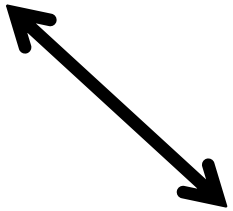
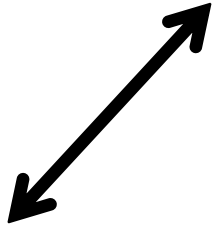
깃 허브(Git Hub)

소스트리(SourceTree)





깃 (Git)

여기서 잠깐!

이 자료에선

깃베쉬(Git-Bash) 나 깃허브 데스크탑(GitHub Desktop)이 아닌

소스트리(Sourcetree)를 사용해서 깃(Git)과 깃허브(GitHub)를 연동합니다.

목 차

Ⅰ. 깃 허브(GitHub) 아이디 생성

1. 깃 허브에서 회원가입을 진행한다. < https://github.com/ >

Sing up 버튼을 클릭해서 회원가입을 진행하면 된다.

만약 아이디가 있는 사람들은 여기 과정을 건너 뛰어도 된다.

* 아이디 이메일은 자신이 이메일을 받을 수 있는 이메일로 설정해야 한다.
(임의로 이메일을 만들어서 사용 X)

일반적인 아이디 회원가입과 같기 때문에 이하 생략.

닉네임은 깃허브 내에서 사용하는 닉네임이기때문에 너무 신중히 고를 필요 없다.

본인 이름으로 만든 후 수정가능.

Ⅱ. 소스트리(Sourcetree) 설치

1. 소스트리 홈페이지에 접속한다. < https://www.sourcetreeapp.com/ >

2. 메인 화면에 보이는 Download For Windows 를 클릭한다. (본인 운영체제가 윈도우일경우)

   본인이 사용하고 있는 운영체제를 클릭하면 된다.

3. 소스트리 설치가 끝나면 설정 창이 자동으로 뜬다.

   1) 우리는 BitBucket은 사용하지 않으므로 하단에 건너뛰기를 눌러준다.

   2) Git만 사용할것이기 때문에 Git만 체크해 준 상태에서 넘어간다.

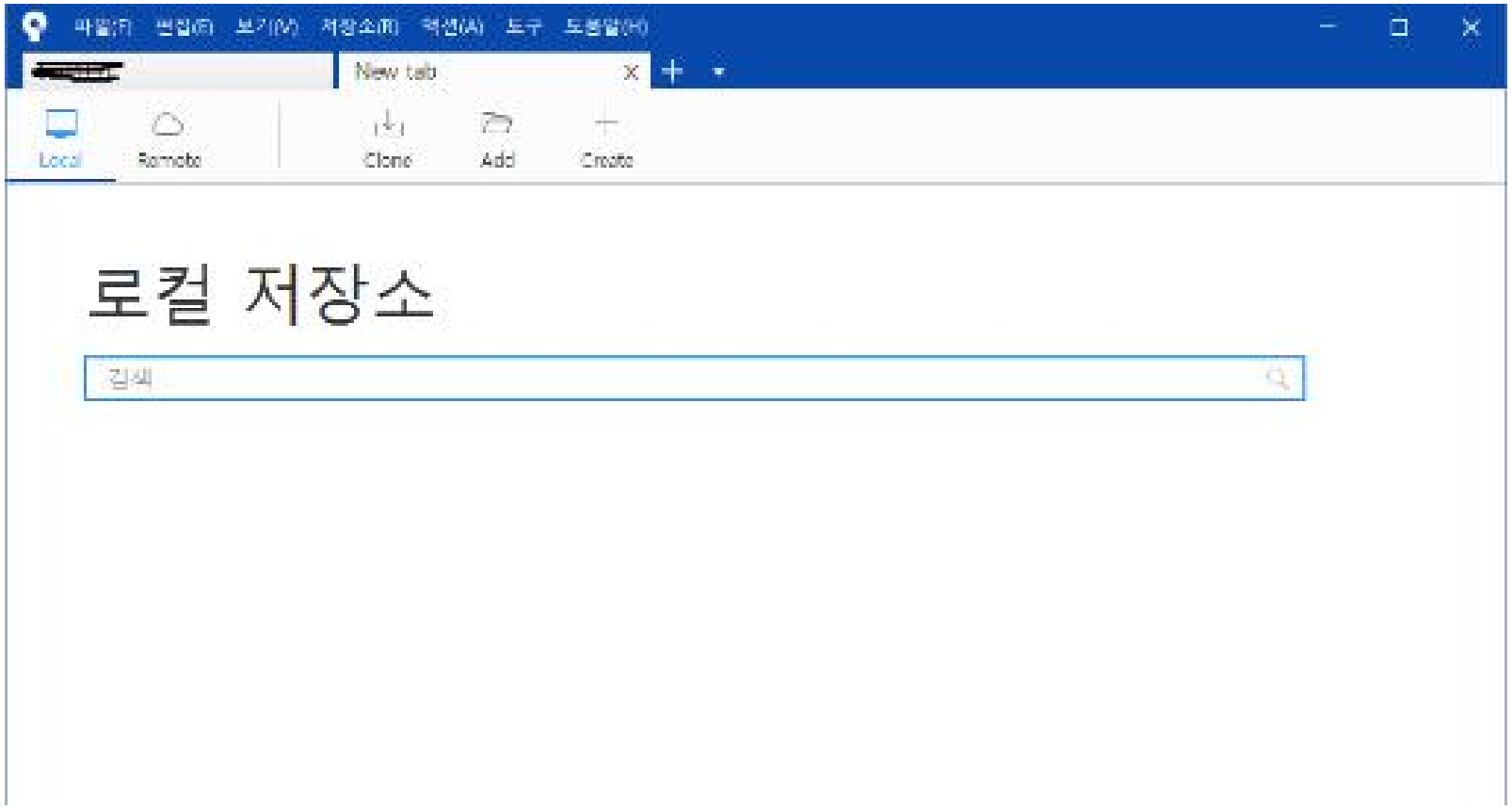
   3) 첫 번째 빈 칸에 본인의 깃 허브 닉네임을 적어준다.

   4) 두 번째 빈 칸에 본인의 깃 허브 이메일을 적어준다.

   5) SSH key는 가지고 있지 않으니 아니요 누르고 넘어간다.

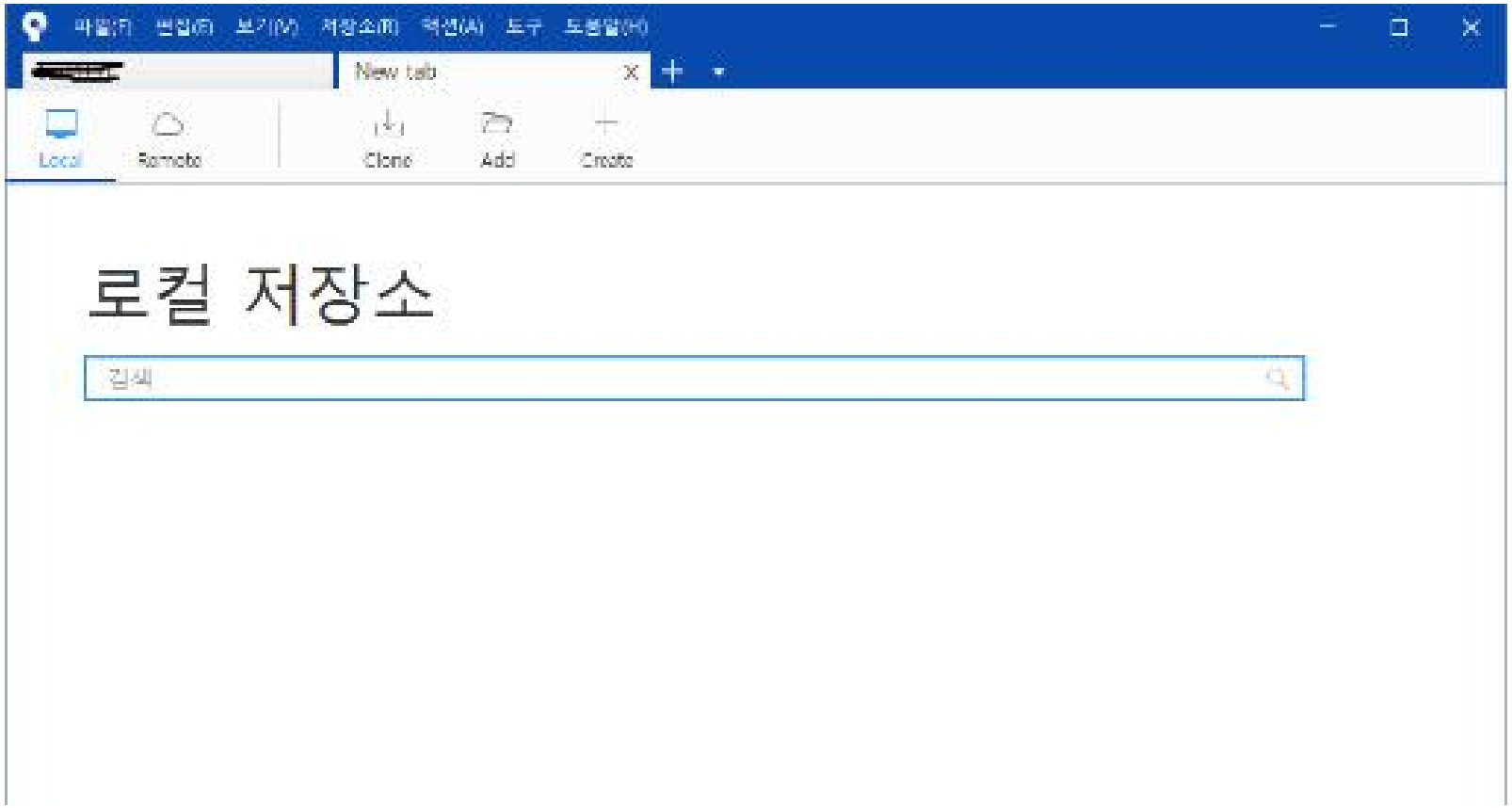
Ⅱ. 소스트리(Sourcetree) 설치

다음과 창이 나왔다면 설치가 완료된거다.



Ⅱ. 소스트리(Sourcetree) 설치

여기서 궁금한 점이 이걸 어떻게 사용하는 것이라 생각한다.



그 전에 중요한 것이 용어 설명 이다.

용어의 뜻을 이해하지 못하고 넘어가면

어떻게 깃허브를 사용해야하는지

어떻게 파일을 올리는 지 알수 없다.

Ⅲ. 용어 설명

용어 설명을 시작하기에 앞서

용어에 대한 자세한 설명을 원한다면 구글 등 에서 설명을 찾아보는 것을 추천한다.

이 자료에서의 용어 설명은 정확한 개념 설명보단

프로젝트를 시작하기 전에 간단히 그 용어의 기능이 무엇인지 정도만

쉬운 예시를 통해 설명되어 있으므로

정확한 자료라 볼 수 없다.

Ⅲ. 용어 설명

1) Repository (저장소)

레파지토리라고 불리며, 개발자들 사이에선 짧게 repo(레포) 로 불리기도 한다.

말 그대로 저장소 이다.

내가 작업한 파일 및 자료들을 저장해 둘 수 있는 공간이다.

저장소의 종류로는 원격저장소와 로컬저장소가 있다

원격저장소 : 깃 허브의 저장소를 뜻한다.

로컬저장소 : 내가 작업한 파일들을 모아두는 내PC의 폴더를 의미한다.

Ⅲ. 용어 설명

2) Commit (커밋)

가장 기본중에 기본이다.

파일 및 폴더의 추가&변경사항을 저장소(Repository)에 저장하는 것이다.

쉽게 풀어서 설명하자면 게임에서 Save하기까지의 단계라고 할 수 있다.

사용 방법은 뒷쪽 파일 업로드 부분에서 설명하겠다.

Ⅲ. 용어 설명

3) Push (푸쉬/ 올리기)

푸쉬를 하게 되면 로컬저장소에 있는 내가 작업한 파일들을 원격저장소로 보낼 수 있다.

즉, 내가 작업한 파일들을 깃 허브 저장소로 보내기위한 작업이다.

4) Pull (풀/ 가져와 병합하기)

푸쉬와는 반대로 원격저장소에 있는 파일들을 내 로컬저장소로 내려받는 작업이다.

다른 사람들과 협업을 진행한다면 다른 사람들의 파일 까지 내려받을 수 있다.

5) Fatch (패치/ 가져오기)

패치기능은 쉽게 말해 새로고침이라 생각하면 된다.

깃 허브의 레파지토리에 변경사항이 생겼을 경우 패치를 통해 변경내용을 받아올 수 있다.

Ⅲ. 용어 설명

* 패치(fatch)와 풀(pull)의 차이점

패치(fatch)와 풀(pull)은 겉으로 봤을땐 같은 기능을 수행하는거 같지만 그렇지 않다.

풀은 새로운 변경사항들을 내 로컬 저장소에 병합하고 싶을 때 사용하고,

패치는 내 로컬저장소 안에 원격저장소에 있는 파일들을 병합하지 않고

변경 내역만 확인할 때 사용할 수 있다.

Ⅲ. 용어 설명

6) Branch (브랜치)

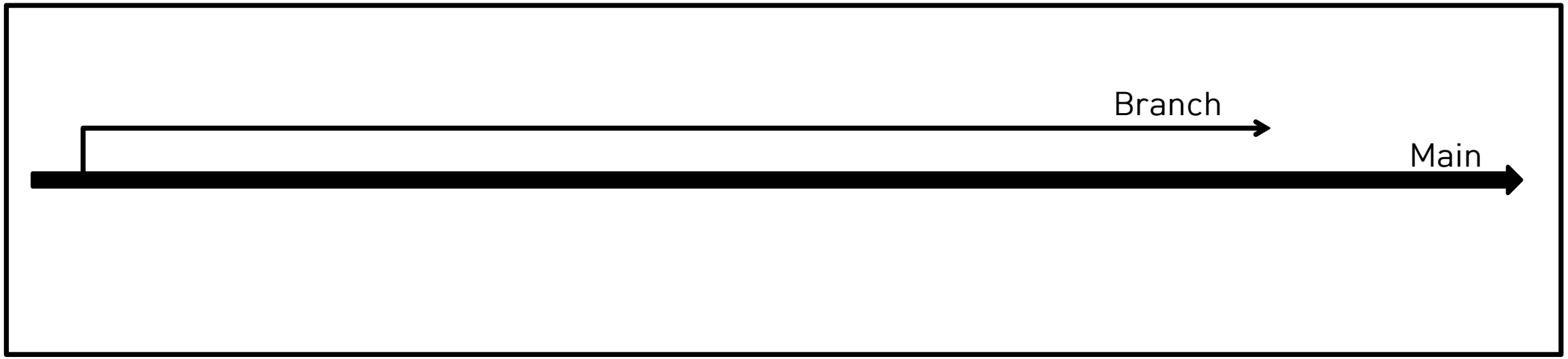
아마 앞선 용어 중에서 가장 이해하기 힘든 부분이라고 생각된다.

Branch의 원래 뜻은 나뭇가지로

여기서 Branch는 독립적인 작업공간을 의미한다고 생각하면 쉽다.

Main이 되는 프로젝트는 건드리지 않는 상태에서

Branch라는 공간을 만들고 그 안에서 내가 작업하는 파일들을 관리할 수 있다.



Ⅲ. 용어 설명

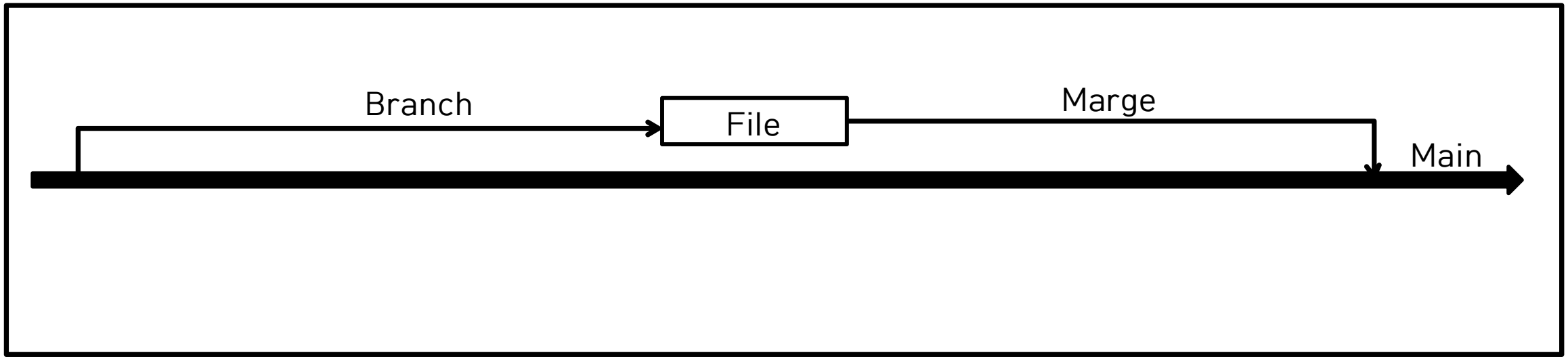
7) Marge (마지/ 병합)

앞선 Branch와 연결되는 기능으로

Branch만 진행했을 경우 Main에는 영향을 주지 않기 때문에 파일이 업로드 된 상태라 볼 수 없다.

따라서, Marge를 통해 내가 Branch에서 작업한 파일들을 Main으로 보내주는 역할을 한다.

주의할 점, 팀 프로젝트 진행 시 아무나 Marge를 해버리면 프로젝트가 망가질 수 있으니 한사람만 하는것이 좋다.



Ⅲ. 용어 설명

8) Stash (스테쉬)

만약 내가 작업하고 있는 도중 다른 요청이 들어와 급하게 내 작업을 내려야 할 때

미완성인 채로 커밋을 진행할 수 없을 때 사용하는 기능이다.

하지만 협업업무중 급히 처리해야 할 일이 있을 경우를 제외하고는 사용할 일이 거의 없다.

따라서, 소규모 프로젝트만 진행하는 우리는 거의 사용하지 않을 기능이다.

9) Conflict (충돌)

여러명이서 작업을 진행할 때 pull받을 때 나는 경우가 많다.

말 그대로 충돌이 일어나 작업을 진행할 수 없는 상태이며 각각의 충돌 이유는 구글링을 추천한다.

Ⅲ. 용어 설명

10) Clone (클론)

클론은 말 그대로 복제품이라 생각하면 된다.

혼자서 개발을 진행하는 경우 깃 저장소에 바로 올리면 되지만,

여러명과 개발을 진행하는 경우 클론을 만들어 진행한다.

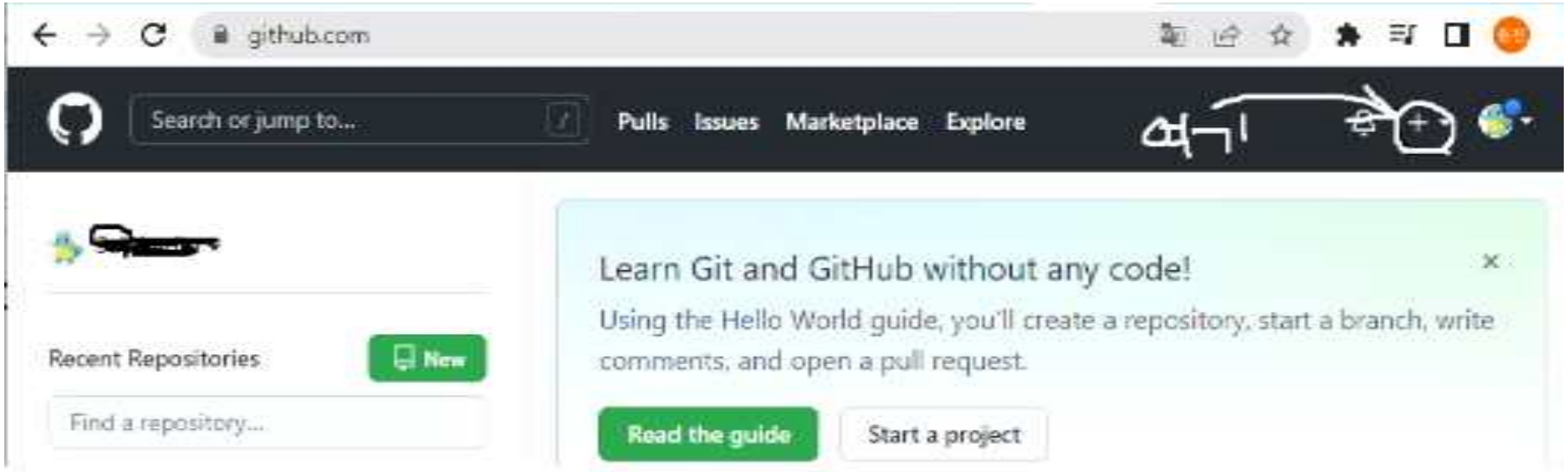
우리가 사용할 소스트리에서도 클론을 만들어 깃 허브 저장소의 주소를 등록해주고

클론 안에서 파일을 관리하며 작업을 진행할 것이다.

Ⅳ. 레파지토리(Repository) 생성

* 깃허브에서 자신의 레파지토리를 생성해보자

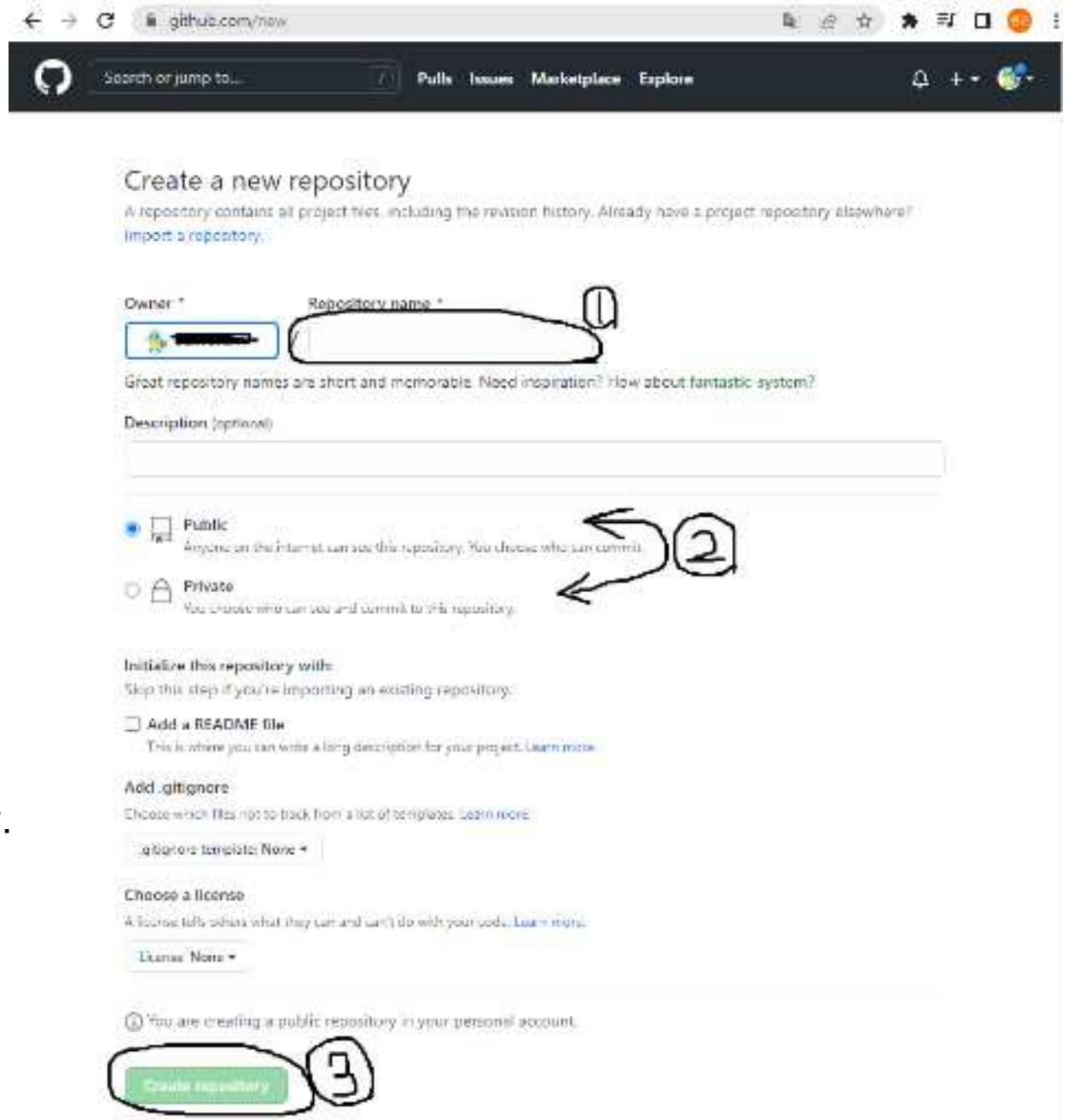
가장먼저 깃허브를 로그인 하면 아래와 같은 창이 나타날것이다.



'여기' 라고 쓰여있는 곳에 + 표시를 누르면 안에 New Repository를 클릭하면 된다.

Ⅳ. 레파지토리(Repository) 생성

New Repository 를 클릭하면 오른쪽과 같은 창이뜨는데



1. 저장소의 이름을 정하는 칸이다.

2. 저장소의 공개범위를 지정하는 칸이다.

   Public : 모두가 볼 수 있다.
   Private : 내가 지정한 사람 외엔 볼 수 없다.

3. 위 설정을 다 마친 뒤에 저장소를 만들어주는 버튼이다.

Ⅴ. 파일업로드

소스트리를 사용해서 파일을 내 레파지토리로 올리고 싶을 때 준비해야 할 것이 있다.

1. 내가 파일을 올리고 싶은 레파지토리의 주소를 가져와야 한다.

    주소는 내가 만든 레파지토리로 들어가면

    < https://github.com/아이디/레파지토리이름.git >

    위 형식의 주소가 있을 것이다. 이것을 레파지토리 주소라 부르겠다.

2. 토큰을 준비해야한다.

    깃허브와 소스트리를 연동해서 사용하려면 소스트리 안에서 깃허브 계정을 연결 시켜주어야한다.

    그때 사용하는 것이 토큰이다.

두개 다 메모장 같은곳에 저장해두자.

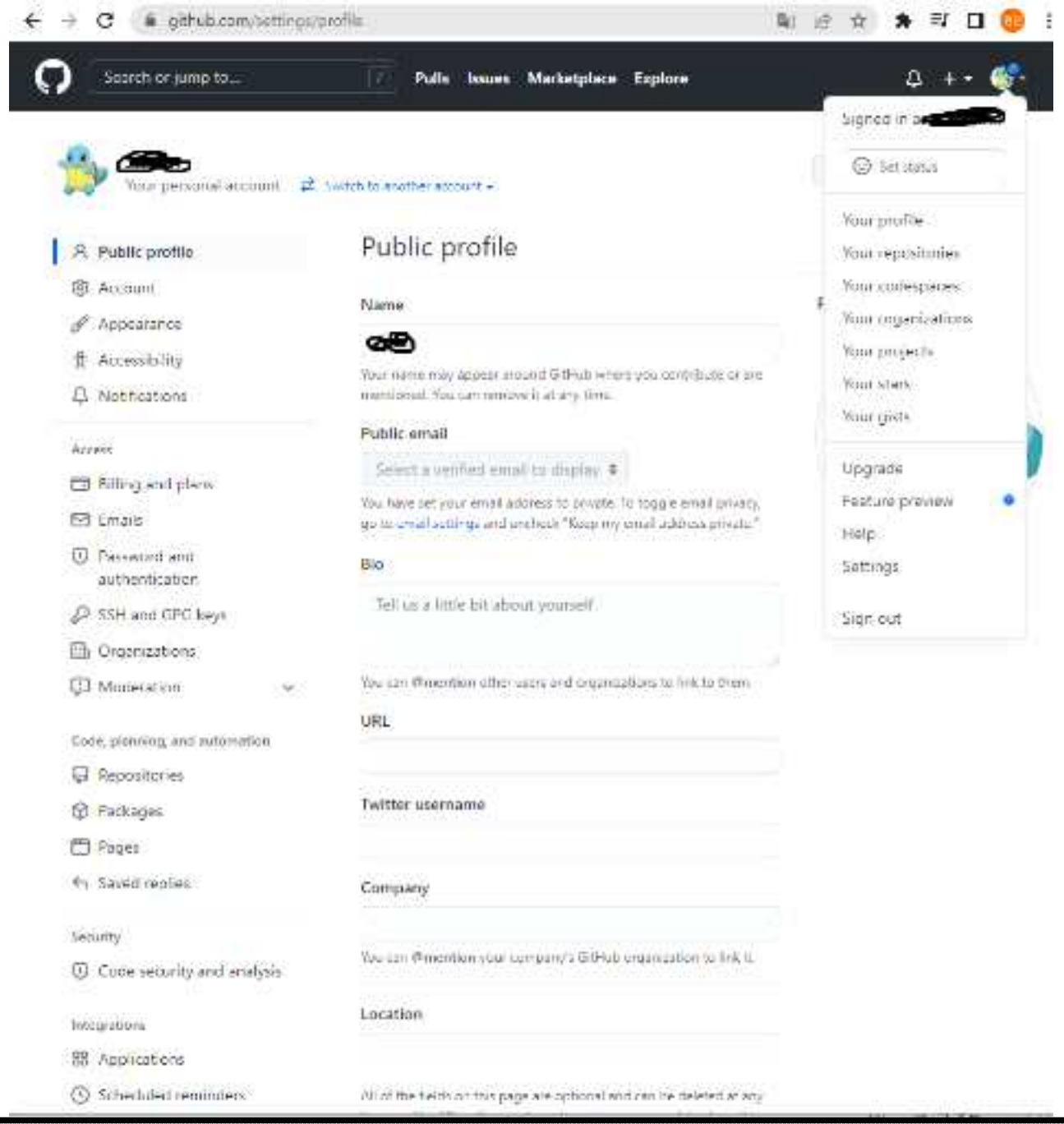
V. 파일업로드

토큰을 받기 위해서는 오른쪽과 같이

자신의 프로필을 눌러서 하단의 Settings로 들어간다.

Settings로 들어온 후 왼쪽 하단을 보면

Developer Settings가 있는데 그것을 클릭해주자



Ⅴ. 파일업로드

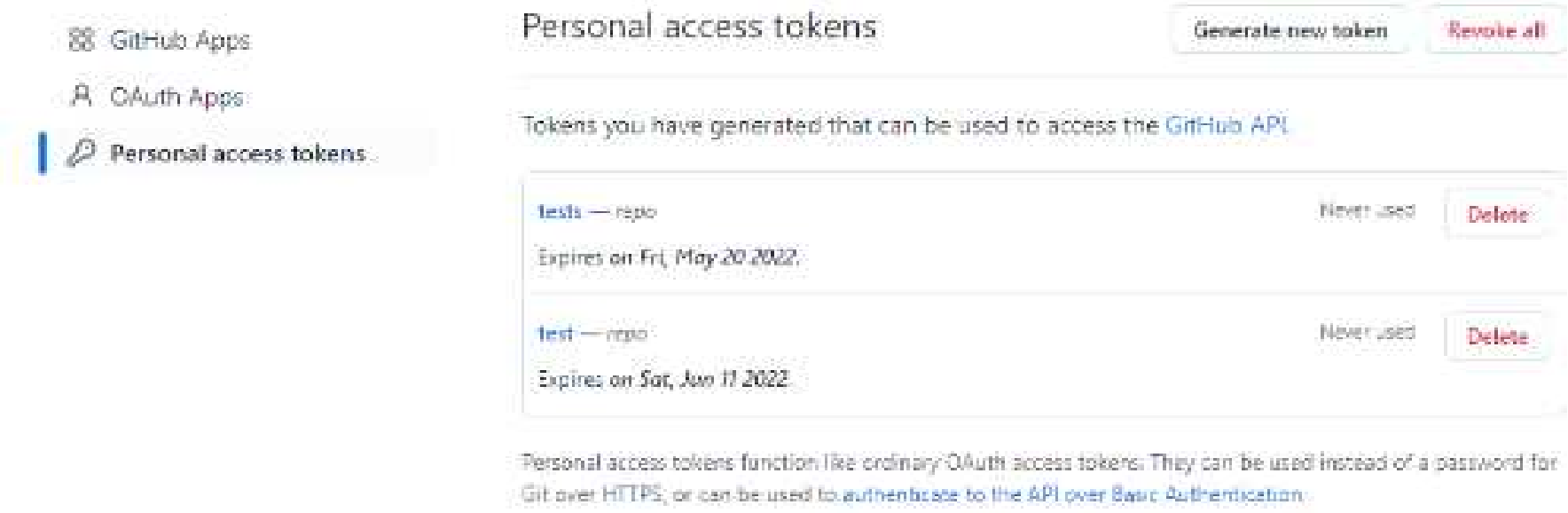
개발자 셋팅으로 들어오면 하단과 같은 창이 뜨는데

메뉴 가장 아래 Personal access tokens에서 토큰을 받을 수 있다.

Generate new token을 클릭해주고

Note부분은 토큰의 사용처를 묻는 공간이니 project정도 적으면 된다. <나는 test라 적어서 아래와 같이 나왔다.>

소스트리에서 사용하는 거면 repo부분만 체크박스를 클릭하고 토큰을 만들면 된다.



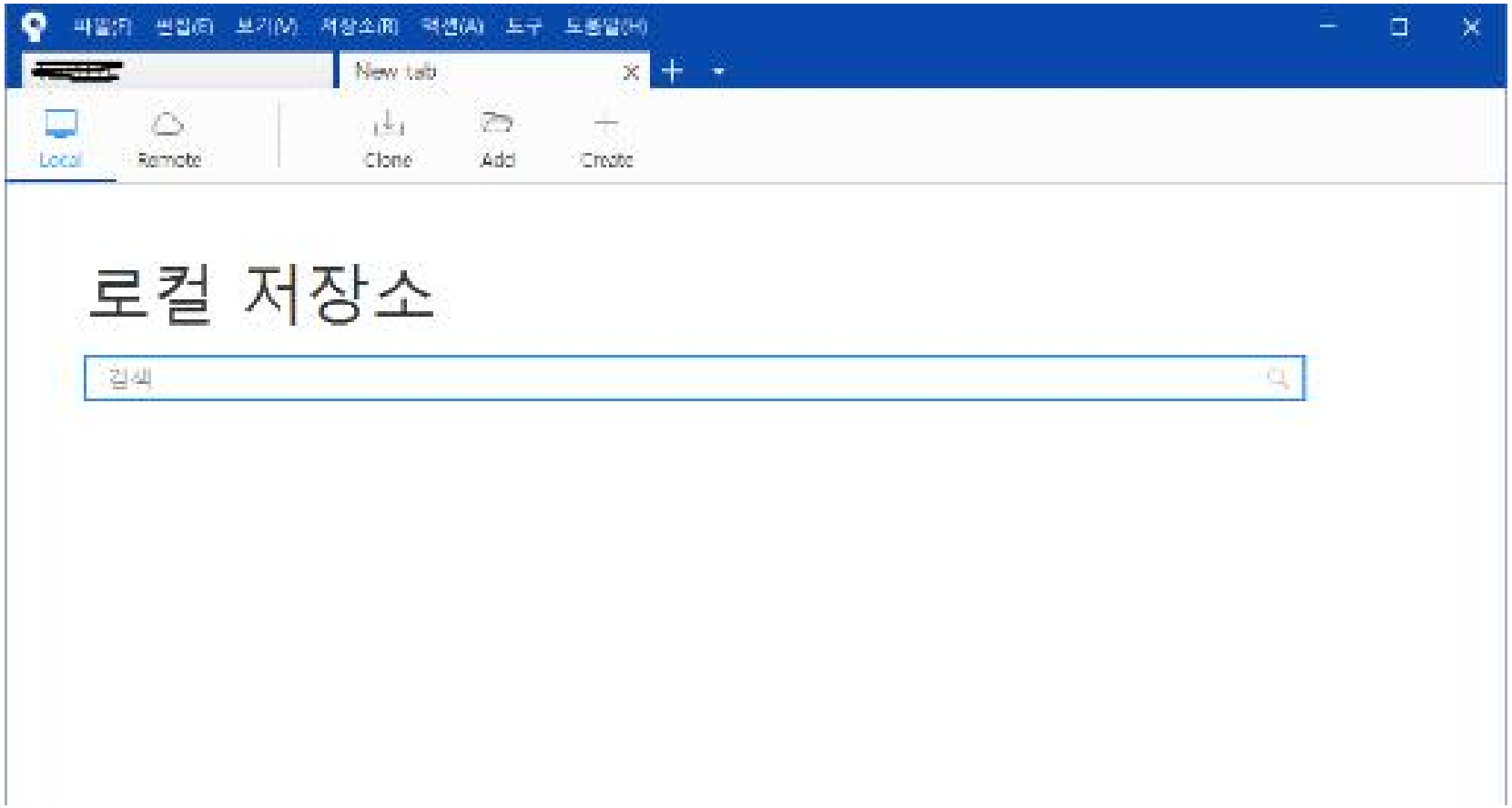
더 궁금한 부분은 구글로

찾아보길 권장한다.

V. 파일업로드

소스트리를 이용해서 파일을 내 저장소로 올려보겠다.

가장 먼저 아까 실행해 둔 소스트리 창을 열어보자

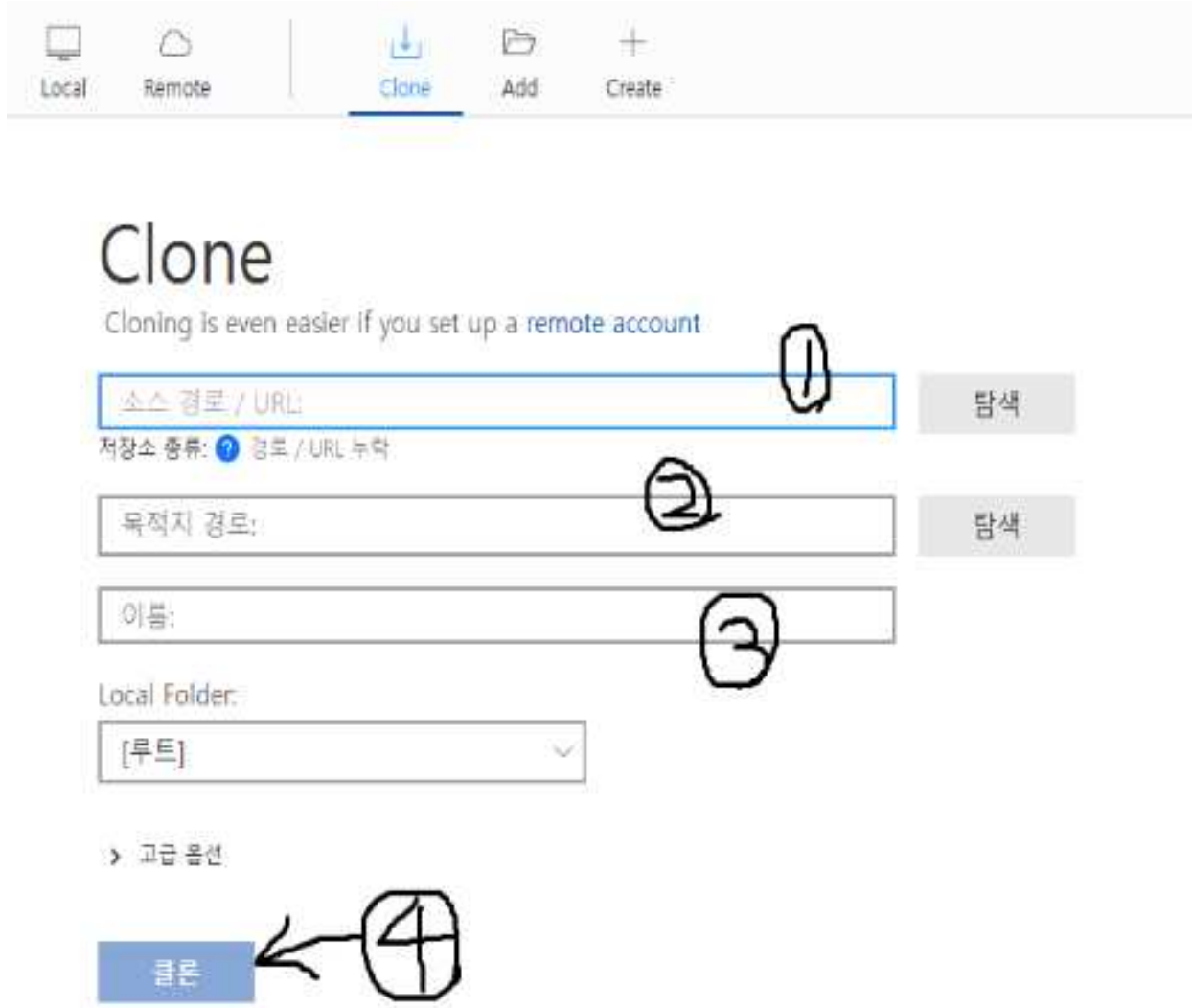


중간에 보이는 Clone을 클릭한다.

Clone을 통해 깃 저장소를 불러온다.

Ⅴ. 파일업로드

클론을 눌렀을 때 보여지는 모습이다.



1. 아까 받아둔 레파지토리 주소를 적는다.

2. 내 로컬 저장소 위치를 설정한다.
   로컬저장소는 내가 파일을 올릴 폴더를 의미한다.

3. 이름부분은 내가 지정한 파일 이름대로 설정되니
   변경해도 되지만 굳이 바꾸지 않아도 된다.

4. 클론 버튼을 눌러준다.

Ⅴ. 파일업로드

클론 버튼을 클릭하면 로그인 창 비슷한게 나올 것이다.

여기서 필요한게 아까 저장해둔 토큰 번호 이다.

로그인창 중간쯤에 보면 token이라는 버튼이 보이고 클래해서 들어가면 글입력 칸 하나가 보일것이다.

그 칸에 아까 저장해둔 토큰 번호를 적어 넣으면 된다.

혹시 토큰 번호를 입력하는 창이 안뜨는 사람들도 있을 것이다.

그럴 땐 클론시 레파지토리저장소 주소를 작성할 때 중간에 내 토큰 번호를 넣어두면 된다.

“https://github.com/a/b.git” >> 라는 저장소가 있을때

“https://토큰번호@github.com/a/b.git” >> 이렇게 중간에 토큰번호를 넣어주면 된다.

Ⅴ. 파일업로드

소스트리에 클론이 생성되었다면 내가 지정한 로컬저장소에 텍스트 파일을 하나 만들어보자 .

텍스트 파일을 만든 후 조금 기다리면 소스트리에서 자동으로 파일을 보여준다.

나타난 파일 오른쪽에 보면 + 버튼이 있다. 눌러주게 되면 내가 이 파일을 깃 허브에 올릴 준비를 하겠다는 의미이다.

하단의 메모칸은 이 파일에 대한 설명을 적어주는 공간이다.

개발자끼리의 약속으로는 메모란에 첫 번째 줄은 변경점 및 추가파일에 대한 설명을 간략하게 적는다.

두 번째 줄은 공백으로 둔 뒤, 세 번째 줄부터 변경점에 대해 자세한 설명을 적어주면 된다.

Ⅴ. 파일업로드

내가 만든 파일이 올라간것이 화면으로 보일것이다.

상단에 Push를 눌러 내 깃 저장소로 보내주자

Push가 완료되면 내 깃허브 계정으로 돌아가보면 내 레파지토리에 파일이 올라와 있는것을 확인 할 수 있다.

이 과정이 파일 올리기의 기초과정이다.

Ⅴ. 파일업로드

두 번째 파일을 만들어서 올려보자.

내 로컬 저장소 파일에 다른 이름의 텍스트 문서를 만들어보자

파일은 만든 후 소스트리로 돌아와 왼쪽 상단의 Commit (커밋)을 클릭해보면

방금 만든 파일이 올라와있다.

앞서 진행한 방법과 동일하게 진행하면 된다.

Ⅴ. 파일업로드

* 깃 이그노어 설정

이그노어는 내가 깃 허브저장소에 올리고 싶지 않은 파일들을 분류해 놓을 수 있다.

소스트리에서 이그노어 설정은 간단하다.

오른쪽 상단에 보이는 설정으로 들어간 뒤

상단에 고급을 클릭해준다.

그럼 첫 번째 줄에 저장소 무시 목록이라고 있는데 편집을 클릭해서 메모장으로 열어준다.

메모장 속에 자신이 포함시키고 싶지 않은 파일들의 확장자 명 혹은 파일 전체 이름을 적어주면

소스트리를 사용해서 Push할 때 지정한 파일들은 제외된다.

Ⅵ. 팀 프로젝트 준비

1) Organization Team생성

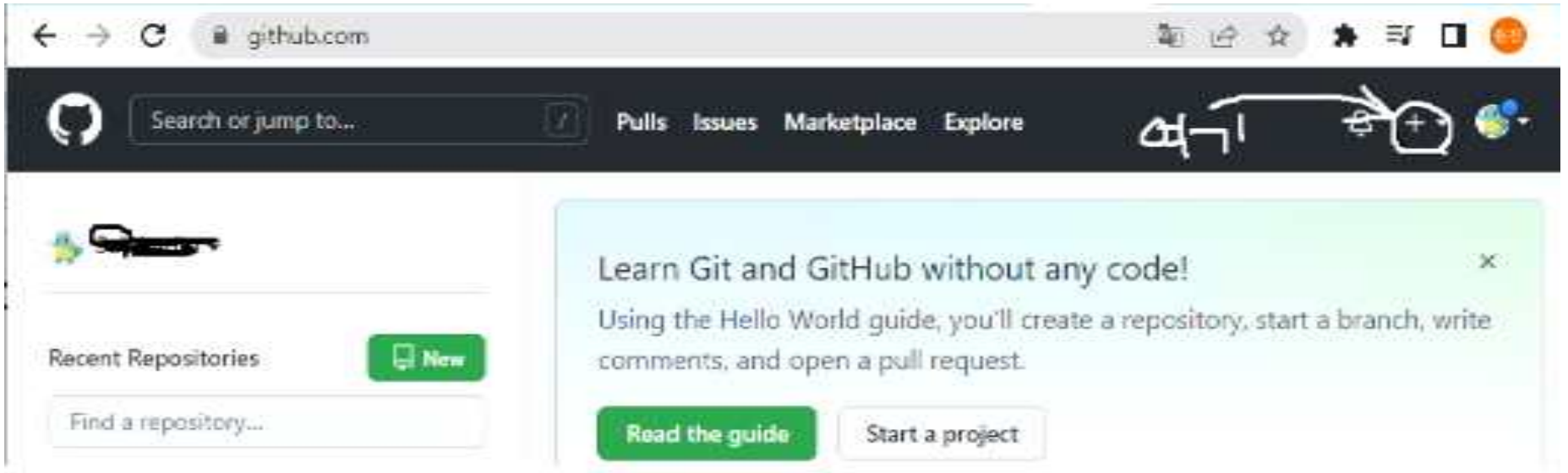
Organization Team은 개인 저장소가 아닌 팀별 저장소를 만들어 줄 수 있다.

개인 레파지트리를 생성했을 때와 동일하게 + 버튼을 클릭해준다.

New Organization을 클릭해 만들어준다.

우리 프로젝트는 무료로 만들어도 좋기때문에 무료로 만들면 된다.

위 계정은 팀별로 1개만 생성하면 되기 때문에 자세하게 알고 싶은 사람은 구글을 추천한다.



Ⅵ. 팀 프로젝트 준비

1) Organization Team생성

조직 계정이 생성이 되면 개인 계정의 레파지토리를 만들듯이 조직 계정으로 레파지토리를 만들고

소스트리에서 연동할때도 조직 레파지토리의 저장소 주소를 가져다가 사용하면 된다.

팀원을 초대한 후 권한을 부여할 수 있는데, 같이 작업을 진행해야 하기 때문에

write정도의 권한을 부여하는 것이 좋다.

다른 권한에 대해 자세하게 알고 싶은 사람은 따로 조사를 해보는 것을 추천한다.

조직 계정의 master권한은 만든 사람에게 주어진다.

Ⅵ. 팀 프로젝트 준비

2) 소스트리 사용해서 팀 프로젝트 진행 예시

2명으로 이루어진 팀이 프로젝트를 진행한다고 가정해 보겠다.

먼저 팀원은 조직 계정의 레파지토리 저장소의 주소를 가져와 소스트리에 연결시켜둔다.

후에 소스트리에서 프로젝트 폴더를 커밋후 푸시해준다.

프로젝트 폴더가 모두에게 공유 된 후 프로젝트는 시작된다고 볼 수 있다.

먼저 각각의 팀원의 브랜치를 설정해 두는 것이 먼저가 되어야 한다.

Ⅵ. 팀 프로젝트 준비

2) 소스트리 사용해서 팀 프로젝트 진행 예시

오른쪽과 같이 로컬 저장소와

프로젝트 워크스페이스와 동일하게 설정했다.

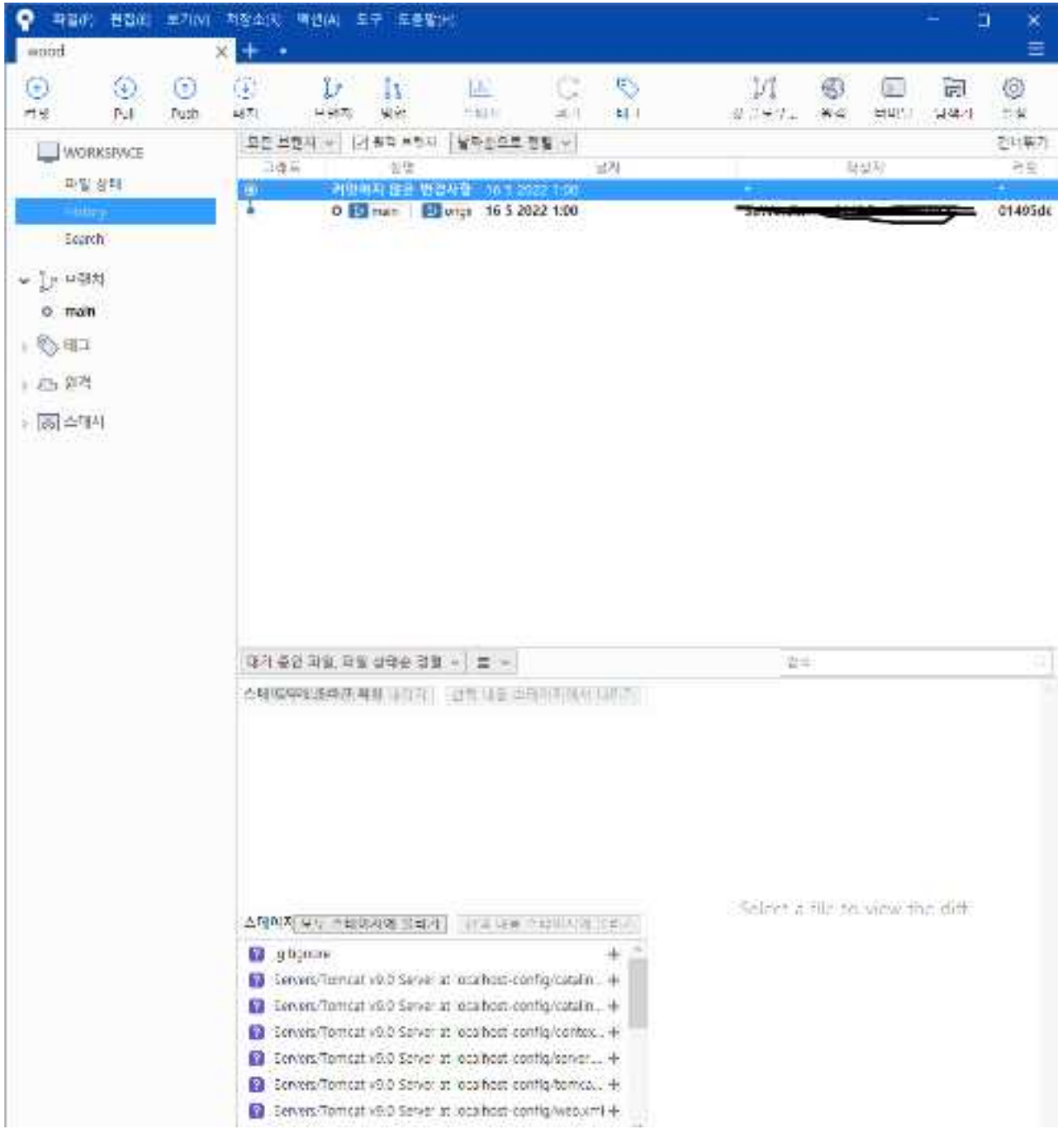
A가 파일을 만들고 B가 Pull을 받으면

A가 올린 파일 그대로를 받을 수 있다.

이 다음 진행할 것은 팀원 별로 브랜치를 만든다.

브랜치는 앞선 설명과 같이

개인이 작업을 진행 할 공간을 만들어 주는 것이다.



Ⅵ. 팀 프로젝트 준비

2) 소스트리 사용해서 팀 프로젝트 진행 예시

윗쪽 상단에 있는 브랜치를 누른다.

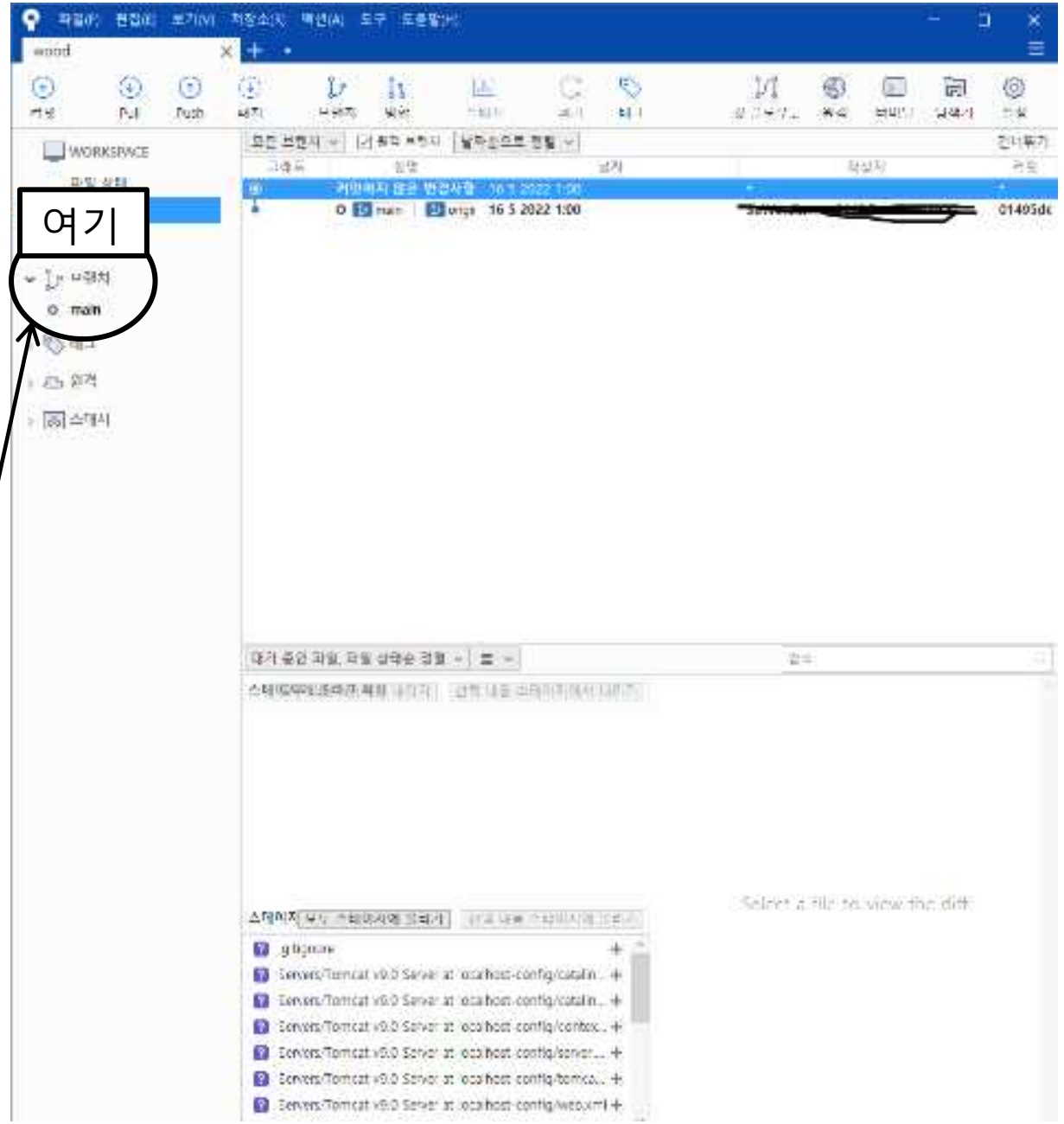
개인이 담당하는 기능별로 브랜치를 만들어도 된다.

브랜치가 만들어 졌으면 왼쪽 부분 브랜치 부분에

자신이 만든 브랜치가 생성되었음이 표시되어 있다.

브랜치가 생성되었으면 이제 그 브랜치에서 작업을

진행하면 된다.



Ⅵ. 팀 프로젝트 준비

2) 소스트리 사용해서 팀 프로젝트 진행 예시

브랜치를 만들었을 때 오른쪽과 같이 나와있을 것이다.

메인과 브랜치 옆 o표시는 현재 작업중인 곳을 나타낸다.

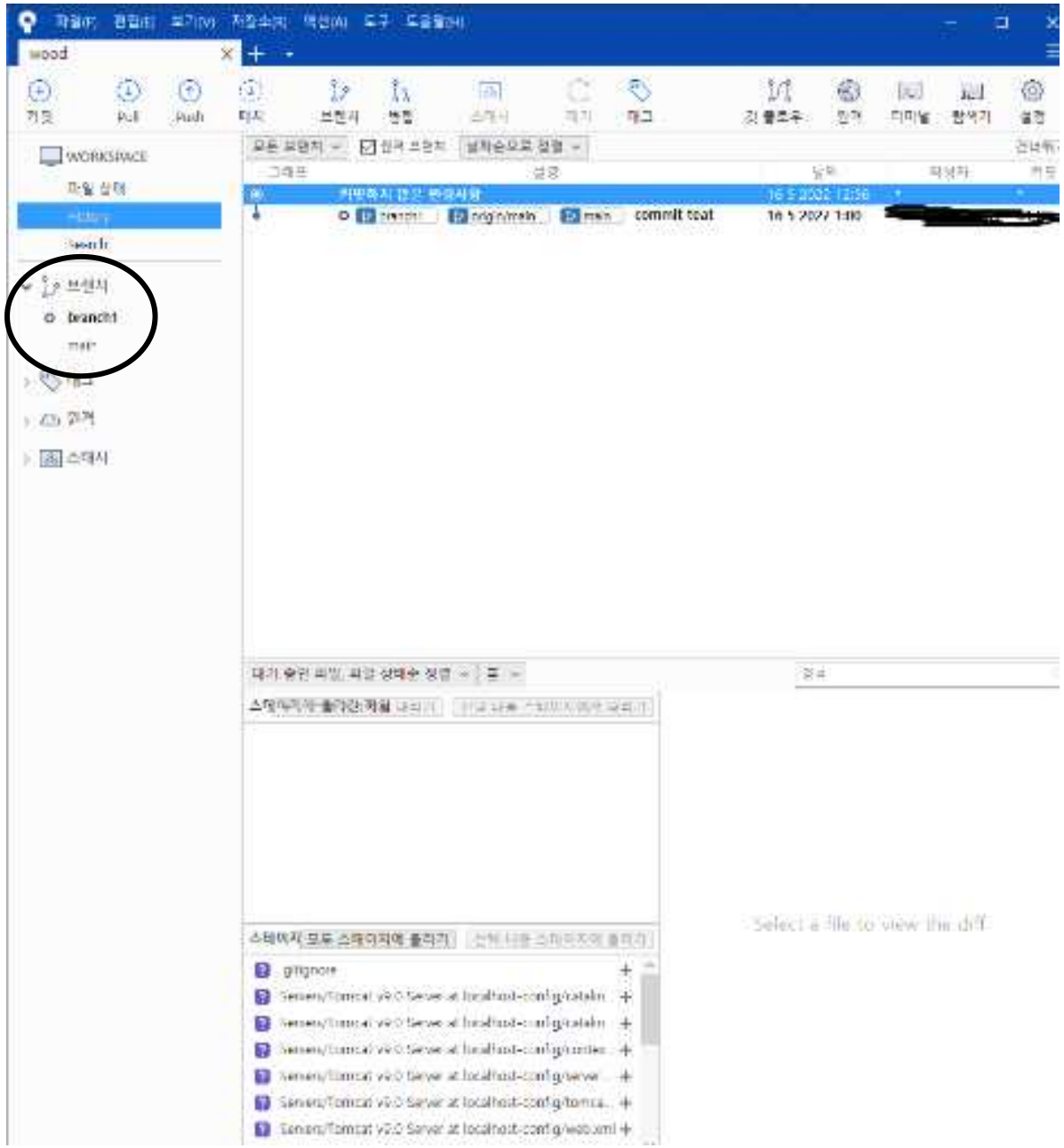
따라서 브랜치에서 작업하고 싶으면

오른쪽과 같이 브랜치에 o표시가 있을때 하면된다.

메인에서 파일을 올리고 싶을땐

브랜치에 있는 메인을 두번 클릭하면 메인으로 넘어간다.

그 상태에서 파일은 Push해주면 된다.



Ⅵ. 팀 프로젝트 준비

2) 소스트리 사용해서 팀 프로젝트 진행 예시

브랜치에서 메인으로 파일을 보내는
예시를 하나 들어주겠다.

1. 먼저 브랜치에 파일을 Commit한다.

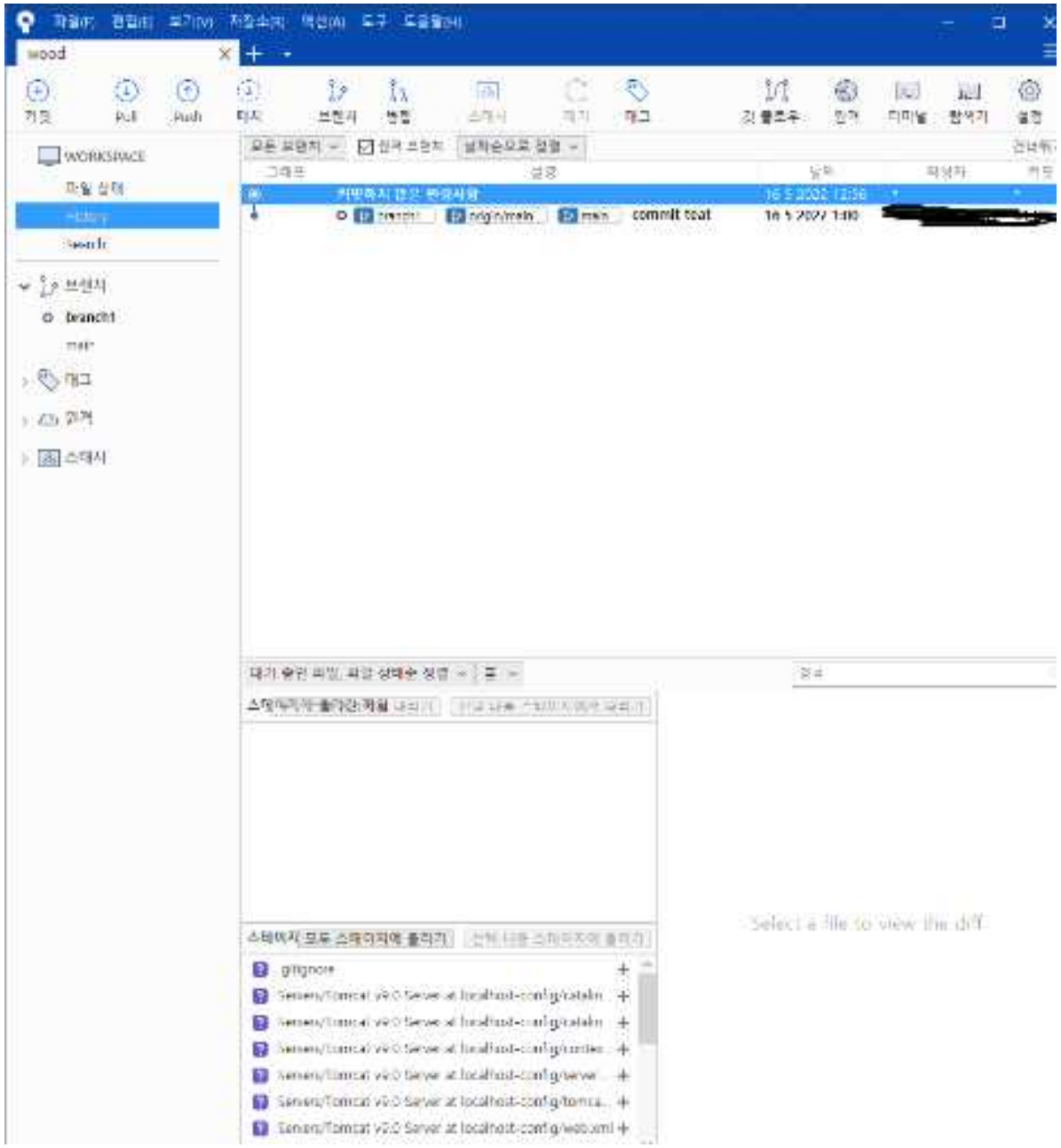
2. 커밋받은 파일을 확인한다
   여기서 Push하면 브랜치에만 파일이 올라간다.

3. 메인으로 넘어간다

4. 병합을 누른 뒤 작업한 파일을 누르고,

5. 하단의 확인을 눌러준다.

6. 마지막으로 Push를 눌러 깃 허브 저장소에 보낸다



Ⅵ. 팀 프로젝트 준비

2) 소스트리 사용해서 팀 프로젝트 진행 예시

하지만 병합(Marge)는 각자 하는것보단 한 사람이 지정되어 있는 상태에서

그 사람만이 병합을 진행하는 것이 좋다.

따라서 프로젝트를 진행할 경우 다른 인원들을 브랜치에서만 작업을 진행하고

담당자 혹은 Master가 main브랜치를 담당해

병합을 따로 진행하는 것이 좋다.

< 병합은 내 파일이 아니더라도 각각의 브랜치에서 Push해주면 병합할 수 있다. >

Ⅶ. 주의할 점

파일을 Push할 때 여러 명이 한번에 올려버리면 충돌이 일어나 파일들이 망가질 수 있다.

한 사람이 올리면 다음 사람이 올리는 방식으로 천천히 진행해야 한다.

따라서 프로젝트 진행할 때는 서로의 소통이 중요하다.

이름이 같은 파일을 올릴경우 덮어쓰기가 된다.

물론 깃허브 내에선 히스토리 기록을 확인할 수 있기 때문에 큰 걱정은 하지 않아도 된다.

하지만 혹시 모를 상황에 대비해 원초적인 백업을 추천한다.

그날 프로젝트 파일을 모두가 올린 뒤 문제 없이 동작했다면 로컬저장소를 zip으로 묶는 것이다.

다음 날에 프로젝트가 망가지면 그 전날 코드를 가져올 수 있게 대비해두고 매일매일 진행해 두는 것이 좋다