

GitHub 으로 협업 프로젝트!

조장과 조원 역할

조장 1명, 역할 : Github Remote Repository **Owner**

조원 n명, 역할 :Github Remote Repository **Member**

- 조장님이 Repo를 하나 파고,
- 조원들이 Push 할 수 있는 권한을 부여해서,
- 함께 개발할 수 있는 저장소를 만든다.

생각해보자.

Public으로 만든 팀 Repository을 만들었다면

- 누구나 소스코드 읽기 권한이 있을까?
- 누구나 Push 할 권한이 있을까?

Private으로 만든 팀 Repository을 만들었다면

- 누구나 소스코드 읽기 권한이 있을까?
- 누구나 Push 할 권한이 있을까?

팀 프로젝트

Text 문서를 사용하여, 채팅을 한다.

1. 조장님이 chat.txt 파일을 생성, 첫 번째 질문을 적고 push 하기
2. 멤버1 님이 최신 chat.txt 파일로 pull 후,
chat.txt 파일에 대답을 적음, 그리고 다음 사람을 위한 질문을 적고 push
3. 멤버2 님이 최신 chat.txt 파일로 pull 후,
chat.txt 파일에 대답을 적음, 그리고 다음 사람을 위한 질문을 적고 push
4. 반복 (조장 > 멤버1 > 멤버2..n > 조장 > 멤버1 > 멤버2 .. n > 반복)

채팅 내용

- 질문에 대한 대답을 하고, 다음사람에게 질문 한다.
- 취미 / 좋아하는 음식 / 운동, 영화, 등 간단히 대답할 수 있는 질문들을 한다.

[조장님만 수행] Repository 만들기

Remote Repo name

- teamHobby

공개여부

- public

The image shows the GitHub 'Create a new repository' form and the resulting repository page. The form is on the left, and the repository page is on the right. Red boxes highlight specific parts of the form: the 'Repository name' field, the 'Description' field, and the 'Add a README file' checkbox. The repository page shows the 'Initial commit' with a 'README.md' file that contains the text 'teamHobby' and '우리 팀원을 알아가는 미션'.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * yts0275 / Repository name * teamHobby ✓

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

Description (optional)

우리 팀원을 알아가는 미션

☒ 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:

Skip this step if you're importing an existing repository.

☒ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None

Choose a license

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

License: None

This will set `master` as the default branch. Change the default name in your [settings](#).

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

Create repository

master 1 branch 0 tags

yts0275 Initial commit

README.md Initial commit

README.md

teamHobby

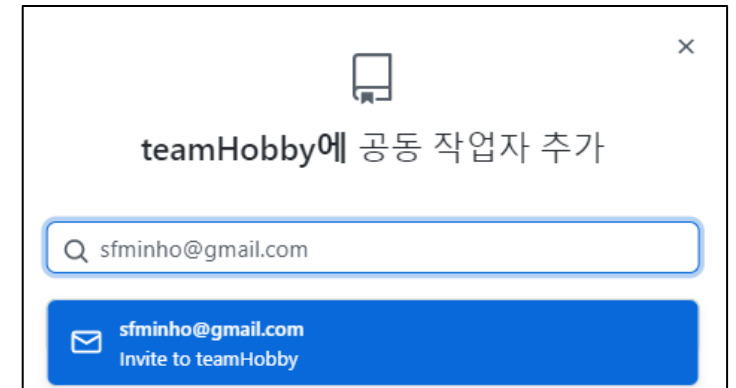
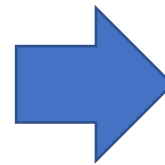
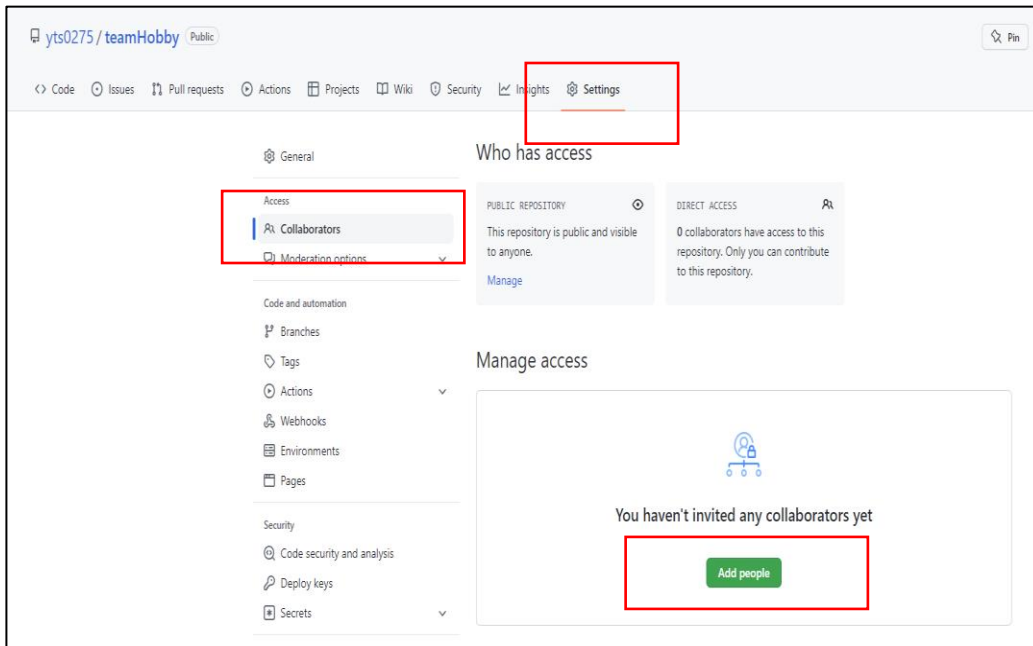
우리 팀원을 알아가는 미션

Description 내용이 자동으로 readme.md 파일에 적혀진다.

[조장님만 수행] 팀원 초대하기

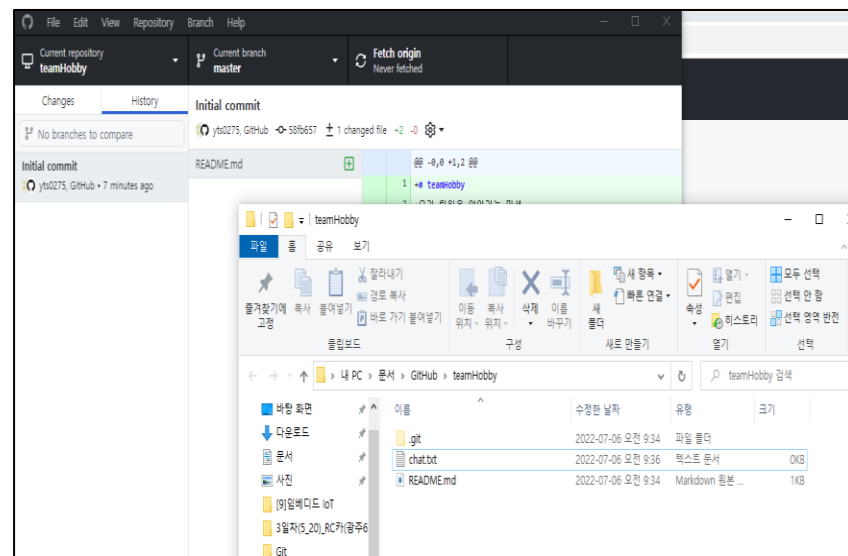
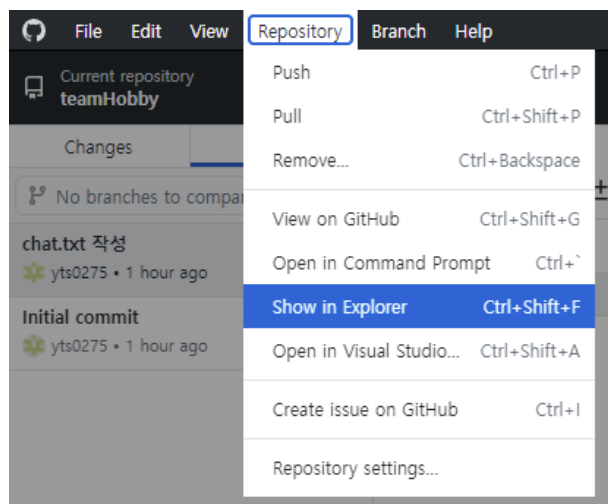
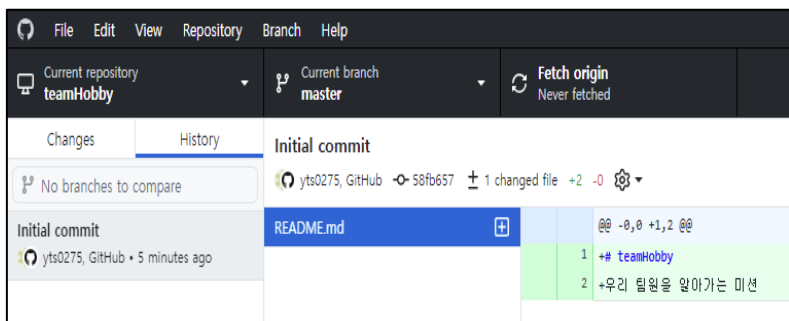
Settings > Manage access 접속

- **조원님들은 조장님께 github 아이디 (이메일주소)를 전달주세요.**



[조장님만 수행] clone 작업폴더

1. Clone
2. Show in Explorer 로 폴더 열기
3. chat.txt 파일 생성



[조장님만 수행] chat.txt 파일 작성

chat.txt 파일을 다음과 같이 작성한다.

```
chat.txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
## 지역 n조의 대화방 규칙
질문과 답변으로 구성됩니다.
1. 최신 파일을 pull 합니다. (fetch > pull)
2. 본인이 답변할 차례가 되었을 때, 질문에 대한 답변을 적습니다.
3. 다음 사람에게 할 질문을 적습니다.

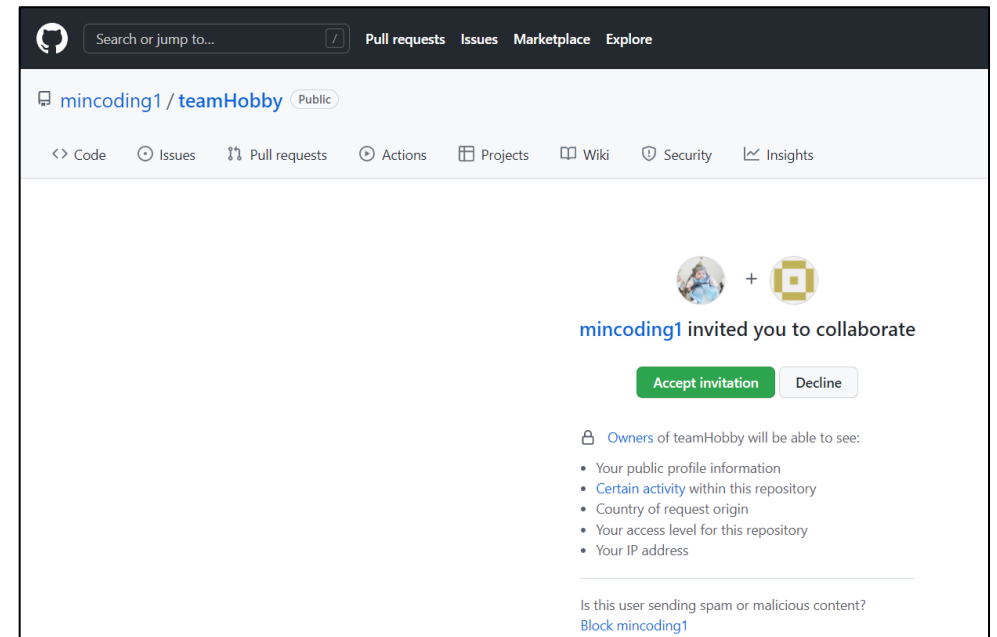
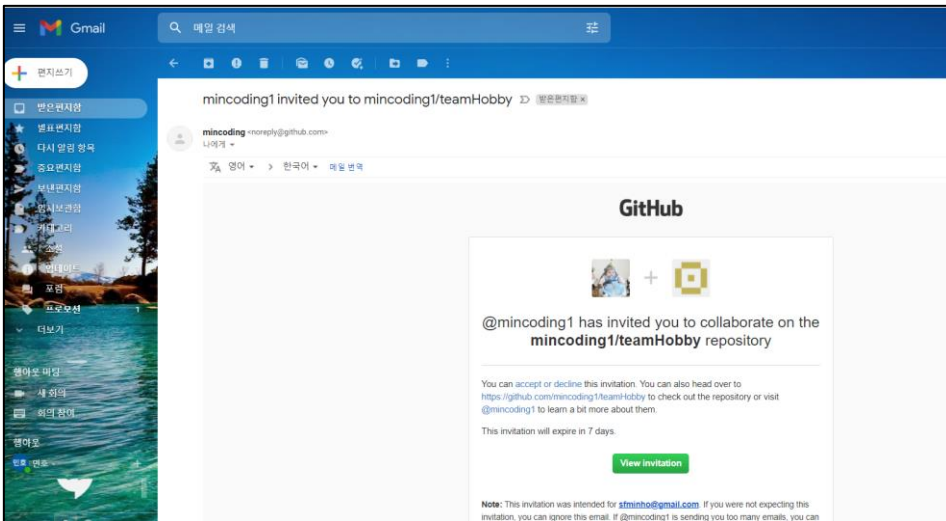
---

## 채팅방

홍길동 : How Are You?
```


[조원님들 수행] 초대에 승락하기

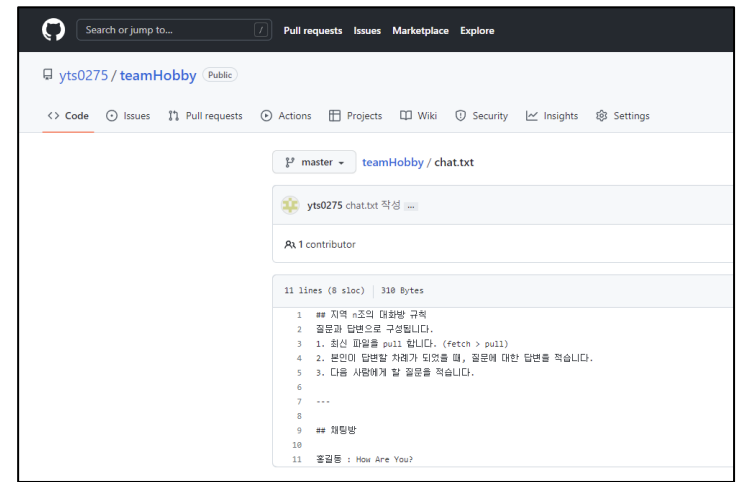
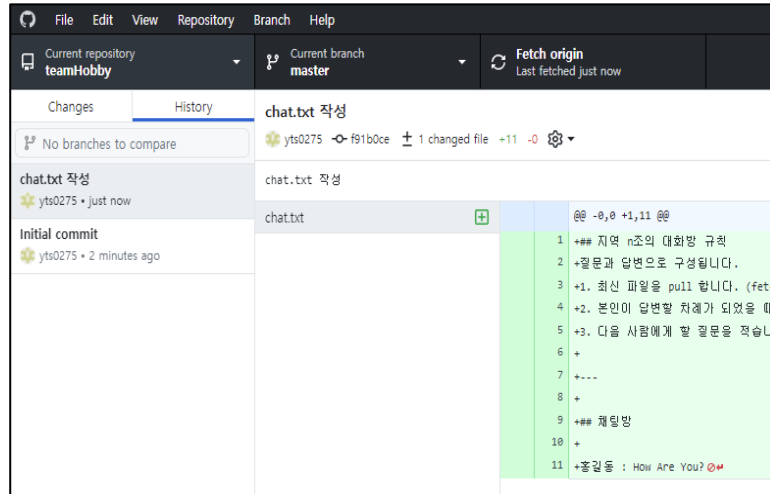
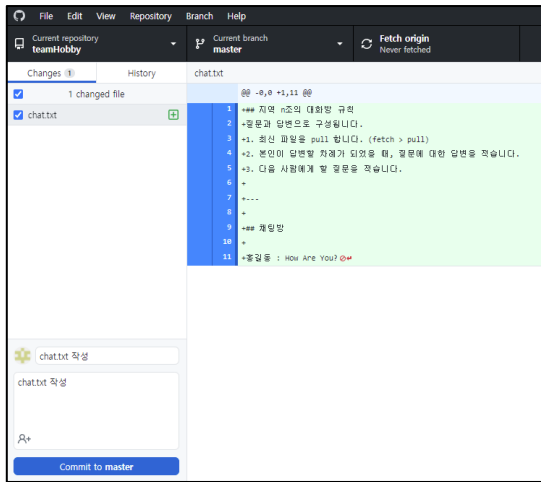
조원님들은
초대에 승락해주세요.



[조장님만 수행] commit 후 push 하기

commit > push > github 결과 확인

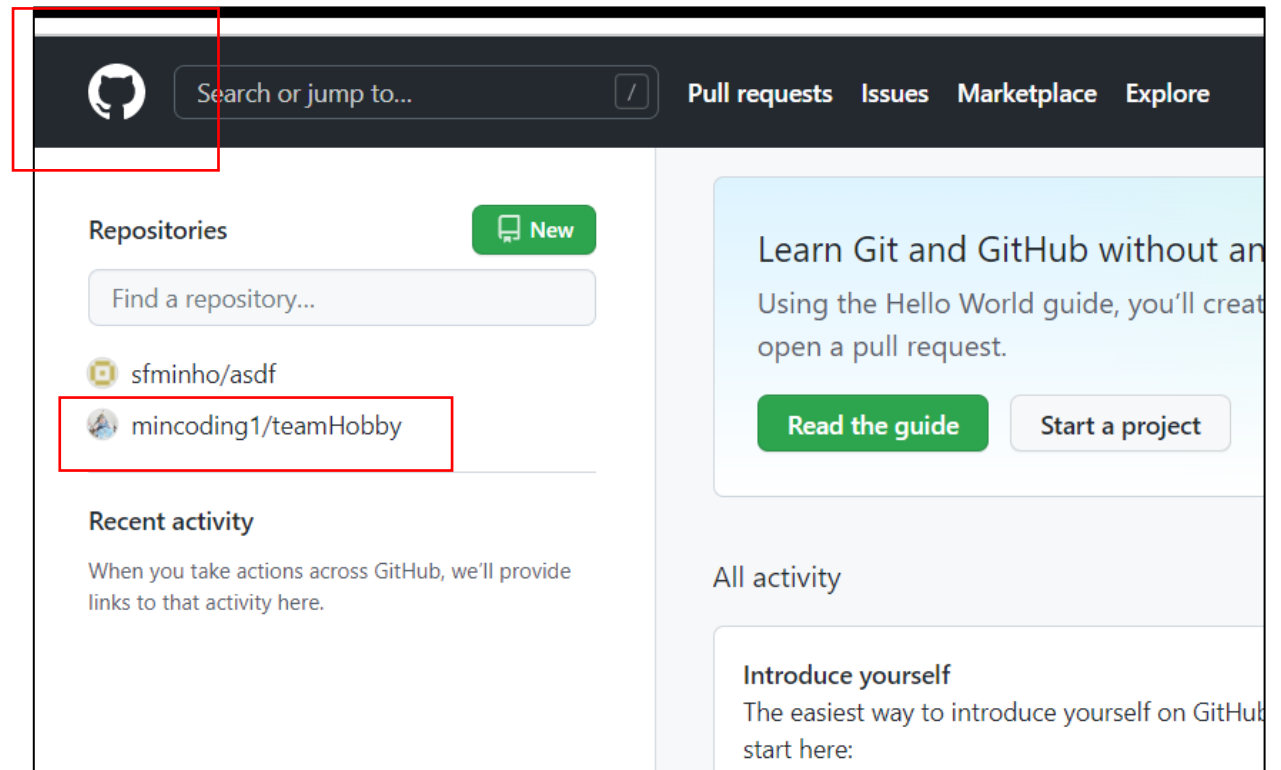
github 주소를, 강사님에게 Maternmost로 알려준다



[조원님들 수행] 팀 Repository로 이동

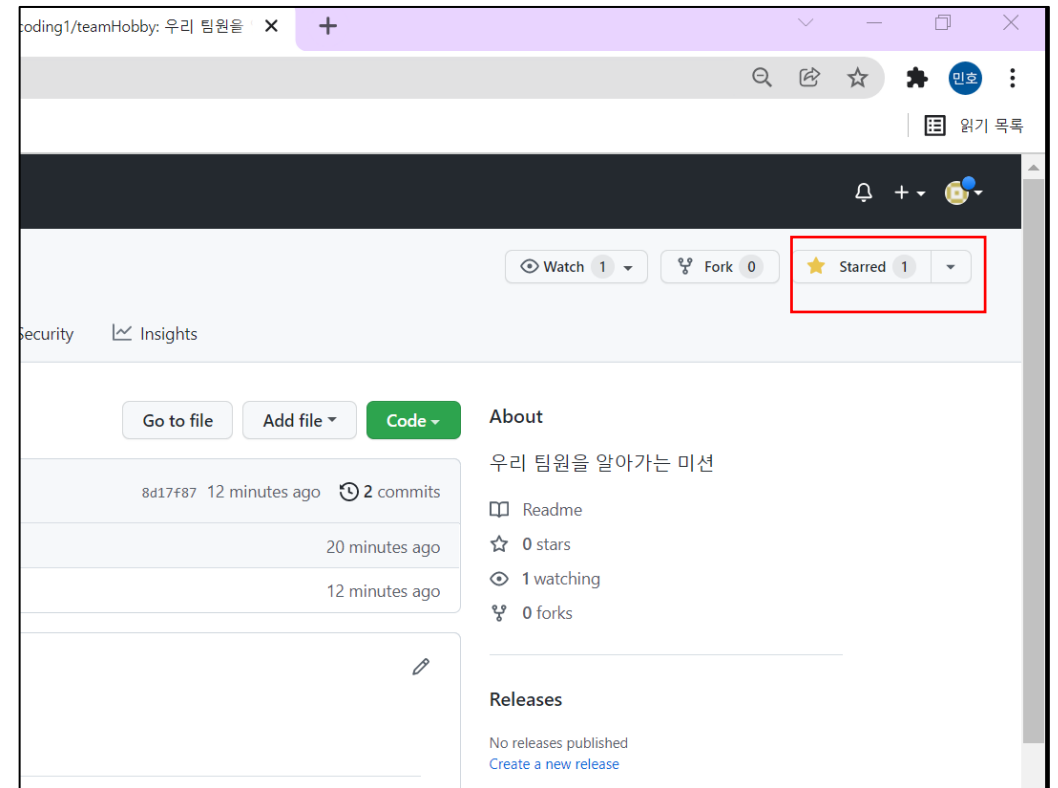
Github 마스코드 클릭

→ 옥토캣(Octocat)
• 고양이머리 + 문어다리



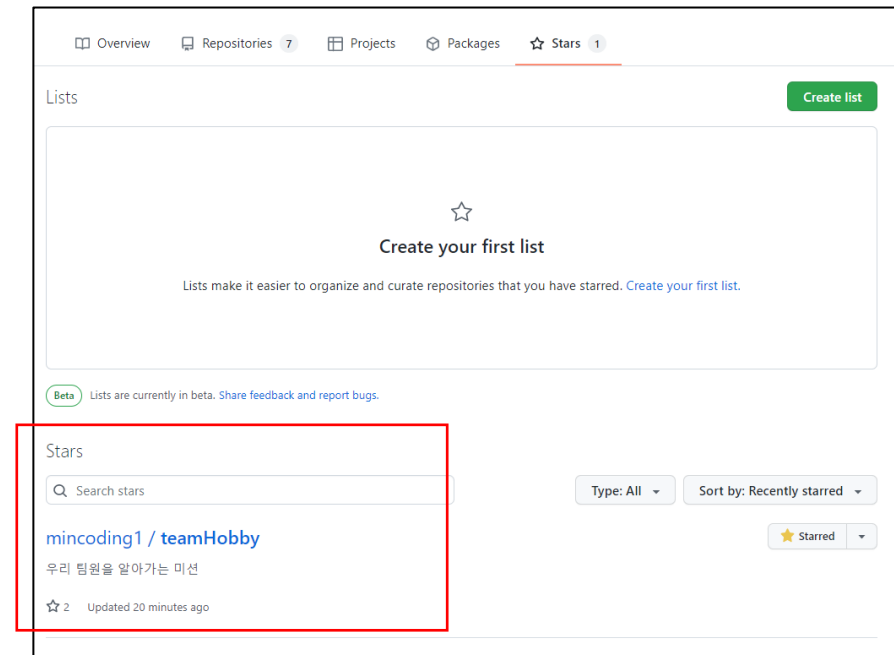
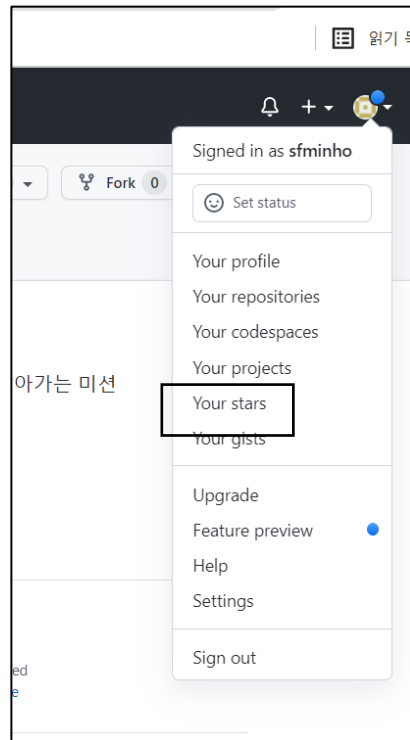
[조원님들 수행] 팀 Repository로 이동

팀 Repo. 오른쪽 최상단,
즐거찾기로 등록
• Star 버튼 클릭



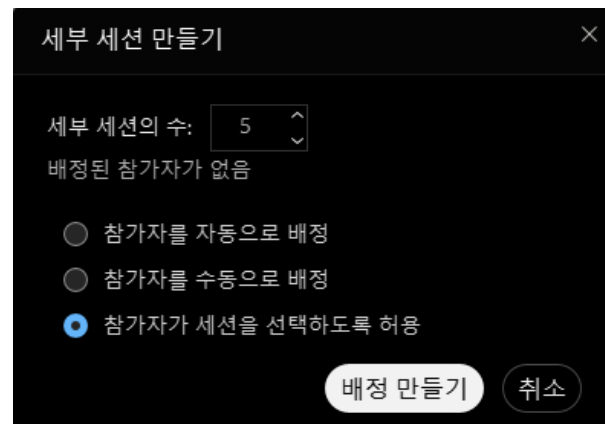
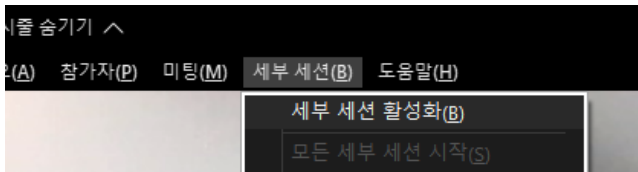
[조원님들 수행] Star 메뉴 확인

Star로 접속하면,
즐거찾기 되어있는 것을 확인할 수 있다.



[강사가 팀별 세션을 만듦] 세션 만들기

팀 프로젝트를 위한, 강사가 세션을 만든다.



팀별로 채팅 시작

프로젝트 시작

File Edit View Repository Branch Help

Current repository teamHobby

Current branch main

Fetch origin
Last fetched just now

Changes History

No branches to compare

답변 완료!
inho.choi • just now

날씨 물어 봅니다
sfminho • 3m

chat.md 파일 첫 생성
inho.choi • 27m

Initial commit
mincoding • 36m

답변 완료!

inho.choi ba97509 1 changed file +6 -3

답변을 했다!

chat.md

8	8	@@ -8,6 +8,9 @@
9	9	## 채팅방
10	10	
11		-홍길동 : How Are You?
12		-김철수 : I AM FINE THANKYOU?
13		-김철수 : HOW'S WEATHER DODAY? ☹️
11		+홍길동 : How Are You?
12		+김철수 : I AM FINE THANKYOU?
13		+김철수 : HOW'S WEATHER DODAY?
14		+Mr.알링 : 날씨는 화창하다 휴먼
15		+Mr.알링 : 소는 어느 나라 것이 맛있어요?
16		+

서로 적극적으로 모르는 부분을 물어본다!!!
질문은 적극적으로, 답변을 열심히 도와주자!!

4단계로 되어있는 Git 구조!

그동안 봐왔던 구조

이렇게 3 단계로 생각을 해왔다.



Remote
Repo.

Local
Repo.

작업
폴더

이제, 4단계 구조로 생각하자

작업 폴더에 파일이 존재한다.

Remote
Repo.

Local
Repo.

Staging Area
<커밋 대기실>

작업
폴더



PullMePullMePullMeUp.txt

Staging Area

작업 폴더에 있는 파일을
커밋을 하기 전 대기실인
Staging Area 에 추가한다.



PullMePullMePullMeUp.txt

Remote
Repo.

Local
Repo.

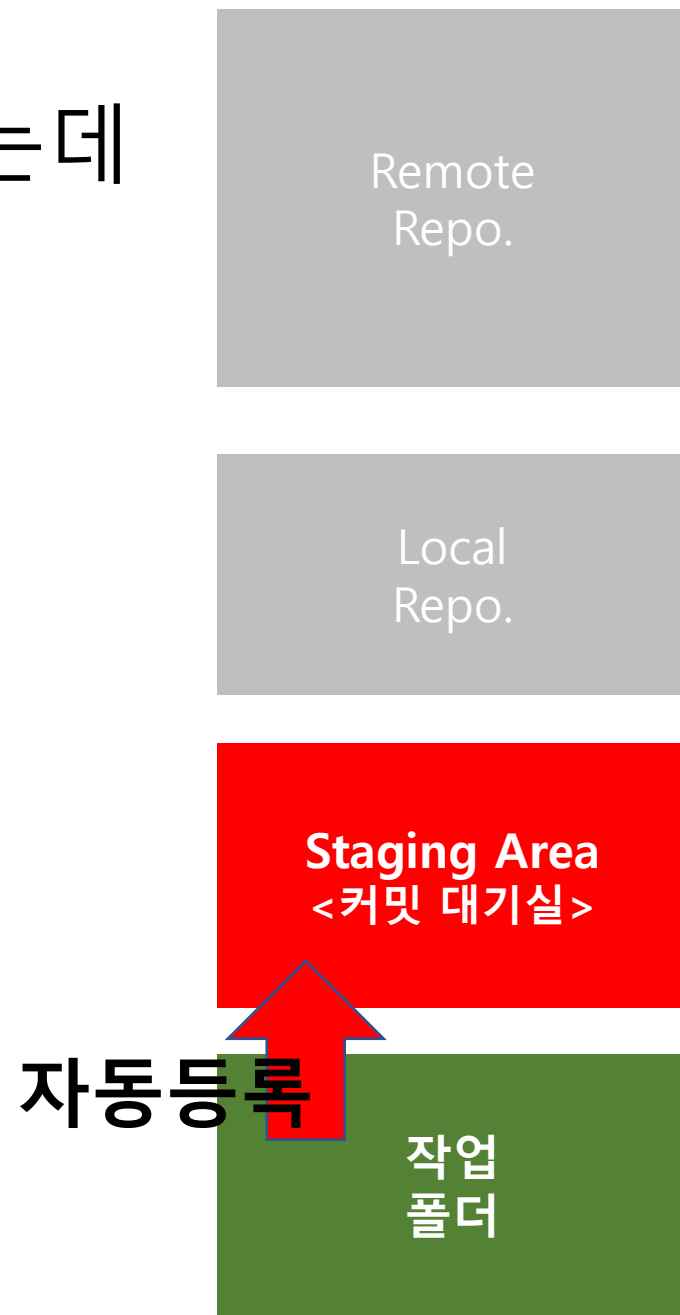
Staging Area
<커밋 대기실>

add

작업
폴더

GitHub Desktop 쓸 땐 이런 개념 없었는데

GitHub Desktop에서는
모든 파일들을 Staging Area로
자동 추가한다.



4 단계 Git 사용방법 살펴보기

기존에는 3 단계로 학습했지만, 이번에 4 단계로 학습해본다.

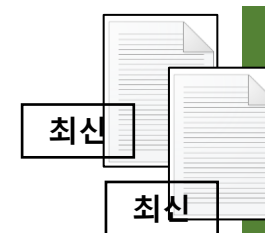
단계 1 : 초기 파일 생성하기

먼저, 작업 폴더에
파일을 두 개 생성한다.

Remote
Repo.

Local
Repo.

Staging Area
<커밋 대기실>

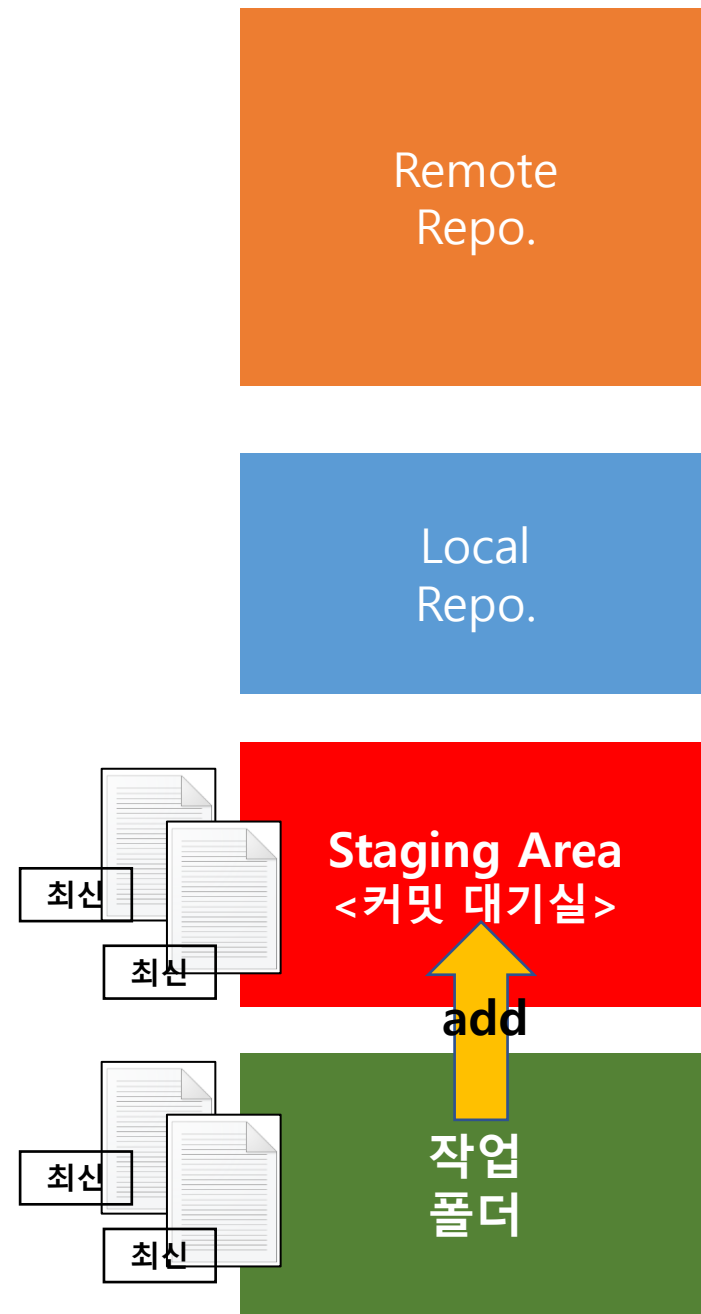


작업
폴더

단계 2 : Staging Area에 올리기

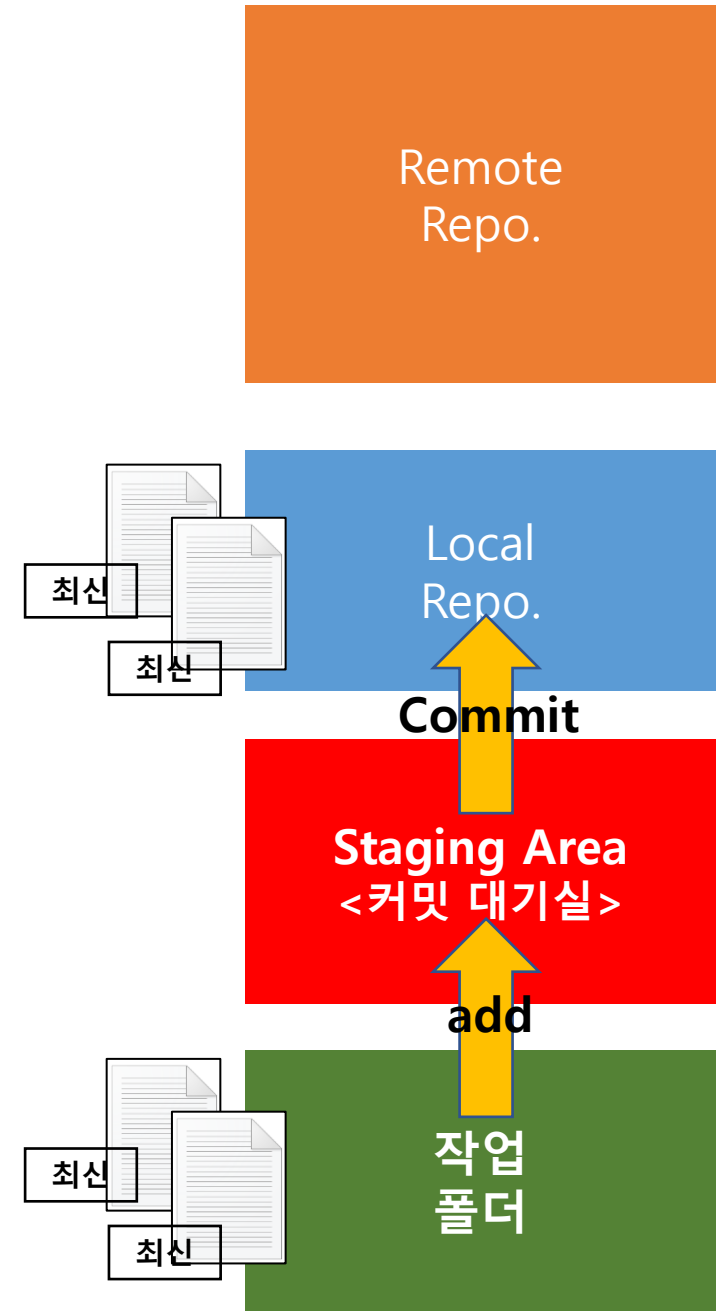
Staging Area 에 파일을 추가하는 것을 **add** 라고 한다.

이 단계는 Github Desktop 이 자동으로 수행한다.



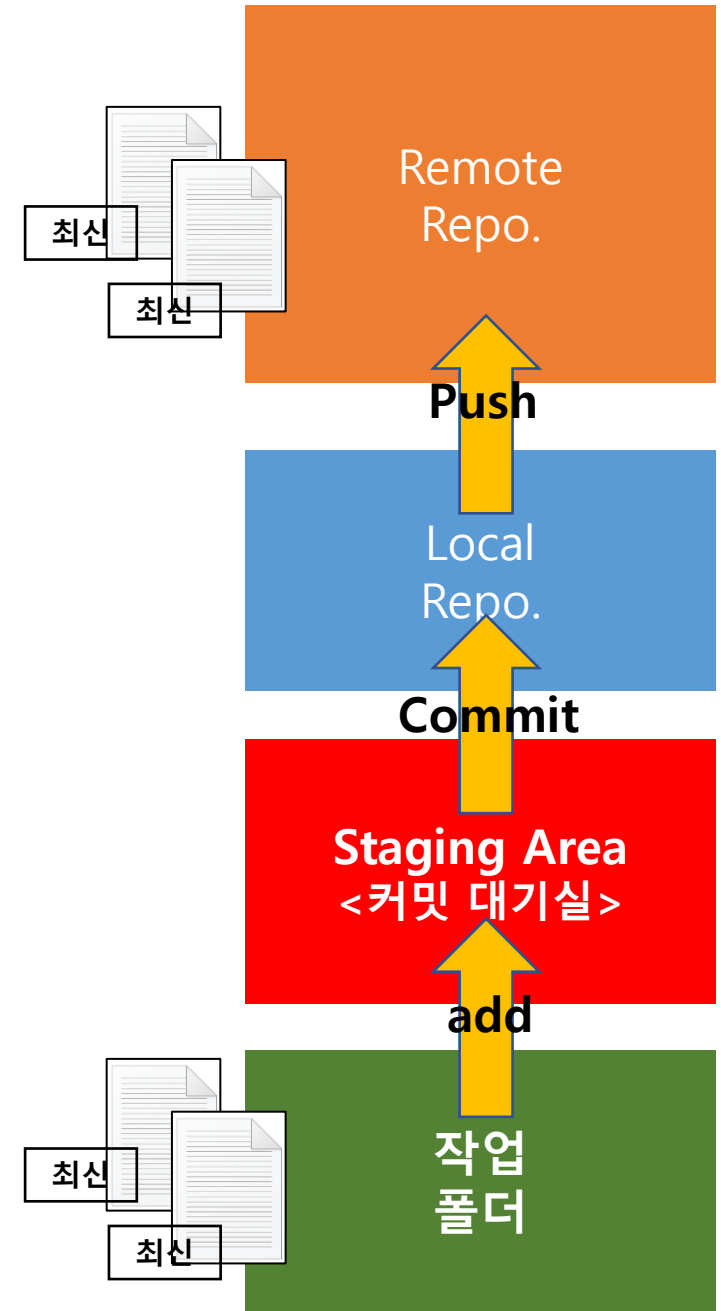
단계 3 : Commit 하기

기존에 우리가 하던
Commit



단계 4 : Push 하기

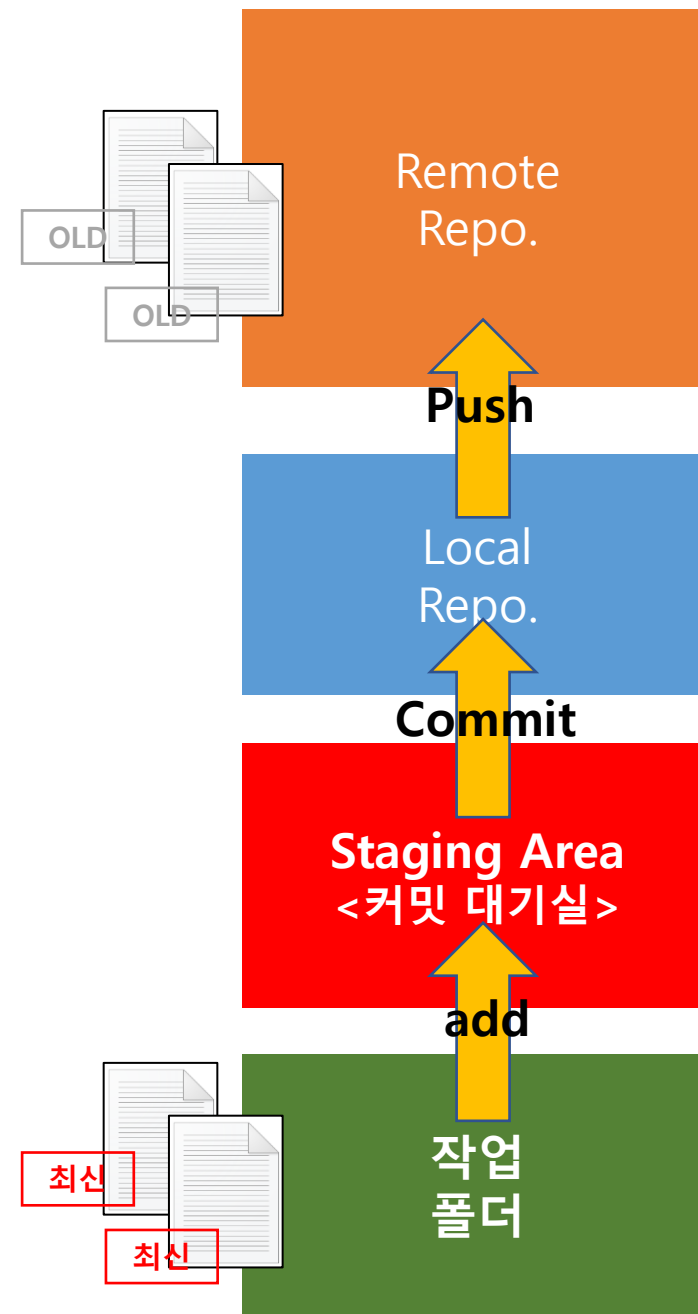
기존에 우리가 하던
Push



추가 동작 살펴보기

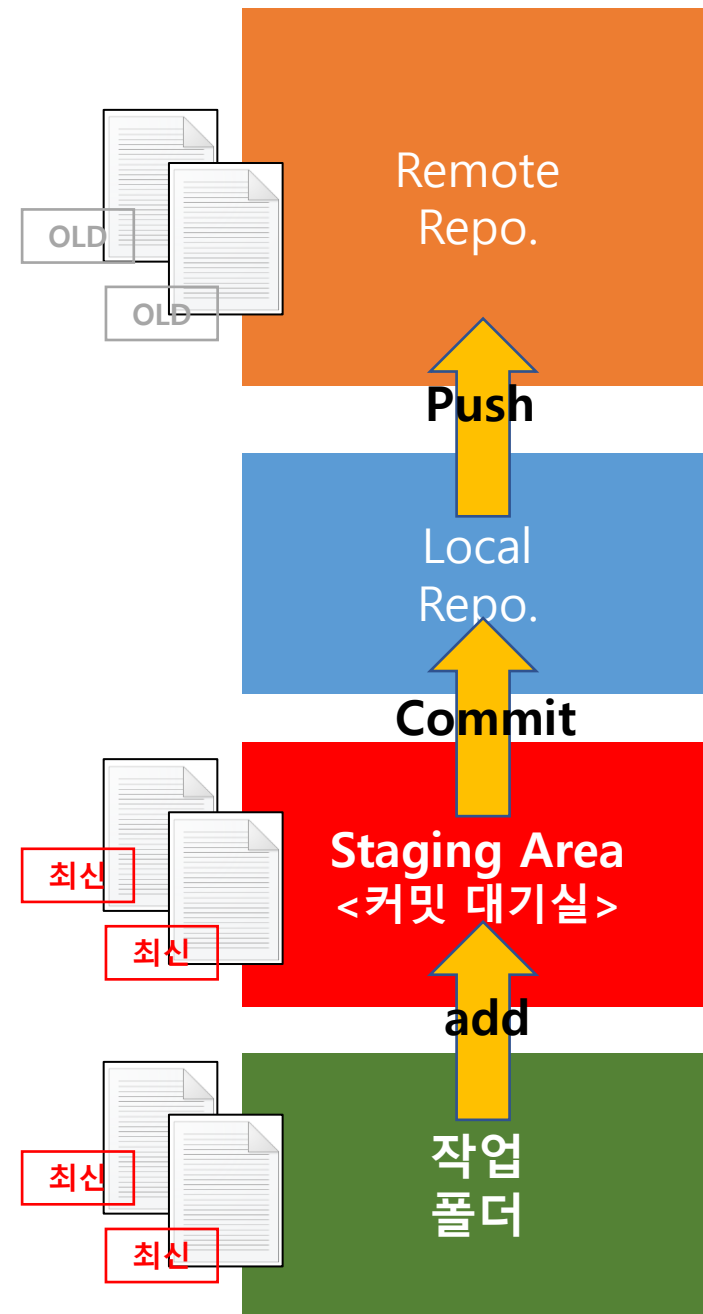
파일 수정

작업 폴더에 있는
파일을 수정함



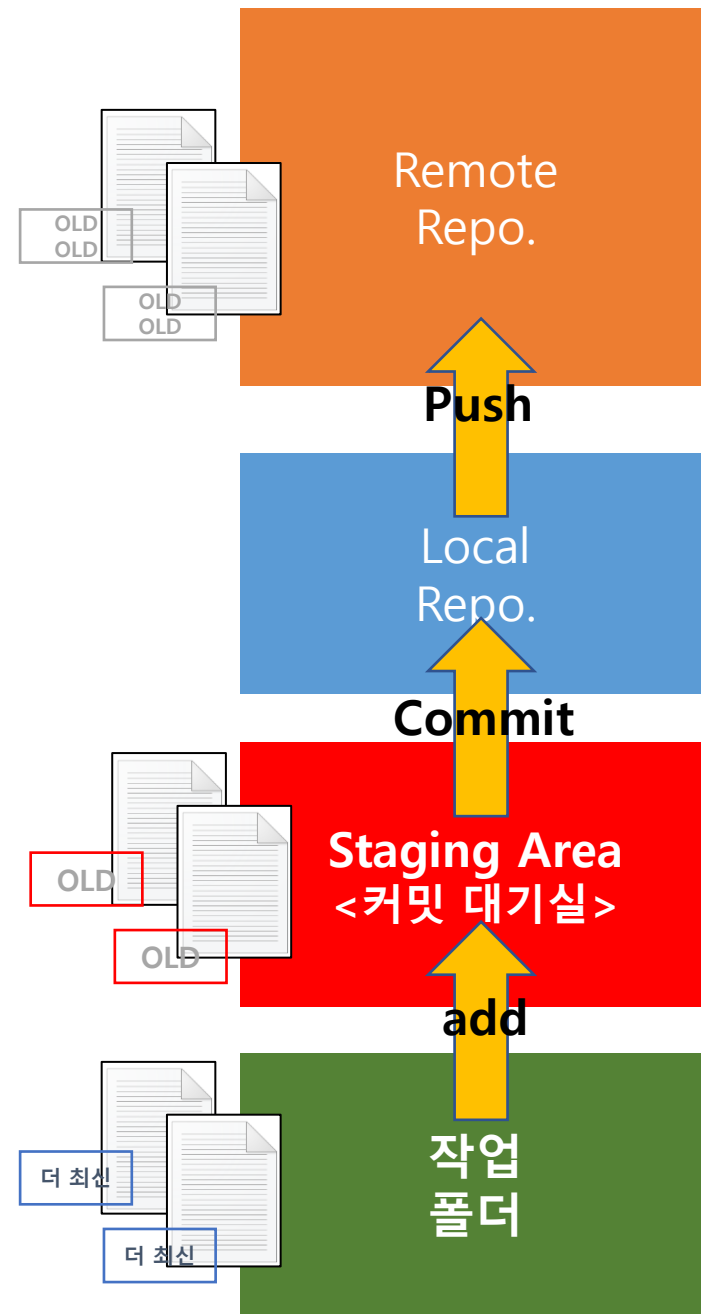
add 수행

Staging Area 에
최신 파일이 추가됨



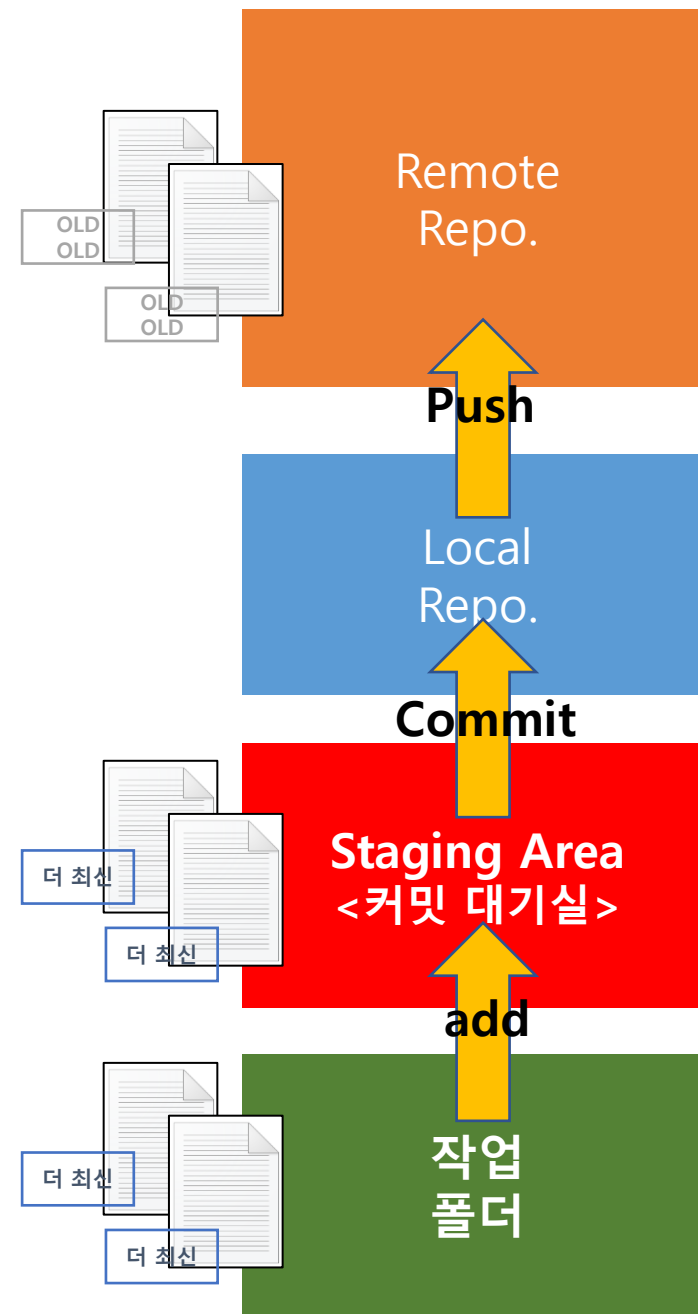
한 번 더 파일 수정

작업 폴더에 있는
파일을 수정함



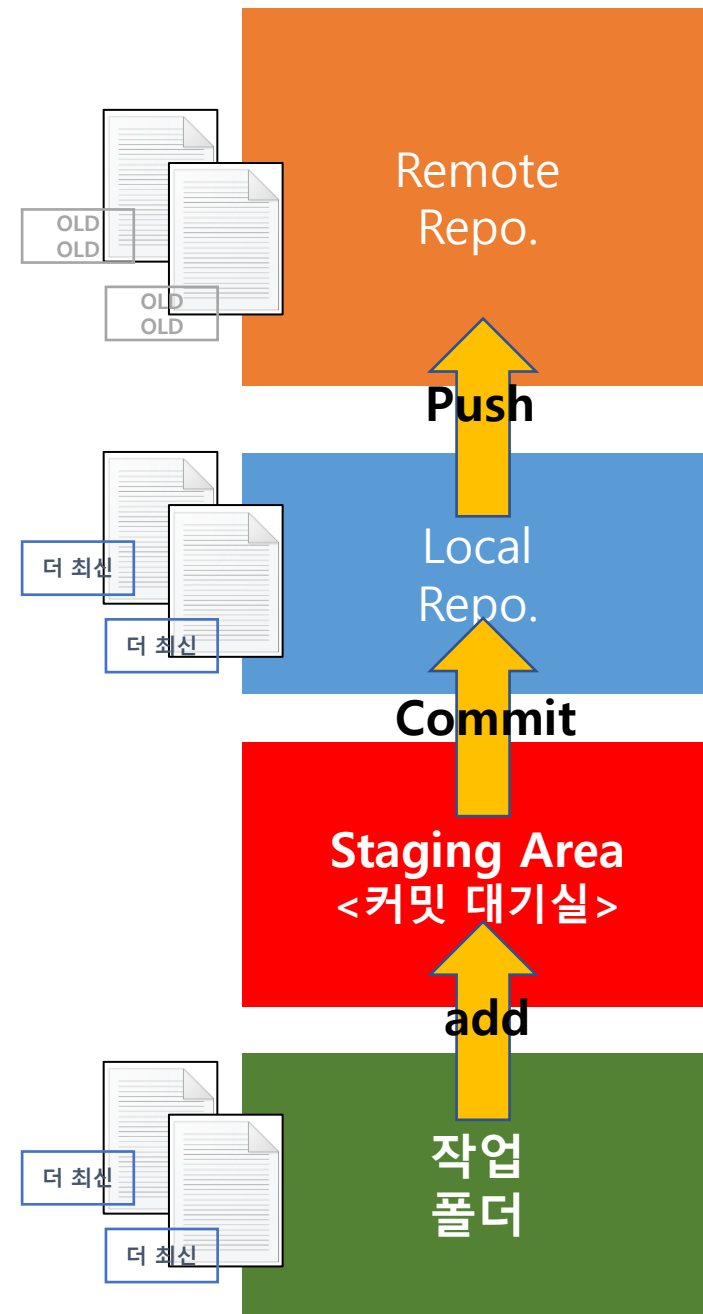
add 수행

Staging Area 에
가장 최신 파일이 추가됨



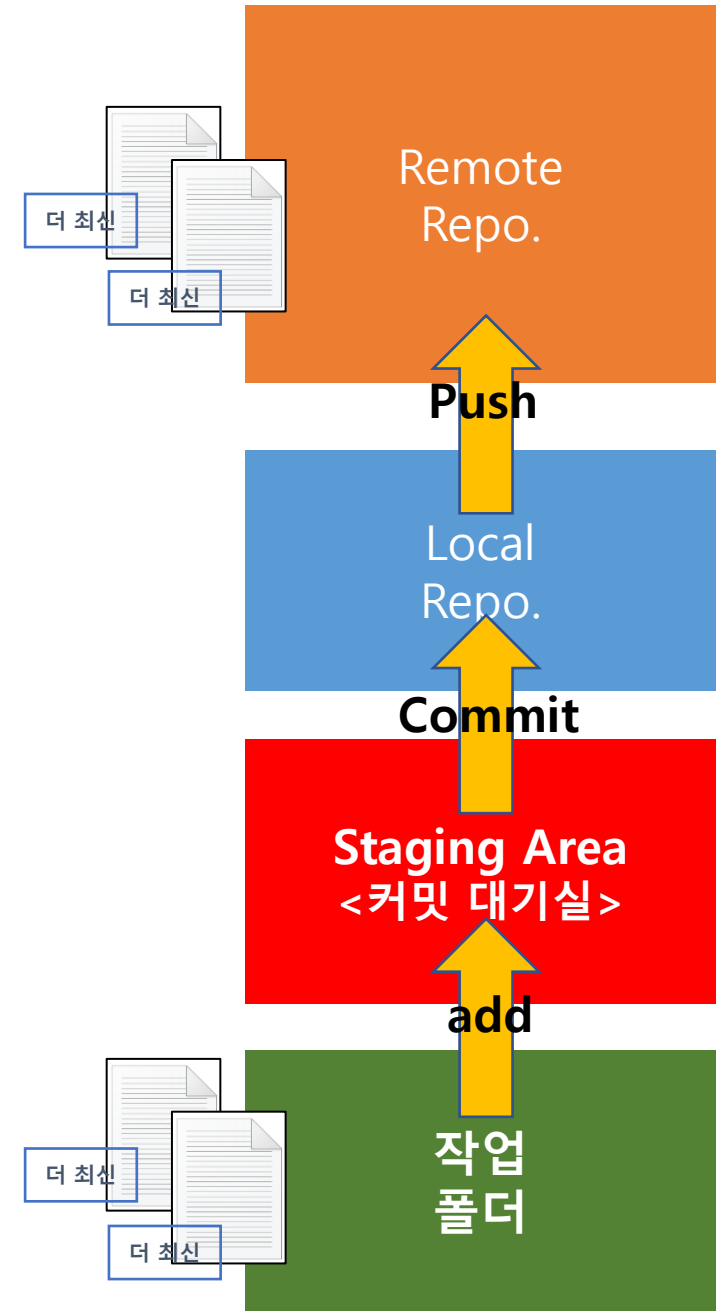
commit 수행

Local Repo. 에 내용 추가 됨



push

Remote Repo. 에 등록 됨



<15 분간 내용을 정리하자!>
곧, Quiz 예정

Git을 잘 이해했는지 확인하는 Quiz!

1번 문제

정답을 비공개 메시지로 보낼 것

이 **Commit 대기실**의 정확한 이름은 무엇인가?

Remote
Repo.

Local
Repo.

????????????????

작업
폴더

2번 문제

정답을 비공개 메시지로 보낼 것

신규 파일을 준비해서, Remote Repo. 까지 올릴 때 사용하는 **명령어**의 3개를 적으시오.

(힌트 : 마지막 정답은 Push)

[] → [] → [Push]

인증 Factor

본인 인증하는 방법

웹사이트에 로그인할 때,
본인 인증을 한다.

가장 흔히 쓰이는 방법은
ID 입력 후,
* **PASSWORD**

Password가 하나의 인증 수단이다.

→ **Single Factor 인증 (1개로 인증)**

Two Factor 인증

조금 더 본인 인증에 심사가 엄격하는 경우
Two Factor 인증을 해야,
서비스 이용이 가능해진다.

예시)

- ID 입력후
 1. Password
 2. 공인인증서 인증

Multi-Factor 인증

2개 이상 Factor로 인증하는 방식을 Multi-Factor라고 한다.

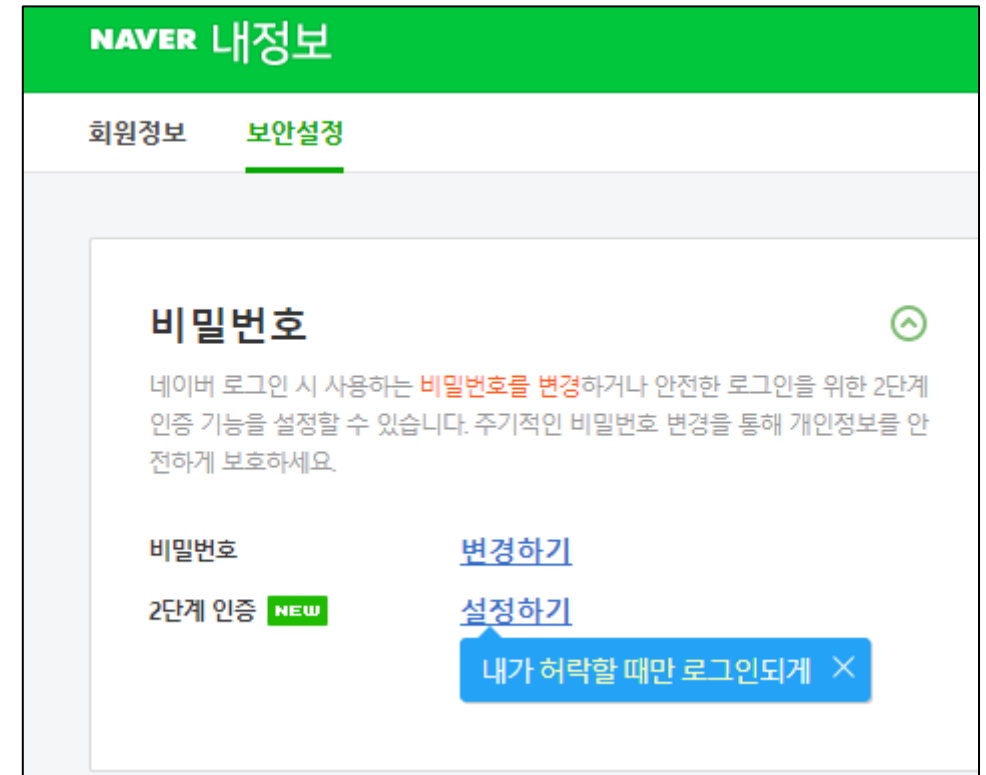
비밀요원이 된다면

1. ID 카드
2. 지문인식
3. 홍채인식

인증 방법은 여러가지가 있다.

네이버 에서 Two Factor 인증

1. Password
2. 핸드폰 인증



채팅창에, 인증 Factor가 어떤 것이 있는지 적어보자!

Password

지문인식

홍채인식

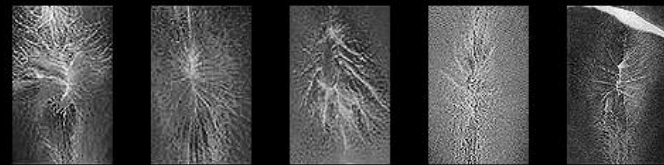
공인인증서 제외

■ 항문 인식 시스템(Anus Recognition System)



ARS를 장착한 ATM에 항문 인식으로 사용자 인증을 받는 장면

이젠 더 이상 카드 분실(도난), 비밀번호 유출 등의 걱정이 필요 없습니다. 저희 (주) 제노프릭스 프리키 아이디어 팀은 사람에 따라 항문의 주름 패턴이 모두 다르다는 원리를 이용하여 항문 인식 시스템(ARS; Anus Recognition System)을 개발하였습니다. 홍채인식, 지문인식 장치는 눈알 빼고, 손가락 잘라가서 사용하면 되므로 무용지물입니다. ARS는 괄약근 수축/이완시의 동적인 주름 패턴을 인식하기 때문에 만에 하나 범죄자가 "똥꼬"를 도려내 가더라도 이용할 수 없습니다.



괄약근 수축/이완을 몇차례 반복하는 동안 항문의 동적인 주름 패턴을 인식하여 본인의 정보와 함께 데이터 베이스에 저장됩니다.

ignore 처리의 필요성

만약 이러한 시스템이 있다면!?

인증 방법 : Password 가 아닌, Key File로 인증

Key File을 내 컴퓨터에 보유 해야만,
회원 로그인 이 가능하다.

→실제, AWS 라는 곳에서
이러한 인증 방법을 사용하고 있다.

만약, 실수로 내가 GitHub에 AWS 시스템의 KeyFile을 업로드 했다면?

1. Visual Studio 버그로
내 GitHub Remote Repo. 가 Public으로 바뀌어
나의 Key File이 온 세상에 공개됨
2. Key File을 가진 사람은
AWS 시스템으로 로그인 가능함
3. **비트코인 채굴기로 사용되어,**
시스템 사용료 \$6,500 사용료 나옴



➔ 비트코인 채굴기가 된다.

GitHub에 업로드 안하고 싶은 파일들 처리

ignore 기능을 사용한다!

로그인 인증에 쓰이는 KeyFile 이나,
개인정보 등은 ignore 처리를 해서

**GitHub에 Push가 안되도록
미리 설정을 할. 수. 있. 다**

keyfile만 ignore 처리하면 되냐!?

keyfile만 ignore 해야하는 것이 아니다.

굳이 보관할 (업로드)할 필요가 없는 파일들이 있다면 이것도 ignore 해도 된다!

자유롭게 ignore 할 것들을 대상을 정하면 된다.

git에서 업로드 할 필요 없는 내용
ignore 하기

실습할 Repository 준비

이미 생성된
Repoistory 로 준비

The screenshot displays the GitHub web interface for a pull request. The top navigation bar includes links for File, Edit, View, Repository, Branch, and Help. The current repository is 'publicStudy119' and the current branch is 'main'. The pull origin is 'Last fetched just now'.

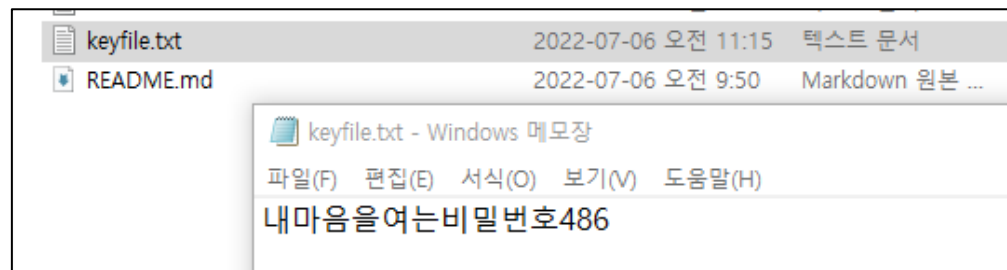
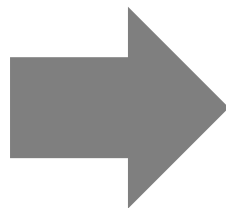
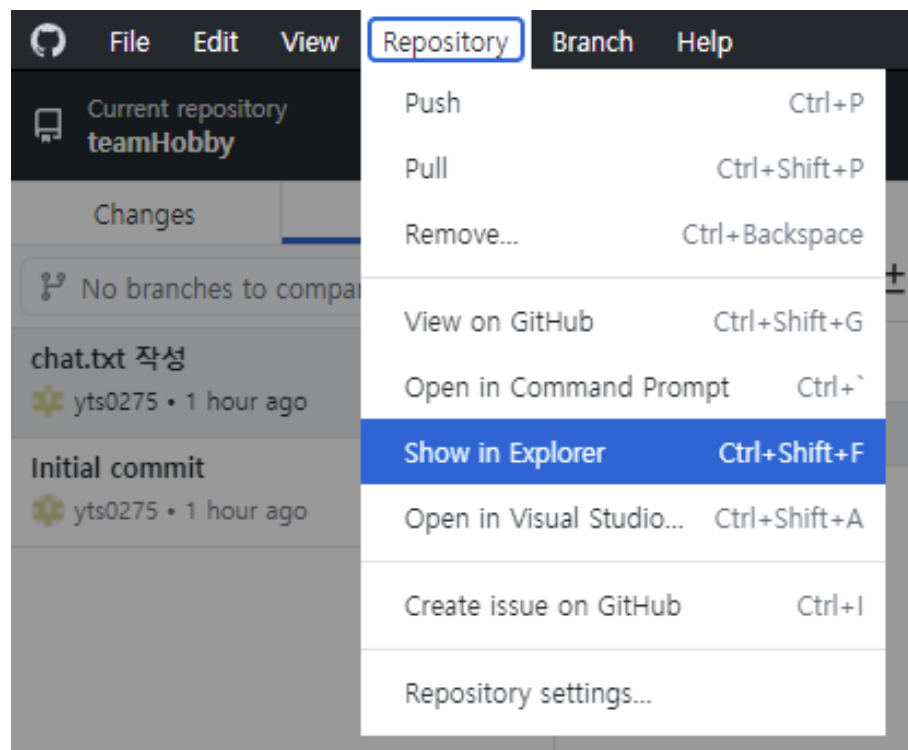
The sidebar shows the 'History' tab selected, indicating 'No branches to compare'. The pull request summary shows it was created by 'inho.choi' 1 day ago. The commit message is 'README 문서를 하나 적어보았습니다.' (I wrote one README document.) by 'mincoding' 1 day ago. The initial commit was also by 'mincoding' 1 day ago.

The pull request details show the title 'PullMePullMePullMeUp.txt' and a description: 'Pull은 최신 변경내용을 가져옵니다. 예전 소스코드를, 최신 소스코드로 업데이트 됩니다.' (Pull gets the latest changes. The old source code, update to the latest source code.)

The diff view shows changes to the file 'PullMePullMePullMeUp.txt'. The changes are as follows:

Line	Original	Changed
1	@@ -1,2 +1,5 @@	@@ -1,2 +1,5 @@
1	나를 가져가요	나를 가져가요
2	-폴미 폴미 폴미 업	-폴미 폴미 폴미 업
2	+폴미 폴미 폴미 업	+폴미 폴미 폴미 업
3	+폴미 폴미 폴미 업	+폴미 폴미 폴미 업
4	+폴미 폴미 폴미 업	+폴미 폴미 폴미 업
5	+폴미 폴미 폴미 업	+폴미 폴미 폴미 업

keyfile.txt 파일을 생성한다.



이제, ignore 할 차례

key file을 local Repository 에도 업로드가 안되도록 해야한다.

왜냐!?

**local Repository 내용들을 push 하면,
그대로 remote Repository 에 저장되기 때문이다.**

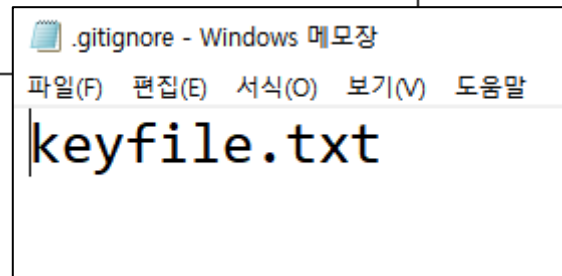
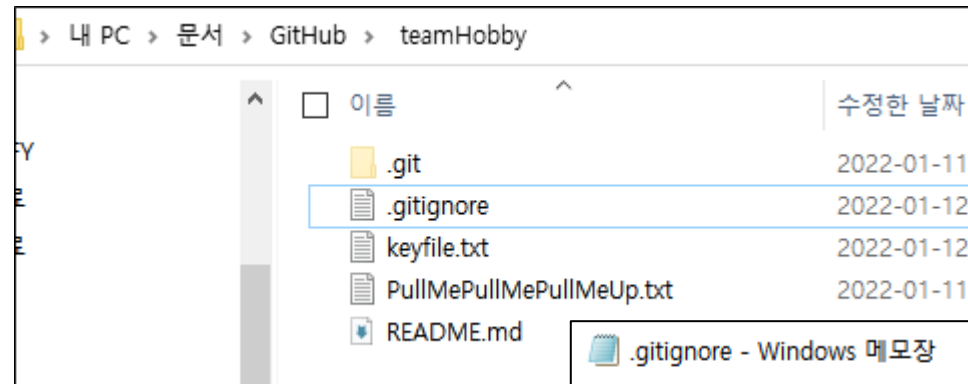
Commit 안함!!

.gitignore 파일에 대해서!

<실습하지 말것!!>

“.gitignore” 라는 파일을
Local Repository에 해당되는 **작업폴더** 안에
만들어두고,

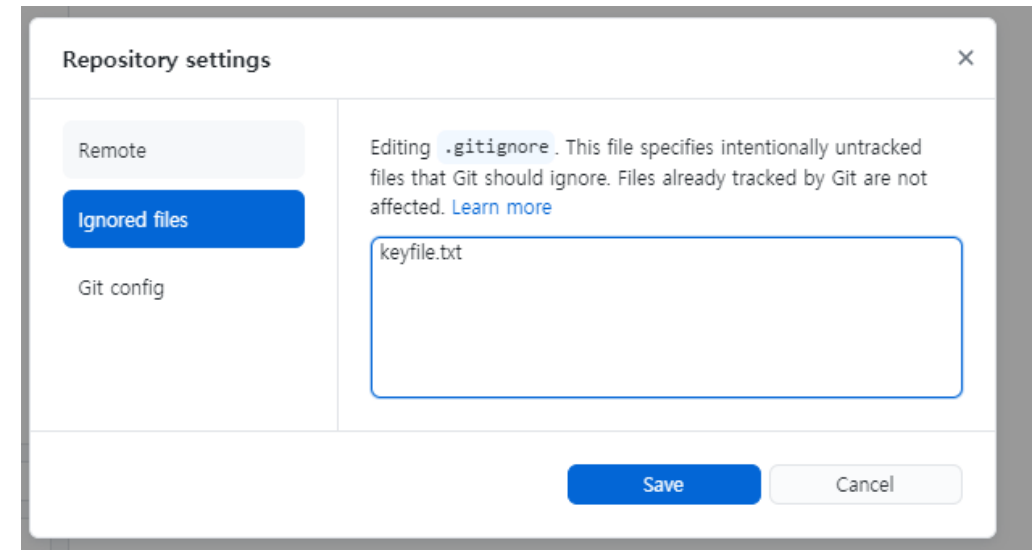
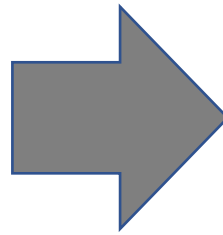
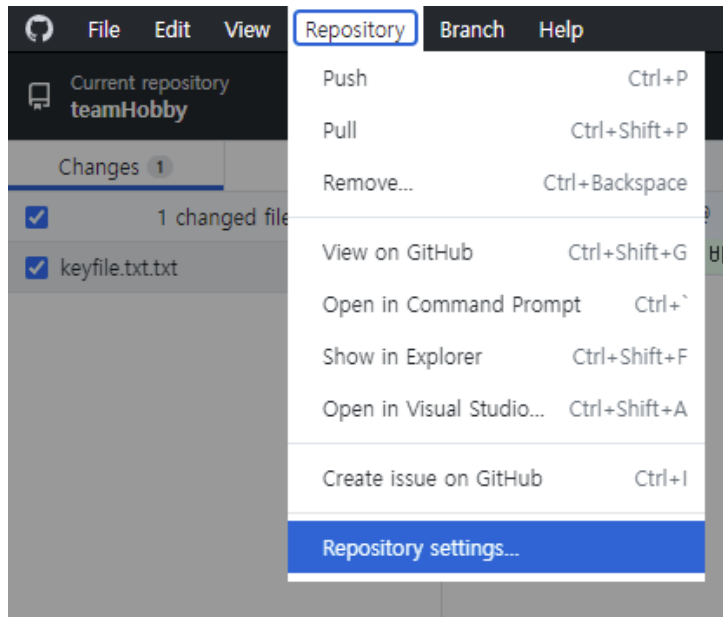
이곳에, 제외시킬 파일 이름들을 적으면 된다.



우리는 Github Desktop 기능을 쓸 것이다.!

ignore 하러 가기

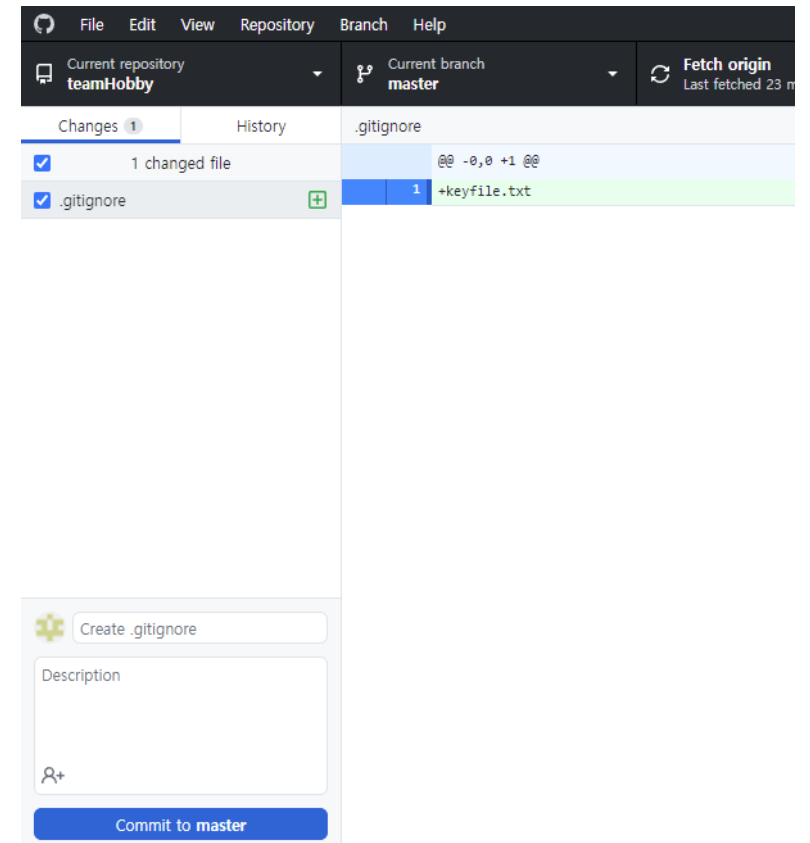
Save 하면,
.gitignore 파일이 자동 생성된다!



ignore 처리 완료된 화면

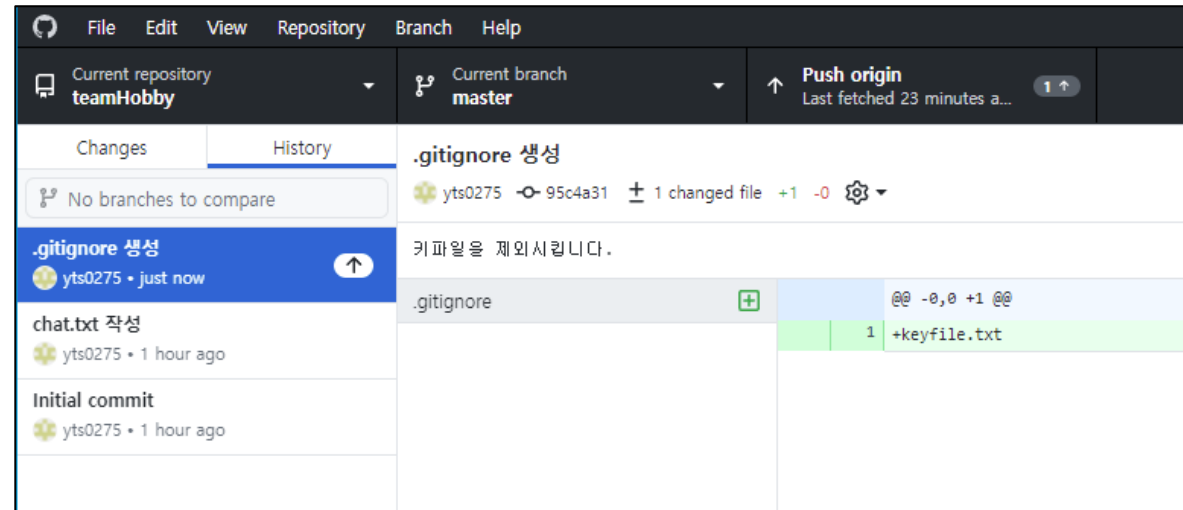
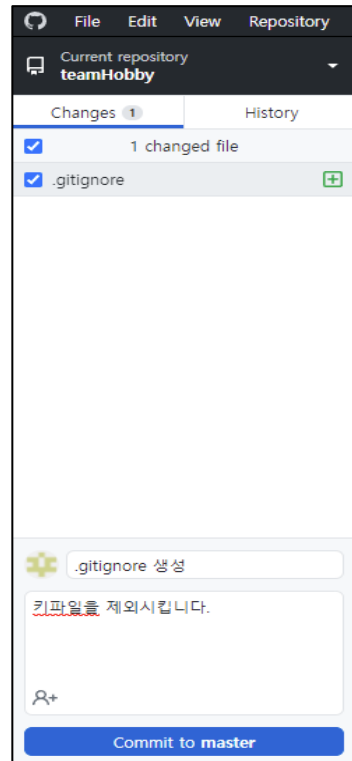
자동으로 .gitignore파일이 생성되었기에
Changes 목록에 하나가 추가되었다.

이 파일은 Commit / Push 해도 된다.



.gitignore 파일 push 하기

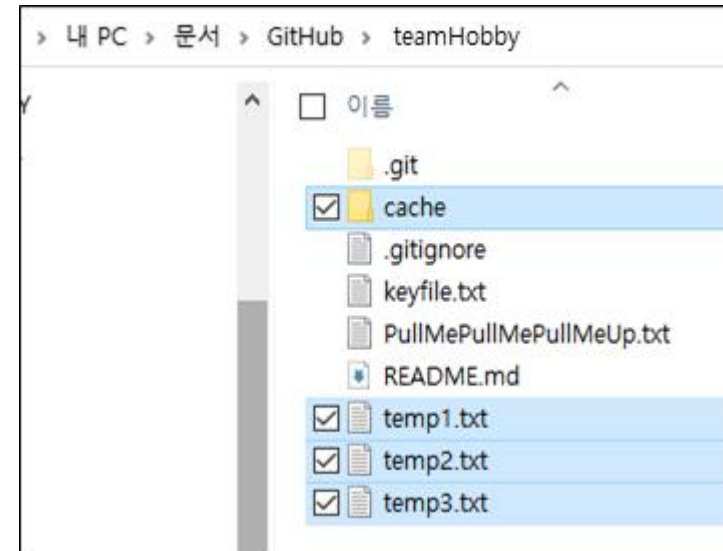
.gitignore 파일은 github에 push 해도 된다. <안전하다>
파일 내용이 노출되는 것이 아니기 때문이다.



ignore 추가 연습을 위한, 실습 준비

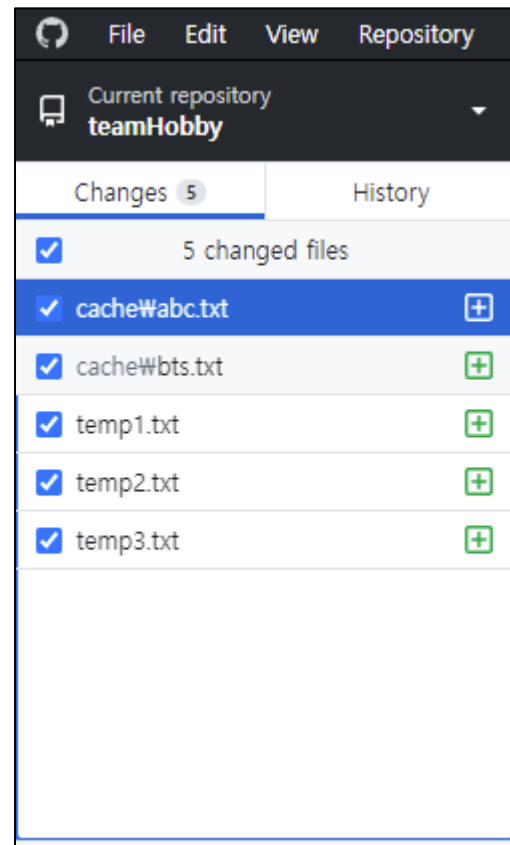
다음과 같이 작업폴더에 파일, 폴더를 준비한다.

1. temp1.txt 파일 생성
2. temp2.txt 파일 생성
3. temp3.txt 파일 생성
4. **cache** 폴더 하나 만들기
5. cache/abc.txt 파일 하나 생성
6. cache/bts.txt 파일 하나 생성



Changes 확인하기

정상적으로 추적이 잘 된다.

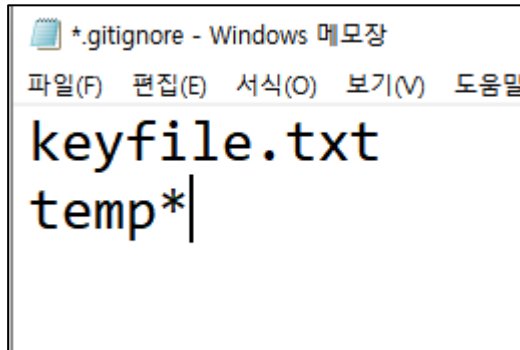


한꺼번에 ignore하기

.gitignore 파일에 다음과 같이 적자.

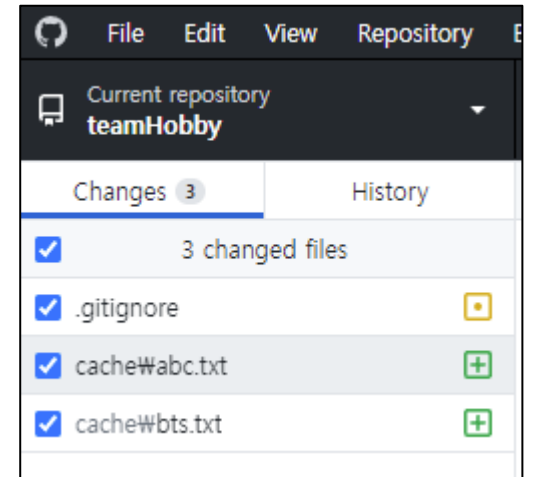
temp*

temp로 시작하는 모든 파일들을 ignore한다.



```
*.gitignore - Windows 메모장
파일(F)  편집(E)  서식(O)  보기(V)  도움말
keyfile.txt
temp*|
```

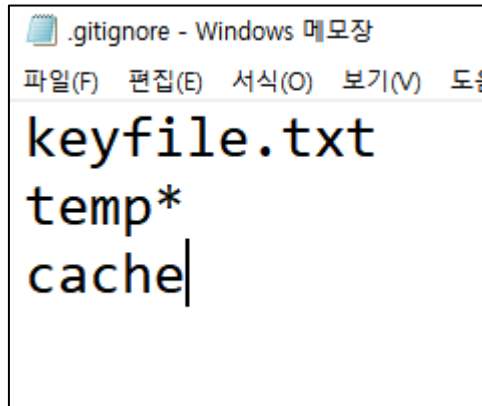
결과확인



temp로 시작되는 모든 파일들이 추적에서 사라졌다.

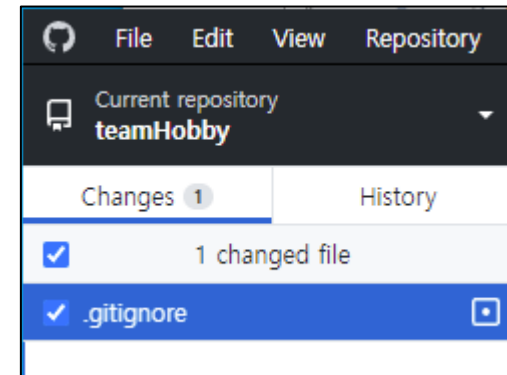
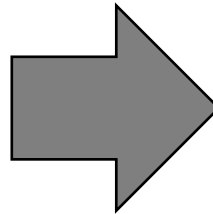
폴더도 ignore 하기

폴더 이름을 적으면 된다.



A screenshot of a Windows Notepad window titled ".gitignore - Windows 메모장". The menu bar includes "파일(F)", "편집(E)", "서식(O)", "보기(V)", and "도움". The text content of the file is:

```
keyfile.txt  
temp*  
cache|
```



깔끔해졌다.

[참고] 디렉토리 vs 폴더

옛날에는 폴더를 디렉토리라고 불렀음

- Windows 대신, DOS 라는 운영체를 사용하던 시절이다.
- 현재 리눅스 라는 운영체제는
폴더라고 부르지 않고, 디렉토리가 부른다.

즉, 디렉토리 = 폴더