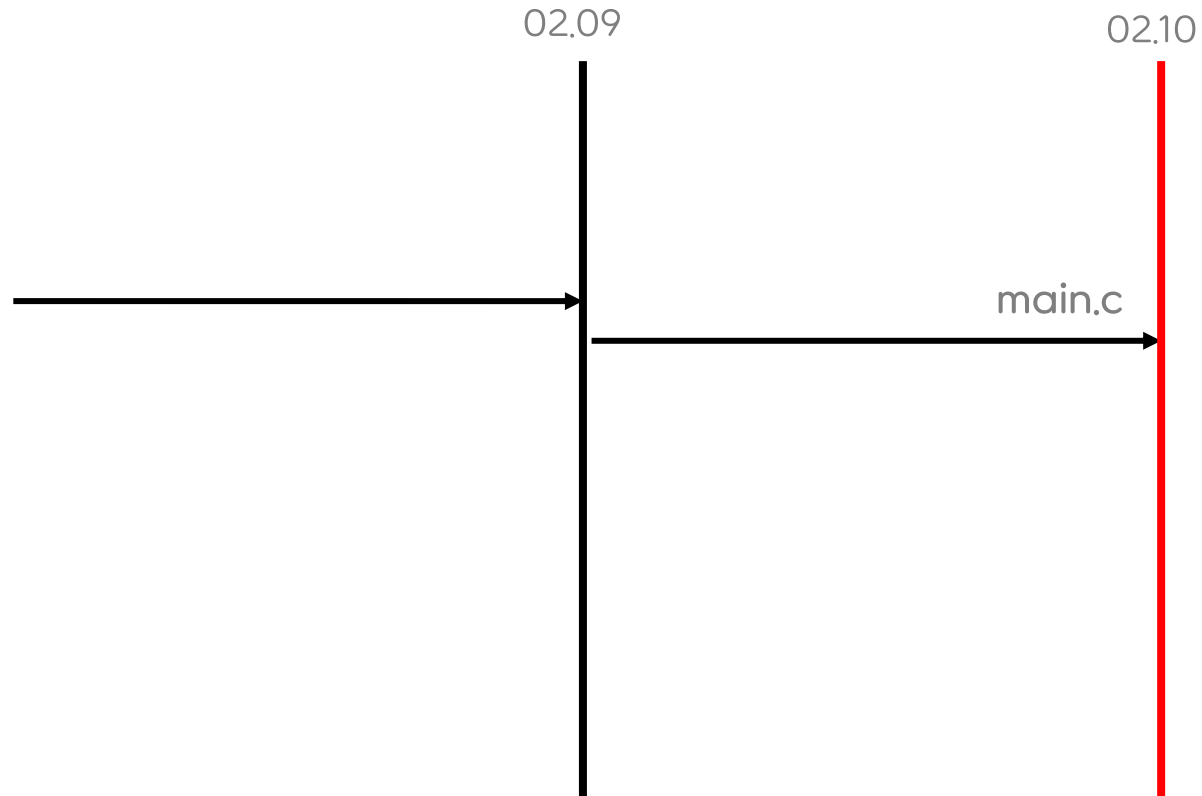
Git의 구조: 형상관리

① 극단적으로 형상관리를 하지 않았을 때...



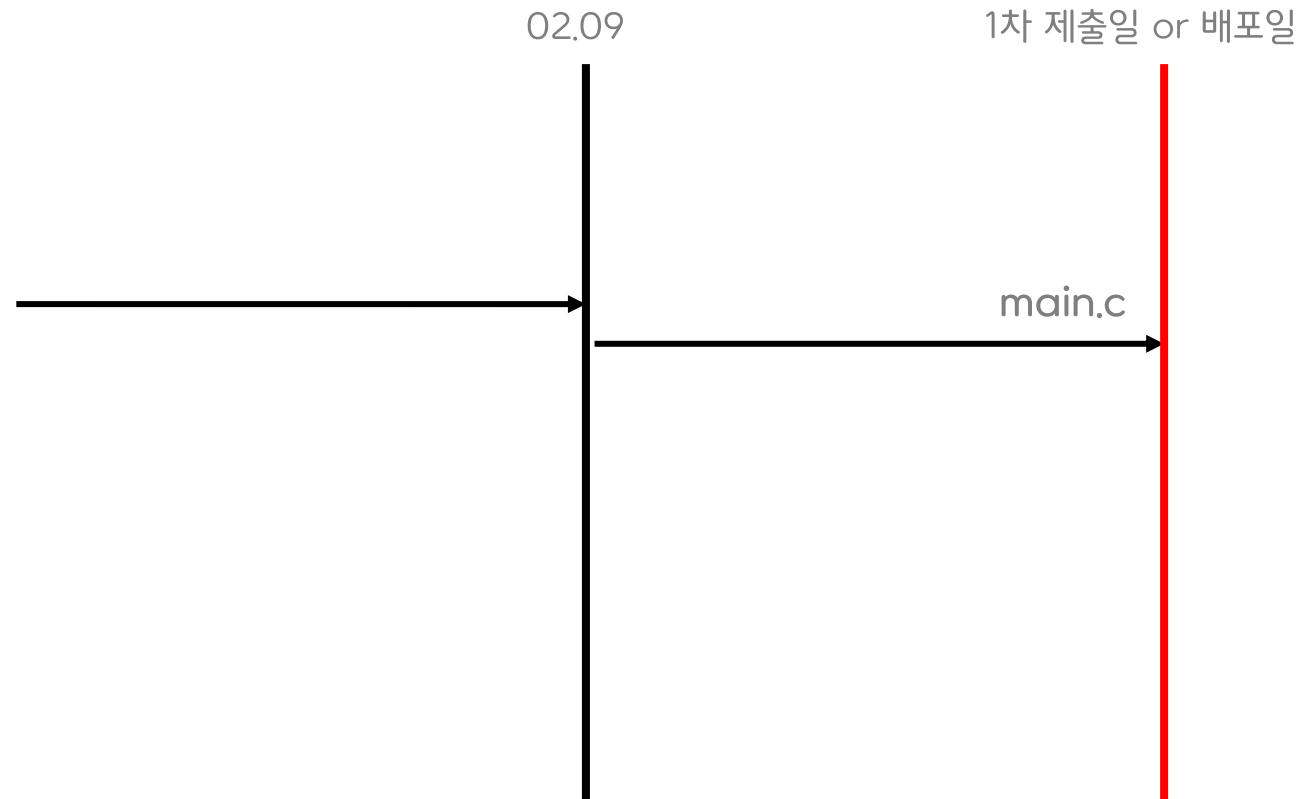
Git의 구조: 형상관리

① 극단적으로 형상관리를 하지 않았을 때...



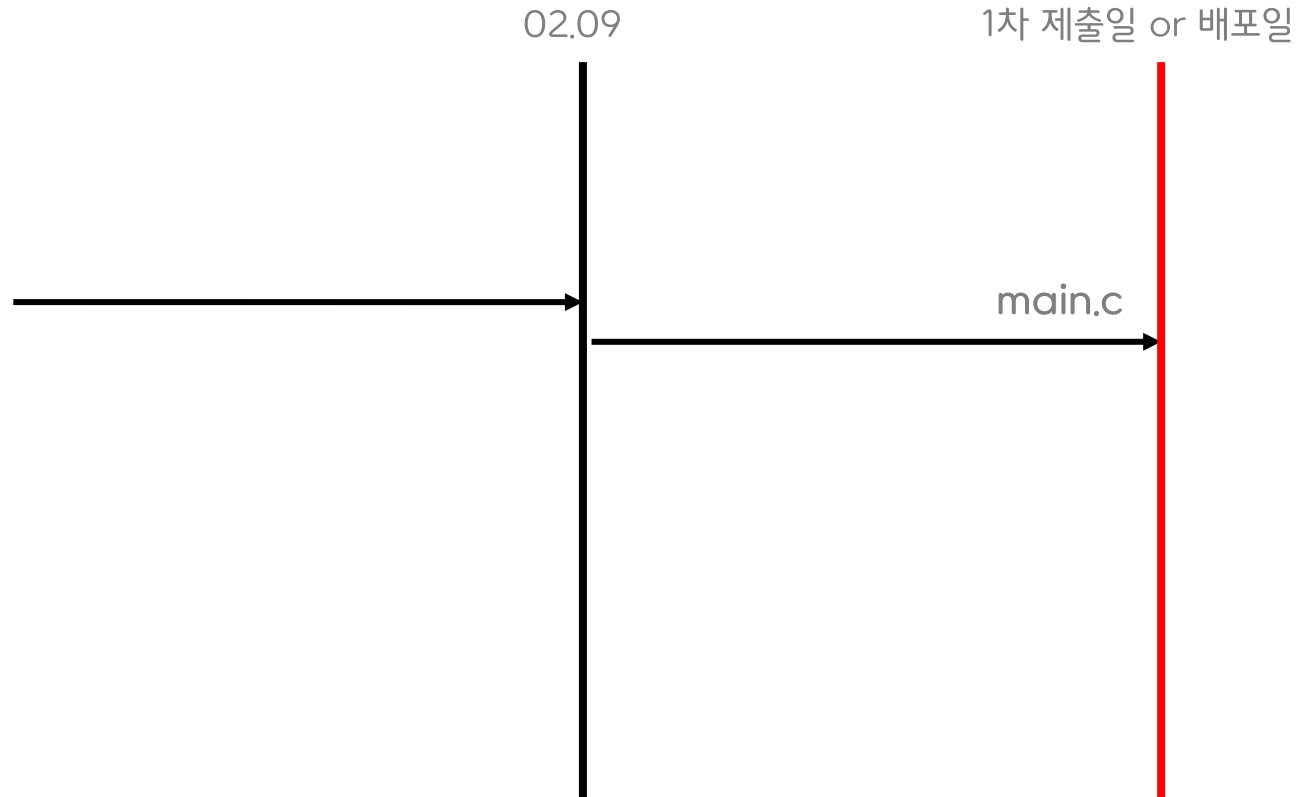
Git의 구조: 형상관리

① 극단적으로 형상관리를 하지 않았을 때...



Git의 구조: 형상관리

① 극단적으로 형상관리를 하지 않았을 때...

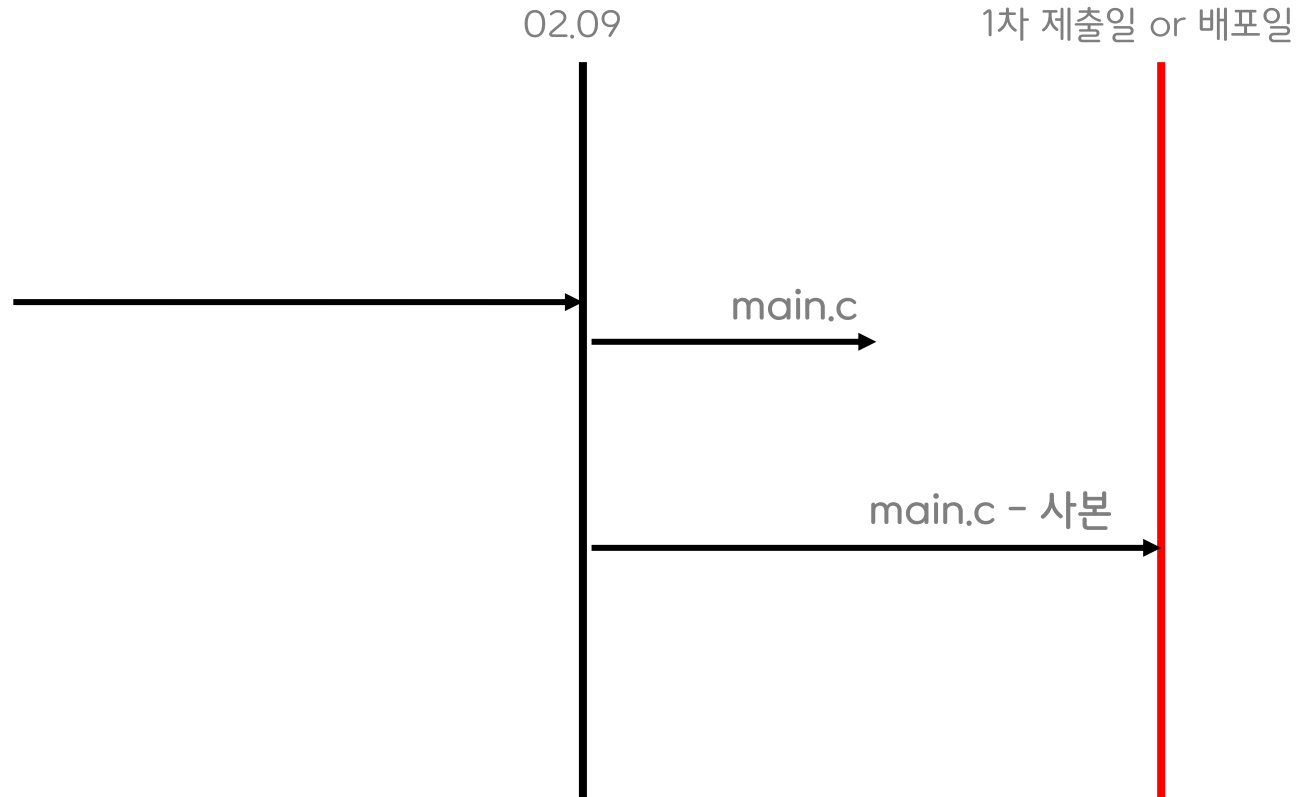


① 그냥 지금 소스코드 제출

1. 진행중인 코드의 버그?
2. 코드의 미완성

Git의 구조: 형상관리

① 극단적으로 형상관리를 하지 않았을 때...



① 그냥 지금 소스코드 제출

1. 진행중인 코드의 버그?
2. 코드의 미완성

② 개발중인 코드는 빼고 제출

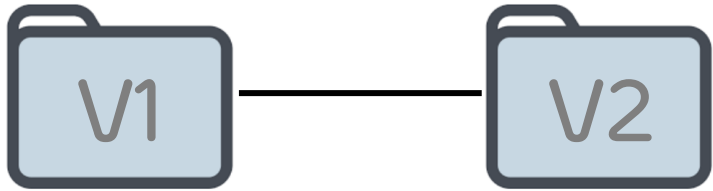
Git의 구조: 형상관리

② 형상관리를 했을 때!



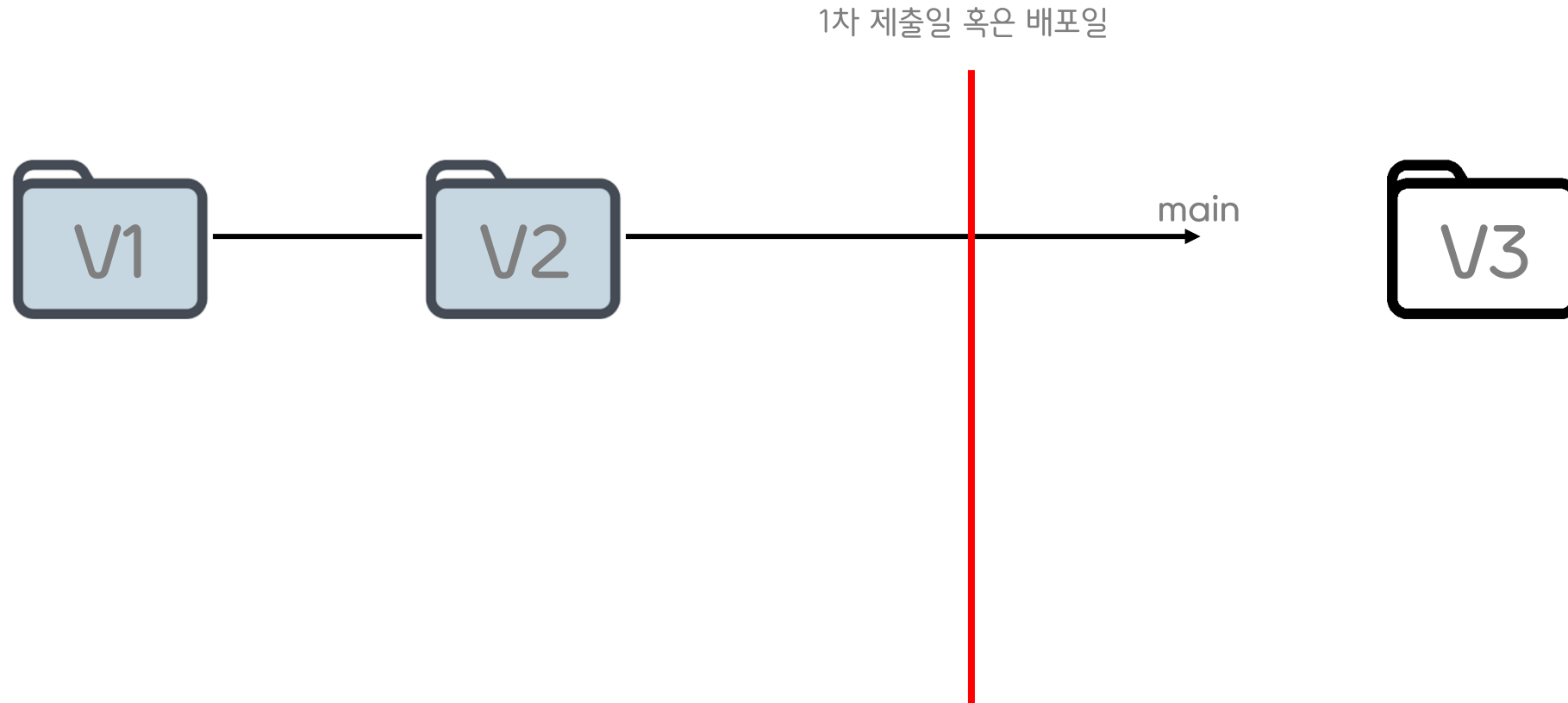
Git의 구조: 형상관리

② 형상관리를 했을 때!



Git의 구조: 형상관리

② 형상관리를 했을 때!



Git의 구조: 형상관리

① 극단적으로 형상관리를 하지 않았을 때...

1. 코드를 백업해야 됨
2. 버그가 안 나던 시점을 기억해야 됨



② 형상관리를 했을 때

1. 개발과 제출을 동시에, 진행 가능
2. 효율적인 이슈 트래킹



설계문서 - V1
이슈 - V1

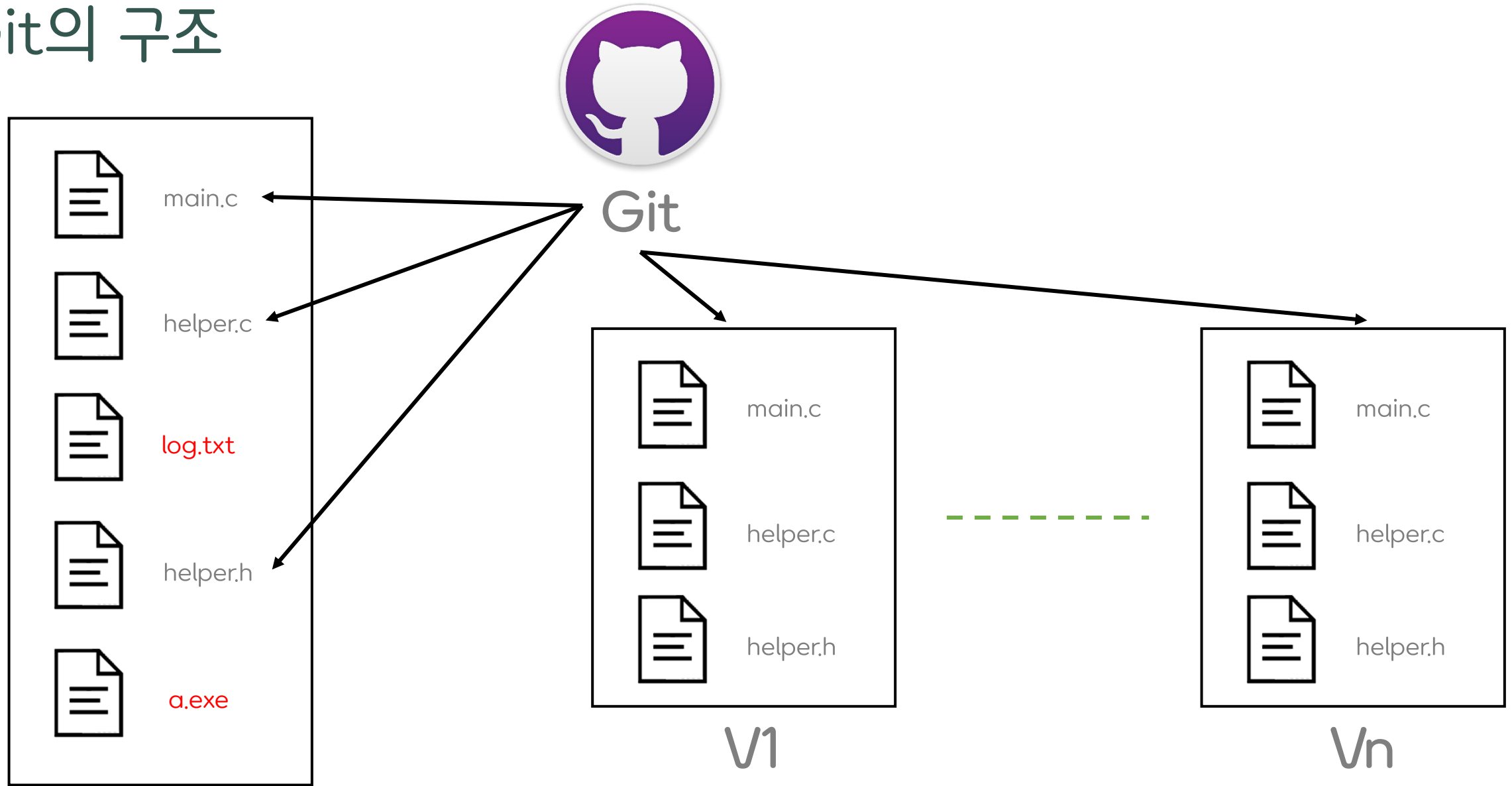


설계문서 - V2
이슈 - V2

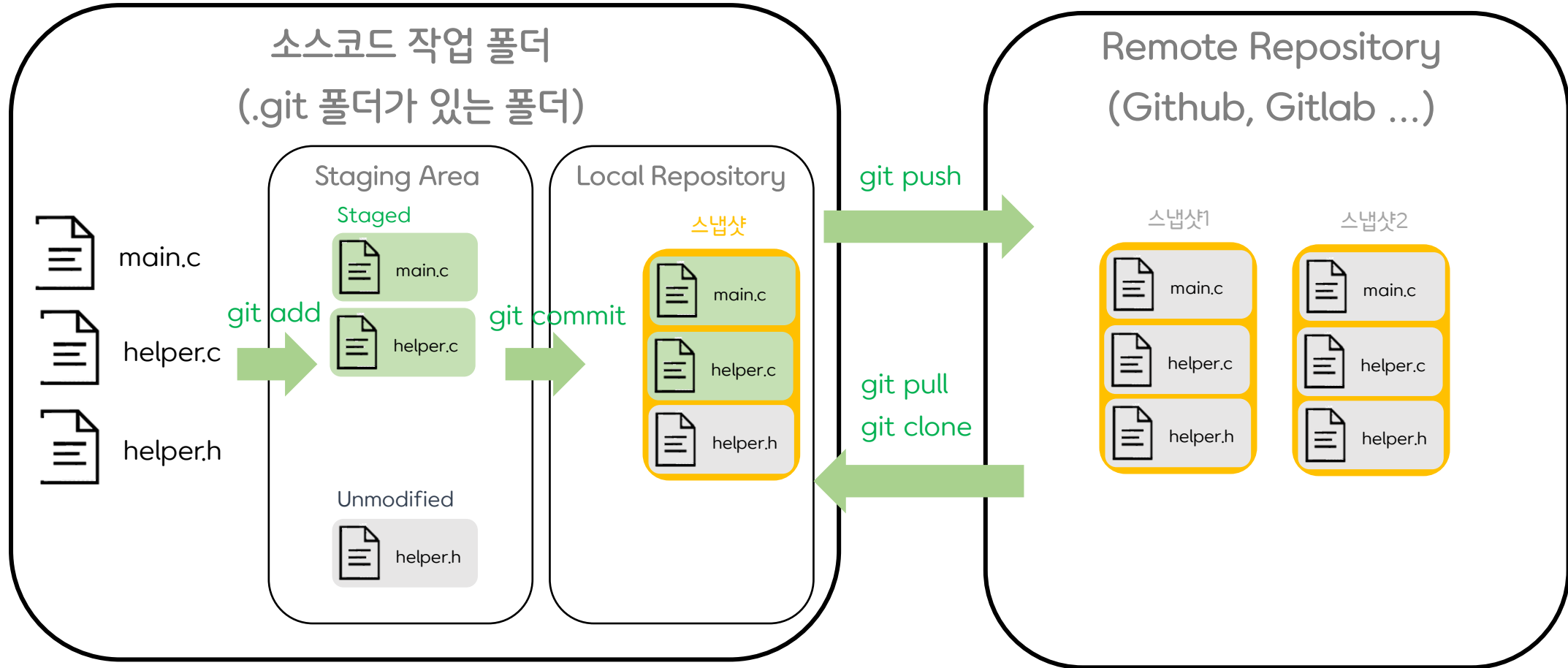


설계문서 - V3
이슈 - V3

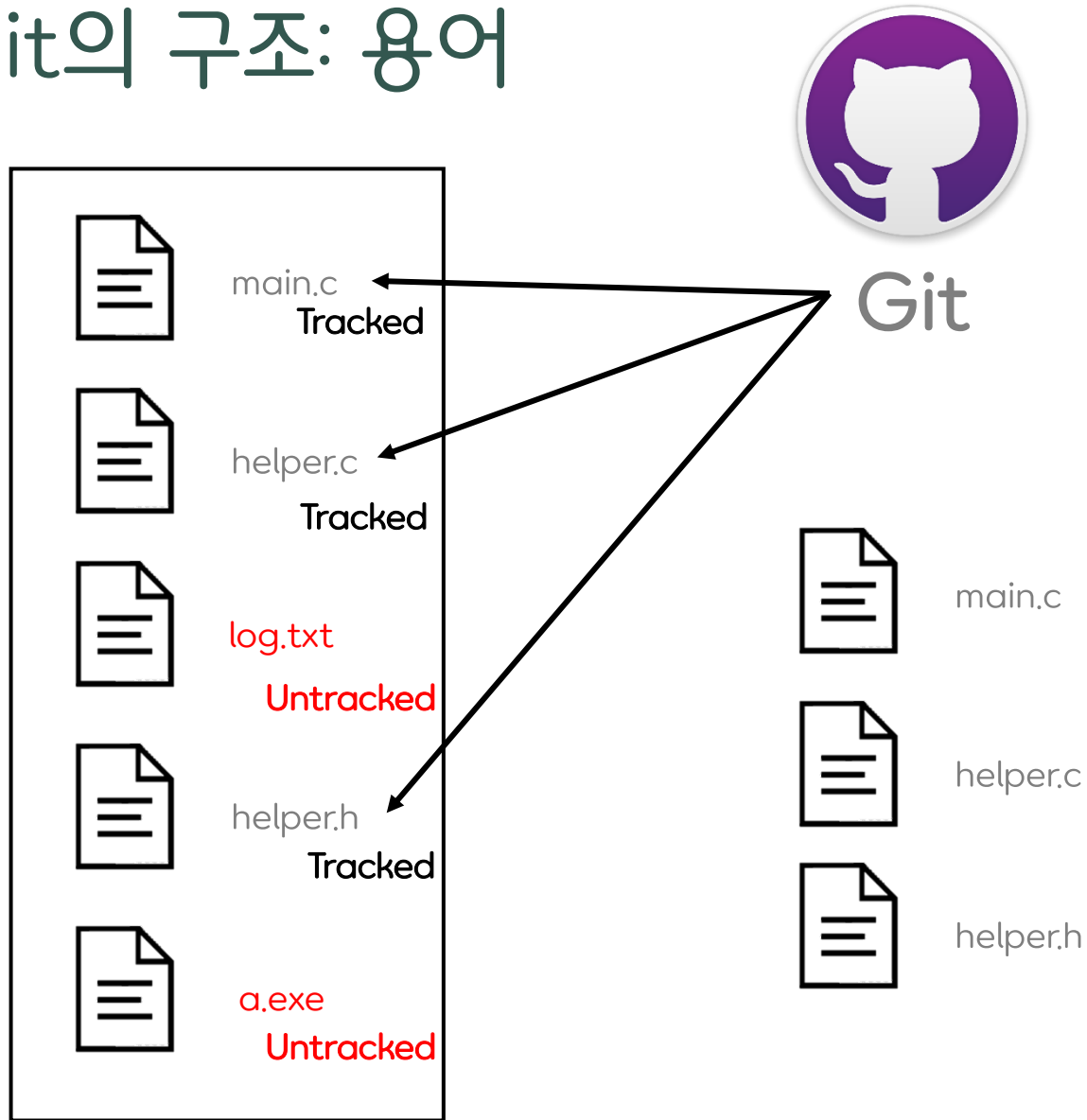
Git의 구조



Git의 구조



Git의 구조: 용어



파일관리(추적)의 상태

Tracked	추적함
Untracked	추적 안함

Git의 구조: 용어



Git

파일관리(추적)의 상태

Tracked
modified

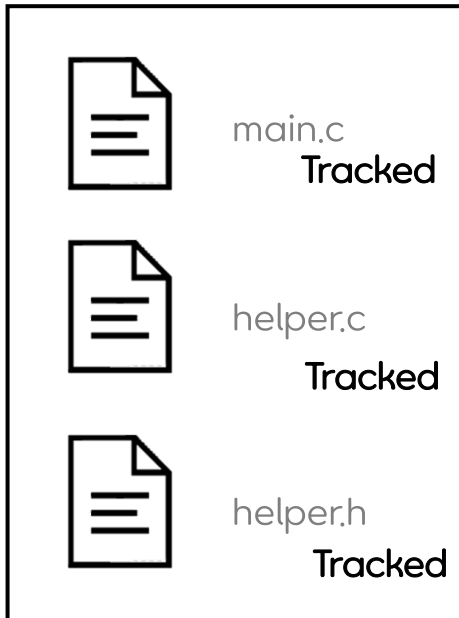
추적함

수정됨

Untracked

추적 안함

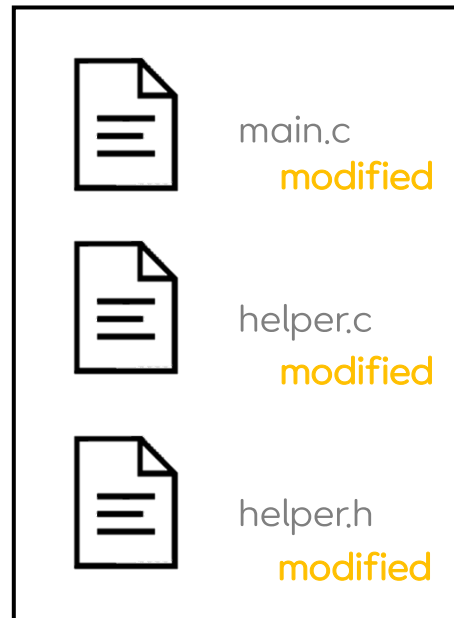
컴퓨터 작업 폴더



파일 수정



컴퓨터 작업 폴더



Git의 구조: 기본 용어



Git

파일관리(추적)의 상태

Tracked

추적함

modified

수정됨

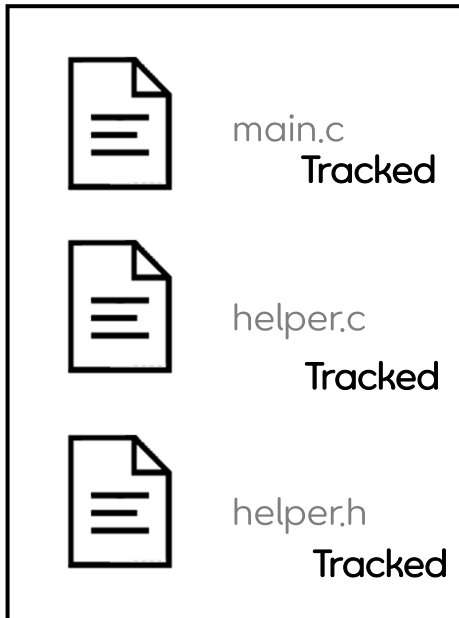
staged

스테이지됨

Untracked

추적 안함

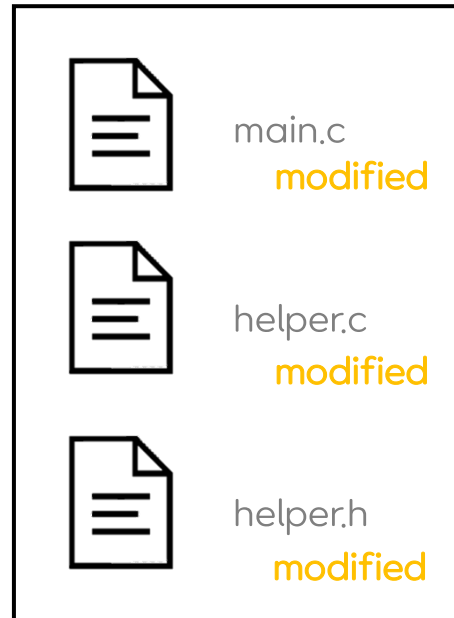
컴퓨터 작업 폴더



파일 수정



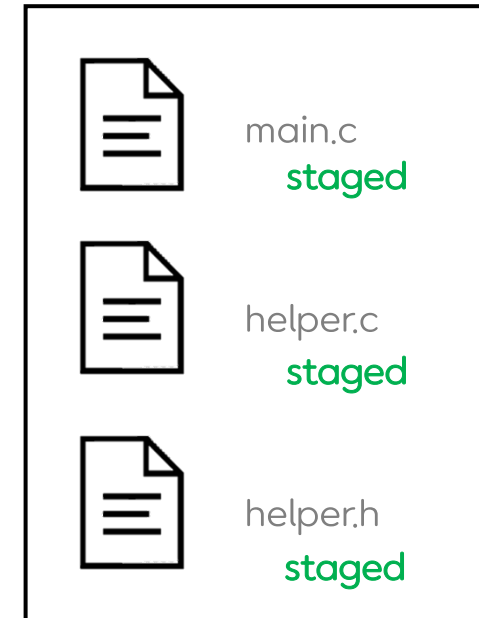
컴퓨터 작업 폴더



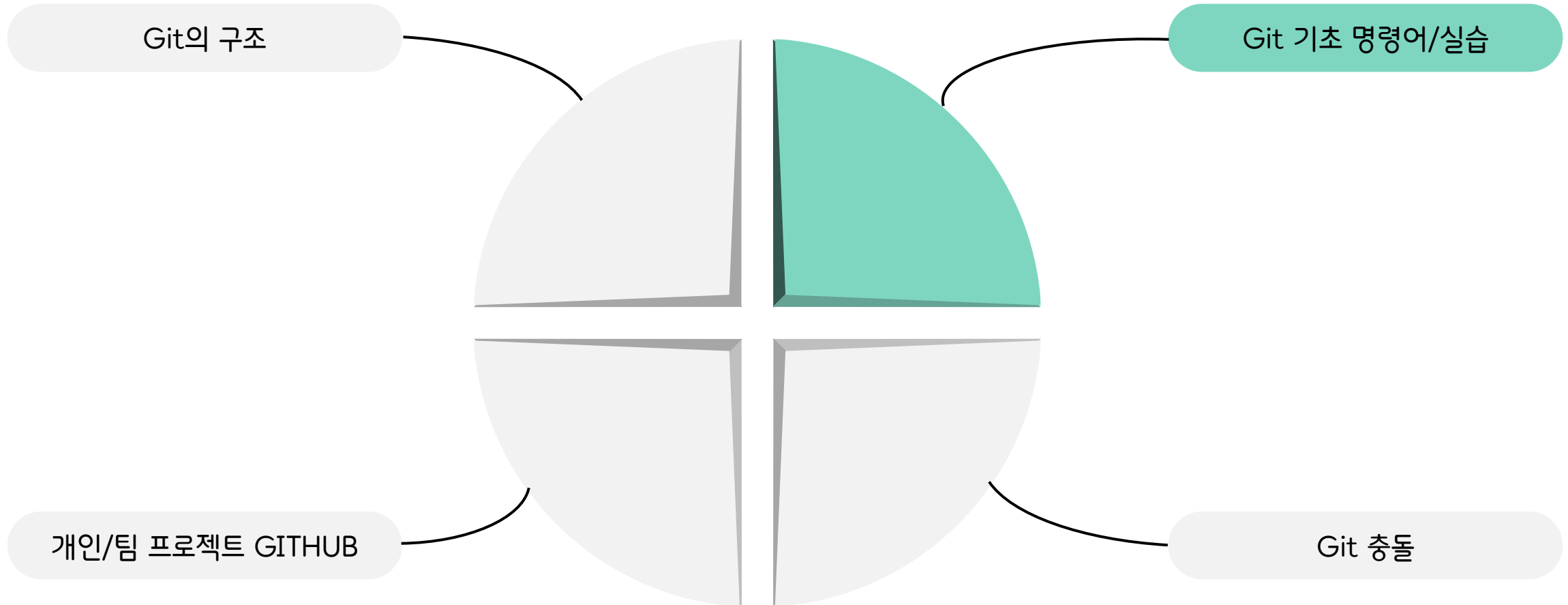
git add



Staging Area



Section2. Git 기초 명령어/실습

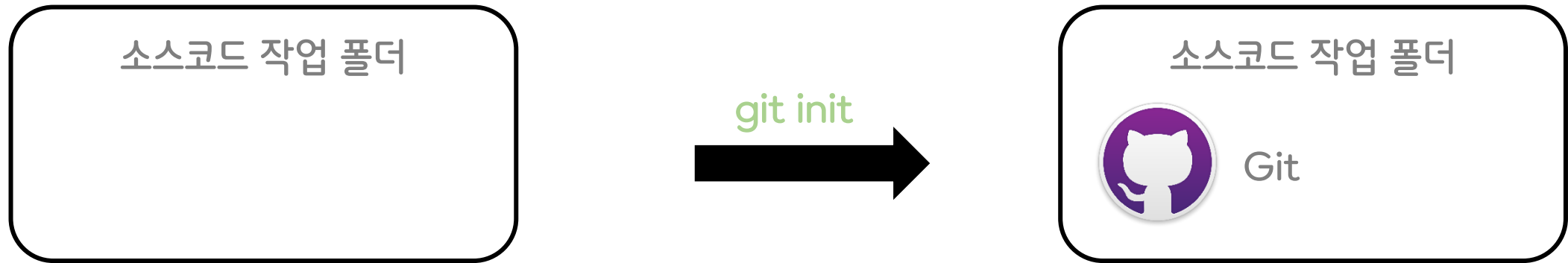


Git 실습: git init

- ① 폴더를 새로 만들어주세요! (이름은 뭐... 상관없어요)
- ② vscode로 ①에서 만든 폴더를 열어주세요!
- ③ vscode에서 명령팔레트를 열어주세요! (단축키 : 컨트롤+시프트+P)
- ④ 명령팔레트에서 *git init* 를 입력해주세요!

Git 기초 명령어: git init

해당 폴더에 Git을 추가하고, 관리를 하겠다는 뜻!



Git 실습: git status

① 해당 폴더에 다음과 같은 파일을 만들어주세요!

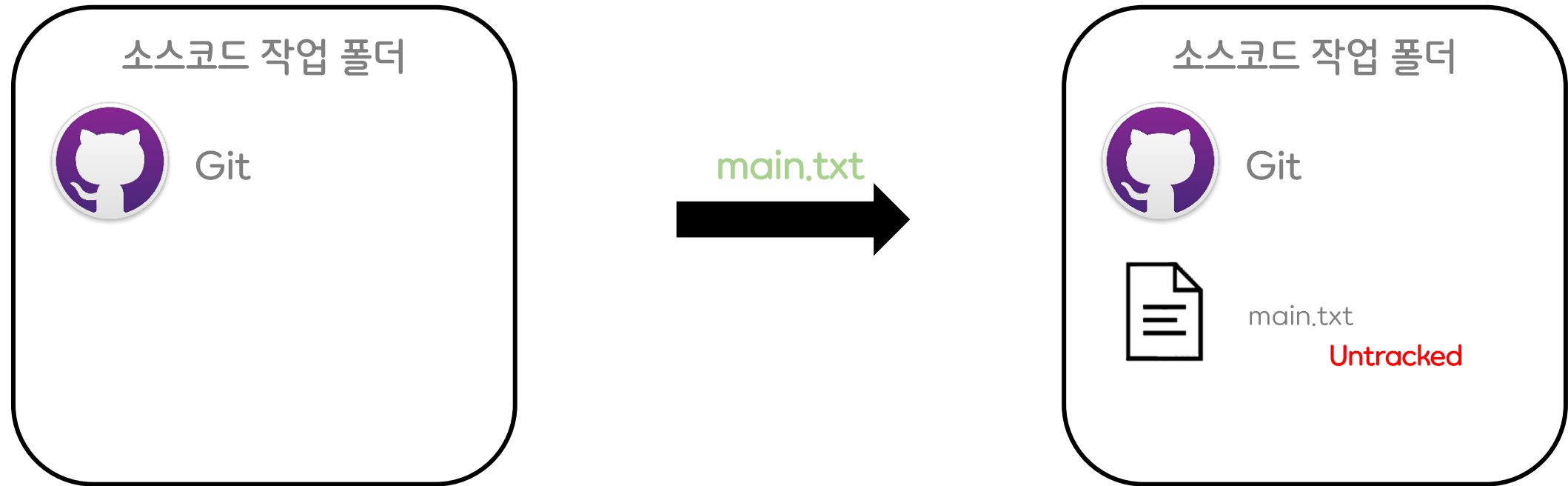
① main.txt : 해당 파일에 “{현재 시간} : **First Commit**” 을 입력 후 저장해주세요!

② 명령프롬프트에서 git status 를 입력해주세요!

```
Untracked files:
(use "git add <file>..." to include in what will be committed)
main.txt
```

Git 기초 명령어: git status

Git이 현재 Tracking 하고 있는 파일 목록들을 보여줍니다



Git 실습: git add {filename}

① 명령프롬프트에서 `git add main.txt` 를 입력해주세요!

① main.txt : 해당 파일에 “{현재 시간} : **First Commit**” 을 입력 후 저장해주세요!

② 명령프롬프트에서 `git status` 를 입력해주세요!

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
   new file:   main.txt
```

Git 기초 명령어: git add {filename}

파일을 Staging Area로 등록하는 과정



git add main.txt





Git 실습: git commit -m {메시지명}

- ① 명령팔레트에서 git commit -m “First commit” 를 입력해주세요!
- ② 명령팔레트에서 git status 를 입력해주세요!

```
On branch main
nothing to commit, working tree clean
```

- ③ vscode 실행창 (컨트롤 + 시프트 + p) 에서 git graph: View Git Graph 클릭!

Graph	Description	Date	Author	Commit
  main	First commit	10 Feb 2022 02:...	BEMELON	34c15e92

Git 기초 명령어: git commit -m {메시지명}

Staging된 파일들을 모두 하나의 스냅샷으로 저장합니다.



Git 실습: git add .

① 새로운 파일들을 만들어보려고 해요!

㉠ main.txt : 해당 파일에 “[현재 시간]: add main2.txt” 을 추가 입력 후 저장해주세요!

㉡ main2.txt : 해당 파일에 “[현재 시간]: First Commit” 을 입력 후 저장해주세요!

② 명령프롬프트에서 git status 를 입력해주세요!

```
On branch main
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:   main.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        main2.txt

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

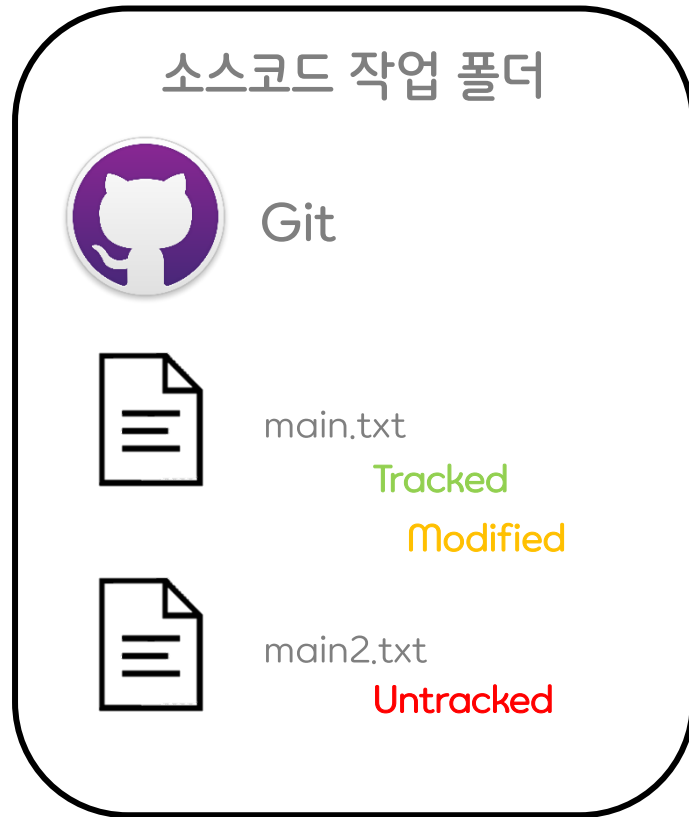
③ 명령프롬프트에서 git add . 를 입력해주세요!

④ 명령프롬프트에서 git status 를 입력해주세요!

```
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.txt
        new file:   main2.txt
```

Git 기초 명령어: git add .

작업 폴더에 있는 모든 내역을 모두 Staging 하는 과정





Git 실습: git commit -m {메시지명}

- ① 명령팔레트에서 git commit -m “Second commit” 를 입력해주세요!
- ② 명령팔레트에서 git status 를 입력해주세요!

```
On branch main
nothing to commit, working tree clean
```

- ③ vscode 실행창 (컨트롤 + 시프트 + p) 에서 git graph: View Git Graph 클릭!

Graph	Description	Date	Author	Commit
	 main Second commit	10 Feb 2022 02:...	BEMELON	ee1ea361
	First commit	10 Feb 2022 02:...	BEMELON	34c15e92

Git 실습: .gitignore

① 새로운 파일 2개를 만들어보려고 해요!!

㉠ a.dummy 이라는 파일에 “a” 문자를 원하는 만큼 입력해주세요!

㉢ b.dummy 이라는 파일에 “b” 문자를 원하는 만큼 입력해주세요!

② 명령프롬프트에서 `git status` 를 입력해주세요!

```
Untracked files:
(use "git add <file>..." to include in what will be committed)
a.dummy
b.dummy
```

③ 파일 하나 더!

㉠ .gitignore 이라는 파일에 *.dummy 를 입력하고 저장해주세요!

④ 명령프롬프트에서 `git status` 를 입력해주세요!

```
On branch main
Untracked files:
(use "git add <file>..." to include in what will be committed)
.gitignore
```

Git 기초 : .gitignore

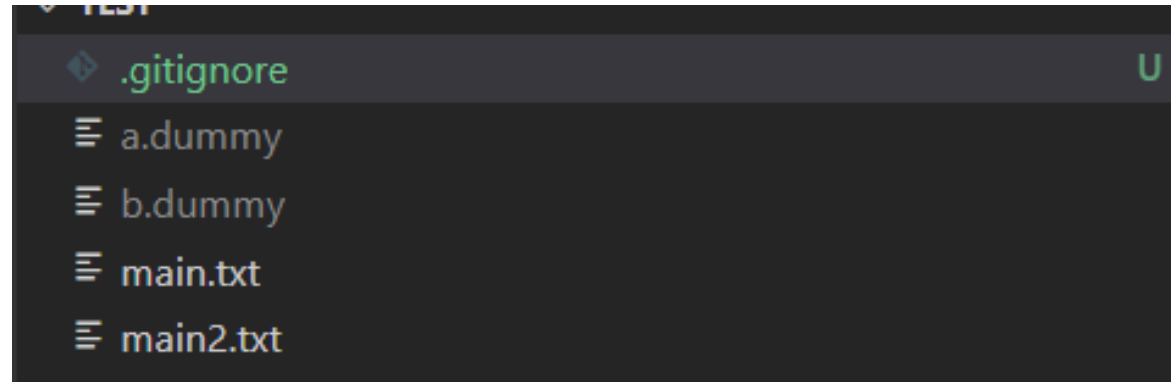
Git이 관리하지 않는 파일들을 지정하는 방법!

*.dummy는 확장자가 dummy인 모든 파일을 의미.

git add . 명령어 사용시 가장 유용하게 쓰이는 파일!

프로젝트별 .gitignore 레퍼런스:

<https://www.toptal.com/developers/gitignore>



① .gitignore 파일도 add -> commit 해주세요 ~

* 메시지명은 자유~

Git 실습: git branch

① 명령프롬프트에서 git branch rats_branch 를 입력해주세요!

② 명령프롬프트에서 git branch 를 입력해주세요!

```
PS C:\Users\hgdk\i\Desktop\test> git branch
* main
  rats_branch
```

③ 명령프롬프트에서 git switch rats_branch 를 입력해주세요!

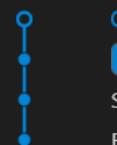


④ 명령프롬프트에서 git branch를 입력해주세요!

```
PS C:\Users\hgdk\i\Desktop\test> git branch
main
* rats_branch
```

⑤ main.txt 에 {현재 시간} : [rats_branch] update main2.txt 를 추가해주세요!

⑥ main2.txt 에 {현재 시간} : [rats_branch] Second commit 를 추가해주세요!

⑦ add / commit 해주세요!

Graph	Description	Date	Author	Commit
	 rats_branch [rats_branch] commit	10 Feb 2022 03:45	BEMELON	e50fd687
	 main add .gitignore	10 Feb 2022 02:47	BEMELON	3ea8164f
	Second commit	10 Feb 2022 02:32	BEMELON	ee1ea361
	First commit	10 Feb 2022 02:00	BEMELON	34c15e92

Git 기초 명령어: git branch

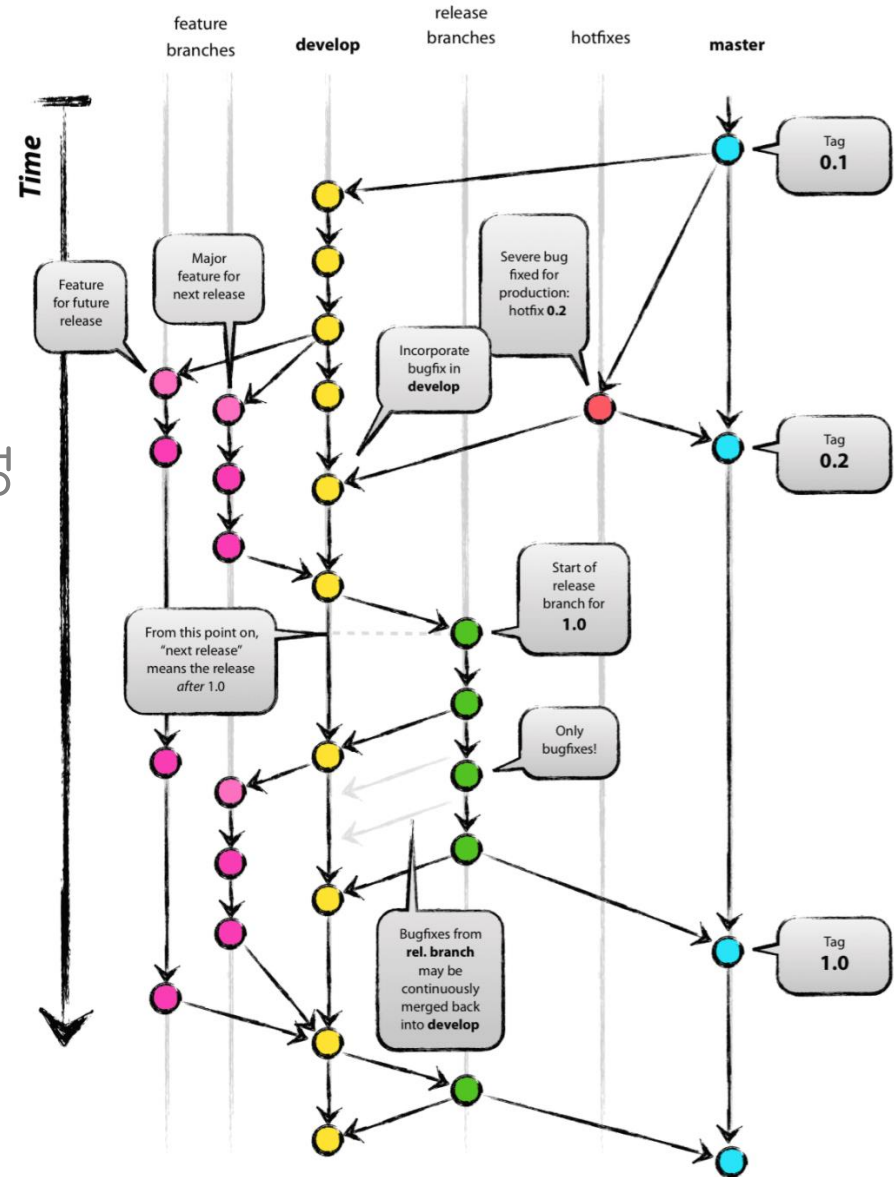
하나의 레포지토리어서 동시에 여러명이 작업을 하는 경우

git branch {branchname}

{branchname} 으로 로컬 레포지토리에 새로운 Branch 생성




git switch {branchname}

{branchname} 으로 메인 브랜치를 설정



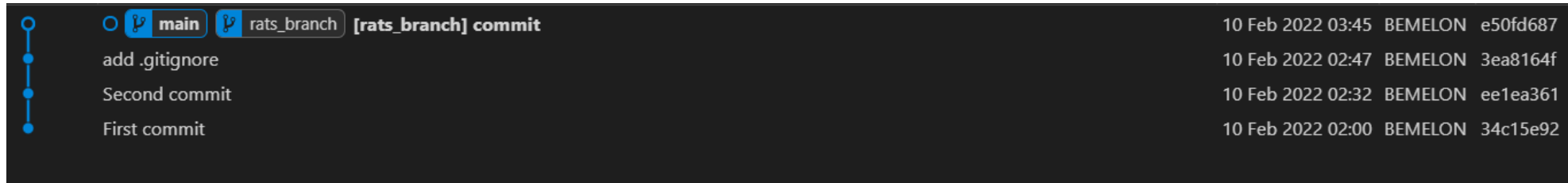
Git 실습: git branch2

- ① 명령프롬프트에서 `git branch main` 를 입력해주세요!
- ② `main.txt`, `main2.txt` 파일을 확인해보세요!
- ③ 명령프롬프트에서 `git switch rats_branch` 를 입력해주세요!
- ④ `main.txt`, `main2.txt` 파일을 확인해보세요!

Graph	Description	Date	Author	Commit
	 rats_branch [rats_branch] commit	10 Feb 2022 03:45	BEMELON	e50fd687
	 main add .gitignore	10 Feb 2022 02:47	BEMELON	3ea8164f
	Second commit	10 Feb 2022 02:32	BEMELON	ee1ea361
	First commit	10 Feb 2022 02:00	BEMELON	34c15e92

Git 실습: git merge

- ① git branch를 main 으로 이동시켜주세요!
- ② 명령팔레트에서 git merge rats_branch 를 입력해주세요!
- ③ main.txt, main2.txt 파일을 확인해보세요!
- ④ 그래프



Git 실습: git checkout

① 명령프롬프트에서 git log 를 입력해주세요!

```
commit ee1ea361e852c7165c87958418b51ae3ba46f655
Author: [REDACTED]
Date: Thu Feb 10 02:32:01 2022 +0900

    Second commit
```

② 여기서 “Second commit”의 커밋 아이디 (ee1ea36) 7글자를 기억합니다.

③ 그리고, 명령프롬프트에서 git checkout {커밋아이디 7글자} 를 입력해주세요!

④ main.txt와 main2.txt를 확인해주세요

Git 기초 명령어 : git checkout

특정 Commit 의 시점에서의 코드를 확인하고 싶을 때!

* HEAD = 현재 코드의 위치

```
* commit e50fd6874ae057043cf5ba0251526e2d50e55b13 (rats_branch, main)
Author: [redacted]
Date: Thu Feb 10 03:45:30 2022 +0900

[rats_branch] commit

* commit 3ea8164fee5640be1c355af641bcac0ec9456288
Author: [redacted]
Date: Thu Feb 10 02:47:49 2022 +0900

add .gitignore

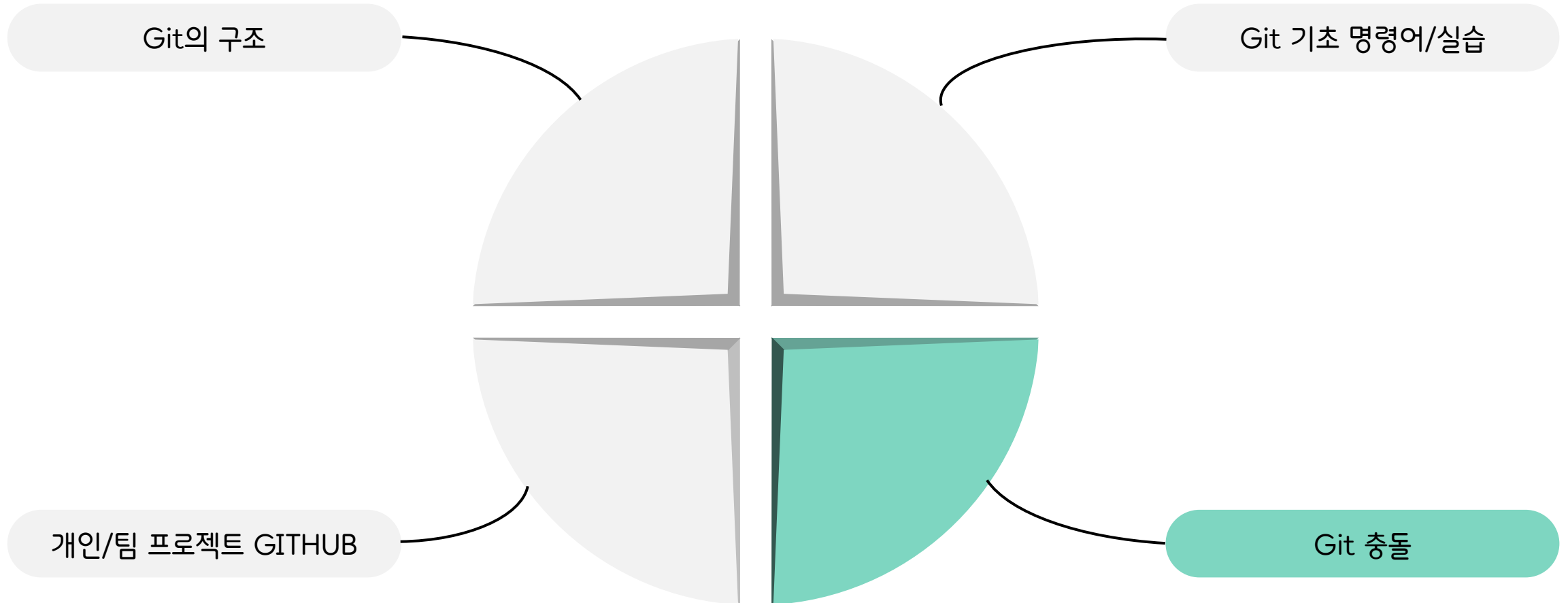
* commit ee1ea361e852c7165c87958418b51ae3ba46f655 (HEAD)
Author: [redacted]
Date: Thu Feb 10 02:32:01 2022 +0900

Second commit

* commit 34c15e9204f2789a9a9125ec49fd7aa264916a84
Author: [redacted]
Date: Thu Feb 10 02:00:39 2022 +0900

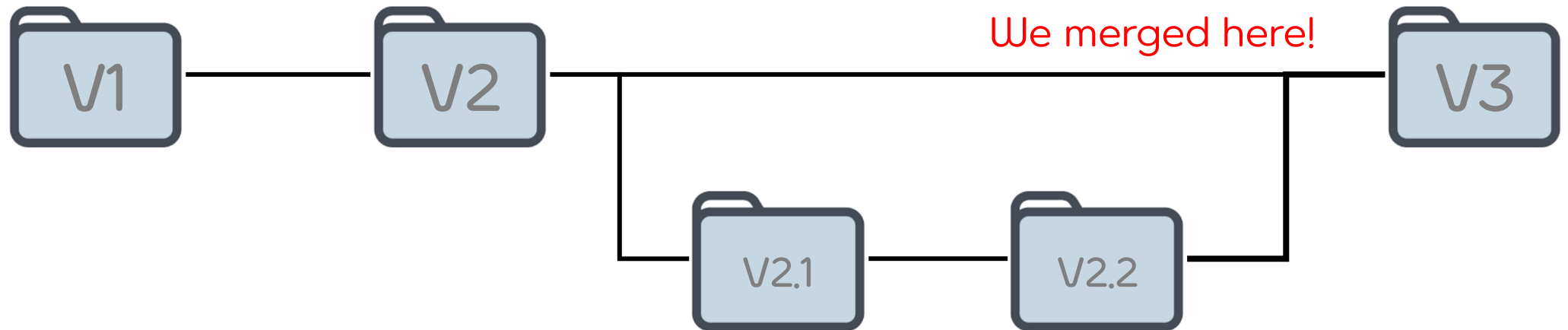
First commit
```

Section3. Git 충돌



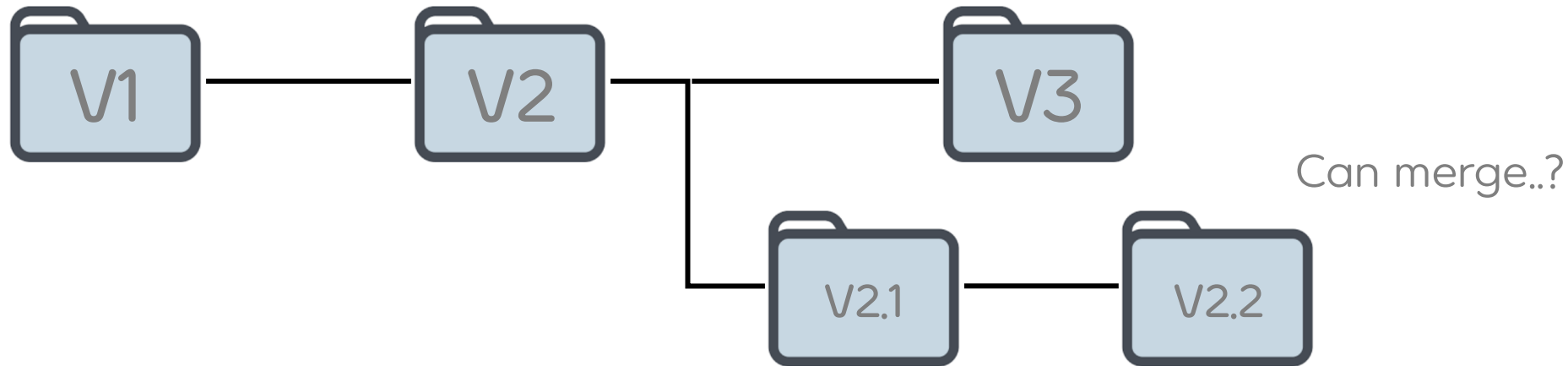
Git 충돌: merge (희망편)

- ① 앞 전에 merge를 진행했었습니다.
- ② 근데 이게 생각보다.... 무서운 케이스들이 존재
- ③ 우리가 했던 이상적인 merge



Git 충돌: merge (절망편)

① 학부생의 git merge (일단 누군가의 코드가 날라가지 않으면 다행)



② V2_1, V2_2 버전들은 모두 V2의 코드를 기반으로 작성된 코드

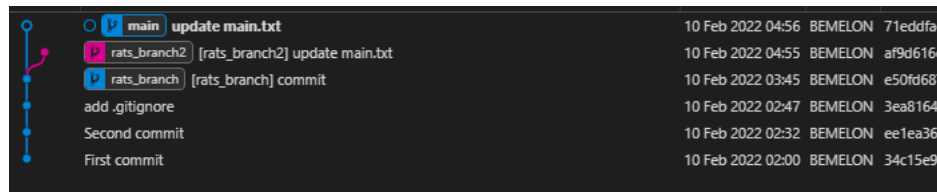
③ 막상 merge 하려고 보니, 베이스 코드가 V3로 변경됨.

* 집으로 다시 오려고 했더니 집은 이사갔고 아무도 안알려줌.

Git 충돌: 실습^^

- ① git HEAD를 최신으로 이동시켜주세요.
- ② 이름이 rats_branch2 인 브랜치를 만들고 rats_branch2로 switch 해주세요.
- ③ main.txt에 {현재 시간} : [rats_branch2] update main2.txt 를 추가해주세요!
- ④ main2.txt에 {현재 시간} : [rats_branch2] Third commit 를 추가해주세요!
* Add / commit 해주세요!
- ⑤ 다시 main으로 브랜치를 switch 해주세요
- ⑥ main.txt에 {현재 시간} : update main2.txt 를 추가해주세요!
- ⑦ main2.txt에 {현재 시간} : Third commit 를 추가해주세요!
* Add / commit 해주세요!

- ⑧ **git merge rats_branch2 ...**



Git 충돌: 실습^^

⑧ `git merge rats_branch2` ...

```
Auto-merging main.txt
CONFLICT (content): Merge conflict in main.txt
Auto-merging main2.txt
CONFLICT (content): Merge conflict in main2.txt
Automatic merge failed; fix conflicts and then commit the result.
```

⑨ vscode에서 main.txt main2.txt를 살펴보면 ...

```
02:00 : First Commit
02:26 : add main2.txt
03:44 : [rats_branch] update main2.txt
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<< HEAD (Current Change)
04:55 : update main2.txt
=====
04:51 : [rats_branch2] update main2.txt
>>>>>> rats_branch2 (Incoming Change)
```

```
02:26 : First Commit
03:44 : [rats_branch] Second commit
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<< HEAD (Current Change)
04:55 : Third commit
=====
04:51 : [rats_branch2] Third commit
>>>>>> rats_branch2 (Incoming Change)
```





⑩ <<<<<< HEAD / ===== / >>>>>> {branchname} 를 잘 설정하는게 포인트

Git 충돌: 해결방법

```
02:00 : First Commit
02:26 : add main2.txt
03:44 : [rats_branch] update main2.txt
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<< HEAD (Current Change)
04:55 : update main2.txt
=====
04:51 : [rats_branch2] update main2.txt
>>>>>> rats_branch2 (Incoming Change)
```

```
02:26 : First Commit
03:44 : [rats_branch] Second commit
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<< HEAD (Current Change)
04:55 : Third commit
=====
04:51 : [rats_branch2] Third commit
>>>>>> rats_branch2 (Incoming Change)
```

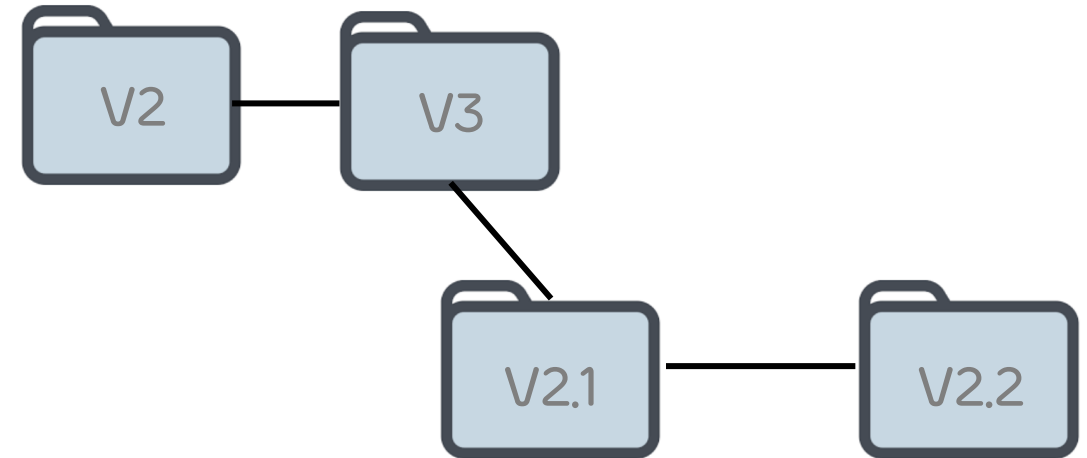
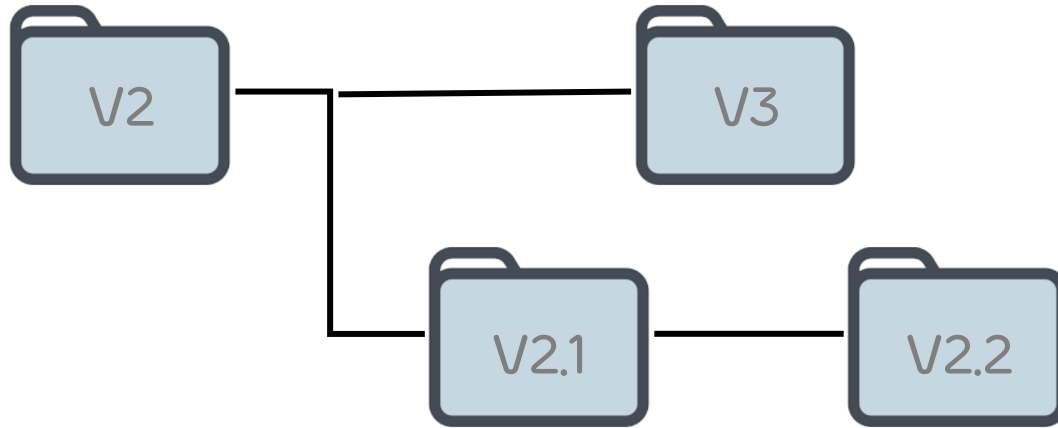
- ① <<<<<< HEAD / ===== / >>>>>> {branchname} 를 잘 설정하는게 포인트
- ② 어떤 코드를 살려야 되는지 분석 후 적절하게 살려야 됨.













	 main solve conflict	10 Feb 2022 05:06	BEMELON	f848342b
	update main.txt	10 Feb 2022 04:56	BEMELON	71eddfac
	 rats_branch2 [rats_branch2] update main.txt	10 Feb 2022 04:55	BEMELON	af9d616d
	 rats_branch [rats_branch] commit	10 Feb 2022 03:45	BEMELON	e50fd687
	add .gitignore	10 Feb 2022 02:47	BEMELON	3ea8164f
	Second commit	10 Feb 2022 02:32	BEMELON	ee1ea361
	First commit	10 Feb 2022 02:00	BEMELON	34c15e92

Git 충돌: rebase 실습

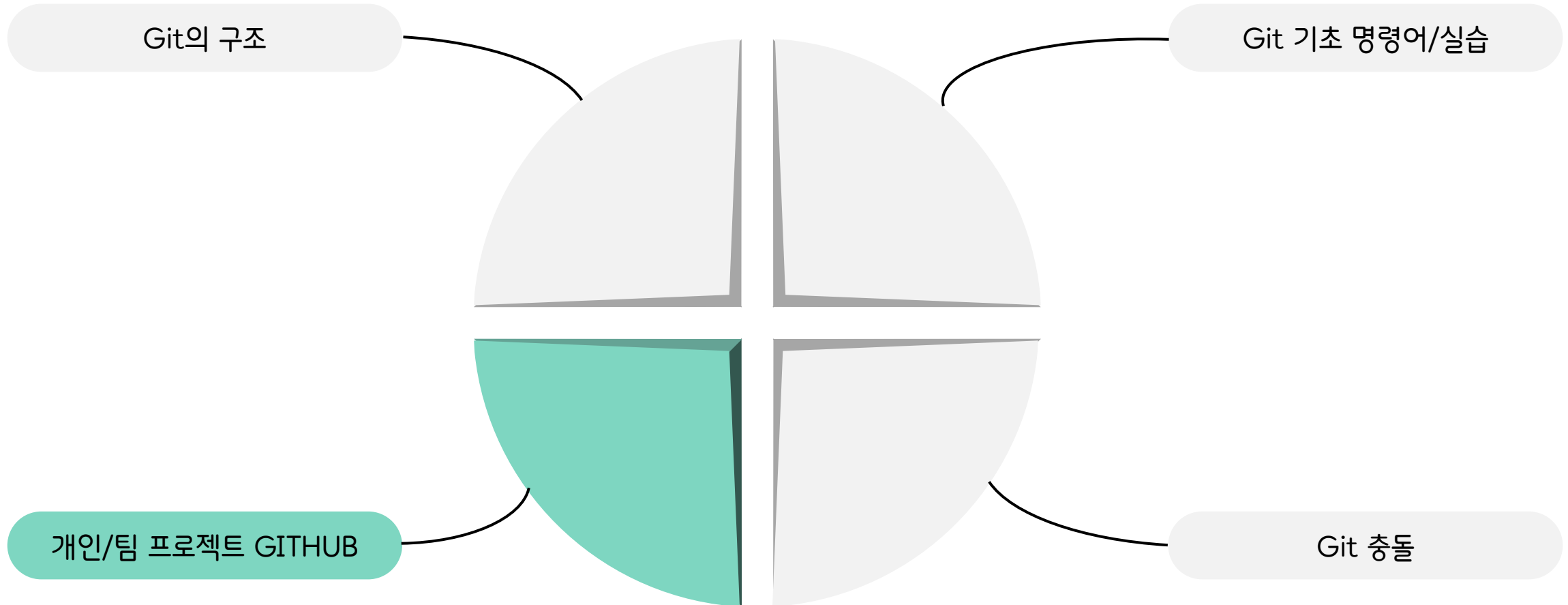
- ① git HEAD를 최신으로 이동시켜주세요.
- ② 이름이 rats_branch3 인 브랜치를 만들고 rats_branch3로 switch 해주세요.
- ③ main.txt에 {현재 시간} : [rats_branch3] update main2.txt 를 추가해주세요!
- ④ main2.txt에 {현재 시간} : [rats_branch3] Fourth commit 를 추가해주세요!
* Add / commit 해주세요!
- ⑤ 다시 main으로 브랜치를 switch 해주세요
- ⑥ main.txt에 {현재 시간} : update main2.txt 를 추가해주세요!
- ⑦ main2.txt에 {현재 시간} : Fourth commit 를 추가해주세요!
* Add / commit 해주세요!
- ⑧ 명령팔레트에 git rebase rats_branch3 을 입력해주세요
- ⑨ 명령팔레트에 git merge rats_branch3 를 입력해주세요

Git 충돌: rebase

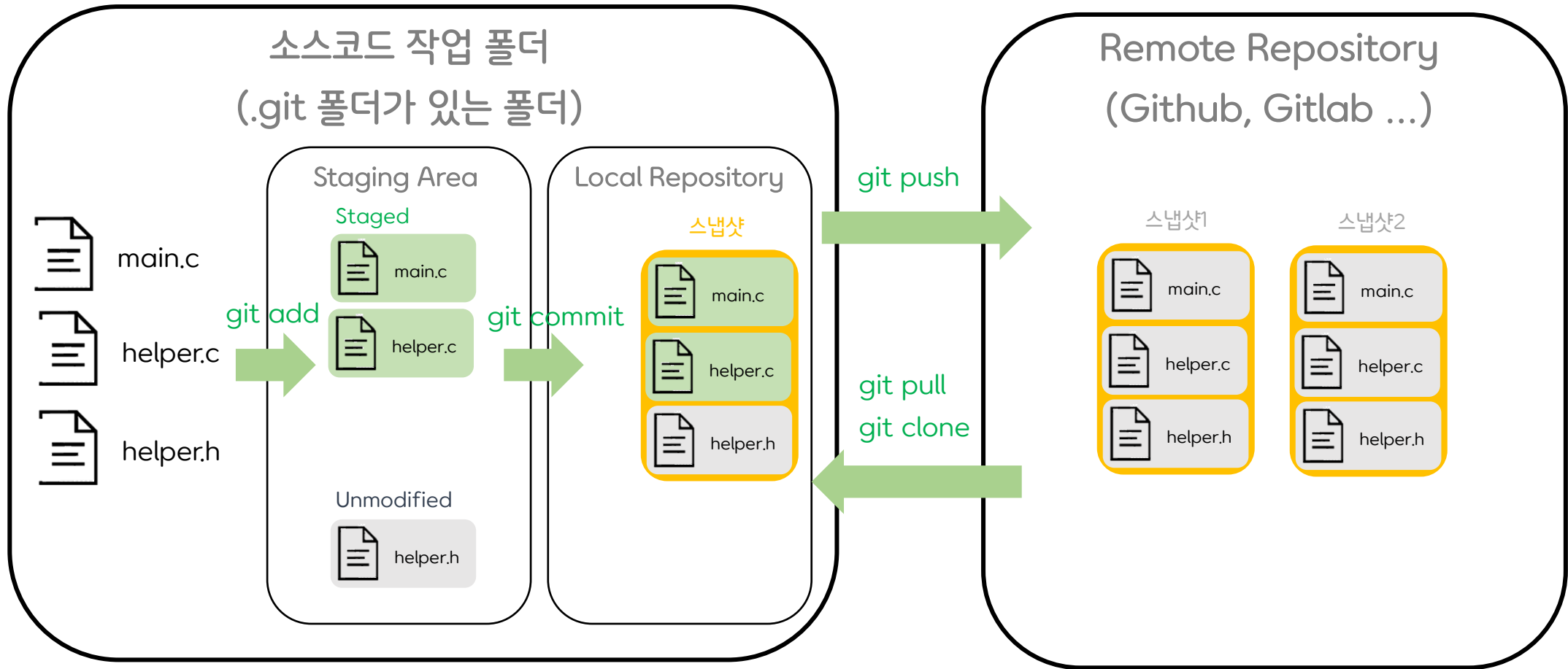


Graph	Description	Date	Author	Commit
	 rats_branch3 [rats_branch3] commit	10 Feb 2022 05:44	BEMELON	0c38d0df
	main commit	10 Feb 2022 05:45	BEMELON	a354231c
	solve conflict	10 Feb 2022 05:06	BEMELON	f848342b
	update main.txt	10 Feb 2022 04:56	BEMELON	71eddfac
	 rats_branch2 [rats_branch2] update main.txt	10 Feb 2022 04:55	BEMELON	af9d616d
	 rats_branch [rats_branch] commit	10 Feb 2022 03:45	BEMELON	e50fd687
	add .gitignore	10 Feb 2022 02:47	BEMELON	3ea8164f
	Second commit	10 Feb 2022 02:32	BEMELON	ee1ea361
	First commit	10 Feb 2022 02:00	BEMELON	34c15e92

Section4. 개인/팀 프로젝트 GITHUB

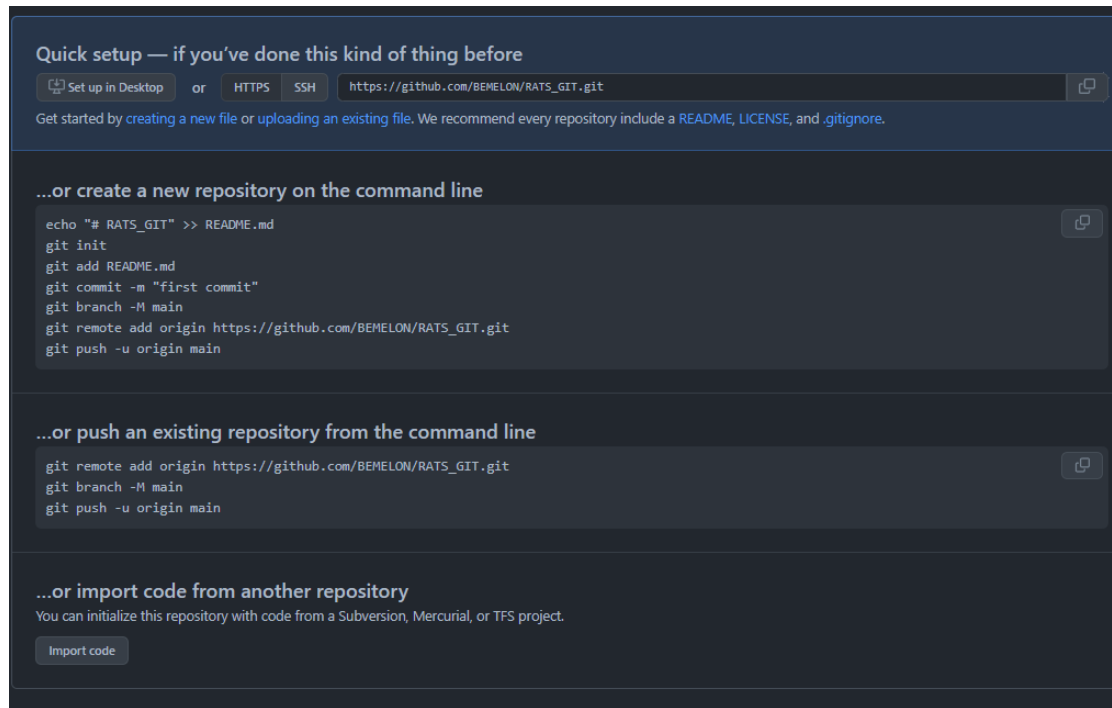


개인/팀 프로젝트 Github



개인/팀 프로젝트 Github

- ① Github에서 레포지토리 하나를 만들어주세요 ~ (이름은 자유!)
- ② HTTPS/SSH 중 하나를 선택해서 복사하면 되는데, SSH 쪽으로 접근하시는게 요즘 추세긴 합니다



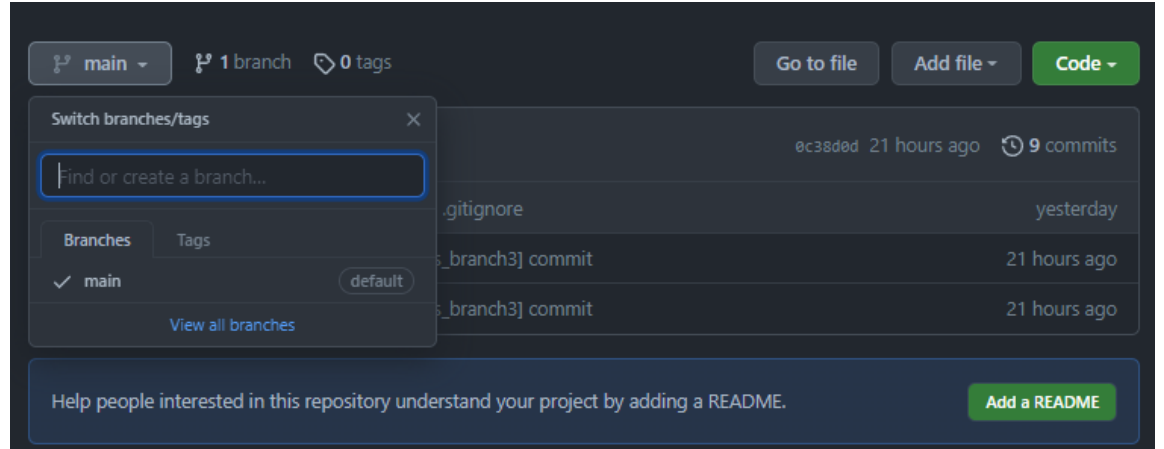
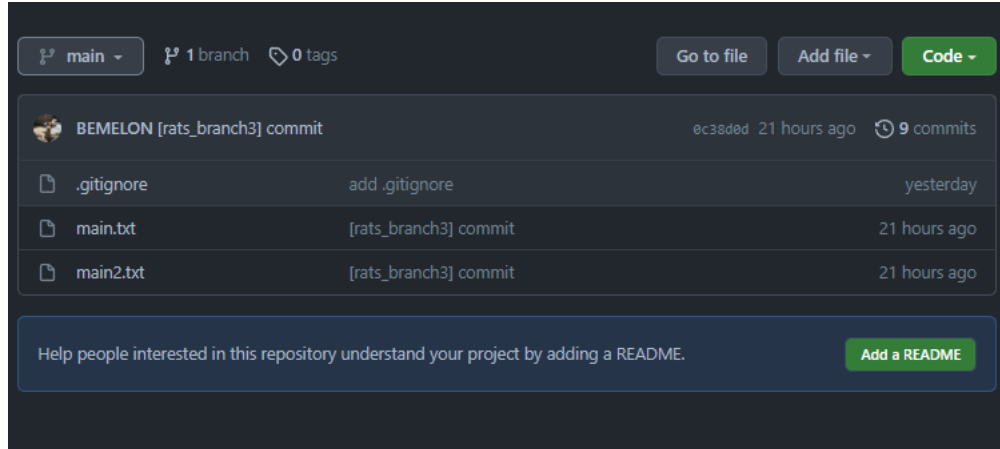
개인/팀 프로젝트 Github

③ 다시 vscode 명령팔레트에서 `git remote add origin {복사한 주소}` 를 입력해주세요!

④ vscode 명령팔레트에서 `git remote -v` 를 입력해주세요!

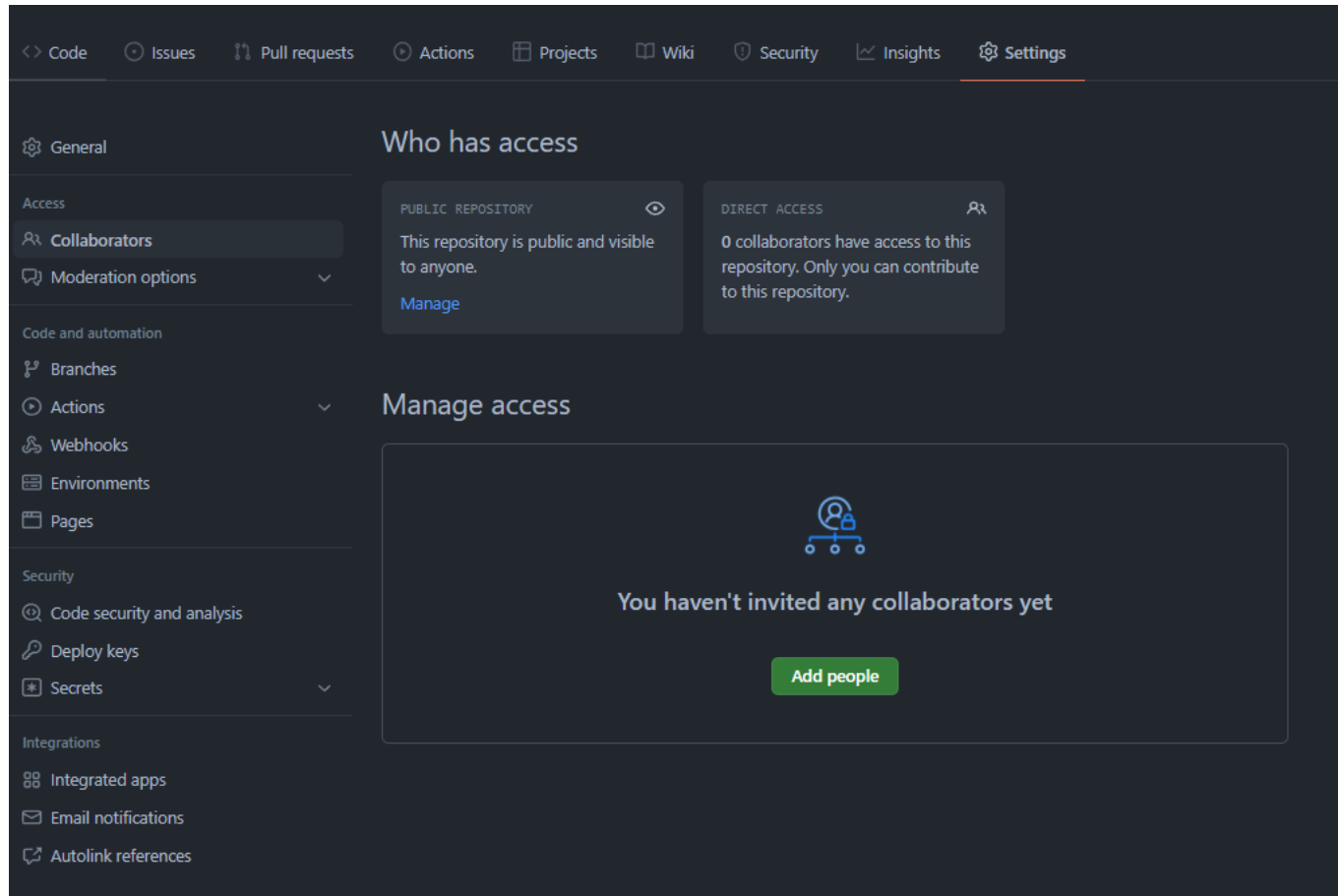
```
origin https://github.com/BEMELON/RATS_GIT.git (fetch)
origin https://github.com/BEMELON/RATS_GIT.git (push)
```

⑤ vscode 명령팔레트에서 `git push origin main` 를 입력해주세요!



프로젝트에서 같이 작업하는 법!

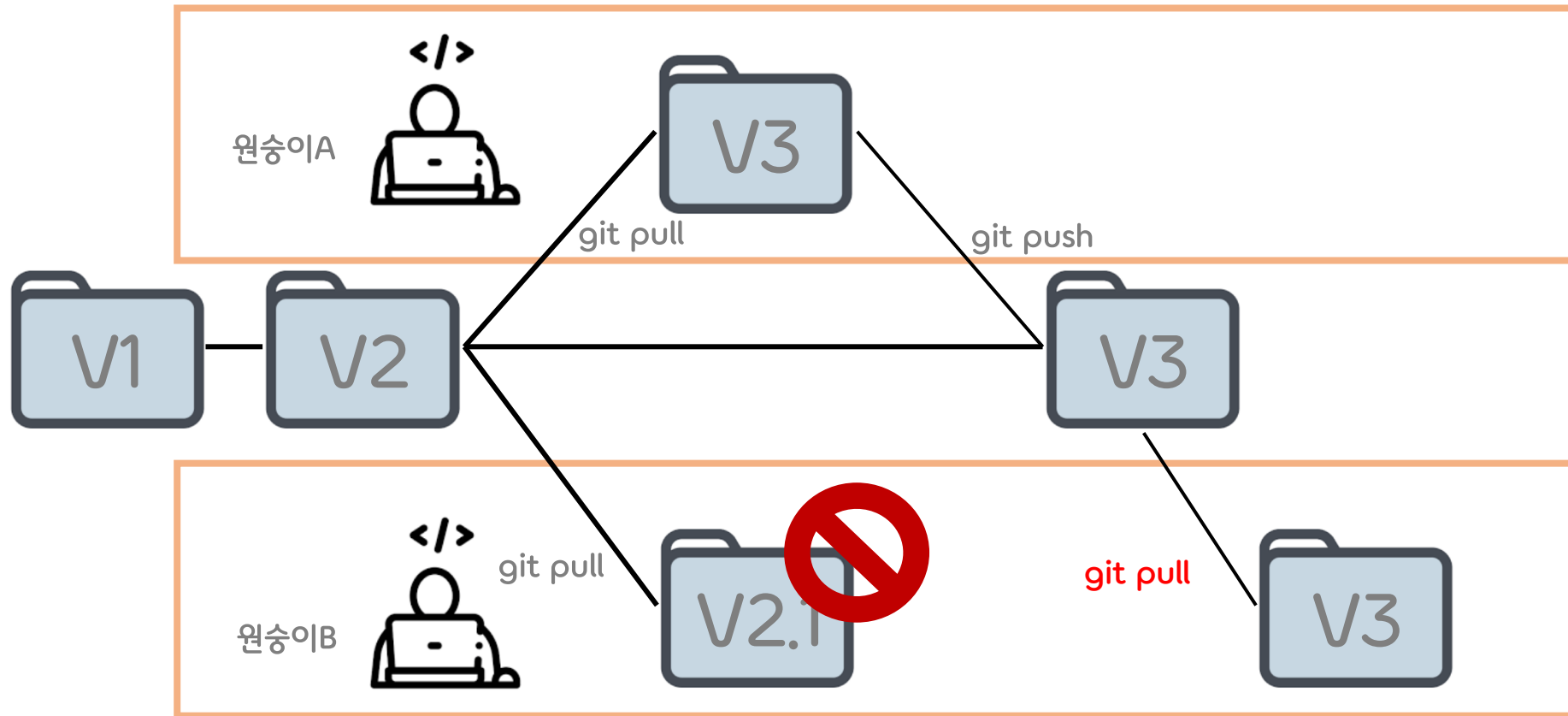
① Github 레포지토리 - Settings - Collaborators 에 동료들을 추가해야함~!



협업 프로젝트에서 충돌....

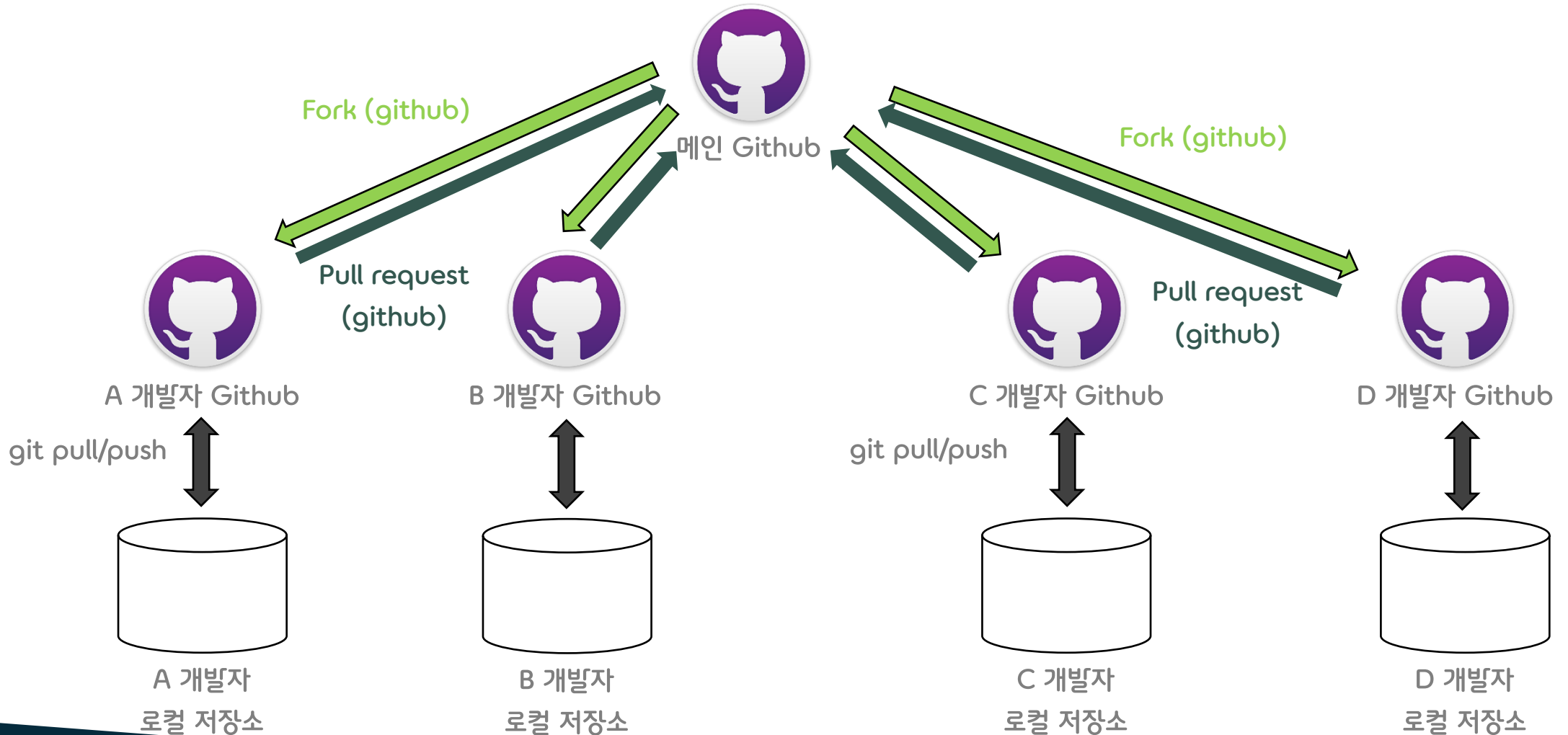
진짜 정말 **git rebase** 정말 좋아요!
정말로, conflict 문제 대부분 해결 가능!
진짜임.

- ① 이전에는 혼자 작업했기 때문에 충돌 날 일이 극히 드물었음.
- ② 동시에 작업하기 때문에 대부분의 경우가 충돌 가능성을 가지고 있음....

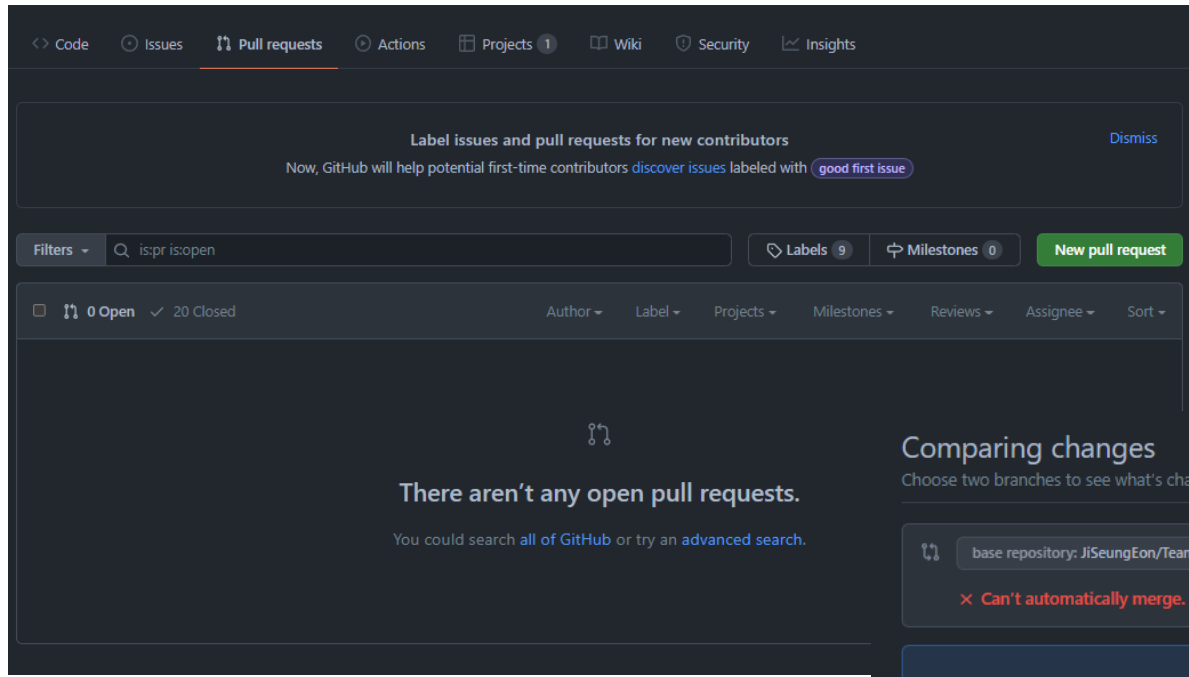


그럼 Github는..?

Github는 영리적인 서비스와 오픈소스를 위한 무상서비스



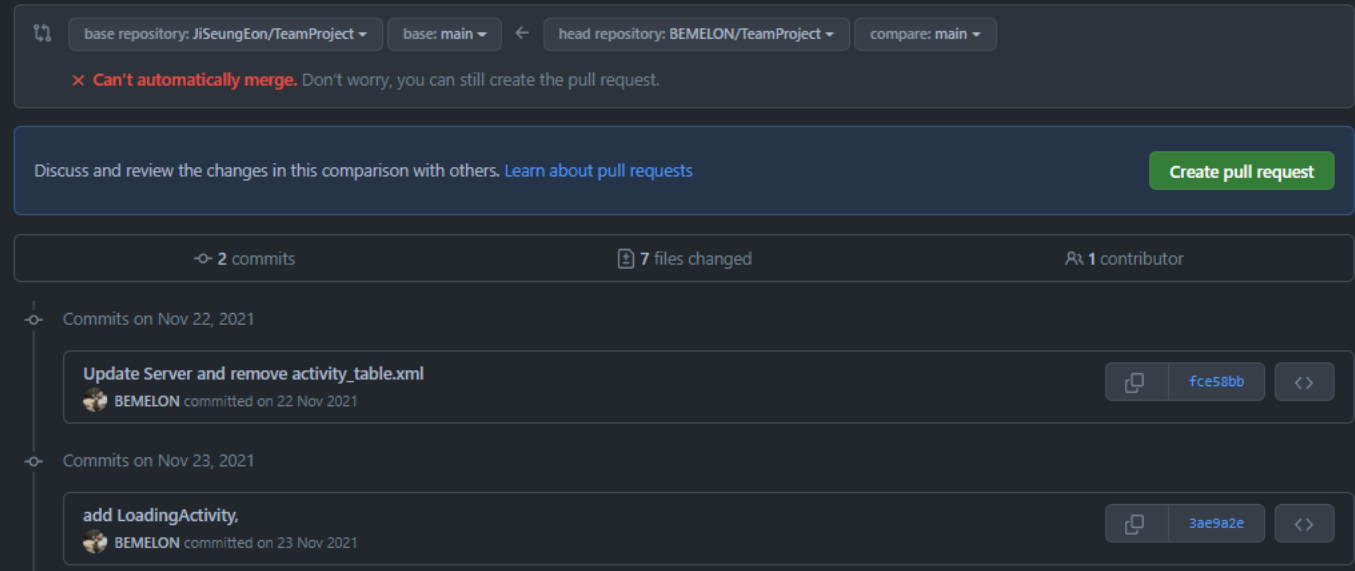
Pull Request



The screenshot shows the GitHub interface with the 'Pull requests' tab selected. At the top, there's a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and Insights. Below this, a message states: 'Label issues and pull requests for new contributors. Now, GitHub will help potential first-time contributors discover issues labeled with `good first issue`.' A 'Dismiss' link is to the right. The main area has a search bar with 'is:pr is:open' and buttons for 'Labels 9' and 'Milestones 0'. A green 'New pull request' button is on the right. Below the search bar, there's a summary bar showing '0 Open' and '20 Closed' pull requests, along with filters for Author, Label, Projects, Milestones, Reviews, Assignee, and Sort. The main content area displays a message: 'There aren't any open pull requests. You could search all of GitHub or try an advanced search.'

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](#).



The screenshot shows the 'Comparing changes' interface. At the top, there's a search bar with 'is:pr is:open' and buttons for 'Labels 9' and 'Milestones 0'. A green 'New pull request' button is on the right. Below the search bar, there's a summary bar showing '0 Open' and '20 Closed' pull requests, along with filters for Author, Label, Projects, Milestones, Reviews, Assignee, and Sort. The main content area displays a message: 'There aren't any open pull requests. You could search all of GitHub or try an advanced search.'

The 'Comparing changes' section shows a comparison between 'base repository: JiSeungEon/TeamProject' and 'head repository: BEMELON/TeamProject'. The base branch is 'main' and the head branch is 'main'. A message states: 'Can't automatically merge. Don't worry, you can still create the pull request.'

Below the comparison, there's a section for 'Discuss and review the changes in this comparison with others. [Learn about pull requests](#)'. A green 'Create pull request' button is on the right.

The comparison shows 2 commits, 7 files changed, and 1 contributor. The commits are listed as follows:

- Commits on Nov 22, 2021
 - Update Server and remove activity_table.xml
 - BEMELON committed on 22 Nov 2021
- Commits on Nov 23, 2021
 - add LoadingActivity.
 - BEMELON committed on 23 Nov 2021

Pull Request 실습! (선택)

① 제 레포지토리를 Fork 해주세요!

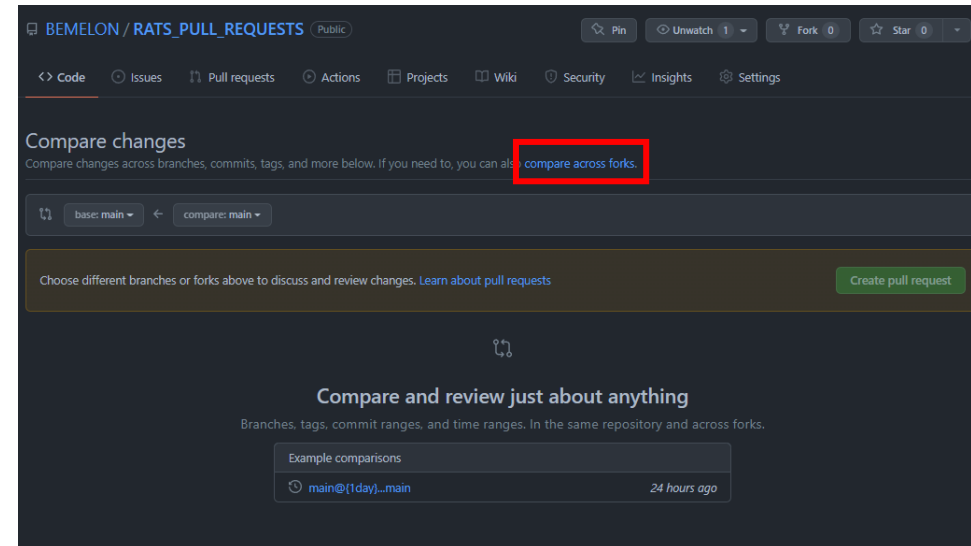
* (https://github.com/BEMELON/RATS_PULL_REQUESTS)

② 로컬 레포지토리에 Fork한 레포지토리를 clone 해주세요!

`git clone {자기 레포지토리 주소}`

③ 자기 학번으로 되어있는 파일을 만들고 push 해주세요!

④ 제 레포지토리에서 pull request를 열어주세요!!



감사합니다!

RATS 25기 황규도

